NATURAL LANGUAGE

A Two-Stage Method for Active Learning of Statistical Grammars

Markus Becker
School of Informatics
University of Edinburgh
m.becker@ed.ac.uk

Miles Osborne
School of Informatics
University of Edinburgh
miles@inf.ed.ac.uk

Abstract

Active learning reduces the amount of manually annotated sentences necessary when training state-of-the-art statistical parsers. One popular method, *uncertainty sampling*, selects sentences for which the parser exhibits low certainty. However, this method does not quantify confidence about the current statistical model itself. In particular, we should be less confident about selection decisions based on low frequency events. We present a novel two-stage method which first targets sentences which cannot be reliably selected using uncertainty sampling, and then applies standard uncertainty sampling to the remaining sentences. An evaluation shows that this method performs better than pure uncertainty sampling, and better than an ensemble method based on bagged ensemble members only.

1 Introduction

State-of-the-art parsers [Collins, 1997; Charniak, 2000] require large amounts of manually annotated training material, such as the Penn Treebank [Marcus *et al.*, 1993], to achieve high performance levels. However, creating such labelled data sets is costly and time-consuming. Active learning promises to reduce this cost by requesting only highly informative examples for human annotation. Methods have been proposed that estimate example informativity by the degree of uncertainty of a single learner as to the correct label of an example [Lewis and Gale, 1994] or by the disagreement of a committee of learners [Seung *et al.*, 1992]. This paper is concerned with reducing the manual annotation effort necessary to train a state-of-the-art lexicalised parser [Collins, 1997].

Uncertainty-based sampling has been successfully applied to the same problem problem [Hwa, 2000]. Here, sentences are selected for manual annotation when the entropy over the probability distribution of competing analyses is high. Entropy quantifies the degree of uncertainty as to the correct analysis of a sentence.

A problem with active learning methods such as uncertainty sampling is that they have no method for dealing with the consequences of low counts. For example, the parse tree probability of the most likely reading in a peaked distribution may depend on a probability which has been unreliably estimated from an as yet rarely observed event. In this case, the model would indicate certainty about a particular analysis where indeed it is not confident. In general, we would like to place less confidence in selection decisions based on entropy over probability distributions involving low frequency events. However, sentences whose predicted parse was selected on the basis of infrequent events may well be informative. Since entropy will not in itself always allow us to reliably select such examples for labelling, we need to consider other mechanisms.

We propose a novel two-stage method which first selects unparsable sentences according to a bagged parser, and applies uncertainty sampling to the remaining sentences using a fully trained parser. Evaluation shows that this method performs better than single parser uncertainty sampling, and better than an ensemble method with bagged ensemble members.

To explain our results, we show empirically that entropy and f-measure are negatively correlated. Thus, selection according to entropy tends to acquire annotations of sentences with low f-measure under the current model. An oracle-based experiment demonstrates that preferably selecting low f-measure sentences is indeed beneficial and explains why uncertainty sampling is successful in general. Furthermore, we find that exactly those sentences which our proposed methods targets show no such correlation between entropy and f-measure. In other words, entropy will not reliably identify informative examples from this subset, even though these sentences have below average f-measure and should be particularly useful. These findings help to explain why the proposed method is a successful strategy.

2 Active Learning Methods

Popular methods for active learning estimate example informativity with the uncertainty of a single classifier or the disagreement of an ensemble of classifiers.

Uncertainty-based sampling (or *tree entropy*) chooses examples with high entropy of the probability distribution for a single parser [Hwa, 2000]:

$$f_M^{te}(s,\tau) = -\sum_{t\in\tau} P_M(t|s)\log P_M(t|s) \qquad (1)$$

where τ is the set of parse trees assigned to sentence s by a stochastic parser with parameter model M. Less spiked

distributions have a higher entropy and indicate uncertainty of the parse model as to the correct analysis. Thus, it will be useful to know their true parse tree.

Ensemble-based methods for active learning select examples for which an ensemble of classifiers shows a high degree of disagreement. *Kullback-Leibler divergence to the mean* quantifies ensemble disagreement [Pereira *et al.*, 1993; McCallum and Nigam, 1998]. It is the average Kullback-Leibler divergence between each distribution and the mean of all distributions:

$$f^{kl}_{\mathcal{M}}(s,\tau) = \frac{1}{k} \sum_{M \in \mathcal{M}} D(P_M || P_{avg}) \quad (2)$$

where $\mathcal{M}$ denotes the set of k ensemble models, P_{avg} is the mean distribution over ensemble members in $\mathcal{M}$, $P_{avg} = \sum_M P_M(t|s)/k$, and $D(\cdot||\cdot)$ is KL-divergence, an information-theoretic measure of the difference between two distributions. It will be useful to acquire the manual annotation of sentences with a high *KL-divergence to the mean*. This metric has been applied for active learning in the context of text classification [McCallum and Nigam, 1998].

3 A Novel Two-Stage Selection Method

Acquiring the correct analysis of a sentence of which the predicted analysis was selected on the basis of infrequent events may well be informative. Since entropy itself will not allow us to reliably select such examples for labelling, we need to consider other mechanisms. A simple, but effective method is to eliminate some infrequent events from the parsing model. Simply bagging the current training set, and retraining the parser on this set allows to identify such examples for labelling.

Bagging is a general machine learning technique that reduces variance of the underlying training methods [Breiman, 1996]. It aggregates estimates from classifiers trained on bootstrap replicates (bags) of the original training data. Creating a bootstrap replicate entails sampling with replacement n examples from a training set of n examples. A bootstrap replicate will not only perturb all event counts to some degree, but will inevitably eliminate some of the low frequency event types.

The proposed method operates in two stages. We first select sentences which are unparsable according to a single bagged version of the parser, but (possibly) parsable under the current fully trained model. From the remaining sentences, we select those with the highest entropy as determined by the fully trained model. We can express this formally as follows:

$$f^{two}_{M,M'}(s,\tau) = \max(f^{te}_M(s,\tau_M), failure(s,M')) \quad (3)$$

where f^{te}_M is tree entropy according to a fully trained model M, defined in (1). The function $failure(s,M')$ returns infinity when sentence s is parsable given bagged parser model M', and 0 otherwise.

4 Experimental Setup

For our experiment, we employ a state-of-the-art lexicalised parser [Collins, 1997].[1] We employ default settings without expending any effort to optimise parameters towards the considerably smaller training sets involved in active learning.

In common with almost all active learning research, we compare the efficacy of different selection methods by performing simulation experiments. We label sentences of sections 02 - 22 from the Penn WSJ treebank [Marcus *et al.*, 1993], ignoring sentences longer than 40 words.

We report the average over a 5-fold cross-validation to ensure statistical significance of the results. In a single fold, we randomly sample (without replacement) an initial labelled training set of a fixed size – 500 or 2,000 sentences, depending on the experiment – and a test set of 1,000 sentences. The remaining sentences constitute the global pool of unlabelled sentences (ca. 37,000 sentences). For a realistic experiment, we tag the test set with the TnT part-of-speech tagger as input for the parser [Brants, 2000]. We train TnT on 30,000 sentences in the global resource. In a 5-fold cross-validation, the parser has 88.8% labelled precision and 88.6% labelled recall, when trained on 37k sentences and applied to test sets of 1,000 sentences.[2]

We randomly sample (without replacement) a subset of 1,000 sentences from the global pool in each iteration. From this subset, 100 sentences are selected for manual annotation according to the current sample selection method. Then, annotated sentences are added to the training set.

For consistent comparison across methods, we evaluate test set performance of a single parser trained on the entirety of the labeled training data at each step, regardless of the selection method being a single or an ensemble method.

Length balanced sampling For situations such as active learning for parsing, the sentences in question need a variable number of labelling decisions. This may confound sample selection metrics and it is therefore necessary to normalise. For example, tree entropy may be directly normalised by sentence length [Hwa, 2000], or by the binary logarithm of the number of parser readings [Hwa, 2001].

We use the following method to control for sentence length in sample selection: Given a batch size b, we randomly sample b sentences from the pool and record the number e_l of selected examples for sentence length l. Then, for all lengths $l = 1, 2, \ldots 40$, we select from all sentences in the pool of length l the e_l examples with the highest score according to our sample selection metric. Of course, the union of these sets will have b examples again. Since we randomly sampled the batch from the pool, we may assume that the batch distribution reflects the pool distribution, in particular wrt. the distribution of sentence lengths.

[1] We use the flexible reimplementation of the parser by Dan Bikel, developed at the University of Pennsylvania [Bikel, 2004]. It can be obtained at `http://www.cis.upenn.edu/~dbikel/`

[2] It would be desirable for methodological reasons to automatically tag the global resource, too. However, our corpus split scheme does not leave enough disjoint training material for the tagger, so we use the gold standard tags for the pool sentences.

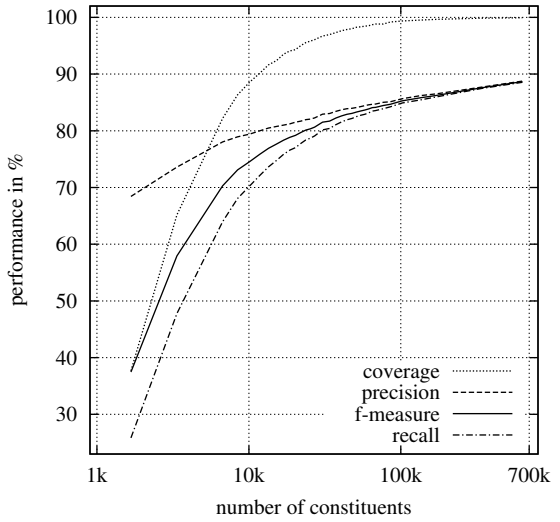


Figure 1: A random sampling learning curve. The maximal training set has 37k sentences (630k constituents). The number of constituents is given on a log scale.

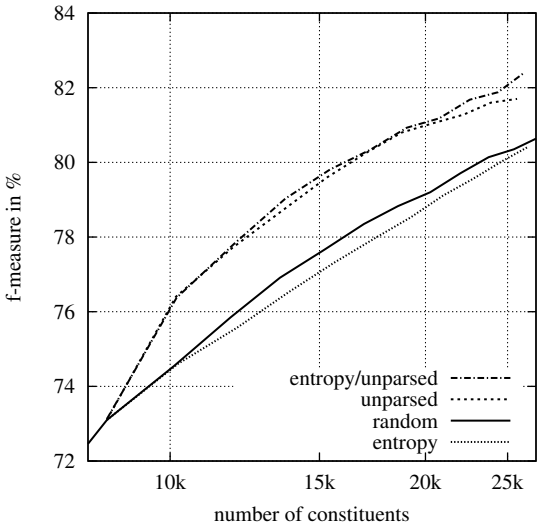


Figure 2: Comparison of f-measure learning curves. Number of constituents is given on a log scale.

This method effectively reproduces the sentence length profile of the original corpus by construction and therefore guards against the selection of sentence length biased subsets. Furthermore, it is equally applicable for all metrics and allows a direct comparison between metrics. Note that this method is applicable in general for the sample selection of sequential data where one may expect to find correlation between sample length and score.

Relevant evaluation measures Active learning for parsing is typically evaluated in terms of achieving a given f-measure for some amount of labelling expenditure. The cost of acquiring manually annotated training material is given in terms of the number of constituents. F-measure itself is a composite term, being composed of precision and recall [Black *et al.*, 1991]. Fig. 1 shows a learning curve for a random sampling experiment. We see that precision and recall do not increase at the same rate. For this reason, it may well be advantageous to aggressively increase recall with minimal impact on precision (formulate stronger: still want to increase precision). One way of achieving this is to pursue sentences which cannot be parsed.

5 Results

The experiments in this section address the following questions. Is it generally useful to select unparsable sentences for manual annotation? What is the gain of using the novel two-stage method over a state-of-the-art uncertainty-based sample selection method? Given that the two-stage method has a bagged component, how does it compare against a state-of-the-art ensemble-based method which employs bagged ensemble members?

Method	Cost	Reduction
random	23200	N/A
entropy/unparsed	16100	30.6%
unparsed	16400	29.4%
entropy	24500	-5.7%

Table 1: Annotation cost to reach 80% f-measure, and reduction over random sampling.

Including/excluding unparsed sentences In this experiment, we compare methods which do or do not include unparsed sentences in the batch of selected examples. Acquiring the correct parse tree of an unparsable sentence increases the size of the model structure of the grammar and, presumably, helps to increase coverage in the test set.

A very simple method, **unparsed**, preferably includes unparsed pool sentences in the batch. Should the number of unparsed sentences fall short of the batch size, we randomly sample parsable sentences from the pool to fill the batch. By contrast, **entropy** only selects parsable sentences with high entropy. The method **entropy/unparsed** preferably selects unparsed pool sentences, and fills the batch with high entropy examples. We may view this method as being composed of a binary parsability component, and a gradual uncertainty component. The baseline is **random**, a parser trained on randomly sampled training sets of different sizes.

We start with an initial training set of 500 randomly sampled sentences, containing 8,400 labeled constituents and continue for 10 rounds until 1,500 sentences have been sampled (ca. 26k constituents). Here, as in all subsequent experiments we employ length balanced sampling, cf. Sec. 4.

Methods **unparsed** and **entropy/unparsed** perform consistently better than **random** (Fig. 2). Note that their performance is nearly identical until more than 20k constituents have been labelled. Method **entropy** performs consistently

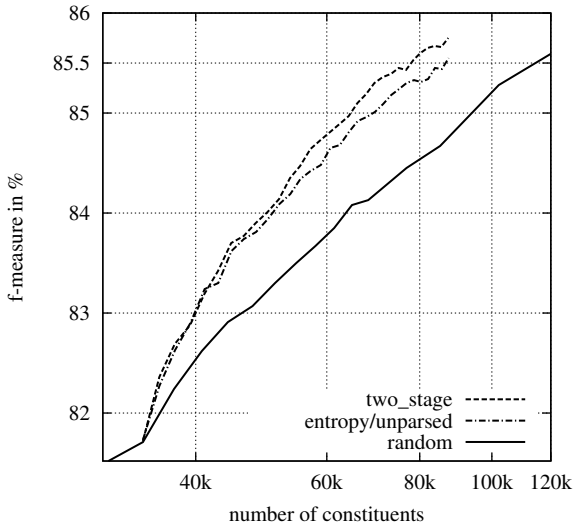


Figure 3: The new two-stage method versus state-of-the-art uncertainty sampling.

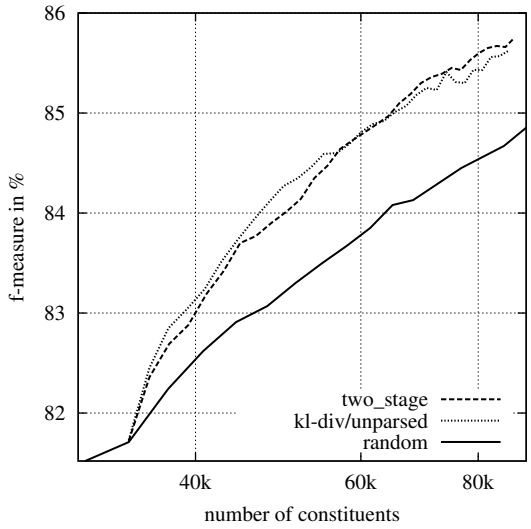


Figure 4: The new two-stage method versus a state-of-the-art ensemble-based method.

Method	Cov	Prec	Rec	Fm
random	95.8	82.0	78.6	80.3
entropy/unparsed	98.3	82.9	81.2	82.1
unparsed	98.5	82.4	80.9	81.7
entropy	94.4	82.8	77.6	80.1

Table 2: Parseval values for different metrics after 25,000 constituents have been annotated.

Method	Cost	Reduction
random	115600	N/A
two_stage	77900	32.6%
entropy/unparsed	86700	25.0%

Table 3: Annotation cost to reach 85.5% f-measure, and reduction over random sampling.

worse than **random**.

Methods **unparsed** and **entropy/unparsed** reduce the amount of labeled constituents necessary to achieve 80% f-measure by around 30% as compared to **random**, while **entropy** actually increases the cost by 5.7% (Tab. 1).

We also compare performance across methods for the same amount of annotation effort. Tab. 2 shows precision, recall, and f-measure after labelling 25k constituents. Methods **unparsed** and **entropy/unparsed** have considerably higher coverage than **random**, **entropy** has lower coverage. While all methods show comparable values for precision, they differ decidedly in their recall values. The two methods which aggressively pursue unparsed sentences, **unparsed** and **entropy/unparsed**, have more than 3% points higher recall than **entropy**, and accordingly higher f-measures than **random** and **entropy**.

These results confirm the importance of including unparsed sentences. Doing so helps achieving better coverage and a higher recall value which directly translates into higher f-measure. Accordingly, all of the following experiments will include unparsed sentences in the batch. The negative result for the purely entropy-based method shows clearly that a naive application of uncertainty sampling may have adverse consequences. It is extremely important to consider which phenomena a selection method is targeting.

A two-stage selection method The novel method addresses problems with sentences which cannot be reliably selected with popular active learning methods. Therefore, we expect a gain in performance. Method **two_stage** preferably includes sentences which are unparsable according to a parser trained on a bagged version of the current training set. Then, the batch is filled with (parsable) sentences which have high entropy according to a second, fully trained parser.

Given that composite methods which preferably select unparsed sentences perform nearly uniformly well, we will now use a considerably larger initial training set in order to be able to observe differences between these methods. We start with 2,000 sentences (34k constituents), and continue for 30 rounds until a total of 5,000 sentences has been sampled (ca. 87k constituents).

Method **two_stage** performs consistently better than both **random** and **entropy/unparsed** (Fig. 3). It reduces the amount of labelled data necessary to reach 85.5% f-measure by 32.6% as compared to **random** (Tab. 3). The central result of this paper is that, to reach this level, **two_stage** reduces the number of constituents by 8,800 constituents against the state-of-the-art method **entropy/unparsed**: a reduction by a further 10.1%. Also, it has consistently higher precision and recall than **entropy/unparsed** after the labelling of 80k constituents (Tab. 4).

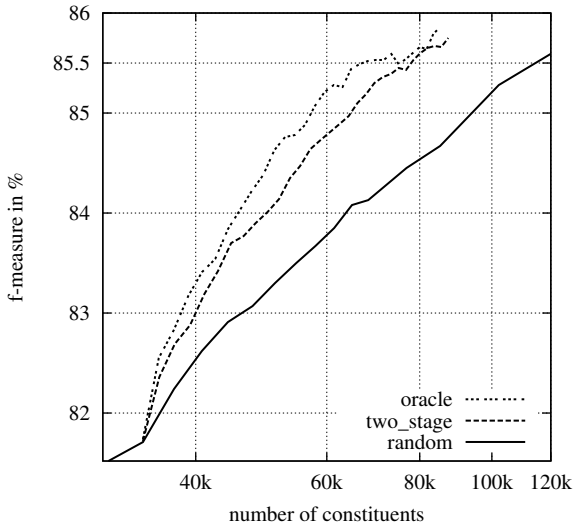


Figure 5: Comparison of f-measure learning curves. Number of constituents is given on a log scale.

Method	Cov	Prec	Rec	Fm
random	99.0	85.1	84.0	84.5
two_stage	99.7	86.0	85.2	85.6
entropy/unparsed	99.6	85.7	84.9	85.3

Table 4: Parseval values for different metrics after 80,000 constituents have been annotated.

An ensemble method We now compare performance of the new method against a state-of-the-art ensemble method, namely the KL-divergence to the mean for an ensemble of n bagged parsers, cf. Subsec. 2. The method **kl-div/unparsed** preferably selects unparsed pool sentences, and fills the batch with sentences having a high mean KL-divergence. (We consider a sentence as unparsed when at least one of the ensemble members fails to deliver an analysis.) As in the previous experiment, we start with 2,000 sentences and continue for 30 rounds. We set the ensemble size to $n = 5$.

Fig. 4 shows ensemble method **kl-div/unparsed** to perform better than **two_stage** until 57k constituents. After this point, **two_stage** performs clearly better than **kl-div/unparsed**. This is actually a surprising result, given that both methods perform similar jobs: They select unparsable sentences according to one or more bagged parsers, and then apply an information-theoretic measure, either entropy or KL-divergence. We conjecture that, after having filtered the difficult examples in the first stage, the second stage should make use of the information to be had from a fully trained parser. At any rate, our proposed method is conceptually simpler and also quicker to compute than an ensemble-based method.

6 Understanding the New Method

Acquiring the annotation of objectively difficult sentences should improve the parser. We employ an oracle-based experiment to test this claim. Method **oracle** selects sentences whose preferred parse tree (according to the current grammar) has low f-measure as determined against a gold-standard tree. Fig. 5 shows that method **oracle** performs consistently better than our best result, the new **two_stage** method. This suggests that a selection method which successfully targets difficult sentences (low f-measure) will perform well.

	Sentences	F-measure	Pearson
parsable	4,919	83.9%	-0.37
unreliable	112	71.5%	0.05

Table 5: Average f-measure and correlation coefficients between entropy and f-measure

In another experiment, we train the parser on a randomly sampled training set of 2,000 sentences, and apply it to a test set of 5,000 sentences. We are interested in the degree of correlation between the variables f-measure (preferred tree against gold-standard tree) and tree entropy. A correlation analysis over all 4,919 **parsable** sentences shows that the two variables are indeed (negatively) correlated, cf. Tab. 5. Pearson's coefficient is -0.37. Given the size of the considered data set, correlation is highly significant ($p = 0.01$). Selection according to entropy will thus tend to pick low f-measure sentences. Given the observation from the oracle experiment that it is beneficial to target low f-measure sentences, this finding explains why entropy is a useful selection method.

If we now apply a bagged version of the parser to the same test set, more sentences become unparsable since we eliminate some infrequent parse events. Focusing on the 112 **unreliable** sentences which are parsable under a fully trained model, but not under a bagged model, we find a Pearson coefficient between entropy and f-measure close to 0 (Tab. 5). In other words, entropy and f-measure are uncorrelated, and entropy cannot reliably select difficult examples within this class of sentences. What is more, the average f-measure within these **unreliable** sentences is more than 12 percentage points below average, indicating that acquiring their true parse trees will be particularly useful.

Note that the first stage of our new method targets exactly these kind of unreliable sentences. The above experiments demonstrate why the new method is indeed successful.

7 Related Work

Preferably selecting high entropy examples has been shown to be an effective method for parsing [Hwa, 2000]. Selecting unparsed sentences has been previously suggested because of their high uncertainty, e.g. in [Thompson *et al.*, 1999]. However, to the best of our knowledge, this effect has not been quantified before. Bagging ensemble members (or alternatively random perturbation of event counts) has been explored in the context of active learning by [Argamon-Engelson and Dagan, 1999; McCallum and Nigam, 1998]. In particular, Argamon-Engelson and Dagan have indicated that these methods target low frequency events. Bagging (and boosting) a parser ensemble has been employed to increase parser performance [Henderson and Brill, 2000]. However, the application of a bagged parser ensemble to active learning is

new. Density estimation has been suggested as a method to guard against selecting of outliers, e.g [McCallum and Nigam, 1998; Tang *et al.*, 2002]. This approach is orthogonal to our suggested new method, and combining the two may well result in even better performance.

8 Conclusion

We demonstrated a number of points in this paper. First, we investigated the effect of targeting unparsed sentences. This is a simple, but very effective way to increase labelled recall and thereby f-measure. This method has been used implicitly before, but to our knowledge the effect of this strategy has not been quantified previously. Secondly, we presented a novel, two-stage method which particularly targets sentences which cannot be reliably selected using popular active learning methods. We showed that the proposed method works better than uncertainty sampling alone. Also it compares favourably against a state-of-the-art ensemble method based on bagging. Finally, an oracle-based experiment indicated that targeting (objectively) difficult sentences is a good strategy. Furthermore, we demonstrated that entropy and f-measure are significantly correlated in general. However, they are uncorrelated for exactly the class of sentences of which our new method takes care. This explains why the new two-stage method performs well. In future work, we would like to investigate if the proposed two-stage method can be applied to applications other than parsing.

Acknowledgments

We would like to thank James Henderson, Mirella Lapata, and Andrew Smith for valuable discussions and comments on this work.

References

[Argamon-Engelson and Dagan, 1999] Shlomo Argamon-Engelson and Ido Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

[Bikel, 2004] Daniel M. Bikel. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511, 2004.

[Black *et al.*, 1991] E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings, Speech and Natural Language Workshop*, 1991.

[Brants, 2000] Thorsten Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, 2000.

[Breiman, 1996] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–14, 1996.

[Charniak, 2000] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, 2000.

[Collins, 1997] Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 1997.

[Henderson and Brill, 2000] John C. Henderson and Eric Brill. Bagging and boosting a treebank parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, 2000.

[Hwa, 2000] Rebecca Hwa. Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 45–52, 2000.

[Hwa, 2001] Rebecca Hwa. *Learning Probabilistic Lexicalized Grammars for Natural Language Processing*. PhD thesis, Harvard University, 2001.

[Lewis and Gale, 1994] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[Marcus *et al.*, 1993] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[McCallum and Nigam, 1998] Andrew McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*, pages 350–358, 1998.

[Pereira *et al.*, 1993] Fernando C.N. Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, 1993.

[Seung *et al.*, 1992] H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Computational Learning Theory*, pages 287–294, 1992.

[Tang *et al.*, 2002] M. Tang, X. Luo, and S. Roukos. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 120–127, 2002.

[Thompson *et al.*, 1999] C. A. Thompson, M. E. Califf, and R. J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning*, pages 406–414, 1999.

TimeML-Compliant Text Analysis for Temporal Reasoning

Branimir Boguraev and Rie Kubota Ando
IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA
bran@us.ibm.com, rie1@us.ibm.com

Abstract

Reasoning with time[1] needs more than just a list of temporal expressions. TimeML—an emerging standard for temporal annotation as a language capturing properties and relationships among time-denoting expressions and events in text—is a good starting point for bridging the gap between temporal analysis of documents and reasoning with the information derived from them. Hard as TimeML-compliant analysis is, the small size of the only currently available annotated corpus makes it even harder. We address this problem with a hybrid TimeML annotator, which uses cascaded finite-state grammars (for temporal expression analysis, shallow syntactic parsing, and feature generation) together with a machine learning component capable of effectively using large amounts of unannotated data.

1 Temporal Analysis of Documents

Many information extraction tasks limit analysis of time to identifying a narrow class of time expressions, which literally specify a temporal point or an interval. For instance, a recent (2004) ACE task is that of temporal expression recognition and normalisation (TERN; see http://timex2.mitre.org/tern.html). It targets absolute date/time specifications (*e.g. "June 15th, 1998"*), descriptions of intervals (*"three semesters"*), referential (relative) expressions (*"last week"*), and so forth. A fraction of such expressions may include a relational component (*"the two weeks since the conference"*, *"a month of delays following the disclosure"*), making them event-anchored; however, the majority refer only to what in a more syntactic framework would be considered as a 'temporal adjunct'. The TERN task thus does not address the general question of associating a time stamp with an event.

Deeper document analysis requires awareness of temporal aspects of discourse. Several applications have recently started addressing some issues of time. Document summarisation tackles identification and normalisation of time expressions [Mani & Wilson, 2000], time stamping of event clauses [Filatova and Hovy, 2001], and temporal ordering of events in news [Mani *et al.*, 2003]. Operational question answering (QA) systems can now (under certain conditions) answer *e.g. 'when'* or *'how long'* questions [Prager *et al.*, 2003].

Beyond manipulation of temporal expressions, advanced content analysis projects are beginning to define operational requirements for, in effect, temporal reasoning. More sophisticated QA, for instance, needs more than just information derived from 'bare' temporal markers [Pustejovsky *et al.*, 2003; Schilder & Habel, 2003]. Intelligence analysis typically handles contradictory information, while looking for mutually corroborating facts; for this, temporal relations within such an information space are essential. Multi-document summarisation crucially requires temporal ordering over events described across the collection.

A temporal reasoner requires a framework capturing the ways in which relationships among entities are described in text, anchored in time, and related to each other. Related are questions of defining a representation that can accommodate components of a temporal structure, and implementing a text analysis process for instantiating such a structure.

This paper describes an effort towards an analytical framework for detailed time information extraction. We sketch the temporal reasoning component which is the ultimate 'client' of the analysis. We motivate our choice of TimeML, an emerging standard for annotation of temporal information in text, as a representational framework; in the process, we highlight TimeML's main features, and characterise a mapping from a TimeML-compliant representation to an isomorphic set of time-points and intervals expected by the reasoner.

We develop a strategy for time analysis of text, a synergistic approach deploying both finite-state (FS) grammars and machine learning techniques. The respective strengths of these technologies are well suited for the challenges of the task: complexity of analysis, and paucity of examples of TimeML-style annotation. A complex cascade of FS grammars targets certain components of TimeML (time expressions, in particular), identifies syntactic clues for marking other components (related to temporal links), and derives features for use by machine learning. The training is on a TimeML annotated corpus; given the small—and thus problematic for training—size of the only (so far) available reference corpus (TimeBank), we incorporate a learning strategy developed

[1]This work was supported by the ARDA NIMD (Novel Intelligence and Massive Data) program PNWD-SW-6059.

to leverage large volumes of unlabeled data.

To our knowledge, this is the first attempt to use the representational principles of TimeML for practical analysis of time. This is also the first use of a TimeML corpus as reference data for implementing temporal analysis.

2 Motivation: Reasoning with Time

We are motivated by developing a useful, and reusable, temporal analysis framework, where 'downstream' applications are enabled to reason and draw inferences over time elements.

A hybrid reasoner [Fikes *et al.*, 2003], to be deployed in intelligence analysis, maintains a directed graph of time points, intervals defined via start and end points, and temporal relations such as BEFORE, AFTER, and EQUAL_POINT. The graph is assumed generated via a mapping process, external to the reasoner, from a (temporal) text analysis. Relations are operationalised, and temporal algebra evaluates instances, draws inference over goals, and broadens a base of inferred assertions on the basis of relational axioms. An example within the reasoner's inferential capability is:

(*find instances of* `?int` *such that* `(during ?int 2003)`).

Reasoning with relations such as *during* (associating an event with an interval), *costarts* (associating two events), instantiated for the example fragment: *"On 9 August Iran accuses the Taliban of taking 9 diplomats and 35 truck drivers hostage in Mazar-e-Sharif. The crisis began with that accusation."* would infer, on the basis of predicates like:

(during Iran-accuses-Taliban-take-hostages August-9-1998)
(costarts Iran-accuses-Taliban-take-hostages Iran-Taliban-Crisis)

that the answer to the question *"When did the Iranian-Taliban crisis begin?"* is *"August 9, 1998"*.

Details of this inferential process need not concern us here. We gloss over issues like enumerating the range of temporal relations and axioms, describing the reasoner's model of events (*e.g. Iran-accuses-Taliban-take-hostages*), and elaborating its notion of 'a point in time' (subsuming both literal expressions and event specifications). Operationally, a separate component maps temporal analysis results to a suitably neutral, and expressive, ontological representation of time (DAML-Time [Hobbs *et al.*, 2002]). This allows for a representation hospitable to first-order logic inference formalism—like the one assumed in Hobbs *et al.*—to be kept separate from surface text analysis: much like the traditional separation along the syntax-semantics interface.

We start from the belief that the representation for the reasoner is derivable from a TimeML-compliant text analysis.[2] TimeML is a proposal for annotating time information;*e.g.* the first example sentence above would be marked up as:

```
<SIGNAL sid="s1"> On </SIGNAL> <TIMEX3 tid="t1" type="DATE" value=
"1998-08-09"> 9 August </TIMEX3> Iran <EVENT eid= "e1" class= "I_ACTION">
accuses </EVENT> the Taliban of <EVENT eid="e4" class="OCCURRENCE"> taking
</EVENT> 9 diplomats and 35 truck drivers hostage in Mazar-e-Sharif. The <EVENT
eid="e8" class="OCCURRENCE"> crisis </EVENT> <EVENT eid="e12" class=
"ASPECTUAL"> began </EVENT> <SIGNAL sid="s2" type="DATE" mod="START"> with
</SIGNAL> that <EVENT eid="e16" class="I_ACTION"> accusation </EVENT>.
<MKINSTANCE eiId="ei1" evId="e1"/> <MKINSTANCE eiId="ei2" evId="e8"/>
<MKINSTANCE eiId="ei3" evId="e12"/> <MKINSTANCE eiId="ei4" evId="e16"/>
<TLINK eiId="ei1" relToTime="t1" relType="IS_INCLUDED"/>
<TLINK eiId="ei4" relToEIId="ei1" relType="IDENTITY"/>
<ALINK eiId="ei2" relToEIId="ei4" relType="INITIATES"/>
```

TimeML is described in Section 3. Essentially, it promotes explicit representation and typing of time expressions and events, and an equally explicit mechanism for linking these with temporal links, using a vocabulary of temporal relations.

In addition to in-line mark-up, explicit links are marked. Event instance identifiers, ei1, ei2, and ei4 refer to, respectively, the accusation in the first sentence, the crisis, and the reference to *"that accusation"* in the second sentence. The relType attributes on the link descriptions define temporal relationships between event instances and time expressions; in this particular example, an IDENTITY link encodes the coreferentiality between the event instances (mentions) in the two sentences of the *accusation* event of the earlier example.

It is the combination of event descriptors, their anchoring to time points, and the semantics of relational links, which enable the derivation of *during* and *costarts* associations that the reasoner understands.

3 TimeML: a Mark-up Language for Time

Most content analysis applications to date do not explicitly incorporate temporal reasoning, and their needs can be met by analysis of simple time expressions (dates, intervals, *etc*). This is largely the motivation for TERN's TIMEX2 tag; at the same time it explains why TIMEX2 is inadequate for supporting the representational requirements outlined earlier.[3]

TimeML aims at capturing the richness of time information in documents. It marks up more than just temporal expressions, and focuses on ways of systematically anchoring event predicates to a time denoting expressions, and on ordering such event expressions relative to each other.

TimeML derives higher expressiveness from explicitly separating representation of temporal expressions from that of events. Time analysis is distributed across four component structures: TIMEX3, SIGNAL, EVENT, and LINK; all are rendered as tags, with attributes [Saurí *et al.*, 2004].[4]

[2] We are not alone: work on temporal reasoning from formal inference point of view reaches a similar conclusion: "... the [TimeML] annotation scheme itself, due to its closer tie to surface texts, can be used as the first pass in the syntax-semantics interface of a temporal resolution framework such as ours. The more complex representation, suitable for more sophisticated reasoning, can then be obtained by translating from the annotations." [Han & Lavie, 2004].

[3] For a notable extension to TIMEX2, see [Gaizauskas & Setzer, 2002]. An attempt to codify some relational information linking the TIMEX with an event, it is still limited, both in terms of scope (only links with certain syntactic shape can be captured) and representational power (it is hard to separate an event mention from possibly multiple event instances); see [Pustejovsky *et al.*, 2003].

[4] Additionally, a MKINSTANCE tag embodies the difference between event *tokens* and event *instances*: for example, the analysis of *"Max taught on Monday and Tuesday"* requires two different instances to be created for a *teaching* EVENT. Even if typically there is a one-to-one mapping between an EVENT and an instance, the language requires that a realisation of that event is created.

TIMEX3 extends[5] the TIMEX2 [Ferro, 2001] attributes: it captures temporal expressions (commonly categorised as DATE, TIME, DURATION), both literal and intensionally specified. SIGNAL tags are (typically) function words indicative of relationships between temporal objects: temporal prepositions (*for*, *during*, *etc*.) or temporal connectives (*before*, *while*). EVENT, in TimeML nomenclature, is a cover term for situations that happen or occur; these can be punctual, or last for a period of time. TimeML posits a refined typology of events [Pustejovsky *et al.*, 2003]. All classes of event expressions—tensed verbs, stative adjectives and other modifiers, event nominals—are marked up with suitable attributes on the EVENT tag. Finally, the LINK tag is used to encode a variety of relations that exist between the temporal elements in a document, as well as to establish an explicit ordering of events. Three subtypes to the LINK tag are used to represent strict temporal relationships between events or between an event and a time (TLINK), subordination between two events or an event and a signal (SLINK), and aspectual relationship between an aspectual event and its argument (ALINK).

TimeML's richer component set, in-line mark-up of temporal primitives, and non-consuming tags for temporal relations across arbitrarily long text spans, make it highly compatible with the current paradigm of annotation-based encapsulation of document analysis.

4 TimeML and Temporal Analysis

TimeML's annotation-based representation facilitates integration of time analysis with the analysis of other syntactic and/or discourse phenomena; it also naturally supports exploitation of larger contextual effects by the temporal parser proper (see 4.4) . This is a crucial observation, given that the prominently attractive characteristic of TimeML—its intrinsic richness of expression—makes it challenging for analysis.

There are two broad categories of problems for developing an automated TimeML analyser: of substance and of infrastructure. Substantive issues include normalising time expressions to a canonical representation (TIMEX3's value attribute), identifying a broad range of events (*e.g.* event nominals and predicative adjectives acting as event specifiers), linking time-denoting expressions (typically a TIMEX3 and an EVENT), and typing of those LINKs.

The infrastructure problems—small size and less than consistent mark-up of the TimeBank corpus—are due to the fact that this, first, version is largely a side product of a small number of annotators trying out TimeML's expressive capabilities. TimeBank is thus intended as a reference, and not for training. Our hybrid approach to temporal parsing, combining finite-state (FS) recognition with machine learning from sparse data (4.2), is largely motivated by this nature of TimeBank.

4.1 The TimeBank corpus

TimeBank has only 186 documents (68.5K words). If we held out 10% of the corpus as test data, we have barely over 60K words for training. Below we show counts of (EVENT-TIMEX3) TLINK[6] and EVENT types [Saurí *et al.*, 2004]. TLINK examples are particularly sparse; the data also shows highly uneven distribution of examples of different types.

In comparison, the Penn TreeBank corpus for part-of-speech tagging contains >1M words (> 16 times larger than TIMEBANK); the CoNLL'03 named entity chunking training set (at `http://cnts.uia.ac.be/conll2003/ner/`) has over 200K words with 23K examples (15 times more than TLINK examples) over just 4 name classes (compared to the 13 TLINK classes defined by TimeML). TERN's training set—almost 800 documents/300K words—is considered to be somewhat sparse, with over 8K TIMEX examples.

TLINK type	# occurrences	EVENT type	# occurrences
IS_INCLUDED	866	OCCURRENCE	4,452
DURING	146	STATE	1,181
ENDS	102	REPORTING	1,010
SIMULTANEOUS	69	I_ACTION	668
ENDED_BY	52	I_STATE	586
AFTER	41	ASPECTUAL	295
BEGINS	37	PERCEPTION	51
BEFORE	35		
INCLUDES	29		
BEGUN_BY	27		
IAFTER	5		
IDENTITY	5		
IBEFORE	1		
Total :	1,451	Total :	8,243

4.2 Analytical strategy

Minimally, the reasoner would require that the analytical framework supports time stamping and temporal ordering of events; thus we target the analysis tasks of finding TIMEX3's, assigning canonical values, marking and typing EVENTs, and associating (some of them) with TIMEX3 tags.

TIMEX3 expressions are naturally amenable to FS description. FS devices can also encode some larger context for time analysis (temporal connectives for marking putative events, clause boundaries for scoping possible event-time pairs, *etc*; see 4.4). To complement such analysis, a machine learning approach can cast the problem of marking EVENTs as chunking. Recently, [Ando, 2004] has developed a framework for exploiting large amounts of unannotated corpora in supervised learning for chunking. In such a framework, mid-to-high-level syntactic parsing—typically derived by FS cascades—can produce rich features for classifiers.

Thus, we combine FS grammars for temporal expressions, embedded in a general purpose shallow parser, with machine learning trained with TimeBank and unannotated corpora.

4.3 FS-based parser for temporal expressions

Viewing TIMEX3 analysis as an information extraction task, a cascade of finite-state grammars with broad coverage (compiled down to a single TIMEX3 automaton with 500 states and over 16000 transitions) targets abstract temporal entities such

[5] TIMEX2 and TIMEX3 differ substantially in their treatment of event anchoring and sets of times.

[6] In all of our experiments we exclude TIMEX3 markup in metadata; the TLINK counts only reflect links to temporal expressions in the body of documents.

as UNIT, POINT, PERIOD, RELATION, *etc*; these may be further decomposed and typed into *e.g.* MONTH, DAY, YEAR (for a UNIT); or INTERVAL or DURATION (for a PERIOD).

Fine-grained analysis of temporal expressions, instantiating attributes like `granularity`, `cardinality`, `ref_direction`, and so forth, is crucially required for normalising a TIMEX3: representing *"the last five years"* as illustrated below facilitates the derivation of a value for the TIMEX3 value attribute.

```
[timex : [relative      : true ]
         [ref_direction : past ]
         [cardinality   : 5 ]
         [granularity   : year ] ]
```

Such analysis amounts to a parse tree under the TIMEX3. (Not shown above is additional information, anchoring the expression into the larger discourse and informing other normalisation processes which emit the full complement of TIMEX3 attributes—type, temporalFunction, anchorTimeID, *etc*). TimeBank does not contain such fine-grained mark-up: the grammars thus perform an additional 'discovery' task, for which no training data currently exists, but which is essential for discourse-level post-processing, handling *e.g.* ambiguous and/or underspecified time expressions or the relationship between document-internal and document-external temporal properties (such as 'document creation time').

4.4 Shallow parsing for feature generation

In principle, substantial discourse analysis can be carried out from a shallow syntactic base, and derived by means of FS cascading [Kennedy & Boguraev, 1996]. Our grammars interleave shallow parsing with named entity extraction. They specify temporal expressions in terms of *linguistic* units, as opposed to simply lexical cues (as many temporal taggers to date do). This point cannot be over-emphasised. One of the complex problems for TimeML analysis is that of event identification. A temporal tagger, if narrowly focused on time expressions only (cf. [Schilder & Habel, 2003]), offers no clues to what events are there in the text. In contrast, a temporal parser aware of the syntax of a time phrase like *"during the long and ultimately unsuccessful war in Afghanistan"* is very close to knowing—from configurational properties of a prepositional phrase—that the nominal argument (*"war"*) of the temporal preposition (*"during"*) is an event nominal.

Ultimately, syntactic analysis beyond TimeML components is used to derive features for the classifiers tasked with finding EVENTs and LINKs (Section 5).

Feature generation typically relies on a mix of lexical properties and some configurational syntactic information (depending on the complexity of the task). Our scheme additionally needs some semantic typing, knowledge of boundaries of longer syntactic units (typically a variety of clauses), and some grammatical function. An example (simplified) of the FS cascade output is:

```
[Snt [svoClause
  [tAdjunct In [NP [timex3 the 1988 period timex3] NP] tAdjunct],
  [SUB [NP the company NP] SUB]
  [VG [GrmEventOccurrence earned grmEventOccurrence] VG]
  [OBJ [NP [Money $20.6 million Money] NP] OBJ] svoClause] ... Snt]
```

Most of the above is self-explanatory, but we emphasise a few key points. The analysis captures the mix of syntactic chunks, semantic categories, and TimeML components used for feature generation. It maintains local TIMEX3 analysis; the time expression is inside of a larger clause boundary, with internal grammatical function identification for some of the event predicates. The specifics of mapping configurational information into feature vectors is described in Section 5.

4.5 Machine learning for TimeML components

TimeML parsing is thus a bifurcated process of TimeML components recognition: TIMEX3's are marked by FS grammars; SIGNALs, EVENTs and LINKs are identified by classification models derived from analysis of both TimeBank and large unannotated corpora. *Features* for these models are derived from common strategies for exploiting local context, as well as from mining the results—both mark-up and configurational—from the FS grammar cascading, as illustrated in the previous section. (More details on feature generation follow in Section 5 below.)

Classifiers and feature vectors

The classification framework we adopt for this work is based on a principle of *empirical risk minimization*. In particular, we use a *linear classifier*, which makes classification decisions by thresholding inner products of feature vectors and weight vectors. It learns weight vectors by minimizing classification errors (*empirical risk*) on annotated training data.

For our experiments (Section 6), we use the *Robust Risk Minimization* (RRM) classifier [Zhang *et al.*, 2002], which has been shown useful for a number of text analysis tasks such as syntactic chunking, named entity chunking, and part-of-speech tagging.

In marked contrast to generative models, where assumptions about features are tightly coupled with algorithms, RRM—as is the case with discriminative analysis—enjoys clear separation of feature representation from the underlying algorithms for training and classification. This facilitates experimentation with different feature representations, since the separation between these and the algorithms which manipulate them does not require change in algorithms. We show how choice of features affects performance in Section 6.

Word profiling for exploitation of unannotated corpora

In general, classification learning requires substantial amount of labeled data for training—considerably more than what TimeBank offers (cf. 4.1). This characteristic of size is potentially a limiting factor in supervised learning approaches. We, however, seek to improve performance by exploiting unannotated corpora, with their natural advantages of size and availability. We use a *word profiling* technique, developed specially for exploiting a large unannotated corpus for tagging/chunking tasks [Ando, 2004]. Word profiling identifies, and extracts, word-characteristic information from unannotated corpora; it does this, in essence, by collecting and compressing feature frequencies from the corpus.

Word profiling turns co-occurrence counts of words and features (*e.g.* 'next word', 'head of subject', *etc*) into new feature vectors. For instance, observing that *"extinction"* and *"explosion"* are often used as syntactic subject to *"occur"*,

and that *"earthquakes" "happen"*, helps to predict that *"explosion"*, *"extinction"*, and *"earthquake"* all function like event nominals. Below (6.1) we demonstrate the effectiveness of word profiling, specifically for EVENT recognition.

5 Implementation

To use classifiers, one needs to design *feature vector representation* for the objects to be classified. This entails selection of some predictive attributes of the objects (in effect promoting these to the status of *features*) and definition of mappings between vector dimensions and those attributes (*feature mapping*). In this section we describe the essence of our feature design for EVENT and TLINK recognition.[7]

5.1 EVENT recognition

Similarly to named entity chunking, we cast the EVENT recognition task as a problem of sequential labeling of tokens by encoding chunk information into token tags. For a given *class*, this generates three tags: E:*class* (the last, end, token of a chunk denoting a mention of *class* type), I:*class* (a token inside of a chunk), and O (any token outside of any target chunk). The example sequence below indicates that the two tokens *"very bad"* are spanned by an event-state annotation.

$\cdots$ *another*/O *very*/I:event-state *bad*/E:event-state *week*/O $\cdots$

In this way, the EVENT chunking task becomes a $(2k+1)$-way classification of tokens where k is the number of EVENT types; this is followed by a Viterbi-style decoding. (We use the same scheme for SIGNAL recognition.)

The feature representation used for EVENT extraction experiments mimics the one developed for a comparative study of entity recognition with word profiling [Ando, 2004]. The features we extract are:

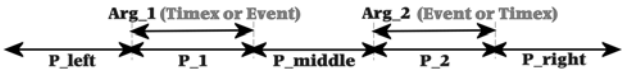
- token, capitalization, part-of-speech (POS) in 3-token window;
- bi-grams of adjacent words in 5-token window;
- words in the same syntactic chunk;
- head words in 3-chunk window;
- word uni- and bi-grams based on subject-verb-object and preposition-noun constructions;
- syntactic chunk types (noun or verb group chunks only);
- token tags in 2-token window to the left;
- tri-grams of POS, capitalization, and word ending;
- tri-grams of POS, capitalization, and left tag.

5.2 TLINK recognition

TLINK is a relation between events and time expressions which can link two EVENTs, two TIMEX3's, or an EVENT and a TIMEX3. Presently (see 4.2) we focus on TLINKs between events and time expressions.

As a relational link, TLINK does not naturally fit the tagging abstraction for a chunking problem, outlined above. Instead, we formulate a classification task as follows. After posting EVENT and TIMEX3 annotations (by the event classifier and the FS temporal parser, respectively), for each pairing between an EVENT and a TIMEX3, we ask whether it is a certain type of TLINK. This defines a $(\ell+1)$-way classification problem, where ℓ is the number of TLINK types (BEFORE, AFTER, *etc*; Section 4.1). The adjustment term '$+1$' is for the *negative* class (not-a-temporal-link).

The relation-extraction nature of the task of posting TLINKs requires a different feature representation, capable of encoding the syntactic function of the relation arguments (EVENTs and TIMEX3's), and some of the larger context of their mentions. To that end, we consider the following five *partitions* (defined in terms of tokens): spans of arguments (P_1 or P_2); two tokens to the left/right of the left/right argument (P_left/P_right); and the tokens between the arguments (P_middle). From each partition, we extract tokens and parts-of-speech as features.



We also consider *segments* (syntactic constructions derived by FS analysis: 'when-clause', 'subject', *etc*) in certain relationship to partitions: contained in P_1, P_2, or P_middle; covering P_1 (or P_2) but not overlapping with P_2 (or P_1); occurring to the left of P_1 (or the right of P_2); or covering both P_1 and P_2. We use uni- and bi-grams of types of these segments as features.

In this feature representation, segments play a crucial role by capturing the syntactic functions of EVENTs and TIMEX3's, as well as the syntactic relations between them.

Thus in the example analysis on p. 4, `svoClause` is the smallest segment containing both an EVENT and a TIMEX3, indicative of a direct syntactic relation between the two. In the next example, the TIMEX3 and EVENT chunks are contained in different clauses (a `thatClause` and a `svoClause`, respectively), which structurally prohibits a TLINK relation between the two.

```
[Snt
 Analysts have complained
 [thatClause that [timex3 third-quarter timex3] corporate earnings
    have n't been very good thatClause]
 [svoClause , but the effect [event hit event] ... svoClause] Snt]
```

Thus our feature representation is capable of capturing this information via the types of the segments that contain each of EVENT and TIMEX3 without overlapping.

6 Experiments

We present here performance results on EVENT and TLINK recognition only. This is largely because the primary focus of this paper is to report on how effective our analytical strategy is in leveraging the reference nature of the small TimeBank corpus for training classifiers for TimeML. Of these, SIGNAL was briefly mentioned earlier (see footnote 7), and TIMEX3 recognition, driven by FS grammars, belongs to a different paper. Since this is the first attempt to build a TimeML-compliant analyser (cf. Section 1), there are no comparable results in the literature.

The results (micro-averaged F-measure) reflect experiments with different settings, against the TimeBank corpus, and produced by 5-fold cross validation.

[7] We do not discuss SIGNAL recognition here, as the `signal` tag itself contributes nothing to EVENT or TLINK recognition, beyond what is captured by a lexical feature over the temporal connective.

6.1 EVENT recognition

It should be clear, by looking at the example analysis (p. 4), how local information and syntactic environment both contribute to the feature generation process. Figure 1 shows performance results with and without word profiling for exploiting an unannotated corpus. For word profiling, we extracted

features	with typing	w/o typing
basic	61.3	78.6
basic + word-profiling	64.0 (+2.7)	80.3 (+1.7)

Figure 1: Event extraction results, with/without typing. Parentheses show contribution of word profiling, over using basic features only.

feature co-occurrence counts from 40M words of 1991 *Wall Street Journal*. The proposed event chunks are counted as correct only when both the chunk boundaries and event types are correct. While word profiling improves performance, 64.0% F-measure is lower than typical performance of, for instance, named entity chunking. On the other hand, when we train the EVENT classifiers without typing, we obtain 80.3% F-measure. This is indicative of the inherent complexity of the EVENT typing task.

6.2 TLINK recognition

In this experimental setting, we only consider the pairings of EVENT and TIMEX3 which appear within a certain distance in the same sentences.[8]

For comparison, we implement the following simple baseline method. Considering the text sequence of EVENTs and TIMEX3's, only 'close' pairs of potential arguments are coupled with TLINKs; EVENT e and TIMEX3 t are close if and only if e is the closest EVENT to t and t is the closest TIMEX3 to e. For all other pairings, no temporal relation is posted. Depending on the 'with-'/'without-typing' setting, the baseline method either types the TLINK as the most populous class in TimeBank, IS_INCLUDED, or simply marks it as 'it exists'.

Results are shown in Figure 2. Clearly, the detection of

distance (# of TLINKs)	features	with typing	w/o typing
distance $\leq$ 64 tokens (1370 TLINKs)	baseline	21.8	34.9
	basic	52.1	74.1
	basic+FS	53.1 (+1.0)	74.8 (+0.7)
distance $\leq$ 16 tokens (1269 TLINKs)	baseline	38.7	61.3
	basic	52.8	75.8
	basic+FS	54.3 (+1.5)	76.5 (+0.7)
distance $\leq$ 4 tokens (789 TLINKs)	baseline	49.8	76.1
	basic	57.0	80.1
	basic+FS	58.8 (+1.8)	81.8 (+1.7)

Figure 2: TLINK extraction results, with/without typing. Parentheses show contribution of grammar-derived features, over using basic ones only. Baseline posts TLINKs over 'close' EVENT/TIMEX3 pairs.

temporal relations between events and time expressions requires more than simply coupling the closest pairs within a sentence (as the baseline does). It is also clear that the baseline method performs poorly, especially for pairings over relatively long distances. For instance, it produces 34.9% when we consider the pairings within 64 tokens without typing. In the same setting, our method produces 74.8% in F-measure, significantly outperforming the baseline.

We compare performance in two types of feature representation: 'basic' and 'basic+FS grammar', which reflect the without- and with-segment-type information obtained by the grammar analysis, respectively. As the positive delta's show, configurational syntactic information can be exploited beneficially by our process. Focusing on within-4-tokens pairings, we achieve 81.8% F-measure without typing of TLINKs, and 58.8% with typing. (The task without typing is a binary classification to detect whether the pairing has a TLINK relation or not, regardless of the type.) As the figure shows, the task becomes harder when we consider longer distance pairings. Within a 64 token distance, we obtain figures of 74.8% and 53.1%, without and with typing respectively.

While we are moderately successful in detecting the *existence* of temporal relations, the noticeable differences in performance between the task settings with and without typing indicate that we are not as successful in distinguishing one type from another. In particular, the relatively low performance of TLINK typing highlights the difficulty in distinguishing between DURING and IS_INCLUDED.

The guidelines (and common sense analysis) suggest that IS_INCLUDED type should be assigned if the time point or duration of EVENT *is included* in the duration of the associated TIMEX3. DURING, on the other hand, should be assigned as a type if some relation represented by the EVENT *holds during* the duration of the TIMEX3. We note that for this particular typing problem, the subtle distinctions are hard even for human annotators: the TimeBank corpus displays a number of occasions where inconsistent tagging is evident.

7 Conclusion

TimeML is a significant development in time analysis, as it captures detailed information, anchored in eventuality and linguistic structure, and shown to be crucial inferential and reasoning tasks. In addition to defining annotation guidelines, the TimeML effort notably created the first reference corpus illustrative of expressiveness of the language.

Unfortunately, the small size of the TimeBank corpus prevents its straightforward use as a training resource, a problem further exacerbated by the inherent complexity of TimeML-compliant analysis. And yet, for reasoning engines to function, TimeML analysers need to be built.

[Mani *et al.*, 2004] discuss some pioneering work in linking events with times, and ordering events, indicative of productive strategies for posting (some) TLINK information. However, the nature of these efforts is such that differences in premises, representation, and focus make a direct performance comparison impossible. Furthermore, the work pre-dates TimeML, and cannot be conveniently mapped to TimeBank data; this, in effect, precludes a quantitative comparison with our work.

In a first systematic attempt at TimeML-compliant analysis,

[8] To evaluate the TLINK classifier alone, we use the EVENT and TIMEX3 annotations in TimeBank.

and leveraging the TimeBank corpus, we have developed a strategy which synergistically blends finite-state analysis over linguistic annotations with a state-of-the-art machine learning technique. Particularly effective are: aggressive analysis, by complex grammars, of both TimeML components and syntactic structure; coupled with a learning algorithm capable of training over unannotated data, in addition to exploiting arbitrarily small amounts of labeled data. While work remains (notably refining the TLINK recogniser, targeting other types of LINKs, and enhancing EVENT recognition with external lexical resources), this is a significant step in instantiating a deeper time analysis, capable of satisfying the needs of reasoning engines.

References

[Ando, 2004] R.K. Ando. Exploiting unannotated corpora for tagging and chunking. In *Proceedings of ACL-04*.

[Ferro, 2001] L. Ferro. TIDES: Instruction manual for the annotation of temporal expressions. MTR 01W0000046V01, The MITRE Corporation, 2001.

[Fikes *et al.*, 2003] R. Fikes, J. Jenkins, and G. Frank. JTP: A system architecture and component library for hybrid reasoning. TR KSL-03-01, Stanford University, 2003.

[Filatova and Hovy, 2001] E. Filatova & E. Hovy. Assigning time-stamps to event-clauses. In *Proceedings of the 10th Conference of the EACL*, Toulouse, France, 2001.

[Gaizauskas & Setzer, 2002] R. Gaizauskas and A. Setzer, editors. *Annotation Standards for Temporal Information in NL*, Las Palmas, Spain, 2002.

[Han & Lavie, 2004] B. Han and A. Lavie. Framework for resolution of time in natural language. *TALIP Special Issue, Spatial and Temporal Information Processing*, 2004.

[Hobbs *et al.*, 2002] J.R. Hobbs, G. Ferguson, J. Allen, P. Hayes, and A. Pease. A DAML ontology of time, 2002.

[Kennedy & Boguraev, 1996] C. Kennedy & B. Boguraev. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of COLING-96*, Copenhagen, DK, 1996.

[Mani & Wilson, 2000] I. Mani and G. Wilson. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting of the ACL*, Hong Kong, 2000.

[Mani *et al.*, 2003] I. Mani, B. Schiffman, and J. Zhang. Inferring temporal ordering of events in news. In *Proceedings of ACL-41 (HLT-NAACL)*, Edmonton, Canada, 2003.

[Mani *et al.*, 2004] I. Mani, J. Pustejovsky, and B .Sundheim. Introduction: special issue on temporal information processing. *ACM Transactions Asian Language Information Processing*, 3(1):1–10, 2004.

[Prager *et al.*, 2003] J. Prager, J. Chu-Carroll, E. Brown, and C. Czuba. Question answering using predictive annotation. In *Advances in Question Answering*, 2003.

[Pustejovsky *et al.*, 2003] J. Pustejovsky, J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, G. Katz, and D. Radev. TimeML: Robust specification of event and temporal expressions in text. In *AAAI Spring Symposium on New Directions in Question-Answering*, pages 28–34, Stanford, CA, 2003.

[Saurí *et al.*, 2004] R. Saurí, J. Littman, R. Gaizauskas, A. Setzer, and J. Pustejovsky. TimeML annotation guidelines, Version 1.1, TERQAS Workshop, 2004.

[Schilder & Habel, 2003] F. Schilder and C. Habel. Temporal information extraction for temporal QA. In *AAAI Spring Symposium on New Directions in Question-Answering*, pages 35–44, Stanford, CA, 2003.

[Zhang *et al.*, 2002] T. Zhang, F. Damerau, and D.E. Johnson. Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637, 2002.

Viewing Referring Expression Generation as Search

Bernd Bohnet* **and Robert Dale†**
*Institute for Intelligent Systems, University of Stuttgart, Germany
†Centre for Language Technology, Macquarie University, Sydney, Australia
Bohnet@iis.uni-stuttgart.de, Robert.Dale@mq.edu.au

Abstract

Almost all natural language generation (NLG) systems are faced with the problem of the generation of referring expressions (GRE): given a symbol corresponding to an intended referent, how do we work out the semantic content of a referring expression that uniquely identifies the entity in question? This is now one of the most widely explored problems in NLG: over the last 15 years, a number of algorithms have been proposed for addressing different aspects of this problem, but the different approaches taken make it very difficult to compare and contrast the algorithms provided in any meaningful way. In this paper, we show how viewing the problem of referring expression generation as a search problem allows us to recast existing algorithms in a way that makes their similarities and differences clear.

1 Introduction

A major component task in natural language generation (NLG) is the generation of referring expressions: given an entity that we want to refer to, how do we determine the content of a referring expression that uniquely identifies that intended referent? This is now a widely explored problem in the NLG literature; since at least [Dale, 1989], the standard conception of this task has been as follows:

1. We assume we have a knowledge base that characterises the entities in the domain in terms of a set of attributes and the values that the entities have for these attributes; so, for example, our knowledge base might represent the fact that entity e_1 has the value *cup* for the attribute *type*, and the value *red* for the attribute *colour*.
2. In a typical context where we want to refer to some e_i, which we call the *intended referent*, there will be other entities from which the intended referent must be distinguished; these are generally referred to as *distractors*. So, for example, we may want to distinguish a particular cup from all the other items present in the context of a dining table.
3. The goal of referring expression generation is therefore to find some collection of attributes and their values which distinguish the intended referent from all the potential distractors in the context.

Over the last 15 years, a wide variety of algorithms have been proposed to deal with specific aspects of this problem. For example, while earlier algorithms focussed on the use of attributes that correspond to simple one-place predicates (as might be realised, for example, by means of adjectives such as *red* or *large*), later work attempts to address the use of relational predicates (as might be realised by prepositions such as *in* and *next to*), and other work looks at the incorporation of boolean operators such as *not* and *and*. Consequently, we now have a considerable body of research in this area; however, it is difficult to establish just how these different algorithms relate to each other.

This paper represents a first step towards consolidating the results in this area, with the aim of developing a framework within which different algorithms can be compared and assessed. The structure of the paper is as follows. In Section 2, we provide a brief overview of work on the generation of referring expressions to date. In Section 3, we borrow a standard approach used in Artificial Intelligence (AI) to represent problems in an elegant and uniform way (see, for example, [Simon and Newell, 1963]; [Russell and Norvig, 2003]), sketching how GRE algorithms can be expressed in terms of *problem-solving by search*. In Section 4, we explore how the most well-known algorithms can be expressed in this framework. In Section 5, we discuss how this approach enables a more fruitful comparison of existing algorithms, and we point to ways of taking this work further.

2 A Brief Review of Work To Date

Although the task of referring expression generation is discussed informally in earlier work on NLG (in particular, see [Winograd, 1972; McDonald, 1980; Appelt, 1981], the first formally explicit algorithm was introduced in [Dale, 1989]. This algorithm, which we will refer to as the Full Brevity (FB) algorithm, is still frequently used as a basis for other GRE algorithms. The FB algorithm searches for the best solution amongst all possible referring expressions for an entity; the algorithm derives the smallest set of attributes for the referent

in question, producing a referring expression that is both *adequate* (in the sense that it provides enough information to do what it needs to do) and *efficient* (in the sense that it does not provide any more information than it needs to).

This initial algorithm limited its application to one-place predicates. [Dale and Haddock, 1991] introduced a constraint-based procedure that could generate referring expressions involving relations (henceforth IR), using a greedy heuristic to guide the search.

As a response to the computational complexity of greedy algorithms, [Reiter and Dale, 1992; Dale and Reiter, 1995] introduced the psycholinguistically motivated Incremental Algorithm (IA). The most used and adapted algorithm, this is based on the observation that people often produce referring expressions which are informationally redundant; the algorithm uses a preference ordering over the attributes to be used in a referring expression, incorporating those attributes which rule out at least one potential distractor.

In recent years there have been a number of important extensions to the IA. The Context-Sensitive extension (CS; [Krahmer and Theune, 2002]) is able to generate referring expressions for the most salient entity in a context; the Boolean Expressions algorithm (BE; [van Deemter, 2002]) is able to derive expressions containing boolean operators, as in *the cup that does not have a handle*; and the Sets algorithm (SET; [van Deemter, 2002]) extends the basic approach to references to sets, as in *the red cups*. Some approaches reuse parts of other algorithms: the Branch and Bound (BaB; [Krahmer *et al.*, 2003]) algorithm uses the Full Brevity algorithm, but is able to generate referring expressions with both attributes and relational descriptions using a graph-based technique.

We have identified here what we believe to be the most cited strands of research in this area, but of course there are many other algorithms described in the literature: see, for example, [Horacek, 1997; Bateman, 1999; Stone, 2000]. Space limitations prevent a complete summary of this other work here, but our intention is to extend the analysis presented in this paper to as many of these other algorithms as possible.

All these algorithms focus on the generation of definite references. In principle, they would be embedded in a higher-level algorithm that includes cases for when the entity has not been previously mentioned (thus leading to an initial indefinite reference) or when the referent is in focus (thus leading to a pronominal reference). However, in practice, most work tends to focus exclusively on definite reference; see [Dale, 1989; Krahmer and Theune, 2002; Dale, 2003] for further discussion of the more general case.

3 GRE from the Perspective of Problem Solving

With so many algorithms to choose from, it would be useful to have a uniform framework in which to discuss and compare algorithms; unfortunately, this is rather difficult given the variety of different approaches that have been taken to the problem.

Within the wider context of AI, [Russell and Norvig, 2003] present an elegant definition of a general algorithm for problem solving by search. The search graph consists of *nodes* with the components *state* and *path-cost*; a problem is represented by an *initial-state*, an *expand-method* which identifies new states in the search space, a *queuing-method* which determines the order in which the states should be considered, and a *path-cost-function* which determines the cost of reaching a given state. In this framework, the search strategy is determined by the combination of *queuing-method* and *path-cost-function* used.

In the following, we use this framework to provide a characterisation of existing GRE algorithms in terms of problem solving by search. Given an intended referent, a set of properties true of the intended referent,[1] and the set of distractor entities from which we wish to distinguish the intended referent, we can conceptualise the search space as consisting of states that correspond to possible descriptions of the intended referent. Each state has three components: a description that is true of the intended referent, the set of distractor entities that the description also applies to besides the intended referent, and the set of properties of the intended referent that have not yet been considered in describing the referent.

1. The *initial-state* is of the form $\langle\{\}, C, P\rangle$, where C is the set of distractors in the initial context, and P is the set of all properties true of the intended referent.

2. The goal state is of the form $\langle\{\lambda x P_1, \lambda x P_2, \ldots\}, \{\}, P'\rangle$, where the first term contains a set of properties of the intended referent that, by virtue of the second term (the set of distractors) being an empty set, distinguish the intended referent; P' contains any properties of the intended referent not yet used in the description.

3. All other states in the search space are then intermediate states through which an algorithm will move as it adds new properties to the description.

4. The search strategy is carried out by the *expand-method* and the *queuing-method*, which together characterise the specific GRE algorithm (for example, FB, GH or IA) that is used.

5. The *path-cost-function* allows us to route the search as required; this can be used to take account of salience weights, or to embody some kind of heuristic search.

For any given algorithm, not all of the methods and functions need to be implemented; in particular, we will see that some algorithms do not require a *path-cost-function*.

[1] We will use the notion of a property to correspond to an attribute and its value; this will considerably simplify the notation we provide.

4 GRE Algorithms in Terms of Problem Solving

We adopt here an object oriented formalism,[2] since this allows the representation of dependencies between the algorithms by means of inheritance and overwriting.

To enable more fruitful comparison of the different GRE algorithms, we want to distinguish those aspects of the algorithms which are true of all algorithms, and those which are unique to each particular algorithm. In Section 4.1, we first describe the elements that are shared by all the algorithms; we then go on to describe the distinct aspects of each algorithm in turn.

4.1 Common Elements

Following from the previous section, the definitions of the *node* and *state* classes are as shown in Definition 1. This figure also shows the definitions for *initial-state* and *goal*, which remain constant across the algorithms.

Definition 1: The Node and State Classes

```
class Node {
  s // State
  path-cost // Cost of the path to this node
  getState()
    {return s} // returns the state of the node
}
class State {
  L // Set of chosen properties and/or relations
  C // Set of distractors
  P // Set of available properties and/or relations
}
initialState() {return new State(∅,C,P)}
// the goal is the empty set of distractors
goal(s) {
   if s.C = ∅ then return true
   else return false
}
```

Given these components, the main method *makeRefExp* is then as represented in Definition 2. This takes two arguments, which serve as the parameters that distinguish one algorithm from another: an *expand* method to create the successors of a given state, and a *queue* method, which defines how to insert nodes into the node queue. Depending on the order in which the nodes are inserted, different search strategies can be realized: for example, when the nodes are inserted at the front of the queue, the search strategy is depth-first; when the nodes are inserted at the end of the queue, the search strategy is breadth-first; when the nodes of the queue are sorted by the estimated distance to the goal, then the search type is best-first; and so on.

In addition, we may require a number of general-purpose methods which can be used by a number of different algorithms. One such 'utility function' is the method *rulesOut*, which takes a property or relation p and a set of distractors, and returns the set of distractors which are ruled out by p.

[2]We follow the code conventions typically used in OO-languages, where the names of classes start with upper case characters, and the names of methods and variables start with lower case characters.

Definition 2: The Basic Algorithm Structure

```
makeRefExp() {
   // create a initial queue with a single node
   nodeQueue ← [new Node(initialState())]
   while nodeQueue ≠ ∅ do
      node ← removeFront(nodeQueue)
      if goal(node.getState()) then
         return node // success
      end
      nodeQueue ← queue(nodeQueue,expand(node))
   end
   return nil // failure
}
```

With this machinery in place, we can now redefine the existing algorithms in terms of their core differences, which correspond essentially to different ways of expanding the search space.

4.2 The Full Brevity Algorithm

The distinctive property of the Full Brevity (FB) algorithm is that it computes all combinations of the available properties P with increasing length, so that it may find the shortest combination that succeeds in identifying the intended referent.

This behaviour is captured by the *expand* method shown in Definition 3. The method creates a set of successors by creating a node for each property p_i which has not so far been checked, provided that p_i rules out at least one distractor.

The FB algorithm uses a breadth-first search implementation of the queue, as shown in Definition 4. Consequently, any solution for which *goal* returns *true* will have a minimal number of properties, since the breadth-first search considers smaller combinations of properties first.

The FB algorithm uses the *expand* method, and *createNode* method which are shown in Definition 3 and it is invoked by a call of *makeRefExp* method which is shown in Definition 2.

4.3 The Incremental Algorithm

The distinctive property of the Incremental Algorithm is that it reduces the computational complexity of constructing a referring expression by considering properties to use in sequence from a predefined ordering of the available properties. The implementation of the *expand* method shown in Definition 5 provides this behaviour.

If the set of properties of the current state $s.P$ is not empty, then the first property p according to the given order O is chosen from the set of properties of the current state $s.P$, and a node is created with a new state by the method *createNode*. Note that the *createNode* method is the same as that used in the FB algorithm and shown in Definition 3.

Definition 3: The Full Brevity Algorithm

expand($node$) {
  $N \leftarrow \emptyset$
  $s \leftarrow node.getState()$
  foreach $p \in s.P$ **do**
    $N \leftarrow N \cup \{$ *createNode* $(node, p)\}$
  end
  return N
}
createNode($node$, p) {
  $s \leftarrow node.getState()$
  $out \leftarrow$ *rulesOut*(p, $s.C$)
  if $out \neq \emptyset$ **then**
    return new Node($s.C - out$, $s.L \cup \{p\}$, $s.P - \{p\}$)
  else return new Node($s.C$, $s.L$, $s.P - \{p\}$)
}

Definition 4: Breadth-first Queueing

queue($actNodes, newNodes$) {
  // append the nodes at the end
  return $actNodes \cup newNodes$
}

Unlike the *expand* method used in the FB algorithm, however, the set of nodes returned here contains only one node. The main method applies the *goal* predicate to this node; if this returns true, then the node containing the state with the list of properties for the referring expression is returned.

Definition 5: The Incremental Algorithm

O // Predefined constant order of properties
expand($node$) {
  $N \leftarrow \emptyset$
  $s \leftarrow node.getState()$
  if $s.P \neq \emptyset$ **then**
    $p \leftarrow$ choose the first p in O, where $p \in s.P$
    $N \leftarrow N \cup$ { *createNode*($node, p$) }
  end
  return N
}

4.4 Extension of the IA to Sets

The algorithms considered so far have been concerned with constructing descriptions for individual referents; [van Deemter, 2002] introduced an algorithm which extends the IA to sets. The extension is shown in terms of our framework in Definition 6.

Note that, precisely because this algorithm is an extension of the IA algorithm, we reuse the *expand* method from that algorithm. Consequently, the extension requires only the rewriting of the *createNode* method, whereby an attribute p_i is only chosen when it does not rule out entities

Definition 6: The Set Algorithm

R // Set of referents
createNode(node, p) {
  $out \leftarrow$ rulesOut($p, s.C$)
  if ($\neg\exists x \in R \& x \in out) \& (\exists x \in C \& x \in out)$ **then**
    return new Node($s.C - out, s.L \cup \{p\}, s.P - \{p\}$)
  else return new Node($s.C$, $s.L$, $s.P - \{p\}$)
}

from the set of intended referents R and when it rules out at least one entity from the set of distractors C. If a property does not fulfil that condition, then a node with the current state is returned and the process is continued, as in the IA, with the next property.

4.5 GRE Involving Relations

The algorithm for GRE Involving Relations (IR) introduced by [Dale and Haddock, 1991] is constraint-based. The search strategy used to fulfil the constraints is a combination of a greedy search, which chooses the relation that leads to the smallest set of distractors, and depth-first search to describe the entities, that is, the intended referent as well as entities which are referenced in the relations.

The strategy can be explained best by means of an AND/OR-tree, as shown in Figure 1. Here, the top node represents a state in which relational properties are to be considered as additions to the set of chosen properties. Each search step consists of two stages: in the first stage, we choose the relation p_i which rules out the largest number of distractors; in the second stage, each entity which is referenced by the chosen relation has to be described by repeating the process recursively. This is done in a depth-first manner, but if the related entity is not uniquely distinguished then the next p_j that the intended referent participates in is chosen, and so on. This process continues until all entities are uniquely described (*success*) or no further relations can be chosen (*failure*).

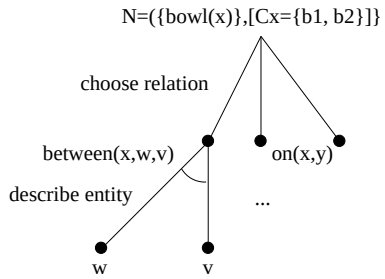


Figure 1: Expansion tree for the IR algorithm

The algorithm is represented in the problem solving paradigm as in Definition 7. Here, the *expand* method chooses a relation which rules out the largest number of distractors; it then calls the method *createNode*, which recursively calls *makeRefExp* for each new referent contained in the relation.

Definition 7: Involving Relations

```
r // Referent
expand(node) {
   s ← node.getState()
   p_c ← nil
   // Chosen relation is p_c, where p rules out
   // the largest number of distractors
   foreach p ∈ s.P do
      if p_c = nil or
         |rulesOut(p, s.C)| > |rulesOut(p_c, s.C)|
      then p_c ← p
      end
   end
   node_c ← createNode(node, p_c)
   if (node_c = nil) then return ∅
   else return {node_c}
}
createNode(node, p) {
   s ← node.getState()
   C ← s.C − rulesOut(p, s.C)
   L ← s.L ∪ {p}
   // Extend Description
   foreach r' ∈ {r_p | r_p ∈ referents(p) & r_p ≠ r} do
      node_r' ← makeRefExp(r')
      if node_r' = nil then return nil // failure
      L ← L ∪ node.getState().L
   end
   return new Node(C, L, s.P − {p})
}
```

Definition 8: Context-Sensitive Algorithm

```
r // Referent
createNode(node, p) {
   s ← node.getState()
   out ← rulesOut(p, s.C)
   C ← s.C − out
   L ← s.L
   if (out ≠ ∅ or contrastive(r, p)) then
      L ← L ∪ {p}
      if v expresses a relation between r and r' then
         node_r' ← makeRefExp(r')
         L ← L ∪ node_r'.getState().L
      end
   end
   if mostSalient(r, L, C) then
      L ← L ∪ { defArt }
      // The most salient rules out all distractors:
      C ← ∅
   end
   return new Node(C, L, s.P − {p})
}
```

4.6 Context-Sensitive GRE

[Krahmer and Theune, 2002] also introduced a number of extensions to the IA: the use of salience weights in order to add a definite article to the description for the most salient entity; contrastive properties in order to add properties which impose a contrast between two entities; and a relational extension, similar in spirit but not in form to that in the IR algorithm described above.

Again, as for the SETS algorithm, the commonality with the IA algorithm surfaces as the reuse of the latter algorithm's *expand* method; only the *createNode* method needs to be rewritten, as in Definition 8. To model this variant in our framework, we introduce the following additional methods (cf. [Krahmer and Theune, 2002]):

- *contrastive* takes a referent r and a property p_i; it checks whether the property under consideration is contrastive.
- *mostSalient* takes a referent r, a set of properties, and a set of distractors; it checks whether every entity in the set of distractors has a lower salience weight than r.

5 Conclusions and Future Work

In the foregoing, we have shown how a number of the most frequently discussed algorithms for the generation of referring expressions can be represented within a common framework. The framework is, we believe, an intuitively appealing one: the process of constructing a referring expression is viewed as a search problem, where we effectively build a search space consisting of possible descriptions.

There are three significant advantages to this approach.

First, it allows us to determine what the algorithms have in common. This is particularly interesting in that it allows us to begin to assemble a collection of core functionalities that are usable in a variety of different approaches to GRE. This is apparent not only in terms of the general framework (where, for example, the notions of states and their initialisation, the definition of what it is to be a goal state, and the overall algorithmic pattern) are shared, but in terms of 'helper' routines (such as *rulesOut* and *mostSalient*) which can be modularised out of the essence of different algorithms and reused elsewhere.

Second, it makes it possible to see more clearly what the essential differences between the algorithms really are. In their original forms, these differences are obscured, due to the absence of a common vocabulary for expressing the algorithms; by representing the algorithms within a common framework, it becomes easier to see where the algorithms differ, and where the differences are simply due to differences in notation or presentation. By using the framework of problem solving as search, we have effectively decomposed the algorithms into a number of key elements: a search srategy, represented by the *queuing-method*, and an *expand-method*, which encompasses two aspects of each algorithm: the basic strategy adopted and the particular kinds of referring expressions covered. Furthermore, the *expand-method* decomposes into a general strategy for expansion (as found in, for example, the Full Brevity algorithm and the Incremental Algorithm), and a *createNode* method, which varies depending upon the kind of referring expression targetted.

Third, it allows us to see more clearly the logical space within which the algorithms reside, and to see ways of

combining aspects of different algorithms. At its simplest, this is clearest with respect to the kind of search strategy used in the algorithms. Present formulations conflate the choice of search strategy with the other aspects of the algorithm (such as how subsequent nodes in the search space are computed); our approach separates out these different facets of the algorithms, and makes it much easier to see that the choice of search strategy is an independent decision. Consequently, for example, we can easily experiment with a variant of the IR algorithm that uses breadth-first search rather than depth-first search.

So far, we have used the framwework to express the most widely-known algorithms in the literature. Preliminary examination of the algorithms in [Krahmer *et al.*, 2003], [van Deemeter and Krahmer, forthcoming], and [Horacek, 2004] suggests that these will also be expressible within the framework described here. As we capture more algorithms in the framework, our intention is to tease out an inventory of basic constituent elements which can then be reassembled and integrated in different ways, so that we can derive a better understanding of the nature of the problem of referring expression generation.

Acknowledgements

This work was carried out while the first author was a visiting researcher at Macquarie University's Centre for Language Technology; the visit was supported by Macquarie University's Research Development Scheme. We are grateful to the members of the Centre for Language Technology who provided insightful comments on earlier presentations of this work, and to the anonymous IJCAI reviewers for their helpful comments.

References

[Appelt, 1981] D. E. Appelt. *Planning Natural Language Utterances to Satisfy Multiple Goals*. PhD thesis, Stanford University, 1981.

[Bateman, 1999] J. Bateman. Using Aggregation for Selecting Content when generating Referring Expressions. In *Proceedings of the 37th Annual Meeting of the ACL*. University of Marylad, 1999.

[Dale and Haddock, 1991] R. Dale and N. Haddock. Generating referring expressions involving relations. In *Proceedings of the 5th EACL*, pages 161–166, Berlin, Germany, 1991.

[Dale and Reiter, 1995] R. Dale and E. Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263, 1995.

[Dale, 1989] R. Dale. Cooking up referring expressions. In *Proceedings of the Twenty-Seventh Annual Meeting of the ACL*, pages 68–75, Vancouver, British Columbia, 1989.

[Dale, 2003] R Dale. One-anaphora and the case for discourse-driven referring expression generation. In *Proceedings of the Australasian Language Technology Workshop*. University of Melbourne, 2003.

[Horacek, 1997] H. Horacek. An Algorithm for Generating Referential Descriptions with Flexible Interfaces. In *Proceedings of the 35th Annual Meeting of the ACL*. University of Madrid, 1997.

[Horacek, 2004] H. Horacek. On Referring toSets of Objects Naturally. In H. Bunt and R. Muskens, editors, *Third International Natural Language Generation Conference*, pages 70 – 79. Springer-Verlag Heidelberg, 2004.

[Krahmer and Theune, 2002] E. Krahmer and M. Theune. Efficient context-sensitive generation of referring expressions. In *Information Sharing: Reference and Presupposition in NLG and Interpretation*, pages 223–264. CSLI, 2002.

[Krahmer *et al.*, 2003] E. Krahmer, S. van Erk, and A. Verleg. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72, 2003.

[McDonald, 1980] D. D. McDonald. *Natural Language Generation as a Process of Decision-making Under Constraints*. PhD thesis, Massachusetts Institute of Technology, 1980.

[Reiter and Dale, 1992] E. Reiter and R. Dale. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th ACL*, pages 232–238, 1992.

[Russell and Norvig, 2003] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

[Simon and Newell, 1963] H. Simon and A. Newell. *GPS, a program that simulates thought*, pages 279–293. 1963.

[Stone, 2000] M. Stone. On identifying sets. In *Proceedings of the First International Natural Language Generation Conference*. Mitzpe Ramon, 2000.

[van Deemeter and Krahmer, forthcoming] K. van Deemeter and E. Krahmer. Graphs and booleans: On the generation of referring expressions. In H. Bunt and R. Muskens, editors, *Computing Meaning Vol. III*. Dordrecht: Kluwer, forthcoming.

[van Deemter, 2002] K. van Deemter. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52, 2002.

[Winograd, 1972] T. Winograd. *Understanding Natural Language*. Academic Press, 1972.

Word Sense Disambiguation with Distribution Estimation

Yee Seng Chan and Hwee Tou Ng
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{chanys, nght}@comp.nus.edu.sg

Abstract

A word sense disambiguation (WSD) system trained on one domain and applied to a different domain will show a decrease in performance. One major reason is the different sense distributions between different domains. This paper presents novel application of two distribution estimation algorithms to provide estimates of the sense distribution of the new domain data set. Even though our training examples are automatically gathered from parallel corpora, the sense distributions estimated are good enough to achieve a relative improvement of 56% when incorporated into our WSD system.

1 Introduction

Many words have multiple meanings. The process of identifying the sense of a word in a particular context is known as word sense disambiguation (WSD). WSD is an important task of natural language processing (NLP), and is important for applications such as machine translation and information retrieval. Past research has shown that a good approach to WSD is the use of supervised machine learning. With this approach, a large text corpus in which each ambiguous word has been annotated with the correct sense has to be collected to serve as training data.

This corpus-based approach, however, raises the problem of domain dependence faced by these supervised WSD systems. A system trained on data from one domain (for example, sports), will show a decrease in performance when applied to a different domain (for example, economics). This issue of domain difference affecting WSD performance had been brought up by various researchers. Escudero *et al.* [2000] showed that training a WSD system in one domain and applying it to another could lead to a drop of 12% to 19% in WSD accuracy. It was shown that one of the reasons is due to different distributions of senses across domains, as a 9% improvement in accuracy was observed when they sense-balanced the instances from the different domains. Recently, Agirre and Martinez [2004] conducted experiments with training data for WSD which were automatically gathered from the Internet. They reported a potential improvement of 14% in accuracy if they have access to the sense distribution of their test data.

The work of these researchers showed that the different sense distributions across domains have an important effect on WSD accuracy. Therefore, estimation of the target domain's sense distribution will be crucial, in order to build WSD systems that are portable across different domains. A partial solution to this would be to determine the predominant, or most frequently occurring sense of each word in a corpus. Research has shown that the baseline heuristic of selecting the most frequently occurring sense of a word gives high WSD accuracy. This is due to the often skewed distribution of word senses. McCarthy *et al.* [2004b] addressed this when they presented a method to automatically rank word senses in a corpus, in order to determine the predominant sense of each word. They obtained good results when the method is applied on the SENSEVAL-2 English all-words task [Palmer *et al.*, 2001]. The same method is used in a related work [McCarthy *et al.*, 2004a] to identify infrequently occurring word senses.

In the machine learning literature, algorithms to estimate class a priori probabilities have been developed. In this paper, we present novel application of two distribution estimation algorithms to provide estimates of the sense distribution, or a priori probabilities of senses, and show that they are effective in improving the WSD accuracy of systems trained and applied on different domains. Note that determining the predominant sense is a special case of trying to estimate the complete sense distribution. To the best of our knowledge, these algorithms have not been used in NLP or for WSD.

We recognize that a WSD system should be scalable. This is a problem faced by current supervised learning systems, as they usually relied on manually annotated data for training. To tackle this problem, our prior work[Ng *et al.*, 2003] exploited parallel texts for WSD. Encouraging results were obtained in our evaluation on the nouns of SENSEVAL-2 English lexical sample task [Kilgarriff, 2001], which used the WordNet 1.7 sense inventory [Miller, 1990]. Note that the usage of parallel texts as training data represents a natural domain difference with the test data of SENSEVAL-2 English lexical sample task, of which 91% is drawn from the British National Corpus (BNC). In this paper, we chose to draw training data for our experiments from parallel texts, and similarly base our evaluation on the nouns of SENSEVAL-2 English lexical sample task.

This paper is structured as follows. In the next section,

Parallel corpora	Size of English texts (in million words (MB))	Size of Chinese texts (in million characters (MB))
Hong Kong Hansards	39.9 (223.2)	35.4 (146.8)
Hong Kong News	16.8 (96.4)	15.3 (67.6)
Hong Kong Laws	9.9 (53.7)	9.2 (37.5)
Sinorama	3.8 (20.5)	3.3 (13.5)
Xinhua News	2.1 (11.9)	2.1 (8.9)
English translation of Chinese Treebank	0.1 (0.7)	0.1 (0.4)
Total	72.6 (406.4)	65.4 (274.7)

Table 1: Size of English-Chinese parallel corpora

we discuss the process of gathering training data from parallel texts. In section 3, we describe the two distribution estimation algorithms used. We also give a brief description of the predominant sense method presented by [McCarthy *et al.*, 2004b]. In section 4, we evaluate the effectiveness of the two algorithms and the predominant sense method in improving WSD accuracy. Evaluation results on the nouns in SENSEVAL-2 English lexical sample task are presented. Discussions are made in section 5, before we conclude in section 6.

2 Training Data from Parallel Texts

In this section, we briefly describe the parallel texts used in our experiments, and the process of gathering training data from them.

2.1 Parallel Text Alignment

Table 1 lists the 6 English-Chinese parallel corpora (available from Linguistic Data Consortium) used in our experiments. The sentences of the 6 corpora were already pre-aligned, either manually or automatically, when they were prepared. After ensuring the corpora were sentence aligned, we tokenized the English texts, and perform word segmentation on the Chinese texts. Next, word alignment was done on the parallel corpora using the GIZA++ software [Och and Ney, 2000].

2.2 Target Translations Selection

We took the same approach as described in our previous work [Ng *et al.*, 2003] to select some possible Chinese translations for each sense of an English word. For instance, WordNet 1.7 lists 5 senses for the noun *nature*. Senses will be lumped together if they are translated in the same way in Chinese. For example, sense 2 and 3 of *nature* are lumped together as they are both translated as "大自然" in Chinese. Then, from the word alignment output of GIZA++, we select those occurrences of the noun *nature* which have been aligned to one of the Chinese translations chosen. The English side of these occurrences will then serve as training data for the noun *nature*, as they are considered to have been disambiguated and "sense-tagged" by the appropriate Chinese translations.

The time needed to perform manual target translations of word senses is relatively short, compared to the effort needed to manually sense annotate training examples. Also, this step could potentially be automated with the use of an English-Chinese dictionary.

3 Distribution Estimation

To estimate the sense distribution, or a priori probabilities of the different senses in a new data set, we made use of a confusion matrix algorithm [Vucetic and Obradovic, 2001] and an EM based algorithm [Saerens *et al.*, 2002], which we describe in this section. A brief description of the predominant sense method [McCarthy *et al.*, 2004b] is also given, before we discuss how to make use of the estimated a priori probabilities to improve classification accuracy.

3.1 Confusion Matrix

Let us assume that from a set of labeled data S_L, a classifier is used to compute the conditional probabilities $\widehat{p}_L(\omega_j|\omega_i)$, which is an estimate of the probability of classifying an instance as class ω_j , when in fact it belongs to class ω_i . Then, one can apply this classifier with known conditional probabilities $\widehat{p}_L(\omega_j|\omega_i)$ to a set of unlabeled data S_U to obtain its predictions. From these predictions, one can estimate the probability of predicting class ω_j on S_U, which we will denote as $\widehat{q}(\omega_j)$. The a priori probabilities $\widehat{p}(\omega_i)$ on S_U are then estimated by solving the following equation:

$$\widehat{q}(\omega_j) = \sum_{i=1}^{n} \widehat{p}_L(\omega_j|\omega_i)\widehat{p}(\omega_i), \qquad j = 1, \ldots, n \quad (1)$$

where *n* represents the number of classes. Equation (1) can be represented in matrix form as $\mathbf{q} = \mathbf{P} \cdot \mathbf{p}$, from which the a priori probabilities on S_U, represented by $\mathbf{p}$, can be estimated by solving:

$$\mathbf{p} = \mathbf{P}^{-1} \cdot \mathbf{q} \quad (2)$$

To obtain estimates of $\mathbf{P}$ and $\mathbf{q}$, bootstrap sampling [Efron and Tibshirani, 1993] is employed. We will first describe the basic idea of bootstrap before giving the bootstrap algorithm.

In bootstrap sampling, given an original sample *X* with *n* examples, bootstrap sample X^* is obtained by randomly sampling *n* examples from *X* with replacement. To estimate the conditional probabilities $\widehat{p}_L(\omega_j|\omega_i)$, we need to split S_L into S_{train} and S_{test}. Further, let us define $n^*_{test}(\omega_j, \omega_i)$ as the number of examples in a bootstrap sample S^*_{test} predicted to be of class ω_j when true class is ω_i. Also, we define $n^*_U(\omega_j)$ as the number of examples in a bootstrap sample S^*_U predicted to be of class ω_j. We now give the bootstrap algorithm below:

Given S_{test}, S_U, and a classifier, repeat the following for *B* iterations (In our experiments, we set *B* to 200):

- Generate a bootstrap sample from n_{test} examples of S_{test} and calculate:
  $\widehat{p}^*_L(\omega_j|\omega_i) = \frac{n^*_{test}(\omega_j,\omega_i)}{\sum_j n^*_{test}(\omega_j,\omega_i)}$ for $i, j = 1, \ldots, n$

- Generate a bootstrap sample from n_U examples of S_U and calculate:
  $\widehat{q}^*(\omega_j) = \frac{n^*_U(\omega_j)}{n_U}$ for $j = 1, \ldots, n$

- Use (2) to calculate $\widehat{p}^*(\omega_i)^{(b)}$ for $i = 1, \ldots, n$

After *B* iterations, estimate the a priori probabilities $\widehat{p}(\omega_i) = \frac{1}{B}\sum_{b=1}^{B}\widehat{p}^*(\omega_i)^{(b)}$

3.2 EM based Algorithm

Once again, let us assume we train a classifier on a set of labeled data S_L with *n* classes. Further, suppose we have a set of *N* independent instances, or independent realizations of the stochastic variable $\mathbf{x}$, $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ from a new data set. The likelihood of these *N* instances can be defined as:

$$\begin{aligned} L(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) &= \prod_{k=1}^{N} p(\mathbf{x}_k) \\ &= \prod_{k=1}^{N}\left[\sum_{i=1}^{n} p(\mathbf{x}_k, \omega_i)\right] \\ &= \prod_{k=1}^{N}\left[\sum_{i=1}^{n} p(\mathbf{x}_k|\omega_i)p(\omega_i)\right] \quad (3) \end{aligned}$$

If we assume the generation of the instances within the classes, and thus the within-class densities $p(\mathbf{x}_k|\omega_i)$ (i.e., the probabilities of observing $\mathbf{x}_k$ given the class ω_i) do not change from the training set S_L to the new data set, we can define $p(\mathbf{x}_k|\omega_i) = p_L(\mathbf{x}_k|\omega_i)$. To determine the a priori probability estimates $\widehat{p}(\omega_i)$ of the new data set that will maximize the likelihood of (3) with respect to $p(\omega_i)$, we can apply the iterative procedure of the EM algorithm. In effect, through maximizing the likelihood of (3), we obtain the a priori probability estimates as a by-product.

Before we show the steps of the EM algorithm, it will be helpful to define some notations. When we apply the classifier trained on S_L on an instance $\mathbf{x}_k$ drawn from the new data set S_U, we get $\widehat{p}_L(\omega_i|\mathbf{x}_k)$, which we define as the probability of instance $\mathbf{x}_k$ being classified as class ω_i by the classifier trained on S_L. Further, let us define $\widehat{p}_L(\omega_i)$ as the a priori probabilities of class ω_i in S_L , which can be estimated by the class frequency of ω_i in S_L. Also, we will define $\widehat{p}^{(s)}(\omega_i)$ and $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ as estimates of the new a priori and a posteriori probabilities at step *s* of the iterative EM procedure. Assuming we initialize $\widehat{p}^{(0)}(\omega_i)$ by the a priori probabilities of the classes in the labeled data:

$$\widehat{p}^{(0)}(\omega_i) = \widehat{p}_L(\omega_i)$$

then for each new instance $\mathbf{x}_k$ in S_U, and each class ω_i, the EM algorithm provides the following iterative steps:

$$\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k) = \frac{\widehat{p}_L(\omega_i|\mathbf{x}_k)\frac{\widehat{p}^{(s)}(\omega_i)}{\widehat{p}_L(\omega_i)}}{\sum_{j=1}^{n}\widehat{p}_L(\omega_j|\mathbf{x}_k)\frac{\widehat{p}^{(s)}(\omega_j)}{\widehat{p}_L(\omega_j)}} \quad (4)$$

$$\widehat{p}^{(s+1)}(\omega_i) = \frac{1}{N}\sum_{k=1}^{N}\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k) \quad (5)$$

where (4) represents the expectation E-step, (5) represents the maximization M-step, and *N* represents the number of instances in S_U. Note that in (4), the probabilities $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ and $\widehat{p}_L(\omega_i)$ will stay the same through the iteration steps *s* for each particular instance $\mathbf{x}_k$ and class ω_i . Observe that for (4), the new a posteriori probabilities $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ at step *s* are simply the a posteriori probabilities in the conditions of the labeled data, $\widehat{p}_L(\omega_i|\mathbf{x}_k)$, weighted by the ratio of the new priors $\widehat{p}^{(s)}(\omega_i)$ to the old priors $\widehat{p}_L(\omega_i)$. The denominator in (4) is simply a normalizing factor.

At each iteration step *s*, both the a posteriori $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ and a priori probabilities $\widehat{p}^{(s)}(\omega_i)$ are re-estimated sequentially for each new instance $\mathbf{x}_k$ and each class ω_i, until the convergence of the estimated probabilities $\widehat{p}^{(s)}(\omega_i)$. This iterative procedure will increase the likelihood of (3) at each step.

3.3 Predominant Sense

A method of automatically ranking WordNet senses to determine the predominant, or most frequent sense, of a noun in the BNC corpus is presented in [McCarthy *et al.*, 2004b]. A thesaurus is first acquired from the parsed 90 million words of written English from the BNC corpus to provide the *k* nearest neighbors to each target noun, along with the distributional similarity score (*dss*) between the target noun and its neighbor. The WordNet similarity package [Pedersen *et al.*, 2004] is then used to give a WordNet similarity measure (*wnss*), which is used to weight the contribution that each neighbor makes to the various senses of the target noun. Through a combination of *dss* and *wnss*, a prevalence score for each sense of the target noun is calculated, from which the predominant sense of the noun in BNC is determined.

Though the focus of the method is to determine the predominant sense, we could obtain an estimated sense distribution by normalizing the prevalence score of each sense. As noted in section 1, 91% of the test examples of SENSEVAL-2 English lexical sample task were drawn from BNC. Thus, it is reasonable to expect the sense distribution estimated by this predominant sense method to be applicable to the test examples of SENSEVAL-2 English lexical sample task. We implemented this method in order to evaluate its effectiveness in improving WSD accuracy. Our implementation achieved accuracies close to those reported by [McCarthy *et al.*, 2004b]. McCarthy *et al.* [2004b] reported a *jcn measure* predominant sense accuracy of 54% on the Semcor corpus, while we measured 53.3% using our implementation. On the SENSEVAL-2 English all-words task, a 64% precision and 63% recall was reported. Our implementation gave 66.2% precision and 63.4% recall. The differences could be due to some minor processing steps which were not described in detail in [McCarthy *et al.*, 2004b]. Finally, note that this predominant sense method is only effective on a very large text corpus, as the thesaurus can only be effectively generated from a very large corpus.

Maximum number of parallel examples per noun	L	EM sample	TRUE − L		EM − L		ConfM − L		Predominant − L	
			adjust	sample	adjust	sample	adjust	sample	adjust	sample
150	68.29	69.35	1.76	3.70	0.81	1.06	0.92	1.04	0.84	0.59
200	69.29	71.33	1.58	3.75	0.88	2.04	0.90	1.20	0.59	0.29
300	70.29	72.07	1.55	3.69	0.71	1.78	0.76	0.74	0.57	0.62
500	72.21	73.57	1.37	**2.40**	0.77	**1.36**	0.79	**0.95**	**0.35**	0.09

Table 2: 5 trials micro-averaged scores for 22 SE2 nouns

Maximum number of parallel examples per noun	EM − L		ConfM − L		Predominant − L	
	adjust	sample	adjust	sample	adjust	sample
150	46.02	28.65	52.27	28.11	47.73	15.95
200	55.70	54.40	56.96	32.00	37.34	7.73
300	45.81	48.24	49.03	20.05	36.77	16.80
500	56.20	56.67	57.66	39.58	25.55	3.75

Table 3: Relative improvement percentage from using the a priori probabilities estimated by the 3 methods

3.4 Improving Classification Based on A Priori Estimates

If a classifier was trained to estimate posterior class probabilities $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ when presented with a new instance $\mathbf{x}_k$ from S_U, it can be directly adjusted according to estimated class probabilities $\widehat{p}(\omega_i)$ on S_U.

Denoting predictions of the classifier as $\widehat{p}_L(\omega_i|\mathbf{x}_k)$, a priori probability of class ω_i from S_L as $\widehat{p}_L(\omega_i)$, estimated a priori probability of class ω_i from S_U as $\widehat{p}(\omega_i)$, adjusted predictions $\widehat{p}_{adjust}(\omega_i|\mathbf{x}_k)$ can be calculated as:

$$\widehat{p}_{adjust}(\omega_i|\mathbf{x}_k) = \frac{\widehat{p}_L(\omega_i|\mathbf{x}_k)\frac{\widehat{p}(\omega_i)}{\widehat{p}_L(\omega_i)}}{\sum_j \widehat{p}_L(\omega_j|\mathbf{x}_k)\frac{\widehat{p}(\omega_j)}{\widehat{p}_L(\omega_j)}} \quad (6)$$

Note the similarity of (6) with (4).

4 Evaluation

In our experiments, we implemented the supervised WSD approach of [Lee and Ng, 2002], which was reported to achieve state-of-the-art accuracy. The knowledge sources used include parts-of-speech, surrounding words, and local collocations. We use the naive Bayes algorithm as our classifier, which was reported to achieve good WSD accuracy and it is fast to train and test with the naive Bayes classifier. Also, the naive Bayes classifier outputs posteriori probabilities in its classifications, which can be used directly to adjust the predictions based on (6).

4.1 Results

We used the approach outlined in section 2 to obtain training examples from parallel texts. Two senses s_1 and s_2 of a noun are lumped together if they are translated into the same Chinese word. Although there are 29 nouns in the SENSEVAL-2 English lexical sample task, the senses of 7 nouns are lumped into one sense (i.e., all senses of each of these 7 nouns are translated into one Chinese word). Thus, we limit our evaluation to the remaining 22 nouns.

Saerens *et al.* [2002] reported that increasing the training set size results in an improvement in the a priori estimation. To investigate this issue, we gradually increased the maximum number of examples per noun selected from the parallel texts. For each condition, we sampled training examples from the parallel text using the *natural distribution* of the parallel text, up to the maximum allowed number of examples per noun. We noted the WSD accuracy achieved by the classifier without any adjustment, in the column labeled *L* in table 2. To provide a basis for comparison, we adjusted the classifier posteriori probability output by the true sense distribution of the test data, using equation (6). The increase in WSD accuracy thus obtained is listed in the column *adjust* under *TRUE−L*. For example, if we knew exactly the true sense distribution of the test data, we would have achieved an accuracy of 68.29% + 1.76% = 70.05%, when trained on a maximum of 150 examples per noun.

Besides adjusting the classifier posteriori probability output, one could also re-sample training examples from the parallel texts according to the true sense distribution. The increase in WSD accuracy obtained using this approach is listed in the column *sample* under *TRUE−L*. Note that scores listed under *TRUE−L* indicate the upper-bound accuracy achievable had we known the true sense distribution of the test data.

Increase in WSD accuracy by adjusting the classifier output using the sense distribution estimated by the EM algorithm is listed in column *adjust* under *EM−L*. Corresponding increase by re-sampling training examples from the parallel texts according to the sense distribution estimated by the EM algorithm is listed in column *sample* under *EM−L*. Similar increases by using sense distribution estimated by the confusion matrix algorithm, and the predominant sense method, are listed under the columns *adjust* and *sample*, under *ConfM−L* and *Predominant−L*. The corresponding relative improvement of the various accuracy improvements (comparing against using the true sense distribution) achieved by the two algorithms and the predominant sense method is given in Table 3. As an example, for 150 parallel examples

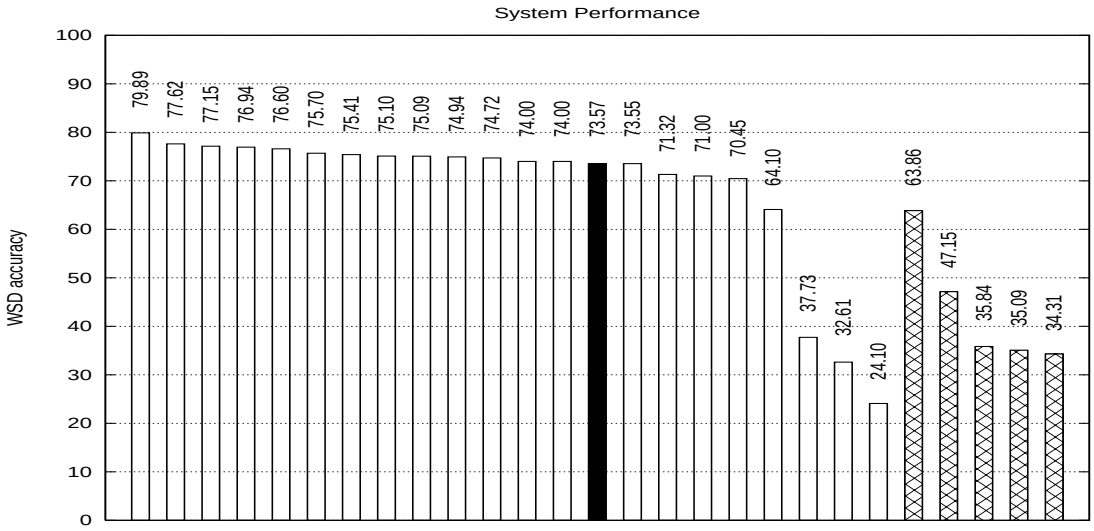


Figure 1: Comparison of **Parallel-EM's** performance against SENSEVAL-2 participating systems. The single shaded bar represents **Parallel-EM**, the empty white bars represent the supervised systems, and the pattern filled bars represent the unsupervised systems.

per noun, *EM adjust* gives an improvement of 0.81%, against an improvement of 1.76% using *TRUE adjust*. Thus, relative improvement is $0.81/1.76 = 46.02\%$.

The results show that the confusion matrix and EM algorithms are effective in improving WSD accuracy. Also, the two algorithms are most effective when a large training data set (which is 500 examples per noun in our experiments) is used. The results also indicate the potential of parallel text scaling up to achieve better WSD accuracy as more training examples are used, as shown by the increasing accuracy figures under the *L* column of Table 2. This is a phenomenon not explored in [Ng *et al.*, 2003], which used the same number of parallel text examples as the official SENSEVAL-2 training data. At the same time, however, this caused a progressive reduction in the absolute amount of possible WSD accuracy improvement, as indicated by the reduction under the columns *adjust* and *sample* of *TRUE*$-L$ in Table 2 when more parallel texts examples were used. Our experiments also indicate that application of the two algorithms achieved higher increases in WSD accuracy when compared against the predominant sense method, which has the primary aim of determining the predominant sense of a word in a corpus.

We note that re-sampling the training examples gives better WSD improvements than merely adjusting the posteriori probabilities output by the classifier. The number of training examples for a sense will affect the accuracy of the classifier trained for that sense. Thus, if one has a reliable estimate of the sense distribution, it might be more beneficial to re-sample the training examples. The results show that sampling a maximum of 500 examples per noun from the parallel texts according to the distribution estimated by the EM algorithm gives our best achievable WSD accuracy of **73.57%** (72.21% + 1.36%). In the column *EM sample* of Table 2, we show the WSD accuracy resulting from sampling parallel text examples according to the EM estimated sense distribution, with different maximum number of examples per noun (i.e., this column is obtained by adding the figures from the columns *L* and *EM*$-L$ *sample*).

4.2 Comparison with SE2 Systems

We compare our WSD accuracy achieved with the SENSEVAL-2 English lexical sample task participating systems. We performed sense lumping on the publicly available predictions of the SENSEVAL-2 participating systems. The WSD accuracy of each system is then calculated over the set of 22 nouns. In the SENSEVAL-2 English lexical sample task, there were 21 supervised and 5[1] unsupervised systems. Figure 1 shows the relative ranking of our system (which we named **Parallel-EM**) against the various SE2 systems.

We note that our system achieved better performance than all the unsupervised systems, and would have ranked at the 64% percentile among the list of 22 (including ours) supervised systems.

5 Discussion

The score of 73.57% achieved by **Parallel-EM** is competitive, considering that the SE2 supervised systems are trained with the manually annotated official data, while our system was trained with examples automatically extracted from parallel texts. Moreover, the following factors will also have to be considered.

Official Training Data has Similar Sense Distribution as Test Data

This is an advantage that the participating SENSEVAL-2 supervised systems enjoy. Note that the results in Table 2 show that if we sample according to the *true distribution* with a maximum of 500 examples per noun, we could achieve a score of 74.61%.

[1]There were 4 systems from IIT (2 were resubmissions), but system answers were only available for 2 of them.

Some Official Training and Test Examples are from the Same Document

Our prior work [Ng *et al.*, 2003] highlighted the fact that many of the official SENSEVAL-2 training and test examples are extracted from the same document, and that this has inflated the accuracy figures achieved by SENSEVAL-2 participating systems.

To investigate this effect, we trained our WSD classifier using the *official training examples*. When evaluated on the official test examples, a WSD accuracy of 79.71% was achieved. Next, we followed the steps taken in [Ng *et al.*, 2003]: We took the official training and test examples of each noun *w* and combined them together. The combined data is then randomly split into a new training and a new test set such that no training and test examples come from the same document. While performing the split, we ensured the sense distribution of the new training and test data sets followed those of the official data sets. A WSD classifier was then trained on this new training set, and evaluated on this new test set. 5 random trials were conducted, and our WSD classifier achieved an accuracy of 77.83% (a drop of 1.88% from 79.71%).

Lack of Parallel Training Examples for some Senses

A disadvantage faced by **Parallel-EM** is that, though the senses are lumped, we are still lacking training examples for 9.64% of the test data. We believe that with the steady increase in volume and variability of parallel corpora, this problem will improve in the near future.

Domain Difference

Lastly, we note that 91% of the SENSEVAL-2 English lexical sample task test examples were drawn from BNC. This represents a domain difference with our parallel corpora, which is mainly a mix of legislative, law, and news articles. Examples from different domains will contain different information, thus presenting different classification cues to the learning algorithm.

6 Conclusion

Differences in sense distribution between the training and test sets will result in a loss of WSD accuracy. Addressing this issue is important, especially if one is to build WSD systems that are portable across different domains, of which the difference in sense distribution is an inherent issue. In this paper, we used two distribution estimation algorithms to provide estimates of the sense distribution in a new data set. When applied on the nouns of the SENSEVAL-2 English lexical sample task, we have shown that they are effective in improving WSD accuracy.

Acknowledgements

The first author is supported by a Singapore Millennium Foundation Scholarship (ref no. SMF-2004-1076). This research is also partially supported by a research grant R252-000-125-112 from National University of Singapore Academic Research Fund.

References

[Agirre and Martinez, 2004] E. Agirre and D. Martinez. Unsupervised WSD based on automatically retrieved examples: The importance of bias. In *Proceedings of EMNLP04*, pages 25–32, Barcelona, Spain, July 2004.

[Efron and Tibshirani, 1993] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

[Escudero *et al.*, 2000] G. Escudero, L. Marquez, and G. Rigau. An empirical study of the domain dependence of supervised word sense disambiguation systems. In *Proceedings of EMNLP/VLC'00*, Hong Kong, October 2000.

[Kilgarriff, 2001] A. Kilgarriff. English lexical sample task description. In *Proceedings of SENSEVAL-2*, pages 17–20, Toulouse, France, July 2001.

[Lee and Ng, 2002] Y. K. Lee and H. T. Ng. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of EMNLP02*, pages 41–48, Philadelphia, Pennsylvania, USA, July 2002.

[McCarthy *et al.*, 2004a] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. Automatic identification of infrequent word senses. In *Proceedings of COLING04*, pages 1220–1226, Geneva, Switzerland, August 2004.

[McCarthy *et al.*, 2004b] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. Finding predominant word senses in untagged text. In *Proceedings of ACL04*, pages 280–287, Barcelona, Spain, July 2004.

[Miller, 1990] G. A. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.

[Ng *et al.*, 2003] H. T. Ng, B. Wang, and Y. S. Chan. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of ACL03*, pages 455–462, Sapporo, Japan, July 2003.

[Och and Ney, 2000] F. J. Och and H. Ney. Improved statistical alignment models. In *Proceeedings of ACL00*, pages 440–447, Hong Kong, October 2000.

[Palmer *et al.*, 2001] M. Palmer, C. Fellbaum, and S. Cotton. English tasks: All-words and verb lexical sample. In *Proceedings of SENSEVAL-2*, pages 21–24, Toulouse, France, July 2001.

[Pedersen *et al.*, 2004] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet::similarity - measuring the relatedness of concepts. In *Proceedings of AAAI04, Intelligent Systems Demonstration*, San Jose, CA, July 2004.

[Saerens *et al.*, 2002] M. Saerens, P. Latinne, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41, January 2002.

[Vucetic and Obradovic, 2001] S. Vucetic and Z. Obradovic. Classification on data with biased class distribution. In *Proceedings of ECML01*, pages 527–538, Freiburg, Germany, September 2001.

A Probabilistic Learning Method for XML Annotation of Documents

Boris Chidlovskii[1]
[1]Xerox Research Centre Europe
6, chemin de Maupertuis
38240 Meylan, France

Jérôme Fuselier[1,2]
[2]Université de Savoie - Laboratoire SysCom
Domaine Universitaire
73376 Le Bourget-du-Lac, France

Abstract

We consider the problem of semantic annotation of semi-structured documents according to a target XML schema. The task is to annotate a document in a tree-like manner where the annotation tree is an instance of a tree class defined by DTD or W3C XML Schema descriptions. In the probabilistic setting, we cope with the tree annotation problem as a *generalized probabilistic context-free parsing* of an observation sequence where each observation comes with a probability distribution over terminals supplied by a probabilistic classifier associated with the content of documents. We determine the most probable tree annotation by maximizing the joint probability of selecting a terminal sequence for the observation sequence and the most probable parse for the selected terminal sequence.

1 Introduction

The future of the World Wide Web is often associated with the Semantic Web initiative which has as a target a wide-spread document reuse, re-purposing and exchange, achieved by means of making document markup and annotation more machine-readable. The success of the Semantic Web initiative depends to a large extent on our capacity to move from *rendering-oriented* markup of documents, like PDF or HTML, to *semantic-oriented* document markup, like XML and RDF.

In this paper, we address the problem of *semantic annotation of HTML documents according to a target XML schema*. A tree-like annotation of a document requires that the annotation tree be an instance of the target schema, described in a DTD, W3C XML Schema or another schema language. Annotation trees naturally generalize flat annotations conventionally used in information extraction and wrapper induction for Web sites.

The migration of documents from rendering-oriented formats, like PDF and HTML, toward XML has recently become an important issue in various research communities [2; 3; 11; 12]. The majority of approaches either make certain assumptions about the source and target XML documents, like a conversion through a set of local transformations [3], or entail the transformation to particular tasks, such as the semantic annotation of dynamically generated Web pages in news portals [11] or the extraction of logical structure from page images [12].

In this paper, we consider the general case of tree annotation of semi-structured documents. We make *no assumptions about the structure of the source and target documents* or their possible similarity. We represent the document content as a sequence of observations $\mathbf{x} = \{x_1, \ldots, x_n\}$, where each observation x_i is a content fragment. In the case of HTML documents, such a fragment may be one or multiple leaves, often surrounded with rich contextual information in the form of HTML tags, attributes, etc. The tree annotation of sequence $\mathbf{x}$ is given by a pair $(\mathbf{y}, d)$, where $\mathbf{y}$ and d refer to leaves and internal nodes of the tree, respectively. The sequence $\mathbf{y} = \{y_1, \ldots, y_n\}$ can be seen, on one side, as labels for observations in $\mathbf{x}$, and on the other side, as a terminal sequence for tree d that defines the internal tree structure over $\mathbf{y}$ according to the target XML schema.

In supervised learning, the document annotation system includes selecting the tree annotation model and training the model parameters from a training set S given by triples $(\mathbf{x}, \mathbf{y}, d)$. We adopt a probabilistic setting, by which we estimate the probability of an annotation tree $(\mathbf{y}, d)$ for a given observation sequence $\mathbf{x}$ and address the problem of finding the pair $(\mathbf{y}, d)$ of maximal likelihood.

We develop a modular architecture for the tree annotation of documents that includes two major components. The first component is a probabilistic context-free grammar (PCFG) which is a probabilistic extension to the corresponding (deterministic) XML schema definition. The PCFG rules may be obtained by rewriting the schema's element declarations (in the case of a DTD) or element and type definitions (in the case of a W3C XML Schema) and the rule probabilities are chosen by observing rule occurrences in the training set, similar to learning rule probabilities from tree-bank corpora for NLP tasks. PCFGs offer the efficient inside-outside algorithm for finding the most probable parse for a given sequence $\mathbf{y}$ of terminals. The complexity of the algorithm is $O(n^3 \cdot |N|)$, where n is the length of sequence $\mathbf{y}$

and $|N|$ is the number of non-terminals on the PCFG.

The second component is a probabilistic classifier for predicting the terminals y for the observations x_i in $\mathbf{x}$. In the case of HTML documents, we use the maximum entropy framework [1], which proved its efficiency when combining content, layout and structural features extracted from HTML documents for making probabilistic predictions $p(y)$ for x_i.

With the terminal predictions supplied by the content classifier, the tree annotation problem represents *the generalized case of probabilistic parsing*, where each position i in sequence $\mathbf{y}$ is defined not with a specific terminal, but with a terminal probability $p(y)$. Consequently, we consider the sequential and joint evaluations of the maximum likelihood tree annotation for observation sequences. In the joint case, we develop a generalized version of the inside-outside algorithm that determines the *most probable annotation tree* $(\mathbf{y}, d)$ according to the PCFG and the distributions $p(y)$ for all positions i in $\mathbf{x}$. We show that the complexity of the generalized inside-outside algorithm is $O(n^3 \cdot |N| + n \cdot |T| \cdot |N|)$, where n is the length of $\mathbf{x}$ and $\mathbf{y}$, and where $|N|$ and $|T|$ are the number of non-terminals and terminals in the PCFG.

We also show that the proposed extension of the inside-outside algorithm imposes the *conditional independence requirement*, similar to the Naive Bayes assumption, on estimating terminal probabilities. We test our method on two collections and report an important advantage of the joint evaluation over the sequential one.

2 XML annotation and schema

XML annotations of documents are trees where inner nodes determine the tree structure, and the leaf nodes and tag attributes refer to the document content. XML annotations can be abstracted as the class T of *unranked labeled rooted trees* defined over an alphabet Σ of tag names [9]. The set of trees over Σ can be constrained by a *schema* D that is defined using DTD, W3C XML Schema or other schema languages.

DTDs and an important part of W3C XML Schema descriptions can be modeled as extended context-free grammars [10], where regular expressions over alphabet Σ are constructed by using the two basic operations of concatenation $\cdot$ and disjunction $|$ and with occurrence operators $*$ (Kleene closure), ? ($a? = a|\epsilon$) and + ($a+ = a \cdot a*$). An *extended context free grammar* (ECFG) is defined by the 4-tuple $G = (T, N, \mathcal{S}, R)$, where T and N are disjoint sets of terminals and nonterminals in Σ, $\Sigma = T \cup N$; $\mathcal{S}$ is an initial nonterminal and R is a finite set of production rules of the form $A \rightarrow \alpha$ for $A \in N$, where α is a regular expression over $\Sigma = T \cup N$. The language $L(G)$ defined by an ECFG G is the set of terminal strings derivable from the starting symbol S of G. Formally, $L(G) = \{w \in \Sigma^* | S \Rightarrow w\}$, where $\Rightarrow$ denotes the transitive closure of the derivability relation. We represent as a *parse tree* d any sequential form that reflects the derivational steps. The *set of parse trees* for G forms the set $\mathcal{T}(G)$ of unranked labeled rooted trees constrained with schema G.

2.1 Tree annotation problem

When annotating HTML documents accordingly to a target XML schema, the main difficulty arises from the fact that the source documents are essentially layout-oriented, and the use of tags and attributes is not necessarily consistent with elements of the target schema. The irregular use of tags in documents, combined with complex relationships between elements in the target schema, makes the manual writing of HTML-to-XML transformation rules difficult and cumbersome.

In supervised learning, the content of source documents is presented as a sequence of observations $\mathbf{x} = \{x_1, \ldots, x_n\}$, where any observation x_i refers to a content fragment, surrounded by rich contextual information in the form of HTML tags, attributes, etc. The tree annotation model is defined as a mapping $X \rightarrow (Y, D)$ that maps the observation sequence $\mathbf{x}$ into a pair $(\mathbf{y}, d)$ where $\mathbf{y}=\{y_1, \ldots, y_n\}$ is a terminal sequence and d is a parse tree of $\mathbf{y}$ according to the target schema or equivalent PCFG G, $\mathcal{S} \Rightarrow \mathbf{y}$. The training set S for training the model parameters is given by a set of triples $(\mathbf{x}, \mathbf{y}, d)$.

To determine the most probable tree annotation $(\mathbf{y}, d)$ for a sequence $\mathbf{x}$, we maximize the joint probability $p(\mathbf{y}, d|\mathbf{x}, G)$, given the sequence $\mathbf{x}$ and PCFG G. Using the Bayes theorem and the independence of x and G, we have

$$p(\mathbf{y}, d|\mathbf{x}, G) = p(d|\mathbf{y}, G) \cdot p(\mathbf{y}|\mathbf{x}), \tag{1}$$

where $p(\mathbf{y}|\mathbf{x})$ is the probability of terminal sequence $\mathbf{y}$ for the observed sequence $\mathbf{x}$, and $p(d|\mathbf{y}, G)$ is the probability of the parse d for $\mathbf{y}$ according the PCFG G. The most probable tree annotation for $\mathbf{x}$ is a pair $(\mathbf{y}, d)$ that maximizes the probability in (1),

$$(\mathbf{y}, d)_{max} = \underset{(y,d)}{argmax}\, p(d|\mathbf{y}, G) \cdot p(\mathbf{y}|\mathbf{x}). \tag{2}$$

In the following, we build a probabilistic model for tree annotation of source documents consisting of two components to get the two probability estimates in (2). The first component is a probabilistic extension of the target XML schema; for a given terminal sequence $\mathbf{y}$, it finds the most probable parse $p(d|\mathbf{y}, G)$ for sequences according to the PCFG G, where rule probabilities are trained from the available training set. The second component is a probabilistic content classifier C, it estimates the conditional probabilities $p(y|x_i)$ for annotating observations x_i with terminals $y \in T$. Finally, for a given sequence of observations $\mathbf{x}$, we develop two methods for finding a tree annotation $(\mathbf{y}, d)$ that maximizes the joint probability $p(\mathbf{y}, d|\mathbf{x}, G)$ in (1).

3 Probabilistic context-free grammars

PCFGs are probabilistic extensions of CFGs, where each rule $A \rightarrow \alpha$ in R is associated with a real number p in the half-open interval (0; 1]. The values of p obey the

restriction that for a given non-terminal $A \in N$, all rules for A must have p values that sum to 1,

$$\forall A \in N : \sum_{r=A\to\alpha, r\in R} p(r) = 1. \quad (3)$$

PCFGs have a normal form, called the Chomsky Normal Form (CNF), according to which any rule in R is either $A \to B\ C$ or $A \in b$, where A, B and C are non-terminals and b is a terminal. The rewriting of XML annotations requires the *binarization* of source ranked trees, often followed by an extension of the nonterminal set and the underlying set of rules. This is a consequence of rewriting nodes with multiple children as a sequence of binary nodes. The binarization rewrites any rule $A \to B\ C\ D$ as two rules $A \to BP$ and $P \to C\ D$, where P is a new non-terminal.

A PCFG defines a joint probability distribution over Y and D, random variables over all possible sequences of terminals and all possible parses, respectively. Y and D are clearly not independent, because a complete parse specifies exactly one or few terminal sequences. We define the function $p(\mathbf{y}, d)$ of a given terminal sequence $\mathbf{y} \in Y$ and a parse $d \in D$ as the product of the p values for all of the rewriting rules $R(\mathbf{y}, d)$ used in $\mathcal{S} \Rightarrow \mathbf{y}$. We also consider the case where d does not actually correspond to $\mathbf{y}$,

$$p(\mathbf{y}, d) = \begin{cases} \prod_{r\in R(\mathbf{y},d)} p(r), & \text{if } d \text{ is a parse of } y \\ 0, & \text{otherwise.} \end{cases}$$

The values of p are in the closed interval [0; 1]. In the cases where d is a parse of $\mathbf{y}$, all $p(r)$ values in the product will lie in the half open interval (0; 1], and so will the product. In the other case, 0 is in [0; 1] too. However, it is not always the case that $\sum_{d,\mathbf{y}} p(\mathbf{y}, d) = 1$.

The PCFG training takes as evidence the corpus of terminal sequences $\mathbf{y}$ with corresponding parses d from the training set S. It associates with each rule an expected probability of using the rule in producing the corpus. In the presence of parses for all terminal sequences, each rule probability is set to the expected count normalized so that the PCFG constraints (3) are satisfied:

$$p(A \to \alpha) = \frac{count(A \to \alpha)}{\sum_{A\to\beta\in R} count(A \to \beta)}.$$

3.1 Generalized probabilistic parsing

PCFGs are used as probabilistic models for natural languages, as they naturally reflect the "deep structure" of language sentences rather than the linear sequences of words. In a PCFG language model, a finite set of words serve as a terminal set and production rules for non-terminals express the full set of grammatical constructions in the language. Basic algorithms for PCFGs that find the most likely parse d for a given sequence $\mathbf{y}$ or choose rule probabilities that maximize the probability of sentence in a training set, represent [6] efficient extensions of the Viterbi and Baum-Welsh algorithms for hidden Markov models.

The tree annotation model processes sequences of observations $\mathbf{x} = \{x_1, \ldots, x_n\}$ from the infinite set X, where the observations x_i are not words in a language (and therefore terminals in T) but complex instances, like HTML leaves or groups of leaves.

Content fragments are frequently targeted by various probabilistic classifiers that produce probability estimates for labeling an observation with a terminal in T, $p(y|x_i)$, $y \in T$, where $\sum_y p(y|x_i) = 1$. The tree annotation problem can therefore be seen as *a generalized version of probabilistic context-free parsing*, where the input sequence is given by the probability distribution over a terminal set and the most probable annotation tree requires maximizing the joint probability in (2).

A similar generalization of probabilistic parsing takes place in speech recognition. In the presence of a noisy channel for speech streams, parsing from a sequence of words is replaced by parsing from a word lattice, which is a compact representation of a set of sequence hypotheses, given by conditional probabilities obtained by special acoustic models from acoustic observations [4].

4 Content classifier

To produce terminal estimates for the observations x_i, we adopt the maximum entropy framework, according to which the best model for estimating probability distributions from data is the one that is consistent with certain constraints derived from the training set, but otherwise makes the fewest possible assumptions [1]. The distribution with the fewest possible assumptions is one with the highest entropy, and closest to the uniform distribution. Each constraint expresses some characteristic of the training set that should also be present in the learned distribution. The constraint is based on a binary feature, it constrains the expected value of the feature in the model to be equal to its expected value in the training set.

One important advantage of maximum entropy models is their flexibility, as they allow the extension of the rule system with additional syntactic, semantic and pragmatic features. Each feature f is binary and can depend on $y \in T$ and on any properties of the input sequence $\mathbf{x}$. In the case of tree annotation, we include the *content features* that express properties on content fragments, like $f_1(x, y)$ ="1 if y is **title** and x's length is less then 20 characters, 0 otherwise", as well as the *structural and layout features* that capture the HTML context of the observation x, like $f_2(x, y)$="1 if y is **author** and x's father is `span`, 0 otherwise".

With the constraints based on the selected features $f(x, y)$, the maximum entropy method attempts to maximize the conditional likelihood of $p(y|x)$ which is represented as an exponential model:

$$p(y|x) = \frac{1}{Z_\alpha(x)} exp\left(\sum_\alpha \lambda_\alpha \cdot f_\alpha(x, y)\right), \quad (4)$$

where $Z_\alpha(x)$ is a normalizing factor to ensure that all the probabilities sum to 1,

$$Z_\alpha(x) = \sum_y exp\left(\sum_\alpha \lambda_\alpha f_\alpha(x,y)\right). \quad (5)$$

For the iterative parameter estimation of the Maximum Entropy exponential models, we use one of the quasi Newton methods, namely the Limited Memory BFGS method, which is observed to be more effective than the Generalized Iterative Scaling (GIS) and Improved Iterative Scaling (IIS) for NLP and IE tasks [7].

5 Sequential tree annotation

We use pairs $(\mathbf{x}, \mathbf{y})$ from triples $(\mathbf{x}, \mathbf{y}, d)$ of the training set S to train the content classifier C and pairs $(\mathbf{y}, d)$ to choose rule probabilities that maximize the likelihood for the instances in the training set. C predicts the terminal probabilities $p(\mathbf{y}|\mathbf{x})$ for any observation $\mathbf{x}$, while the inside-outside algorithm can find the parse d of the highest probability for a given terminal sequence $\mathbf{y}$.

By analogy with speech recognition, there exists a naive, sequential method to combine the two components C and G for computing a tree annotation for sequence $\mathbf{x}$. First, from C's estimates $p(\mathbf{y}|\mathbf{x})$, we determine the (top k) most probable sequences $\mathbf{y}_{max,j}$ for $\mathbf{x}$, $j = 1, \ldots, k$. Second, we find the most probable parses for all $\mathbf{y}_{max,j}$, $d_{max,j} = \underset{d}{argmax}\, p(d|\mathbf{y}_{max,j}, G)$; and finally, we choose the pair $(\mathbf{y}_{max,j}, d_{max,j})$ that maximizes the product $p(\mathbf{y}_{max,j}) \times p(d_{max,j})$.

The sequential method works well if the noise level is low (in speech recognition) or if the content classifier (in the tree annotation) is accurate enough in predicting terminals y for x_i. Unfortunately, it gives poor results once the classifier C is far from 100% accuracy in $\mathbf{y}$ predictions, as it faces the impossibility of finding any parse for all the top k most probable sequences $\mathbf{y}_{max,j}$.

Example. Consider an example target schema given by the following DTD:

```
<!ELEMENT  Book      (author, Section+)>
<!ELEMENT  Section   (title, (para | footnote)+)>
<!ELEMENT  author    (#PCDATA)>
<!ELEMENT  title     (#PCDATA)>
<!ELEMENT  para      (#PCDATA)>
<!ELEMENT  footnote  (#PCDATA)>
```

The reduction of the above schema definition to the Chomsky Normal Form will introduce extra nonterminals, so we get the PCFG $G = (T, N, \mathcal{S}, R)$, where the terminal set is T={`author, title, para, footnote`}, the nonterminal set is N= {Book, Author, SE, Section, TI, ELS, EL}, $\mathcal{S}$= Book, and R includes twelve production rules.

Assume that we have trained the content classifier C and the PCFG G and have obtained the following probabilities for the production rules in R:

(0.3) Book → AU Section
(0.4) SE → Section Section
(0.8) Section → TI ELS
(0.4) ELS → EL EL
(1.0) AU → `author`
(0.8) EL → `para`
(0.7) Book → AU SE
(0.6) SE → Section SE
(0.2) Section → TI EL
(0.6) ELS → EL ELS
(1.0) TI → `title`
(0.2) EL → `footnote`.

Assume now that we test the content classifier C and PCFG G on a sequence of five unlabeled observations $\mathbf{x} = \{x_1, \ldots, x_5\}$. Let the classifier C estimate the probability for terminals in T as given in the following table:

	x_1	x_2	x_3	x_4	x_5
`author`	0.3	0.2	0.1	0.1	0.2
`title`	**0.4**	**0.4**	0.3	0.3	**0.3**
`para`	0.1	0.2	**0.5**	0.2	0.2
`footnote`	0.2	0.2	0.1	**0.4**	0.2

According to the above probability distribution, the most probable terminal sequence $\mathbf{y}_{max}$ is composed of the most probable terminals for all x_i, $i = 1, \ldots, 5$. It is `'title title para footnote title'` with probability $p(\mathbf{y}_{max}) = p(\mathbf{y}_{max}|x) = \Pi_i \cdot p(\mathbf{y}_i^{max}|x_i) = 0.4 \cdot 0.4 \cdot 0.5 \cdot 0.4 \cdot 0.3 = 0.0096$. However, $\mathbf{y}_{max}$ has no corresponding parse tree in G. Instead, there exist two valid annotation trees for $\mathbf{x}$, $(\mathbf{y}_1, d_1)$ and $(\mathbf{y}_2, d_2)$, as shown in Figure 1. In Figure 1.b, the terminal sequence $\mathbf{y}_2$=`'author title para title para'` with the parse d_2=Book(AU SE(Section (TI EL) Section (TI EL))) maximizes the joint probability $p(\mathbf{y}, d|\mathbf{x}, G)$, with $p(\mathbf{y}_2) = 0.3 \cdot 0.4 \cdot 0.5 \cdot 0.3 \cdot 0.2 = 0.0036$, and

$$\begin{aligned} p(d_2) = & p(\text{Book} \rightarrow \text{AU SE}) \cdot p(\text{AU} \rightarrow \texttt{author}) \times \\ & p(\text{SE} \rightarrow \text{Section Section}) \cdot p(\text{Section} \rightarrow \text{TI EL}) \times \\ & p(\text{TI} \rightarrow \texttt{title})^2 \cdot p(\text{EL} \rightarrow \texttt{para})^2 \times \\ & p(\text{Section} \rightarrow \text{TI EL}) \\ = & 0.7 \cdot 1.0 \cdot 0.4 \cdot 0.2 \cdot 1.0^2 \cdot 0.8^2 \cdot 0.2 = 0.007172. \end{aligned}$$

Jointly, we have $p(\mathbf{y}_2) \times p(d_2) \approx 2.58 \cdot 10^{-5}$. Similarly, for the annotation tree in Figure 1.a, we have $p(\mathbf{y}_1) \times p(d_1) = 0.0048 \cdot 0.0018432 \approx 8.85 \cdot 10^{-6}$.

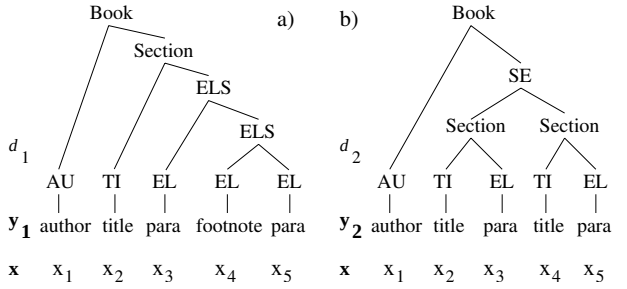


Figure 1: Tree annotations for the example sequence.

6 The most probable annotation tree

As the sequential method fails to find the most probable annotation tree, we try to couple the selection of terminal sequence $\mathbf{y}$ for $\mathbf{x}$ with finding the most probable parse d for $\mathbf{y}$, such that $(\mathbf{y}, d)$ maximizes the probability product $p(d|\mathbf{y}, G) \cdot p(\mathbf{y}|\mathbf{x})$ in (2). For this goal, we extend the

basic inside-outside algorithm for terminal PCFGs [6]. We redefine the *inside probability* as the most probable joint probability of the subsequence of $\mathbf{y}$ beginning with index i and ending with index j, and the most probable partial parse tree spanning the subsequence $\mathbf{y}_i^j$ and rooted at nonterminal A:

$$\beta_A(i,j) = max_{A,\mathbf{y}_i^j}\ p(A_{i,j} \Rightarrow \mathbf{y}_i^j) \cdot p(\mathbf{y}_i^j|\mathbf{x}). \quad (6)$$

The inside probability is calculated recursively, by taking the maximum over all possible ways that the nonterminal A could be expanded in a parse,

$$\beta_A(i,j) = \quad max_{i\le q\le j}\ p(A \to BC) \cdot p(B \Rightarrow \mathbf{y}_i^q) \times p(C \Rightarrow \mathbf{y}_{q+1}^j) \cdot p(\mathbf{y}_i^j|\mathbf{x}).$$

To proceed further, we make the *independence assumption* about $p(\mathbf{y}|\mathbf{x})$, meaning that for any q, $i \le q \le j$, we have $p(\mathbf{y}_i^j|\mathbf{x}) = p(\mathbf{y}_i^q|\mathbf{x}) \cdot p(\mathbf{y}_{q+1}^j|\mathbf{x})$. Then, we can rewrite the above as follows

$$\beta_A(i,j) \quad = max_{i\le q\le j} p(A \to BC) \cdot p(B \Rightarrow \mathbf{y}_i^q) \times \quad (7)$$
$$p(C \Rightarrow \mathbf{y}_{q+1}^j) \cdot p(\mathbf{y}_i^q|\mathbf{x}) \cdot p(\mathbf{y}_{q+1}^j|\mathbf{x}) \quad (8)$$
$$= max_{i\le q\le j} p(A \to BC) \cdot \beta_B(i,q) \cdot \beta_C(q+1,j) \quad (9)$$

The recursion is terminated at the $\beta_{\mathcal{S}}(1,n)$ which gives the probability of the most likely tree annotation $(\mathbf{y}, d)$,

$$\beta_{\mathcal{S}}(1,n) = max\ p(\mathcal{S} \Rightarrow \mathbf{y}_1^n) \cdot p(\mathbf{y}_1^n|\mathbf{x}),$$

where n is the length of both sequences $\mathbf{x}$ and $\mathbf{y}$.

The initialization step requires some extra work, as we select among all terminals in T being candidates for y_k:

$$\beta_A(k,k) = max_{y_k} p(A \to y_k) \cdot p(y_k|\mathbf{x}). \quad (10)$$

It can be shown that the redefined inside function converges to a local maximum in the (Y, D) space. The extra work during the initialization step takes $O(n \cdot |T| \cdot |N|)$ time which brings the total complexity of the extended IO algorithm to $O(n^3 \cdot |N| + n \cdot |T| \cdot |N|)$.

The independence assumption established above represents the *terminal conditional independence*, $p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} p(y_i|\mathbf{x})$ and matches the Naive Bayes assumption. The assumption is frequent in text processing; it simplifies the computation by ignoring the correlations between terminals. Here however it becomes a requirement for the content classifier. While PCFGs are assumed to capture all (short- and long- distance) relations between terminals, the extended inside algorithm (9)-(10) imposes the terminal conditional independence when building the probabilistic model. This impacts the feature selection for the maximum entropy model, by disallowing features that include terminals of neighbor observations y_{i-1}, y_{i+1}, etc, as in the maximum entropy extension with HMM and CRF models [8; 5].

7 Experimental results

We have tested our method for XML annotation on two collections. One is the collection of 39 Shakespearean plays available in both HTML and XML format.[1] 60 scenes with 17 to 189 leaves were randomly selected for the evaluation. The DTD fragment for scenes consists of 4 terminals and 6 non-terminals. After the binarization, the PCFG in CNF contains 8 non-terminals and 18 rules.

The second collection, called TechDoc, includes 60 technical documents from repair manuals. [2] The target documents have a fine-grained semantic granularity and are much deeper than in the Shakespeare collection; the longest document has 218 leaves. The target schema is given by a complex DTD with 27 terminals and 35 nonterminals. The binarization increased the number of non-terminals to 53. For both collections, a content observation refers to a PCDATA leaf in HTML.

To evaluate the annotation accuracy, we use two metrics. The *terminal error ratio* (TER) is similar to the word error ratio used in natural language tasks; it measures the percentage of correctly determined terminals in test documents. The second metric is the *non-terminal error ratio* (NER) which is the percentage of correctly annotated sub-trees.

As content classifiers, we test with the maximum entropy (ME) classifier. For the ME model, we extract 38 *content features* for each observation, such as the number of words in the fragment, its length, POS tags, textual separators, etc. Second, we extract 14 *layout and structural features* include surrounding tags and all associated attributes. Beyond the ME models, we use the maximum entropy Markov models (MEMM) which extends the ME with hidden Markov structure and terminal conditional features [8]. The automaton structure used in MEMM has one state per terminal.

In all tests, a cross-validation with four folds is used. ME and MEMM were first tested alone on both collections. The corresponding TER values for the most probable terminal sequences $\mathbf{y}_{max}$ serve a reference for methods coupling the classifiers with the PCFG. When coupling the ME classifier with the PCFG, we test both the sequential and joint methods. Additionally, we included a special case MEMM-PCFG where the content classifier is MEMM and therefore the terminal conditional independence is not respected.

The results of all the tests are collected in Table 1. The joint method shows an important advantage over the sequential method, in particular in the TechDoc case, where the ME content classifier alone achieves 86.23% accuracy and the joint method reduces the errors in terminals by 1.36%. Instead, coupling MEMM with the PCFG reports a decrease of TER values and a much less important NER increase.

[1] http://metalab.unc.edu/bosak/xml/eg/shaks200.zip.
[2] Available from authors on request.

Method	TechDoc		Shakespeare	
	TER	NER	TER	NER
ME	86.23	–	**100.0**	–
MEMM	78.16	–	99.91	–
Seq-ME-PCFG	86.23	9.38	**100.0**	82.87
Jnt-ME-PCFG	**87.59**	**72.95**	99.97	**99.79**
Jnt-MEMM-PCFG	75.27	56.25	98.09	94.01

Table 1: Evaluation results.

8 Relevant Work

Since the importance of semantic annotation of documents has been widely recognized, the migration of documents from rendering-oriented formats, like PDF and HTML, toward XML has become an important research issue in different research communities [2; 3; 11; 12]. The majority of approaches either constrain the XML conversion to a domain specific problem, or make different kinds of assumptions about the structure of source and target documents. In [11], the source HTML documents are assumed to be dynamically generated through a form filling procedure, as in Web news portals, while a portal subject ontology permits the semantic annotation of the generated documents.

Transformation-based learning is used for automatic translation from HTML to XML in [3]. It assumes that source documents can be transformed into target XML documents through a series of proximity tag operations, including insert, replace, remove and swap. The translation model trains a set of transformation templates that minimizes an error driven evaluation function.

In document analysis research, Ishitani in [12] applies OCR-based techniques and the XY-cut algorithm in order to extract the logical structure from page images and to map it into a pivot XML structure. While logical structure extraction can be automated to a large extent, the mapping from the pivot XML to the target XML schema remains manual.

In natural language tasks, various information extraction methods often exploit the sequential nature of data to extract different entities and extend learning models with grammatical structures, like HMM [8] or undirected graphical models, like Conditional Random Fields [5]. Moreover, a hierarchy of HMMs is used in [13] to improve the accuracy of extracting specific classes of entities and relationships among entities. A hierarchical HMM uses multiple levels of states to describe the input on different level of granularity and achieve a richer representation of information in documents.

9 Conclusion

We propose a probabilistic method for the XML annotation of semi-structured documents. The tree annotation problem is reduced to the *generalized probabilistic context-free parsing* of an observation sequence. We determine the most probable tree annotation by maximizing the joint probability of selecting a terminal sequence for the observation sequence and the most probable parse for the selected terminal sequence.

We extend the inside-outside algorithm for probabilistic context-free grammars. We benefit from the available tree annotation that allows us to extend the inside function in a rigorous manner, and avoid the extension of the outside function which might require approximation.

The experimental results are promising. In future work, we plan to address different challenges in automating the HTML-to-XML conversion. We are particularly interested in extending the annotation model with the source tree structures that have been ignored so far.

10 Acknowledgement

This work is supported by VIKEF Integrated Project co-funded under the EU 6th Framework Programme.

References

[1] A. L. Berger, S. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[2] N. Sundaresan, Ch. Chung, M. Gertz. Reverse engineering for web data: From visual to semantic structures. In *Proc. 18th ICDE'02, San Jose, California*, 2002.

[3] J.R. Curran and R.K. Wong. Transformation-based learning for automatic translation from HTML to XML. In *Proc. 4th Australas. Doc. Computing Symp.*, 1999.

[4] K. Hall and M. Johnson. Language modeling using efficient best-first bottomup parsing. In *IEEE Autom. Speech Recogn. and Understanding Workshop*, 2003.

[5] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of 18th ICML'01, San Francisco, CA*, pages 282–289, 2001.

[6] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[7] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. of 6th Conf. on Natural Language Learning*, pages 49–55, 2002.

[8] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th ICML'00, San Francisco, CA*, pages 591–598, 2000.

[9] Frank Neven. Automata Theory for XML Researchers. *SIGMOD Record*, 31(3):39–46, 2002.

[10] Yannis Papakonstantinou and Victor Vianu. DTD Inference for Views of XML Data. In *Proc. of 19th ACM PODS, Dallas, Texas, USA*, pages 35–46, 2000.

[11] I.V. Ramakrishnan, S. Mukherjee, G. Yang. Automatic annotation of content-rich web documents: Structural and semantic analysis. In *Intern. Sem. Web Conf.*, 2003.

[12] Y. Ishitani. Document transformation system from papers to xml data based on pivot document method. In *Proc. of 7th ICDAR'03*, pages 250–255, 2003.

[13] M. Skounakis, M. Craven, and S. Ray. Hierarchical hidden markov models for information extraction. In *Proc. of 18th IJCAI, Acapulco, Mexico*, 2003.

Maps for Verbs: The Relation Between Interaction Dynamics and Verb Use

Paul R. Cohen and **Clayton T. Morrison**
Center for Research on Unexpected Events (CRUE)
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{cohen,clayton}@isi.edu

Erin Cannon
Department of Psychology
University of Massachusetts
Amherst, MA 01003
ecannon@psych.umass.edu

Abstract

We report a study of word meaning that tests whether dynamical aspects of movies predict word use. The movies were based on a novel representation of verb semantics called *maps for verbs*. We asked preschool-school-age children to describe the movies, and demonstrated that their distributions of words could be predicted by the dynamical aspects of the movies. These results lend support to the empiricist position that word meanings are learned associatively.

1 Introduction

Previous work has shown that robots can learn the meanings of words by associating aspects of the perceptual array with utterances (e.g., [Steels, 1999; Oates, 2002; Siskind, 2001; Vogt, 2002; Cohen and Oates, 1998]). In this paper we test the conjecture that the dynamics of the perceptual array, specifically, the manner in which objects move and interact, explains the production of particular verbs. The work is part of a larger project to model human language development on robot platforms, and is based on the *maps for verbs* framework described in Section 3. A persistent question in the project is whether the perceptual array contains enough information to provide semantics for words, or, in a slightly different formulation, what fraction of the variability of word use is explained by information in the perceptual array? A concrete version of this question is posed here: What fraction of the variability of word use is explained by the dynamical aspects of interactions between two objects?

We developed 18 movies of interactions between objects. Each movie was generated by a program written in *breve* [Klein, 2002], an animation tool with good physics. A vector of parameters of each program characterizes the dynamics of the corresponding movie. We showed the movies to preschool children and asked them to describe the action in the movies. After removing non-content words, we characterized each movie by a distribution of words. We ranked all pairs of movies in two ways: by the similarities of their word distributions and by the similarities of their vectors of program parameters. Then we compared the rankings. The results are highly significant: there is strong dependence between the parameters of the programs that generated the movies and the distributions of words that children use to describe the movies.

2 Related Work

Interest in the perception of the dynamics of whole-body interactions is not new. Heider and Simmel [1944] report a study in which adults were shown a film of animated shapes interacting with each other in and around a box. After watching the film, participants were asked to describe what happened in the film. Heider and Simmel found a strong tendency to attribute a rich set of intentions to the moving objects and a story-line describing the interactions, even though the only information in the stimuli was the shape and size of the objects and their motion dynamics.

More recently, several psychologists and computer scientists have explored dynamic maps as representations of activities, often for machine recognition and modeling of human gesture and bodily movement (e.g., [Thelen and Smith, 1994; Rosenstein, 1998; Intille and Bobick, 1999; Bobick and Davis, 2001].

Blythe, Todd & Miller [1999] and Todd & Barrett [2000] present a preliminary study of adult and child perception of intention based on the dynamics of motion between two simple interacting bodies. Their work also uses the tools of dynamic map representation to characterize interactions. Their results suggest that such dynamics are implicated in people's categorization of interactions, although their focus was on intention, rather than more primitive verb classes.

For a comprehensive review of the psychological literature on dynamics and word meanings see [Cannon and Cohen, 2005].

3 Maps for Verbs

In the *maps for verbs* representation of verb meanings, the denotations of verbs dealing with interactions between two bodies, such as push, hit, chase, and so on, are represented as pathways through a metric space, or map, the axes of which are perceived distance, velocity, and energy transfer [Cohen, 1998]. Verbs with similar meanings have similar pathways. A scene, such as one object chasing another, is thought to be perceived as a pathway through the map. To learn verb meanings, one simply associates verbs that describe scenes with the corresponding pathways.

Although maps are compact and objective representations of some verb meanings, we do not know whether they have psychological reality — whether humans use maps to assign meanings to verbs. Even if they do, the original maps for verbs representation might have the wrong axes, or the axes might be correct but verbs might not be correlated with the particular features of pathways, as we thought. The experiment in this paper does not test whether humans have maps in their heads. Instead it asks, "If one creates movies which are different according to the maps for verbs framework, will human subjects use different distributions of words to describe them?"

In the maps for verbs framework, the dynamics of interaction are split into before, during (contact), and after phases. Figure 1 depicts these phases with illustrative trajectories in eac. The axes of the maps are the same in the *before* and *after* phases: they are relative velocity and distance between the two bodies. Relative velocity is the difference between the velocity of one body, A, and another, B: $Velocity(A) - Velocity(B)$. Many verbs (e.g., transitive verbs) predicate one body as the "actor" and the other as the "target" (or "subject" or "recipient") of the action. For example, in a push interaction, the actor does the pushing, and the target is the body being pushed. By convention, the actor is designated as A and the target is B. Thus, when relative velocity is positive, the actor's velocity is greater than that of the target; and when relative velocity is negative, the target's velocity is greater than that of the actor. Distance, in turn, is simple Euclidean distance between the bodies.

The vertical dimension of the map in the *during* phase is perceived energy transfer (from the actor to the target). If energy transfer is positive, then the actor is imparting to the target more energy than the target originally had; if energy transfer is negative, then the situation is reverse and the target is imparting more energy to the actor. Since energy transfer is not directly perceivable, we approximate it by calculating the acceleration of the actor in the direction of the target while the actor and target are in contact.

The labeled trajectories in Figure 1 characterize the component phases of seven interaction types, as described by the verbs push, shove, hit, harass, bounce, counter-shove and chase.

For example, $\langle \mathbf{b}, \mathbf{b}, \mathbf{b} \rangle$ describes a *shove*. The actor approaches the target at a greater velocity than the target, closing the distance between the two bodies. As it nears the target, the actor slows, decreasing its velocity to match that of the target. Trajectory **b** of the before phase in Figure 1 illustrates these dynamics, showing the decrease in relative velocity, along with decrease in distance. At contact, the relative velocity is near or equal to zero. During the contact phase, the actor rapidly imparts more energy to the target in a short amount of time, as illustrated by **b** of the during/contact phase. And after breaking-off contact with the target, the agent rapidly decreases its velocity while the target moves at a greater velocity due to the energy imparted it.

In Figure 2(b), below, we provide a plot of the dynamics of a simulated shove action. The map in the figure plots the dynamics for a portion of the time between contact phases. The trajectory begins with very low relative velocity, as would be expected just after completing the contact phase of a shove (after phase b in Figure 1), and ends with a high relative velocity that is ramping down (before phase b in Figure 1) just before a new shove occurs.

With this three-phase representation scheme, we define six more interaction types corresponding to common English verbs:

- *Push* $\langle \mathbf{b}, \mathbf{a}, \mathbf{a} \rangle$ – begins like shove, but at contact relative velocity is near or equal to zero and the actor smoothly imparts more energy to the target; after breaking contact, the agent gradually decreases its velocity.
- *Hit* $\langle \mathbf{c}/\mathbf{d}, \mathbf{c}, \mathbf{c} \rangle$ – may begin with the actor already at high velocity relative to the target or increasing in relative velocity, and thus is characterized by **c** or **d** in the before phase.
- *Harass* $\langle \mathbf{c}/\mathbf{d}, \mathbf{c}, \mathbf{d} \rangle$ – is similar to a hit, except the after-phase involves the actor quickly recovering its speed and moving back toward the target, not allowing the distance between the two to get very large (the **d** in the after phase). Harass highlights that interactions may be cyclic: the after phase of one epoch blends into the before phase of the next.
- *Bounce* $\langle \mathbf{c}/\mathbf{d}, \mathbf{d}, \mathbf{e} \rangle$ – along with counter-shove, bounce involves the target making a more reactive response to the actor's actions. Bounce begins like a hit or harass, but at contact, the target transfers a large amount of energy back to the actor.
- *Counter-shove* $\langle \mathbf{b}/\mathbf{c}/\mathbf{d}, \mathbf{e}, \mathbf{e} \rangle$ – is a version of a shove where the target imparts energy to the actor.
- *Chase* $\langle \mathbf{a}, -, - \rangle$ – involves the actor moving toward the target, closing the distance between the two, but never quite making contact, so the during and after phases are not relevant. This is depicted as the circular trajectory **a** in the before phase.

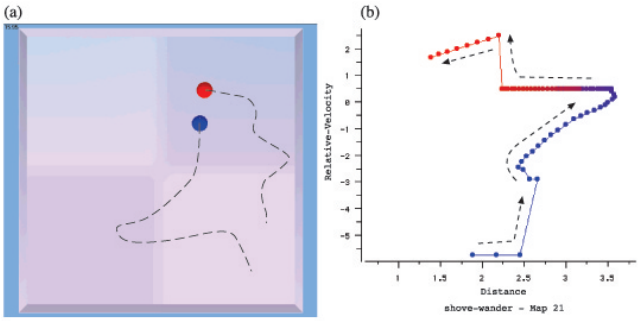


Figure 2: (a) Example of maps-for-verbs simulation running the shove-wander action, as rendered in *breve 1.4*. (Note: dashed lines represent motions of colored patches for demonstration purposes; only the moving color-patches themselves were displayed in the stimuli movies.); (b) Dynamic map plot of shove-wander action before contact, corresponding to the picture in (a) (x-axis = distance between agents, y-axis = relative velocity).

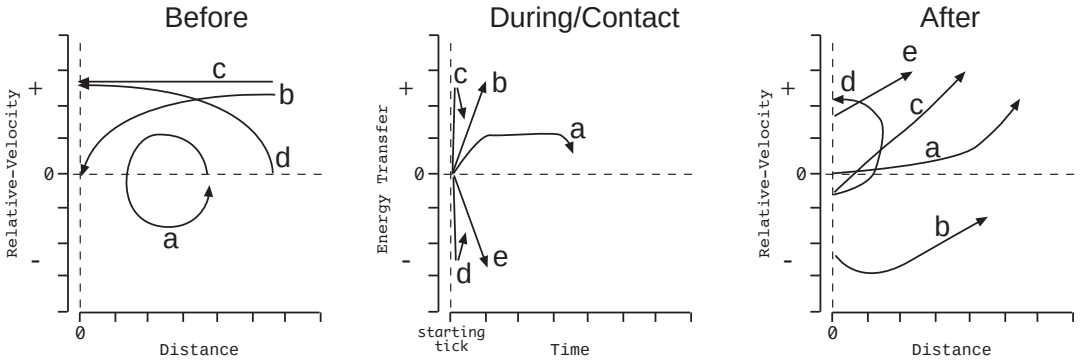


Figure 1: Maps-for-verbs model of the three phases of interaction.

4 Experiment

4.1 Stimuli

We used *breve 1.4*, an environment for developing realistic multi-body simulations in a three dimensional world with physics [Klein, 2002], to implement a model of the seven interaction classes described in the previous section. The models were rendered as two generic objects (a blue ball for the actor and a red ball for the target) moving on a white background — see Figure 2(a). The models allowed us to generate multiple instances of each interaction type.

We generated a set of movies based on each of the *breve* interaction models. For several of the interaction classes we also varied the behavior of the target object, as follows: the target object, (a) did not move except when contacted ("stationary"), (b) moved independently in a random walk ("wander"), or (c) moved according to billiard ball ballistic physics, based on the force of the collision ("coast"). We generated a total of 17 unique movies. For the bounce and countershove interaction types, we only implemented "stationary" and "wander" target behavior, as "coast" would obliterate the effect of the target transferring energy back to the actor. Also, there was only one version of "chase," as the target must always be moving away from the actor. Chase was also unique because it was the only instance in which the two balls never contacted each other.

The 17 movies were recorded and presented on a G3 iMac with 14 inch screen. The children's responses were recorded and later transcribed.

Participants

Sixteen children participated in this study, ranging in age from 26-60 months old (average age = 50 months). Participants were recruited and tested at a local daycare in Amherst, MA.

Procedure

For each child, a total of 18 movies was presented, each movie instance appearing once — with the exception of chase, which the child watched twice. An experimenter told each child that she would be watching movies on the computer screen with two balls, one blue and one red, and that the task was to tell a story about what the balls were doing.

4.2 Analysis

The children used remarkably small vocabularies and very terse sentences to describe the movies; the following transcript is typical:

E: Okay, last one. Can you tell me a good story about this one?
S: Even gooder than all of the other ones?
E: Make it the best story!
S: It's going umm gooder and it's playing but the red is letting the blue push him. And the red is letting the blue one push
E: How come he's letting the blue push?
S: Because he wanted to.
E: Why does he want to?
S: Because he likes to play like that.

All the content words for each trial were extracted and "canonicalized," converting verbs in different tenses or forms (e.g., ending in -ed, -ing, etc.) to a single form. Also, negation phrases, such as "it's not zooming" or "red didn't move," were also transformed into a single token, e.g., not-zooming and not-moving. The total number of unique, canonicalized content words uttered by all the subjects in response to all the movies was 104, of which the following 30 words were uttered more than three times (words are listed with their frequencies): PUSHING, 85; MOVING, 57; BONKING, 54; AWAY, 46; TRYING, 38; PLAYING, 28; FAST, 27; RUNNING, 27; AROUND, 25; UP, 25; GETTING, 21; CHASING, 19; FRIENDS, 18; BUMPING, 17; SLOW, 16; HITTING, 16; DOWN, 16; CIRCLE, 11; CATCHING, 10; STANDING, 7; TAG, 7; ZOOMING, 6; STOPPING, 6; COMING, 4; FLYING, 4; KNOCKING, 4; FOLLOWING, 4; BOUNCING, 4; ABOUT, 4; TOGETHER, 4.

Note that some of these words are not verbs; for instance, "friends" and "away."

Each movie is characterized by a vector of relative frequencies of these 30 words. For example, here is the vector for the movie SHOVE-STATIONARY:

PUSHING, 0.192; MOVING, 0.115; BONKING, 0.0; AWAY, 0.115; TRYING, 0.038; PLAYING, 0.0; FAST, 0.115; RUNNING, 0.077; AROUND, 0.0; UP, 0.0; GETTING, 0.0; CHASING, 0.038; FRIENDS, 0.077; BUMPING, 0.038; SLOW, 0.038; HITTING, 0.0; DOWN, 0.0; CIRCLE, 0.0; CATCHING, 0.038; STANDING,

:BEFORE-SUBJECT-ACTION	'agent-coast
:BEFORE-ACTOR-ACTION	'approach-slow-down
:BEFORE-ACTOR-ORIGINAL-SPEED	2
:BEFORE-ACTOR-DESIRED-SPEED	.5
:BEFORE-ACTOR-DISTANCE-FOR-SLOW-DOWN	1.5
:BEFORE-ACTOR-DISTANCE-FOR-STOP	0
:DURING-ACTOR-ACTION	'agent-approach-speed-up
:DURING-ACTOR-SPEED-UP-START-TIME	'self
:DURING-ACTOR-SPEED-UP-END-TIME	.4
:DURING-ACTOR-ORIGINAL-SPEED	'self
:DURING-ACTOR-DESIRED-SPEED	6
:DURING-ACTOR-SELF-WAIT	.6
:DURING-2-ACTOR-ACTION	'agent-halt
:DURING-2-ACTOR-SELF-WAIT	.5
:AFTER-SUBJECT-ACTION	'agent-coast
:AFTER-ACTOR-ACTION	'agent-run-away-speed-up
:AFTER-ACTOR-SPEED-UP-START-TIME	'self
:AFTER-ACTOR-SPEED-UP-END-TIME	1
:AFTER-ACTOR-ORIGINAL-SPEED	'self
:AFTER-ACTOR-DESIRED-SPEED	2
:AFTER-2-SUBJECT-ACTION	'agent-halt
:AFTER-2-ACTOR-SELF-WAIT	1

Table 1: A vector of parameters for the movie SHOVE-STATIC

0.0; TAG, 0.0; ZOOMING, 0.038; STOPPING, 0.0; COMING, 0.0; FLYING, 0.0; KNOCKING, 0.0; FOLLOWING, 0.0; BOUNCING, 0.038; ABOUT, 0.038; TOGETHER, 0.0.

That is, of the content words used to describe the movie SHOVE-STATIONARY, 19% of them were PUSHING, 11.5% were MOVING, and so on. Let V_{words}(movie) denote the vector of relative frequencies of words used to describe a movie.

The movies also can be characterized by the parameters of the *breve* programs that generated them. These parameters include the current speed and desired speed of the objects, the subroutines that implement patterns of movement, the distance between objects at which one slows down, the latency before starting the next movement, and so on. Table 1 shows the vector for the movie SHOVE-STATIC. Let V_{params}(movie) denote the vector of parameters for the program that generates the movie.

The next step of the analysis is to test whether there is an association between the vectors of parameters for movies, V_{params}(movie), and the vectors of words used to describe the movies, V_{words}(movie). A simple method is to calculate the similarities between pairs of movies i and j, rank the pairs by similarity, and see whether the ranking based on word vectors is predicted by the ranking based on parameter vectors. Let

$$Sim_{words}(i,j) = f(V_{words}(i), V_{words}(j)) \quad (1)$$

$$Sim_{params}(i,j) = g(V_{params}(i), V_{params}(j)) \quad (2)$$

where f and g are methods for comparing word-frequency vectors and parameter vectors, respectively. The question is whether, averaged over pairs of movies i and j, $Sim_{words}(i,j)$ is predicted by $Sim_{params}(i,j)$. We let f be generalized Euclidean distance between the probabilities $p(w_i)$ and $p(w_j)$ of hearing word w uttered in response to movies i and j, respectively:

$$f(i,j) = \sqrt{\sum_{w \in \{pushing, moving, ..\}} (p(w_i) - p(w_j))^2} \quad (3)$$

The function g, for comparing parameter vectors, cannot be simple Euclidean distance because parameter vectors include non-numeric values. We wrote a function that increases the similarity score when symbolic (and numeric values) are identical, decreases the score by a constant when symbolic values don't match, and decreases the score proportional to the mismatch between numeric values. We added some conditions for missing values. We are aware that this function might conceivably be "tuned" to make the parameter vectors better predict the word vectors, so we wrote it once and did not revise it. The results presented below are for the first and only evaluation of this function on these data.

As a final step, we ranked pairs of movies according to $Sim_{words}(i,j)$ and $Sim_{params}(i,j)$.

4.3 Results

Each pair of movies i, j gets two similarity scores, $Sim_{words}(i,j)$ and $Sim_{params}(i,j)$, so we can look at the simple correlation of these scores and the regression of $Sim_{params}(i,j)$ on $Sim_{words}(i,j)$. Figure 3 shows these score plotted against each other and the linear regression line that fits them best. Clearly, the similarity of two movies according to their parameters is a good predictor of the similarity of distributions of words used to describe the movies. The correlation between them is 0.949, which means the similarity of two movies according to their parameters accounts for 90% of the variance in the similarity of the word vectors for the movies. This result is highly significant ($p < .0001$).

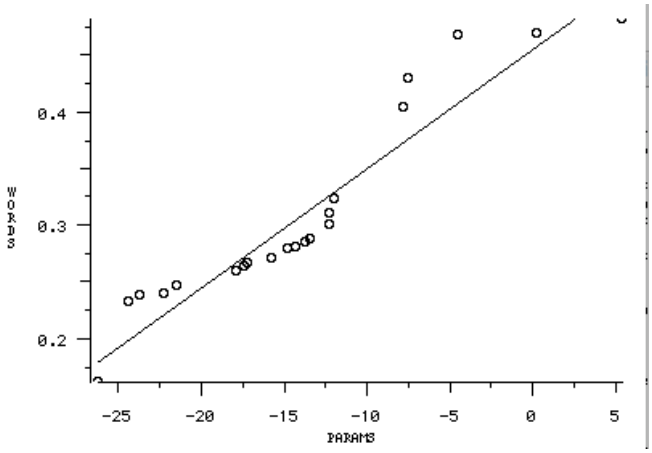


Figure 3: Regression plot of similarity scores for movie pairs according to movie parameters and movie word descriptions.

Another way to analyze the data is to compare the rankings of pairs of movies according to the two similarity scores. To illustrate, suppose we have four movies, a, b, c, d, and so six pairs of movies, ranked by function f from most to least similar: $((a,b)(a,c)(b,d)(a,d)(b,c)(c,d))$. Now suppose function g produces a different ranking: $((b,d)(a,c)(a,d)(c,d)(b,c)(a,b))$. According to f, the most similar movies, with rank 1, are (a,b) whereas the rank of these movies is 6 according to g. We can characterize the difference of two rankings by summing the differences in rank over items; for instance, the item (a,b) contributes $6-1=5$ to the sum.

For the data in Figure 3, the summed rank difference is

$\delta = 104$, but how can we tell whether this is a statistically significant number? We perform a *randomization test* as follows: There are 21 pairs of movies, so construct two vectors, v_1 and v_2, each containing the numbers $1...21$. Now shuffle v_2 thoroughly and calculate $\delta* = \sum_{i=[0,21]} abs(v_{1,i} - v_{2,i})$. Repeat 1000 times. The resulting distribution of $\delta*$ is the *empirical sampling distribution* of the summed rank difference under the *null hypothesis* that the ranks of items in the vectors are unrelated. If the actual summed rank difference, δ, has low probability according to this sampling distribution, we reject the null hypothesis with residual uncertainty (the p value) equal to the quantile of δ in the sampling distribution (see [Cohen, 1995] for details).

As it happens, the ranking of movie pairs according to the movie parameters is not independent of the ranking according to word frequencies. The summed rank difference $\delta = 104$ is just the 0.0016 quantile of the sampling distribution, so we can reject the null hypothesis and conclude with confidence that the rankings are related.

5 Discussion

These results are supported by another study, using the same movies, with adult subjects. (The final paper will present both sets of results.) They show conclusively that the dynamics of interactions between two bodies, as represented by the parameters of the programs that generate movies, predict the distributions of words that children use to describe the movies. This result is surprisingly strong when one considers how few words children actually use. Although our study involved 16 children, only 30 content words were uttered more than three times in the entire study. Nevertheless, by concentrating on the *distributions* of these words for each movie we were able to show a strong dependence on the dynamics of the movies.

Returning to the question that motivates this work, is there sufficient information in the perceptual array, particularly dynamical information, to supply semantics for some words, particularly some verbs? By demonstrating a dependency between dynamical information and word choice (especially in young children) we strengthen the empiricist case that word meanings can be learned as associations between the words and percepts. Our evidence is only suggestive, however, because we demonstrated a dependence of word use not on the child's *percepts* but, rather, on the parameters of the movies shown to the child. Only by assuming that these parameters affect how the movie is perceived can we argue for associative learning of word meanings. The assumption is very reasonable; after all, the parameters were tuned to make the movies look different and distinctive.

Still, we think it likely that associative learning of word meanings "needs help," probably from prior domain knowledge. Although researchers such as Todd and Barrett [Todd and Barrett, 2000] argue that motion is a cue to intention, they do not argue that it is the *only* cue, and one is hard-pressed to see how the intentional aspect of, say, "chasing," can be learned from nothing but the relative motions of two bodies. We think Dennett [Dennett, 1987] is probably right when he says we adopt an intentional stance even to non-intentional scenarios, and the intentional language of our subjects, directed to a couple of colored blobs moving on the screen, lends force to his argument. It seems likely to us that perceived movement is one cue to word meaning, but the intentional aspects of words are generated by the subjects themselves. We are currently designing studies to tease apart these contributors to word meanings.

Finally, we relate an anecdote about our word-learning robot: It learned that "forward" is associated with the wheels rotating in one direction and "backward" with the opposite direction of rotation, but it never learned that forward and backward are antonyms. We realized then that word meanings can be arbitrarily "deep" and that associating percepts and words produces a shallow kind of meaning. But it's a start.

References

[Blythe *et al.*, 1999] P. W. Blythe, P. M. Todd, and G. F. Miller. *How motion reveals intention: Categorizing social interactions*, pages 257–285. Oxford University Press, New York, NY, 1999.

[Bobick and Davis, 2001] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Transaction on Pattern Analysis & Machine Intelligence*, 23(3), 2001.

[Cannon and Cohen, 2005] E.N. Cannon and P.R Cohen. *Word choice as a conditional probability: Motion-based semantics*. Oxford University Press, New York, NY, 2005.

[Cohen and Oates, 1998] P. R. Cohen and T. Oates. A dynamical basis for the semantic content of verbs. In *Grounding of Word Meaning: Data & Models Workshop, AAAI-98*, pages 5–8, 1998.

[Cohen, 1995] Paul R. Cohen. *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, 1995.

[Cohen, 1998] P. R. Cohen. Maps for verbs. In *Proceedings of the Information and Technology Systems Conference, Fifteenth IFIP World Computer Congress*, pages 21–33, 1998.

[Dennett, 1987] D. Dennett. *The Intentional Stance)*. MIT Press, 1987.

[Heider and Simmel, 1944] F. Heider and M. Simmel. An experimental study of apparent behaviour. *American Journal of Psychology*, 57(2):243–59, 1944.

[Intille and Bobick, 1999] S. Intille and A. Bobick. A framework for recognizing multi-agent action from visual evidence. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 518–525, Orlando, FL, July 1999.

[Klein, 2002] J Klein. breve: a 3d simulation environment for the simulation of decentralized systems and artificial life. In *Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems*, 2002. http://www.spiderland.org/breve/.

[Oates, 2002] T. Oates. Peruse: An unsupervised algorithm for finding recurring patterns in time series. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[Rosenstein, 1998] M. T. Rosenstein. Concepts from time series. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 739–745, 1998.

[Siskind, 2001] J. Siskind. Grounding lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of AI Research*, 15:31 – 90, 2001.

[Steels, 1999] Luc Steels. *The Talking Heads Experiment: Volume I. Words and Meanings*. Laboratorium, Antwerpen, 1999. This is a museum catalog but is in preparation as a book.

[Thelen and Smith, 1994] E. Thelen and L. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action*. MIT Press, Cambridge, MA, 1994.

[Todd and Barrett, 2000] P.M. Todd and H. C. Barrett. Judgment of domain-specific intentionality based solely on motion cues. Paper presented at the 12th Annual Meeting of the Human Behavior and Evolution Society, 2000.

[Vogt, 2002] P. Vogt. Anchoring symbols to sensorimotor control, 2002.

Learning to Understand Web Site Update Requests

William W. Cohen
Center for Automated Learning & Discovery
Carnegie Mellon University
wcohen@cs.cmu.edu

Einat Minkov
Language Technologies Institute
Carnegie Mellon University
einat@cs.cmu.edu

Anthony Tomasic
Institute for Software Research
Carnegie Mellon University
tomasic@cs.cmu.edu

Abstract

Although Natural Language Processing (NLP) for *requests* for information has been well-studied, there has been little prior work on understanding requests to update information. In this paper, we propose an intelligent system that can process natural language website update requests semi-automatically. In particular, this system can analyze requests, posted via email, to update the factual content of individual tuples in a database-backed website. Users' messages are processed using a scheme decomposing their requests into a sequence of entity recognition and text classification tasks. Using a corpus generated by human-subject experiments, we experimentally evaluate the performance of this system, as well as its robustness in handling request types not seen in training, or user-specific language styles not seen in training.

1 Introduction

In this paper, we present a natural language system that helps a webmaster maintain the web site for an organization. Specifically, we describe a system for understanding certain natural-language requests to change the factual content on a website. We will assume that the website is based on a database, and focus on requests to update specific facts in this database.

To motivate this, we note that although NLP for requests to *deliver* information(i.e., question-answering) has been well-studied, there has been little prior work on NLP for requests to *update* information. However, NLP for update requests is an attractive research problem, in part because a user can more easily detect an imperfectly-processed utterance.

As a concrete example of update requests, we consider here requests for web-site updates. Such a system would be practically useful, as many organizations maintain a single large database-backed web site that includes information that can be contributed or corrected by many individuals. Since individual users, each of whom may only contribute a few database changes a year, may be reluctant to learn how to interface with the database to make their occasional updates, in many orginazations users submit update requests via email in natural language to a human webmaster. Frequently, there is a waiting period before the human webmaster can incorporate corrections, leading to long processing times, and a web site that is not up to date.

Add the following contact to the Staff list. Arthur Scott ascott@ardra.com Rm 7992 281 1914
On the events page, delete row "December 23 Assembly for Automotive Engineers Conference Room A21"
On the people page under Tommy Lee delete 281 2000
Please delete Kevin Smith's phone number - thanx, Martha
Change Mike Roberts to Michael Roberts.

Figure 1: Example update requests (edited slightly for space and readability)

In this paper, we describe an intelligent system that can process website update requests semi-automatically. First, natural language processing is used to analyze an incoming request. Based on the analysis, the system then constructs an executable version of the proposed change, which is represented as a pre-filled instance of a form. By examining the form, the end user can efficiently determine whether the analysis step was correctly accomplished, and, if necessary, override the results of the agent's analysis by changing values in the form. Prior experiments with human subjects have shown that this process is an effective means of reducing human effort, even if the initial analysis step is imperfect [2004].

This paper focuses on the natural-language processing part of this system. As is typical of informal text like email, users' messages are often ungrammatical, use capitalization patterns inconsistently, use many abbreviations and include typos (as illustrated in Figure 1). As a consequence, standard shallow-NLP tools such as part-of-speech tagging and noun-phrase chunking, which are preliminary steps for text parsing, are quite unreliable. We therefore suggest here a learning approach, where rather than parse the text into a framework of pre-modeled domain knowledge, we decompose the general task into a sequence of entity extraction and classification sub-tasks. All of these sub-tasks can be learned from incoming messages, improving system performance over time.

We will first describe a scheme for decomposing request-understanding into a sequence of learning tasks. Next, we describe the corpus of requests that is used for performance evaluation. We then describe each of the learning sub-tasks

in detail, along with experimental results. We also present experimental results on the robustness of the system – in particular, how the system will perform on request types not seen in training, or on user-specific language usage not seen in training. Finally, we evaluate the end-to-end system's performance, to determine what fraction of messages can be processed completely without errors. We conclude with a review of related work and our conclusions.

2 Understanding Update Requests

2.1 Analysis procedure

Figure 1 gives some example web site update requests that are addressed by the given analysis procedure. General requests that are not for factual update (e.g., "The animated gif in the logo doesn't flash properly when I view it from my home PC") will simply be flagged and forwarded to the real human webmaster.

The analysis procedure contains the following steps.

Request type classification. An informal preliminary analysis of real webmaster request logs suggested that factual-update requests are in one of the following forms: add a new tuple to the database; delete an existing tuple; delete a value from an existing tuple; or alter (add or replace) a value of an existing tuple. One step of the analysis is thus determining the type of request. This is a *text classification* task: each request will be mapped to one of the categories *addTuple*, *deleteTuple*, *deleteValue*, *alterValue*. If it is not in one of these categories, it will be mapped to *otherRequest*.

Named entity recognition (NER). Another step of the analysis is to identify all *entity names* in a request. Figure 2 shows the result of correctly recognizing person names, email addresses, phone numbers, room numbers, and event titles in some sample requests. The subscript after an entity indicates its type (for instance, "person" or "room number").

Role-based entity classification. We distinguish between four different *roles* for an entity in an update request. (a) An entity is a *keyEntity* if it serves to identify the database tuple which is to be modified. In the figure, key entities are marked with a superscript K. An example is the entity "Freddy Smith" in the sentence "please delete Freddy Smith's phone number". (b) An entity is a *newEntity* (marked with a superscript N) if it is a value to be stored in the database. (c) An entity is an *oldEntity* (superscript O) if it is a value currently in the database which the user expects to be replaced with a *newEntity*. (d) Entities unrelated to the execution of the request are considered to be *noiseEntities*. In the figure, they have no superscript marking. Role-based entity classification is an *entity classification* task, in which entities produced by the earlier NER step are given an additional classification.

Target relation classification. The second column of Figure 2 shows the relation associated with each request. For any fixed database schema, there is a fixed set of possible relations, so this is a *text classification* operation.

Target attribute classification. Given entities, the roles of entities, the target relation, and the request type, the semantics of the many tuple-based commands will be often completely determined. One type of request that may still be underspecified is the *deleteValue* request. As an example consider request 4 in the figure: the previous analysis tells us we should delete some attribute value from the tuple of the "person" relation with the key value of "Tommy Lee", but does not specify the value to be deleted. Hence, to complete the analysis for *deleteValue* requests, it is necessary to determine the attribute that needs to be deleted. This is again a text classification task: given a database schema, only a fixed number of attributes need to be considered as possible targets.

For pedagogical reasons, we have described these steps as if they are taken separately. However, the steps are not independent—i.e., information from each step of analysis may affect other steps. In section 6 we describe and evaluate a particular sequence, where outputs of some steps are propagated as inputs to the next steps.

3 The Experimental Corpus

In order to collect an appropriate corpus, a series of controlled human-subject experiments were performed, in which participants were given a series of tasks in pictorial form and asked that they compose and send an appropriate e-mail messages to a webmaster agent. In response to the user's request, the agent returned a *preview* of the updated page, and also *prefilled form* that contained a structured representation of the user's request. The user could correct errors by editing text in various slots of the form, or by choosing from pull-down menus.

Overall, the human-generated corpus contains a total of only 617 example requests, involving approximately 20 subjects, and about 30 different tasks.

Note that the same pictorial task descriptions were presented to multiple users. This sort of duplication can lead to undesirable behavior for a learning system: if a certain pictorial task ,demonstrating addition of a phone number to a person named Greg Johnson for example, is represented by multiple similar examples in the data, then the system might learn a correlation between the phrase "Greg Johnson" and the task of adding a phone number. To address this problem, we manually replaced duplicate entity names with alternative values throughout the corpus, preserving surface features such as capitalization patterns and misspellings.

The requests in the corpus are largely factual updates concerning a single tuple in the database, so we will focus our attention on such requests. Also, the relations in the underlying database schema of the corpus do not contain two attributes or more of the same type, where "type" is defined by the output of the entity recognizer. For instance, personal details might include a home phone number and an office phone number, but our corpus has no such duplications. Duplications of this sort would require an additional entity classifier.

As mentioned, the text itself is often ungrammatical and noisy. We pre-processed the text, annotating it with a version of Brill's part-of-speech tagger [Brill, 1995] and a hand-coded noun-phrase chunker which was tuned for email (using a different corpus). In learning, however, we rely mainly on alternative features that exploit syntactic properties of the messages. These features prove to be informative for the noisy text in our corpus.

	Request	Request Type	Target Relation	Target Attribute
1	Add the following contact to the Staff list. **[Arthur Scott]**$^{N}_{\text{person}}$ **[ascott@ardra.com]**$^{N}_{\text{email}}$ Rm **[7992]**$^{N}_{\text{room}}$ **[412 281 1914]**$^{N}_{\text{phone}}$	addTuple	people	–
2	On the events page, delete row "**[December 23]**$^{K}_{\text{date}}$ **[Assembly for Automotive Engineers Conference]**$^{K}_{\text{eventTitle}}$ Room **[A21]**$^{K}_{\text{room}}$ "	deleteTuple	events	–
3	On the people page under **[Tommy Lee]**$^{K}_{\text{person}}$ delete **[412 281 2000]**$^{O}_{\text{phone}}$	deleteValue	people	phoneNum
4	Please delete **[Freddy Smith's]**$^{K}_{\text{person}}$'s phone number - thanx, **[Martha]**$_{\text{person}}$	deleteValue	people	phoneNum
5	Change **[Mike Roberts]**$^{K}_{\text{person}}$ to **[Michael Roberts]**$^{N}_{\text{person}}$ on the People page.	alterValue	people	personName
6	Please add **[Greg Johnson]**$^{K}_{\text{person}}$'s phone number- **[412 281 2000]**$^{N}_{\text{phone}}$	alterValue	people	phoneNum

Figure 2: Analyzed update requests.

4 Learning

Below we describe each of the individual learning tasks. Relevant experimental results are given for every component.

4.1 Entity Recognition

Named Entity Recognition (NER), or the identification of the substrings of a request that correspond to entity names, is a well-studied yet non-trivial Natural-Language Processing task. We evaluated NER performance for seven linguistic types: time, date, amount, email addresses, phone numbers, room numbers, and personal names. The data includes some mentions of additional entity types (e.g., job titles and organization names) but not in sufficient quantity for learning.

We experimented with two approaches to entity extraction: a rule-based approach, in which hand-coded rules are used to recognize entities; and learning-based extraction. The rule language we used is based on cascaded finite state machines. The learning algorithm we use here is VPHMM, a method for discriminatively training hidden Markov models using a voted-perceptron algorithm [Collins, 2002].

We found that manually constructed rules are best suited for entities such as e-mail addresses and temporal expressions. These types are based on limited vocabularies and fairly regular patterns, and are therefore relatively easy to model manually. Email addresses are an extreme example of this: a simple regular expression matches most email addresses.

Table 1(a) shows the results of extraction using hand-coded rules for email and temporal expressions. We evaluated the rules on the main corpus, which was used for generating the rules, and also on a 96-message "validation set", containing messages which were collected in a second, later series of human-subject experiments (unfortunately, no time expressions were present in this additional set.) As shown in the table, the entity F1 performance is above 95% for all cases that could be evaluated.

In Table 1(b) we show results for learning on the full set of entity types, applying the VPHMM algorithm. Here NER is reduced to the problem of sequentially classifying each token as either "inside" or "outside" the entity type to be extracted. Performance is evaluated by the F1-measure[1], where entities are only counted as correct if both start and end boundaries are correct (i.e., partially correct entity boundaries are given no partial credit.) The left-hand columns in the table (titled "5CV") show F1-measure performance on unseen examples, as estimated using 5-fold cross validation. The right-hand columns will be discussed later.

	Test Set	
Type	Full Corpus	Validation
Time	95.7	n/a
Date	96.1	97.7
Email	100.0	100.0

(a) Rules

	Base f.	Tuned f.	Tuned features	
Type	$5CV$	$5CV$	$5CV_{USR}$	$5CV_{REQ}$
Time	87.7	91.2	88.2	93.9
Date	88.5	94.4	95.8	88.9
Amount	89.7	93.1	93.1	85.4
Phone	87.3	94.2	92.4	82.3
Room#	81.9	90.4	87.1	83.0
Person	80.9	90.3	83.6	88.3

(b) Learning

Table 1: Entity recognition results: F1 measures

Performance is shown for two sets of features. The *base feature* set corresponds to words and capitalization templates over a window including the word to be classified, and the three adjacent words to each side. The second set of features, labeled *tuned features* in the table, is comprised of the base features plus some additional, entity-type specific features,

[1] F1 is the geometric mean of recall and precision.

which are constructed using the same rule language used to build the hand-coded extractors. For example, in extracting dates we added an indicator as to whether a word is a number in the range 1-31; for personal names, we added an indicator for words that are in certain dictionaries of first and last names.

Overall, the level of performance for extraction – better than 90% for every entity type, using the tuned features – is very encouraging, especially considering the irregularity of the text and the relatively small amount of training data available. We found that users tend to use the terminology and formats of the website, resulting in reduced variability.

4.2 Role-based entity classification

Once an entity span has been identified, we must determine its functional role—i.e., whether it acts as a *keyEntity*, *newEntity*, *oldEntity*, or *noiseEntity* (as outlined in Section 2.1). We approach this problem as a classification task, where the extracted entities are transformed into instances to be further classified by a learner.

The features used for the learner are as follows. (a) The closest preceding "action verb". An action verb is one of a few dozen words generally used to denote an update, such as "add", "delete", etc. (b) The closest preceding preposition. (c) The presence or absence of a possessive marker after the entity. (d) An indication whether the entity is part of a determined NP.

The experimental results for the important classes are shown in Table 2, in the column marked "5CV". We used here an SVM learner with a linear kernel [Joachims, 2001]. We show results for each class separately, and in addition to F1 performance for each category, we also show error rate. The "Default Error" is the error obtained by always guessing the most frequent class.

Entity Role	F1/Error $5CV$	$5CV_{USR}$	$5CV_{REQ}$	Default Error
keyEntity	87.0/11.5	83.5/14.4	84.0/14.3	44.2
newEntity	88.8/ 7.5	85.0/10.6	83.4/10.7	34.4
oldEntity	81.0/ 2.5	81.3 /2.5	76.4/ 3.0	6.7

Table 2: Role-based entity classification results

The results for the role determination are almost-surprisingly good, considering the difficult, linguistic nature of this role assignment task. The set of features suggested here is small and simple, and yet very informative, supporting effective learning of roles even for semi-ungrammatical texts.

4.3 Target relation classification

To determine the target relation, we used the same SVM learner. The input features to the classifier are a "bag-of-words" representation of a request, as well as the entity types included in the request (for example, presence of a "phone number" entity in a request indicates a "people" relation, in our database schema.). Results are shown in Table 3 in the "5CV" column. As shown by these results, the task of relation determination is relatively straight-forward, provided sufficient training data.

Target Relation	F1/Error $5CV$	$5CV_{USR}$	$5CV_{REQ}$	Def. Error
people	99.7 / 0.3	99.3 / 0.8	97.3/ 3.4	38.7
budget	100.0 / 0.0	99.2 / 1.6	78.8 / 3.6	10.0
events	99.6 / 0.2	97.4 / 1.1	97.8 / 1.0	22.7
sponsors	100.0 / 0.0	98.6 / 0.2	98.6 / 0.2	6.0

Table 3: Target relation classification results

4.4 Request type classification

In many cases the type of a request can be determined from the roles of the entities in the request. For instance, an *addTuple* request has no *keyEntities* but may have multiple *newEntities*; conversely a *deleteTuple* request has *keyEntities*, but no *newEntities*; and only an *alterValue* request can have both *keyEntities* and *newEntities*. This means that most request types can be determined algorithmically from the set of entity roles found in a request.

The primary need for a request-type classifier is to distinguish between *deleteValue* and *deleteTuple* requests. These types of requests are often syntactically quite similar. Consider for instance the requests "delete the extension for Dan Smith" and "delete the entry for Dan Smith". The first is a *deleteValue* for a phone number, and the second is a *deleteTuple* request. The action verb ("delete") and the included entities, however, are identical. To distinguish the two request-types, it is necessary to determine the direct object of the verb "delete"—which is difficult, since shallow parsing is inaccurate on this very noisy corpus—or else to construct features that are correlated with the direct object of the verb.

Thus, we used the following as features. (a) The counts of *keyEntities*, *oldEntities*, and *newEntities* in a request. (b) The action verbs appearing in a request. (c) The nouns that appear in an NP immediately following an action verb, or that appear in NPs before an action verb in passive form. (d) Nouns from the previous step that also appear in a dictionary of 12 common attribute names (e.g., "phone", "extension", "room", "office", etc).

The results are shown in Table 4. With these features, one can distinguish between these request types quite accurately.

Request Type	F1/Error $5CV$	$5CV_{USR}$	$5CV_{REQ}$	Def. Error
deleteTuple	93.1 / 2.4	92.6 / 2.6	74.7 / 9.2	18.0
deleteValue	82.9 / 3.1	86.0 / 2.4	57.5 / 6.0	9.1

Table 4: Request type classification results

4.5 Target attribute classification

The classification of requests by target attributes is very similar to request type classification, except that rather than determining *if* a delete request concerns an attribute, one must

determine *which* attribute the request concerns. Given our assumptions, this step need only be performed for *deleteValue* requests that do not specify an *oldEntity* value.

Here in fact we learn a vocabulary for attributes names. A simple bag-of-words feature works quite well for this task, as is shown by the results in Table 5 in the "5CV" column. The vocabulary used in the corpus to describe each attribute is fairly small: e.g., phone is usually described as "phone", "line" or "extension". Perhaps this is because users tend to use the terminology of the website, or because the relevant vocabularies are limited by nature.

Request Type	F1/Error $5CV$	$5CV_{USR}$	$5CV_{REQ}$	Def. Error
personal name	77.3 / 2.8	66.7 / 4.2	17.5 / 7.6	7.0
phone#	92.7 / 1.0	92.9 / 1.0	8.2 / 7.3	9.1
room#	87.0 / 2.9	91.5 / 1.0	56.9 / 9.1	18.0
publication	79.6 / 3.7	81.2 / 2.1	44.8 / 5.2	9.1
photo	93.1 / 2.4	78.6 / 3.9	71.3 / 5.3	18.0
CV	82.9 / 3.1	86.5 / 0.8	- / -	9.1
amount	93.1 / 2.4	92.5 / 1.0	92.7 / 1.0	18.0

Table 5: Attribute classification results

5 Robustness Issues

One practically important question is how robust this automated webmaster is to changes in the distribution of users and/or requests. To investigate such questions, one can use a different sampling strategy in performing cross-validation. For instance, to determine how robust the system is to queries from new users, we grouped all the examples generated by each subject into a single set, and then performed a cross-validation constrained so that no set was split between training and test. In other words, in every test fold, all of the example requests were from subjects that had not contributed to the training set. This split thus estimates performance of a system that is used for a very large pool of users. Cross validation by user results are given in the results tables, in the columns marked as "USR".

In the corpus, users usually have some personal stylistic quirks—for instance, a user might consistently give dates, names etc. in a particular format. Thus one would expect that performance with this sort of split will be worse than performance with the default uniform splits. As can be seen from the results, the F1 for most NER task drops only slightly, and is above 80 for all entity types. Slight drops in performance are also seen on two of the three entity-role tasks, and noticible drops are seen on two of the seven attribute-classification tasks (person name and photo). Overall, performance seems to be affected only slightly in this setting.

Similarly, to determine how robust the system is to future requests that are quite different from requests encountered during training, we grouped together examples for the same request type (including all requests generated from a particular pictorial task), and then again performed a cross-validation constrained so that no set was split between training and test. In this scenario, all of the example requests in every test fold are for tasks that were not encountered in the training set. The results of this split are given in the columns titled as "REQ".

To summarize the results, the loss in performance for NER problems is moderate, but larger than that seen when splitting by users. Entity-role classification drops off only slightly, and performance for target-relation classification also remains excellent for most relations. However, performance for request-type classification does drop off noticibly. This drop is almost certainly due to lack of appropriate training data: there are only a handful of tasks updating the "budget" relation, and also only a relatively small number of tasks requiring request-type classification. Similarly, the task of classification by attribute name is practically infeasible for some attribute types in this settings, due to the small number of attribute names mentions in the corpus. However, provided that the system is given sufficient training data for the relevant relation and attribute, it should perform well on different requests.

6 Overall Evaluation

In this section, we complement the component-level evaluations with an evaluation of the entire end-to-end process. We executed the tasks in the following order: NER is run for each entity type; then, roles of the extracted entities are assigned; finally, relation and request types are assigned. Note that the noisy predicted entities (i.e., entities extracted by a NER model) were used as input to the entity-role classifier, as well as to the relation to the request-type classifiers. Here we used VPHMMs and hand-coded rules for extraction, and a non-sequential multi-class voted perceptron [Freund and Schapire, 1998] for classification.[2]

From the user's perspective, it is interesting to note what percentage of the requests can be successfully processed at a message level, at different levels of automation. In the experiments, 79.2% of the messages got both their relation and request type classified correctly. In these cases the user would have received the correct form, with some entries filled out incorrectly. In more than half of the cases (53.4%), the user would have received the correct form, with all entities correctly extracted, but with some entity roles mislabeled. In 39.5% of the messages, the automatic processing encountered no errors at all.

Note that in the end-to-end scenario errors from the entity recognition phase are propagated to role classification task. Also, in order for a message to be considered fully correct, assignments must be accurate for each one of the multiple entities included in this message. That is, many correct decisions must be made for perfect performance per request. Overall, we find these results to be very promising, considering the limited size of our corpus.

7 Related work

Lockerd *et. al* [2003] propose an automated Webmaster called "Mr. Web" which has a similar email-based interface.

[2]The voted perceptron is another margin-based classifier. For implementation reasons, it was more convenient to use in these experiments than an SVM, although its performance was generally quite not as good.

They manually analyzed 325 update requests to assess their linguistic regularity, but do not describe any algorithm for processing the requests.

Our system addresses a fairly general natural-language processing task: learning to understand database updates. As such it might be compared to other systems that use learning in NLP. Previous NLP systems have generally either performed deep semantic analysis using hand-coded grammars in a restricted domain, or else a shallower analysis in a broader domain. While learning has been an important tool for developing broad-coverage NLP components such as POS taggers, parsers, and named entity recognition systems, there have surprisingly few attempts to use learning to perform a complete semantic analysis. Notable exceptions are the CHILL system [Zelle and Mooney, 1996], which learns to parse database queries into a meaning representation language, and the work by Miller *et. al* [1996] on using a combination of generative models to extract facts from text. Work in learning such "semantic parsers" is surveyed and motivated elsewhere [Mooney, 2004].

There are several important differences between the work described in this paper and prior efforts. One difference is that we consider understanding update requests, rather than understanding queries (like Zelle & Mooney) or declaratively stated facts (like Miller *et al*). One advantage of the update-request task is that a partially correct analysis is still useful, and furthermore, is likely to elicit user feedback which can be used for training. In contrast, it is unclear how useful it is to answer an imperfectly analyzed database query, or what could be learned from such an episode. A second difference is that our learning method uses primarily data which can plausibly collected from user feedback. In contrast, Zelle & Mooney's system learns from sentence/query pairs, and Miller *et. al.* use a variety of sources for training data including POS-tagged text, parsed sentences, and semantically annotated text. On the other hand, we limit ourselves to conceptually simple database updates, while Zelle & Mooney consider complex structured queries. There are also numerous smaller differences stemming from the nature of the task and corpus.

Although the purpose and scope of our research is different, the entity role classification step we consider above is broadly similar to recent work on semantic role analysis [Fillmore *et al.*, 2000; Gildea and Jurafsky, 2002], and earlier work on case-role assignment (*e.g.*, [Miikkulainen and Dyer, 1991]).

8 Conclusions

We have described and experimentally evaluated a scheme for processing email requests for certain website facutal updates using a sequence of entity recognition and classification tasks. We showed that the noisy informal email text can be successfully processed applying a learning approach, using relatively small sets of syntactic features. Experimental results show that also with limited amount of data, the system reaches a promising rate of 40% of messages processed perfectly. We expect this rate to improve as the examples set grows. Further, human-subject experiments have also shown that partially correct results are useful in settings described here [Tomasic *et al.*, 2004]. Thus the work in this paper is a realistic evaluation of components of an efficient, adaptive, automatic webmaster assistant.

Open questions that remain to be resolved by future research include relaxing the restriction that each request concerns the update of a single tuple per email and evaluating more complex entity types for the entity recognition component. Improving on entity recognition will both enable expansion of system coverage, as well as boost its overall performance.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

References

[Brill, 1995] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 1995.

[Collins, 2002] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

[Fillmore *et al.*, 2000] C. J. Fillmore, F. C. Baker, and H. Sato. The framenet database and software tools. In *LREC*, 2000.

[Freund and Schapire, 1998] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. In *Computational Learning Theory*, 1998.

[Gildea and Jurafsky, 2002] D. Gildea and D. Jurafsky. Automated labeling of semantic roles. *Computational Linguistics*, 2002.

[Joachims, 2001] Thorsten Joachims. A statistical learning model of text classification with support vector machines. In *SIGIR*, 2001.

[Lockerd *et al.*, 2003] Andrea Lockerd, Huy Pham, Taly Sharon, and Ted Selker. Mr.web: An automated interactive webmaster. In *CHI*, 2003.

[Miikkulainen and Dyer, 1991] R. Miikkulainen and M. G. Dyer. Natural language processing with modular PDP networks and distributed lexicon. *Cognitive Science*, 15:343–399, 1991.

[Miller *et al.*, 1996] S. Miller, D D. Stallard, R. Bobrow, and R. Schwartz. A fully statistical approach to natural language interfaces. In *ACL*, 1996.

[Mooney, 2004] Ray Mooney. Learning semantic parsers: An important but under-studied problem. In *Working notes of the AAAI spring symposium on language learning*, 2004.

[Tomasic *et al.*, 2004] Anthony Tomasic, William Cohen, Susan Fussell, John Zimmerman, Marina Kobayashi, Einat Minkov, Nathan Halstead, Ravi Mosur, and Jason Hum. Learning to navigate web forms. In *IIWEB*, 2004.

[Zelle and Mooney, 1996] J. M. Zelle and R. J. Mooney. Learning database queries using inductive logic programming. In *AAAI*, 1996.

A Probabilistic Model of Redundancy in Information Extraction

Doug Downey, Oren Etzioni, and Stephen Soderland
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
{ddowney,etzioni,soderlan}@cs.washington.edu

Abstract

Unsupervised Information Extraction (UIE) is the task of extracting knowledge from text without using hand-tagged training examples. A fundamental problem for both UIE and supervised IE is assessing the probability that extracted information is correct. In massive corpora such as the Web, the same extraction is found repeatedly in different documents. How does this redundancy impact the probability of correctness?

This paper introduces a combinatorial "balls-and-urns" model that computes the impact of sample size, redundancy, and corroboration from multiple distinct extraction rules on the probability that an extraction is correct. We describe methods for estimating the model's parameters in practice and demonstrate experimentally that for UIE the model's log likelihoods are 15 times better, on average, than those obtained by Pointwise Mutual Information (PMI) and the noisy-or model used in previous work. For supervised IE, the model's performance is comparable to that of Support Vector Machines, and Logistic Regression.

1 Introduction

Information Extraction (IE) is the task of automatically extracting knowledge from text. Unsupervised IE (UIE) is IE in the absence of hand-tagged training data. Because UIE systems do not require human intervention, they can recursively discover new relations, attributes, and instances in a rapid, scalable manner as in KNOWITALL [Etzioni *et al.*, 2004; 2005].

A fundamental problem for both supervised IE and UIE is assessing the probability that extracted information is correct. As explained in Section 5, previous IE work has used a variety of techniques to address this problem, but has yet to provide an adequate formal model of the impact of redundancy—repeatedly obtaining the same extraction from different documents—on the probability of correctness. Yet in massive corpora such as the Web, redundancy is one of the main sources of confidence in extractions.

An extraction that is obtained from multiple, distinct documents is more likely to be a bona fide extraction than one obtained only once. Because the documents that "support" the extraction are, by and large, independently authored, our confidence in an extraction increases dramatically with the number of supporting documents. But by how much? How do we precisely quantify our confidence in an extraction given the available textual evidence?

This paper introduces a combinatorial model that enables us to determine the probability that an observed extraction is correct. We validate the performance of the model empirically on the task of extracting information from the Web using KNOWITALL.

Our contributions are as follows:

1. A formal model that, unlike previous work, explicitly models the impact of sample size, redundancy, and different extraction rules on the probability that an extraction is correct. We analyze the conditions under which the model is applicable, and provide intuitions about its behavior in practice.
2. Methods for estimating the model's parameters in both the UIE and supervised IE tasks.
3. Experiments that demonstrate the model's improved performance over the techniques used to assess extraction probability in previous work. For UIE, our model is a factor of 15 closer to the correct log likelihood than the noisy-or model used in previous work; the model is 20 times closer than KNOWITALL's Pointwise Mutual Information (PMI) method [Etzioni *et al.*, 2004], which is based on Turney's PMI-IR algorithm [Turney, 2001]. For supervised IE, our model achieves a 19% improvement in average log likelihood over the noisy-or model, but is only marginally better than SVMs and logistic regression.

The remainder of the paper is organized as follows. Section 2 introduces our abstract probabilistic model, and Section 3 describes its implementation in practice. Section 4 reports on experimental results in four domains. Section 5 contrasts our model with previous work; the paper concludes with a discussion of future work.

2 The Urns Model

Our probabilistic model takes the form of a classic "balls-and-urns" model from combinatorics. We first consider the

single urn case, for simplicity, and then generalize to the full multiple *Urns Model* used in our experiments. We refer to the model simply as URNS.

We think of IE abstractly as a generative process that maps text to extractions. Extractions repeat because distinct documents may yield the same extraction. For example, the Web page containing "Scenic towns such as Yakima..." and the Web page containing "Washington towns such as Yakima..." both lead us to believe that Yakima is a correct extraction of the relation `City(x)`.

Each extraction is modeled as a labeled ball in an urn. A *label* represents either an instance of the target relation, or an error. The information extraction process is modeled as repeated draws from the urn, with replacement. Thus, in the above example, two balls are drawn from the urn, each with the label "Yakima". The labels are instances of the relation `City(x)`. Each label may appear on a different number of balls in the urn. Finally, there may be balls in the urn with *error labels* such as "California", representing cases where the IE process generated an extraction that is *not* a member of the target relation.

Formally, the parameters that characterize an urn are:

- C – the set of unique target labels; $|C|$ is the number of unique target labels in the urn.
- E – the set of unique error labels; $|E|$ is the number of unique error labels in the urn.
- $num(b)$ – the function giving the number of balls labeled by b where $b \in C \cup E$. $num(B)$ is the multi-set giving the number of balls for each label $b \in B$.

Of course, IE systems do not have access to these parameters directly. The goal of an IE system is to discern which of the labels it extracts are in fact elements of C, based on repeated draws from the urn. Thus, the central question we are investigating is: *given that a particular label x was extracted k times in a set of n draws from the urn, what is the probability that $x \in C$?*

In deriving this probability formally below, we assume the IE system has access to multi-sets $num(C)$ and $num(E)$ giving the number of times the labels in C and E appear on balls in the urn. In our experiments, we provide methods that estimate these multi-sets in both unsupervised and supervised settings. We can express the probability that an element extracted k of n times is of the target relation as follows:

First, we have that

$$P(x \text{ appears } k \text{ times in } n \text{ draws} | x \in C) = \sum_r \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k} P(num(x) = r | x \in C)$$

where s is the total number of balls in the urn, and the sum is taken over possible repetition rates r.

Then we can express the desired quantity using Bayes Rule:

$$P(x \in C | x \text{ appears } k \text{ times in } n \text{ draws}) = \frac{P(x \text{ appears } k \text{ times in } n \text{ draws} | x \in C) P(x \in C)}{P(x \text{ appears } k \text{ times in } n \text{ draws})} \quad (1)$$

Note that these expressions include prior information about the label x – for example, $P(x \in C)$ is the prior probability that the string x is a target label, and $P(num(x) = r | x \in C)$ represents the probability that a target label x is repeated on r balls in the urn. In general, integrating this prior information could be valuable for IE systems; however, in the analysis and experiments that follow, we make the simplifying assumption of uniform priors, yielding the following simplified form:

Proposition 1

$$P(x \in C | x \text{ appears } k \text{ times in } n \text{ draws}) = \frac{\sum_{r \in num(C)} (\frac{r}{s})^k (1 - \frac{r}{s})^{n-k}}{\sum_{r' \in num(C \cup E)} (\frac{r'}{s})^k (1 - \frac{r'}{s})^{n-k}}$$

2.1 The Uniform Special Case

For illustration, consider the simple case in which all labels from C are repeated on the same number of balls. That is, $num(c_i) = R_C$ for all $c_i \in C$, and assume also that $num(e_i) = R_E$ for all $e_i \in E$. While these assumptions are unrealistic (in fact, we use a Zipf distribution for $num(b)$ in our experiments), they are a reasonable approximation for the majority of labels, which lie on the flat tail of the Zipf curve.

Define p to be the precision of the extraction process; that is, the probability that a given draw comes from the target relation. In the uniform case, we have:

$$p = \frac{|C| R_C}{|E| R_E + |C| R_C}$$

The probability that a *particular* element of C appears in a given draw is then $p_C = \frac{p}{|C|}$, and similarly $p_E = \frac{1-p}{|E|}$.

Using a Poisson model to approximate the binomial from Proposition 1, we have:

$$P(x \in C | x \text{ appears } k \text{ times in } n \text{ draws}) \approx \frac{1}{1 + \frac{|E|}{|C|} (\frac{p_E}{p_C})^k e^{n(p_C - p_E)}} \quad (2)$$

In practice, the extraction process is noisy but informative, so $p_C > p_E$. Notice that when this is true, Equation (2) shows that the odds that $x \in C$ increase exponentially with the number of times k that x is extracted, but also decrease exponentially with the sample size n.

A few numerical examples illustrate the behavior of this equation. The examples assume that the precision p is 0.9. Let $|C| = |E| = 2,000$. This means that $R_C = 9 \times R_E$—target balls are nine times as common in the urn as error balls. Now, for $k = 3$ and $n = 10,000$ we have $P(x \in C) = 93.0\%$. Thus, we see that a small number of repetitions can yield high confidence in an extraction. However, when the sample size increases so that $n = 20,000$, and the other parameters are unchanged, then $P(x \in C)$ drops to 19.6%. On the other hand, if C balls repeat much more frequently than E balls, say $R_C = 90 \times R_E$ (with $|E|$ set to 20,000, so that p remains unchanged), then $P(x \in C)$ rises to 99.9%.

The above examples enable us to illustrate the advantages of URNS over the noisy-or model used in previous work [Lin

et al., 2003; Agichtein and Gravano, 2000]. The noisy-or model assumes that each extraction is an independent assertion, correct a fraction p of the time, that the extracted label is "true." The noisy-or model assigns the following probability to extractions:

$$P_{noisy-or}(x \in C | x \text{ appears } k \text{ times}) = 1 - (1-p)^k$$

Therefore, the noisy-or model will assign the same probability— 99.9%—in *all three* of the above examples. Yet, as explained above, 99.9% is only correct in the case for which $n = 10,000$ and $R_C = 90 \times R_E$. As the other two examples show, for different sample sizes or repetition rates, the noisy-or model can be highly inaccurate. This is not surprising given that the noisy-or model ignores the sample size and the repetition rates. Section 4 quantifies the improvements obtained by URNS in practice.

2.2 Applicability of the Urns Model

Under what conditions does our redundancy model provide accurate probability estimates? First, labels from the target set C must be repeated on more balls in the urn than labels from the E set, as in Figure 1. The shaded region in Figure 1 represents the "confusion region" – some of the labels in this region will be classified incorrectly, even by the ideal classifier with infinite data, because for these labels there simply isn't enough information to decide whether they belong to C or E. Thus, our model is effective when the confusion region is relatively small. Secondly, even for small confusion regions, the sample size n must be large enough to approximate the two distributions shown in Figure 1; otherwise the probabilities output by the model will be inaccurate.

An attractive feature of URNS is that it enables us to estimate its expected recall and precision as a function of sample size. If the distributions in Figure 1 cross at the dotted line shown then, given a sufficiently large sample size n, expected recall will be the fraction of the area under the C curve lying to the right of the dotted line.

For a given sample size n, define τ_n to be the least number of appearances k at which an extraction is more likely to be from the C set than the E set (given the distributions in Figure 1, τ_n can be computed using Proposition 1). Then we have:

$$\mathbf{E}[TruePositives] = |C| - \sum_{r \in num(C)} \sum_{k=0}^{\tau_n - 1} \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k}$$

where we define "true positives" to be the number of extracted labels $c_i \in C$ for which the model assigns probability $P(c_i \in C) > 0.5$.

The expected number of false positives is similarly:

$$\mathbf{E}[FalsePositives] = |E| - \sum_{r \in num(E)} \sum_{k=0}^{\tau_n - 1} \binom{n}{k} \left(\frac{r}{s}\right)^k \left(1 - \frac{r}{s}\right)^{n-k}$$

The expected precision of the system can then be approximated as:

$$\mathbf{E}[Precision] \approx \frac{\mathbf{E}[TruePositives]}{\mathbf{E}[FalsePositives] + \mathbf{E}[TruePositives]}$$

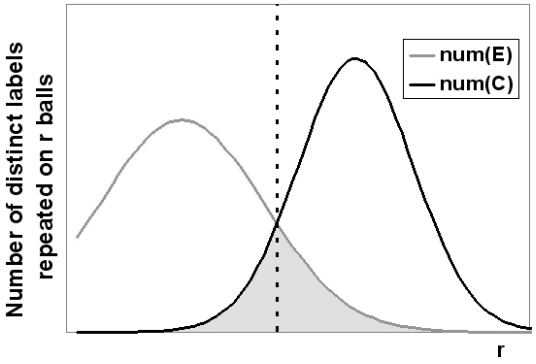


Figure 1: **Schematic illustration of the number of distinct labels in the C and E sets with repetition rate r. The "confusion region" is shaded.**

For example, consider the particular $num(C)$ and $num(E)$ learned (in the unsupervised setting) for the `Film` relation in our experiments. For the sample size $n = 134,912$ used in the experiments, expected number of true positives is 26,133 and expected precision is 70.2%, which is close to the actual observed true positives of 23,408 and precision of 67.7%. Were we to increase the sample size to 1,000,000, we would expect that true positives would increase to 47,609, and precision to 84.0%. Thus, URNS and the above equations enable an IE system to intelligently choose its sample size depending on precision and recall requirements and resource constraints, even in the absence of tagged training data.

2.3 Multiple Urns

We now generalize our model to encompass multiple urns. Information is often extracted using multiple, distinct mechanisms – for example, an IE system might employ several patterns for extracting city names, e.g. "cities including x" and "x and other towns." It is often the case that different patterns have different modes of failure, so extractions appearing across multiple patterns are generally more likely to be true than those appearing for a single pattern. We can model this situation by introducing multiple urns where each urn represents a different extraction mechanism.[1]

Thus, instead of n total extractions, we have a sample size n_m for each urn $m \in M$, with the extraction x appearing k_m times. Let $A(x, (k_1, \ldots, k_m), (n_1, \ldots, n_m))$ denote this event. Further, let $A_m(x, k, n)$ be the event that label x appears k times in n draws from urn m, and assuming that the draws from each urn are independent, we have:

Proposition 2

$$P(x \in C | A(x, (k_1, \ldots, k_m), (n_1, \ldots, n_m))) = \frac{\sum_{c_i \in C} \prod_{m \in M} P(A_m(c_i, k_m, n_m))}{\sum_{x \in C \cup E} \prod_{m \in M} P(A_m(x, k_m, n_m))}$$

With multiple urns, the distributions of labels among balls in the urns are represented by multi-sets $num_m(C)$ and

[1] We may lump several mechanisms into a single urn if they tend to behave similarly.

$num_m(E)$. Expressing the correlation between $num_m(x)$ and $num_{m'}(x)$ is an important modeling decision. Multiple urns are especially beneficial when the repetition rates for elements of C are more strongly correlated across different urns than they are for elements of E—that is, when $num_m(x)$ and $num_{m'}(x)$ tend to be closer to each other for $x \in C$ than for $x \in E$. Fortunately, this turns out to be the case in practice. Section 3 describes our method for modeling multi-urn correlation.

3 Implementation of the Urns Model

This section describes how we implement URNS for both UIE and supervised IE, and identifies the assumptions made in each case.

In order to compute probabilities for extractions, we need a method for estimating $num(C)$ and $num(E)$. For the purpose of estimating these sets from tagged or untagged data, we assume that $num(C)$ and $num(E)$ are Zipf distributed, meaning that if c_i is the ith most frequently repeated label in C, then $num(c_i)$ is proportional to i^{-z_C}. We can then characterize the $num(C)$ and $num(E)$ sets with five parameters: the set sizes $|C|$ and $|E|$, the shape parameters z_C and z_E, and the extraction precision p.

To model multiple urns, we consider different extraction precisions p_m for each urn, but make the simplifying assumption that the size and shape parameters are the same for all urns. As mentioned in Section 2, we expect repetition rate correlation across urns to be higher for elements of the C set than for the E set. We model this correlation as follows: first, elements of the C set are assumed to come from the same location on the Zipf curve for all urns, that is, their relative frequencies are perfectly correlated. Some elements of the E set are similar, and have the same relative frequency across urns – these are the *systematic* errors. However, the rest of the E set is made up of *non-systematic* errors, meaning that they appear for only one kind of extraction mechanism (for example, "Eastman Kodak" is extracted as an instance of `Film` only in phrases involving the word "film", and not in those involving the word "movie."). Formally, non-systematic errors are labels that are present in some urns and not in others. Each type of non-systematic error makes up some fraction of the E set, and these fractions are the parameters of our correlation model. Assuming this simple correlation model and identical size and shape parameters across urns is too restrictive in general— differences between extraction mechanisms are often more complex. However, our assumptions allow us to compute probabilities efficiently (as described below) and do not appear to hurt performance significantly in practice.

With this correlation model, if a label x is an element of C or a systematic error, it will be present in all urns. In terms of Proposition 2, the probability that a label x appears k_m times in n_m draws from m is:

$$P(A_m(x, k_m, n_m)) = \binom{n_m}{k_m} (f_m(x))^{k_m} (1 - f_m(x))^{n_m - k_m} \quad (3)$$

where $f_m(x)$ is the frequency of extraction x. That is,

$$\begin{aligned} f_m(c_i) &= p_m Q_C i^{-z_C} \text{ for } c_i \in C \\ f_m(e_i) &= (1 - p_m) Q_E i^{-z_E} \text{ for } e_i \in E \end{aligned}$$

In these expressions, i is the frequency rank of the extraction, assumed to be the same across all urns, and Q_C and Q_E are normalizing constants such that

$$\sum_{c_i \in C} Q_C i^{-z_C} = \sum_{e_i \in E} Q_E i^{-z_E} = 1$$

For a non-systematic error x which is not present in urn m, $P(A_m(x, k_m, n_m))$ is 1 if $k_m = 0$ and 0 otherwise. Substituting these expressions for $P(A_m(x, k_m, n_m))$ into Proposition 2 gives the final form of our URNS model.

3.1 Efficient Computation

A feature of our implementation is that it allows for efficient computation of probabilities. In general, computing the sum in Proposition 2 over the potentially large C and E sets would require significant computation for each extraction. However, given a fixed number of urns, with $num(C)$ and $num(E)$ Zipf distributed, an integral approximation to the sum in Proposition 2 (using a Poisson in place of the binomial in Equation 3) can be solved in closed form in terms of incomplete Gamma functions. This closed form expression can be evaluated quickly, and thus probabilities for extractions can be obtained efficiently. This solution leverages our assumptions that size and shape parameters are identical across urns, and that relative frequencies are perfectly correlated. Finding efficient techniques for computing probabilities under less stringent assumptions is an item of future work.

3.2 Parameter Estimation

In the event that a large sample of hand-tagged training examples is available for each target relation of interest, we can directly estimate each of the parameters of URNS. We use a population-based stochastic optimization technique to identify parameter settings that maximize the conditional log likelihood of the training data.[2] Once the parameters are set, the model yields a probability for each extraction, given the number of times k_m it appears in each urn and the number of draws n_m from each urn.

As argued in [Etzioni *et al.*, 2005], IE systems cannot rely on hand-tagged training examples if they are to scale to extracting information on arbitrary relations that are not specified in advance. Implementing URNS for UIE requires a solution to the challenging problem of estimating $num(C)$ and $num(E)$ using untagged data. Let U be the multi-set consisting of the number of times each unique label was extracted; $|U|$ is the number of unique labels encountered, and the sample size $n = \sum_{u \in U} u$.

In order to learn $num(C)$ and $num(E)$ from untagged data, we make the following assumptions:

- Because the number of different possible errors is nearly unbounded, we assume that the error set is very large.[3]

[2] Specifically, we use the Differential Evolution routine built into Mathematica 5.0.

[3] In our experiments, we set $|E| = 10^6$. A sensitivity analysis showed that changing $|E|$ by an order of magnitude, in either direction, resulted in only small changes to our results.

- We assume that both $num(C)$ and $num(E)$ are Zipf distributed where the z_E parameter is set to 1.
- In our experience with KNOWITALL, we found that while different extraction rules have differing precision, each rule's precision is stable across different relations [Etzioni *et al.*, 2005]. URNS takes this precision as an input. To demonstrate that URNS is not overly sensitive to this parameter, we chose a fixed value (0.9) and used it as the precision p_m for all urns in our experiments. [4]

We then use Expectation Maximization (EM) over U in order to arrive at appropriate values for $|C|$ and z_C (these two quantities uniquely determine $num(C)$ given our assumptions). Our EM algorithm proceeds as follows:

1. Initialize $|C|$ and z_C to starting values.
2. Repeat until convergence:
   (a) **E-step** Assign probabilities to each element of U using Proposition (1).
   (b) **M-step** Set $|C|$ and z_C from U using the probabilities assigned in the E-step (details below).

We obtain $|C|$ and z_C in the M-step by first estimating the rank-frequency distribution for labels from C in the untagged data. From the untagged data and the probabilities found in the E-step, we can obtain $E_C[k]$, the expected number of labels from C that were extracted k times. We then round these fractional expected counts into a discrete rank-frequency distribution with a number of elements equal to the expected total number of labels from C in the untagged data, $\sum_k E_C[k]$. We obtain z_C by fitting a Zipf curve to this rank-frequency distribution by linear regression on a log-log scale. Lastly, we set $|C| = \sum_k E_C[k] + unseen$, where we estimate the number of unseen labels of the C set using Good-Turing estimation ([Gale and Sampson, 1995]). Specifically, we choose $unseen$ such that the probability mass of unseen labels is equal to the expected fraction of the draws from C that extracted labels seen only once.

This unsupervised learning strategy proved effective for target relations of different sizes; for example, the number of elements of the `Country` relation with non-negligible extraction probability was about two orders of magnitude smaller than that of the `Film` and `City` relations.

Clearly, unsupervised learning relies on several strong assumptions, though our sensitivity analysis has shown that the model's performance is robust to some of them. In future work, we plan to perform a more comprehensive sensitivity analysis of the model and also investigate its performance in a semi-supervised setting.

4 Experimental Results

This section describes our experimental results under two settings: unsupervised and supervised. We begin by describing the two unsupervised methods used in previous work: the noisy-or model and PMI. We then compare URNS with these methods experimentally, and lastly compare URNS with several baseline methods in a supervised setting.

We evaluated our algorithms on extraction sets for the relations `City(x)`, `Film(x)`, `Country(x)`, and `MayorOf(x,y)`, taken from experiments performed in [Etzioni *et al.*, 2005]. The sample size n was 64,581 for `City`, 134,912 for `Film`, 51,313 for `Country` and 46,129 for `MayorOf`. The extraction patterns were partitioned into urns based on the name they employed for their target relation (e.g. "country" or "nation") and whether they were left-handed (e.g. "countries including x") or right-handed (e.g. "x and other countries"). Each combination of relation name and handedness was treated as a separate urn, resulting in four urns for each of `City(x)`, `Film(x)`, and `Country(x)`, and two urns for `MayorOf(x)`.[5] For each relation, we tagged a sample of 1000 extracted labels, using external knowledge bases (the Tipster Gazetteer for cities and the Internet Movie Database for films) and manually tagging those instances not found in a knowledge base. In the UIE experiments, we evaluate our algorithms on all 1000 examples, and in the supervised IE experiments we perform 10-fold cross validation.

4.1 UIE Experiments

We compare URNS against two other methods for unsupervised information extraction. First, in the *noisy-or* model used in previous work, an extraction appearing k times is assigned probability $1 - \prod_{m \in M}(1 - p_m)^k$, where p_m is the extraction precision for urn m. We describe the second method below.

Pointwise Mutual Information

Our previous work on KNOWITALL used Pointwise Mutual Information (PMI) to obtain probability estimates for extractions [Etzioni *et al.*, 2005]. Specifically, the PMI between an extraction and a set of automatically generated *discriminator phrases* (e.g., "movies such as x") is computed from Web search engine hit counts. These PMI scores are used as features in a Naive Bayes Classifier (NBC) to produce a probability estimate for the extraction. The NBC is trained using a set of automatically bootstrapped seed instances. The positive seed instances are taken to be those having the highest PMI with the discriminator phrases after the bootstrapping process; the negative seeds are taken from the positive seeds of other relations, as in other work (e.g., [Lin *et al.*, 2003]).

Although PMI was shown in [Etzioni *et al.*, 2005] to order extractions fairly well, it has two significant shortcomings. First, obtaining the hit counts needed to compute the PMI scores is expensive, as it requires a large number of queries to

[4] A sensitivity analysis showed that choosing a substantially higher (0.95) or lower (0.80) value for p_m still resulted in URNS outperforming the noisy-or model by at least a factor of 8 and PMI by at least a factor of 10 in the experiments described in Section 4.1.

[5] Draws from URNS are intended to represent independent extractions. Because the same sentence can be duplicated across multiple different Web documents, in these experiments we consider only each *unique* sentence containing an extraction to be a draw from URNS. In experiments with other possibilities, including counting the number of unique documents producing each extraction, or simply counting every occurrence of each extraction, we found that performance differences between the various approaches were negligible for our task.

web search engines. Second, the seeds produced by the bootstrapping process tend not to be representative of the overall distribution of extractions. This combined with the probability polarization introduced by the NBC tends to give inaccurate probability estimates.

Discussion of UIE Results

The results of our unsupervised experiments are shown in Figure 2. We plot deviation from the *ideal* log likelihood—defined as the maximum achievable log likelihood given our feature set.

Our experimental results demonstrate that URNS overcomes the weaknesses of PMI. First, URNS's probabilities are far more accurate than PMI's, achieving a log likelihood that is a factor of 20 closer to the ideal, on average (Figure 2). Second, URNS is substantially more efficient as shown in Table 1.

This efficiency gain requires some explanation. KNOWITALL relies on queries to Web search engines to identify Web pages containing potential extractions. The number of queries KNOWITALL can issue daily is limited, and querying over the Web is, by far, KNOWITALL's most expensive operation. Thus, number of search engine queries is our efficiency metric. Let d be the number of discriminator phrases used by the PMI method as explained in Section 4.1. The PMI method requires $O(d)$ search engine queries to compute the PMI of each extraction from search engine hit counts. In contrast, URNS computes probabilities *directly* from the set of extractions—requiring *no* additional queries, which cuts KNOWITALL's queries by factors ranging from 1.9 to 17.

As explained in Section 2, the noisy-or model ignores target set size and sample size, which leads it to assign probabilities that are far too high for the `Country` and `MayorOf` relations, where the average number of times each label is extracted is high (see bottom row of Table 1). This is further illustrated for the `Country` relation in Figure 3. The noisy-or model assigns appropriate probabilities for low sample sizes, because in this case the overall precision of extracted labels is in fact fairly high, as predicted by the noisy-or model. However, as sample size increases relative to the number of true countries, the overall precision of the extracted labels decreases—and the noisy-or estimate worsens. On the other hand, URNS avoids this problem by accounting for the interaction between target set size and sample size, adjusting its probability estimates as sample size increases. Given sufficient sample size, URNS performs close to the ideal log likelihood, improving slightly with more samples as the estimates obtained by the EM process become more accurate. Overall, URNS assigns far more accurate probabilities than the noisy-or model, and its log likelihood is a factor of 15 closer to the ideal, on average. The very large differences between URNS and both the noisy-or model and PMI suggest that, even if the performance of URNS degrades in other domains, it is quite likely to still outperform both PMI and the noisy-or model.

Our computation of log-likelihood contains a numerical detail that could potentially influence our results. To avoid the possibility of a likelihood of zero, we restrict the probabilities generated by URNS and the other methods to lie within the

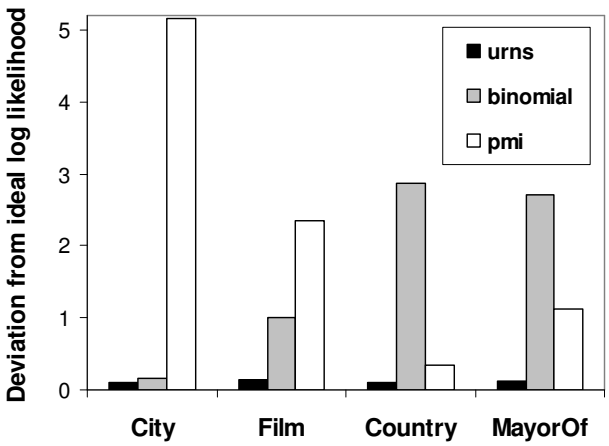


Figure 2: **Deviation of average log likelihood from the ideal for four relations (lower is better). On average,** URNS **outperforms noisy-or by a factor of 15, and PMI by a factor of 20.**

	City	*Film*	*MayorOf*	*Country*
Speedup	17.3x	9.5x	1.9x	3.1x
Average k	3.7	4.0	20.7	23.3

Table 1: **Improved Efficiency Due to** URNS**. The top row reports the number of search engine queries made by** KNOWITALL **using PMI divided by the number of queries for** KNOWITALL **using** URNS**. The bottom row shows that PMI's queries increase with k—the average number of distinct labels for each relation. Thus, speedup tends to vary inversely with the average number of times each label is drawn.**

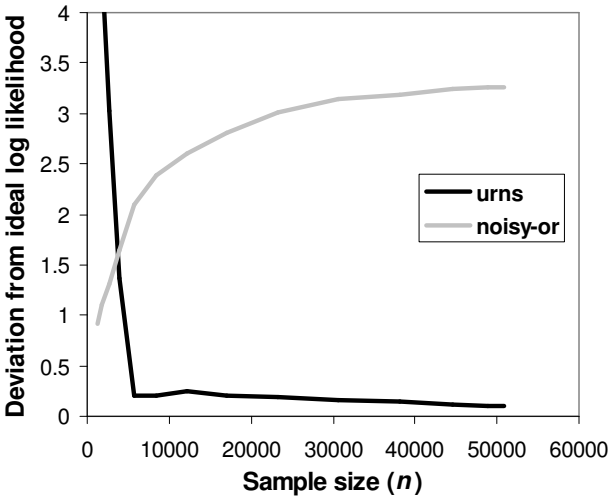


Figure 3: **Deviation of average log likelihood from the ideal as sample size varies for the `Country` relation (lower is better).** URNS **performs close to the ideal given sufficient sample size, whereas noisy-or becomes less accurate as sample size increases.**

range (0.00001, 0.99999). Widening this range tended to improve URNS's performance relative to the other methods, as this increases the penalty for erroneously assigning extreme probabilities—a problem more prevalent for PMI and noisy-or than for URNS. Even if we narrow the range by two digits of precision, to (0.001, 0.999), URNS still outperforms PMI by a factor of 15, and noisy-or by a factor of 13. Thus, we are comfortable that the differences observed are not an artifact of this design decision.

4.2 Supervised IE Experiments

We compare URNS with three supervised methods. All methods utilize the same feature set as URNS, namely the extraction counts k_m.

- **noisy-or** – Has one parameter per urn, making a set of M parameters $(h_1, \ldots, h_M)$, and assigns probability equal to
$$1 - \prod_{m \in M} (1 - h_m)^{k_m}.$$
- **logistic regression** – Has $M + 1$ parameters $(a, b_1, b_2, \ldots, b_M)$, and assigns probability equal to
$$\frac{1}{1 + e^{a + \sum_{m \in M} k_m b_m}}.$$
- **SVM** – Consists of an SVM classifier with a Gaussian kernel. To transform the output of the classifier into a probability, we use the probability estimation built-in to LIBSVM [Chang and Lin, 2001], which is based on logistic regression of the SVM decision values.

Parameters maximizing the conditional likelihood of the training data were found for the noisy-or and logistic regression models using Differential Evolution. In the SVM case, we performed grid search to find the kernel parameters giving the best likelihood performance for each training set – this grid search was required to get acceptable performance from the SVM on our task.

The results of our supervised learning experiments are shown in Table 2. URNS, because it is more expressive, is able to outperform the noisy-or and logistic regression models. In terms of deviation from the ideal log likelihood, we find that on average URNS outperforms the noisy-or model by 19%, logistic regression by 10%, but SVM by only 0.4%.

	City	*Film*	*Mayor*	*Country*	*Average*
noisy-or	0.0439	0.1256	0.0857	0.0795	0.0837
logistic regression	0.0466	0.0893	**0.0655**	0.1020	0.0759
SVM	0.0444	0.0865	0.0659	**0.0769**	0.0684
URNS	**0.0418**	**0.0764**	0.0721	0.0823	**0.0681**

Table 2: **Supervised IE experiments. Deviation from the ideal log likelihood for each method and each relation (lower is better). The overall performance differences are small, with URNS 19% closer to the ideal than noisy-or, on average, and 10% closer than logistic regression. The overall performance of SVM is close to that of URNS.**

5 Related Work

In contrast to the bulk of previous IE work, our focus is on unsupervised IE (UIE) where URNS substantially outperforms previous methods (Figure 2).

In addition to the noisy-or models we compare against in our experiments, the IE literature contains a variety of heuristics using repetition as an indication of the veracity of extracted information. For example, Riloff and Jones [Riloff and Jones, 1999] rank extractions by the number of distinct patterns generating them, plus a factor for the reliability of the patterns. Our work is intended to formalize these heuristic techniques, and unlike the noisy-or models, we explicitly model the distribution of the target and error sets (our $num(C)$ and $num(E)$), which is shown to be important for good performance in Section 4.1. The accuracy of the probability estimates produced by the heuristic and noisy-or methods is rarely evaluated explicitly in the IE literature, although most systems make implicit use of such estimates. For example, bootstrap-learning systems start with a set of seed instances of a given relation, which are used to identify extraction patterns for the relation; these patterns are in turn used to extract further instances (e.g. [Riloff and Jones, 1999; Lin *et al.*, 2003; Agichtein and Gravano, 2000]). As this process iterates, random extraction errors result in overly general extraction patterns, leading the system to extract further erroneous instances. The more accurate estimates of extraction probabilities produced by URNS would help prevent this "concept drift."

Skounakis and Craven [Skounakis and Craven, 2003] develop a probabilistic model for combining evidence from multiple extractions in a supervised setting. Their problem formulation differs from ours, as they classify each occurrence of an extraction, and then use a binomial model along with the false positive and true positive rates of the classifier to obtain the probability that at least one occurrence is a true positive. Similar to the above approaches, they do not explicitly account for sample size n, nor do they model the distribution of target and error extractions.

Culotta and McCallum [Culotta and McCallum, 2004] provide a model for assessing the confidence of extracted information using conditional random fields (CRFs). Their work focuses on assigning accurate confidence values to individual occurrences of an extracted field based on textual features. This is complementary to our focus on *combining* confidence estimates from multiple occurrences of the same extraction. In fact, each possible feature vector processed by the CRF in [Culotta and McCallum, 2004] can be thought of as a virtual urn m in our URNS. The confidence output of Culotta and McCallum's model could then be used to provide the precision p_m for the urn.

Our work is similar in spirit to BLOG, a language for specifying probability distributions over sets with unknown objects [Milch *et al.*, 2004]. As in our work, BLOG models treat observations as draws from a set of balls in an urn. Whereas BLOG is intended to be a general modeling framework for probabilistic first-order logic, our work is directed at modeling redundancy in IE. In contrast to [Milch *et al.*, 2004], we provide supervised and unsupervised learning methods

for our model and experiments demonstrating their efficacy in practice.

6 Conclusions and Future Work

This paper introduced a combinatorial URNS model to the problem of assessing the probability that an extraction is correct. The paper described supervised and unsupervised methods for estimating the parameters of the model from data, and reported on experiments showing that URNS massively outperforms previous methods in the unsupervised case, and is slightly better than baseline methods in the supervised case. Of course, additional experiments and a more comprehensive sensitivity analysis of URNS are necessary.

URNS is applicable to tasks other than IE. For example, PMI computed over search engine hit counts has been used to determine synonymy [Turney, 2001], and for question answering [Magnini *et al.*, 2002]. In the synonymy case, for example, the PMI between two terms is used as a measure of their synonymy; applying URNS to the same co-occurrence statistics should result in a more accurate probabilistic assessment of whether two terms are synonyms. Comparing URNS with PMI on these tasks is a topic for future work.

Acknowledgments

This research was supported in part by NSF grant IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-02-1-0324, and a gift from Google. Google generously allowed us to issue a large number of queries to their XML API to facilitate our experiments. We thank Anna Karlin, Marina Meila, and Dan Weld for helpful discussions, and Jeff Bigham for comments on previous drafts. Also, thanks to Alex Yates for suggesting we consider this problem.

References

[Agichtein and Gravano, 2000] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proc. of the 5th ACM Intl. Conf. on Digital Libraries*, 2000.

[Chang and Lin, 2001] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001.

[Culotta and McCallum, 2004] A. Culotta and A. McCallum. Confidence estimation for information extraction. In *HLT-NAACL*, 2004.

[Etzioni *et al.*, 2004] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in system x: (preliminary results). In *WWW*, 2004.

[Etzioni *et al.*, 2005] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. In *To appear in AIJ*, 2005.

[Gale and Sampson, 1995] W. A. Gale and G. Sampson. Good-turing frequency estimation without tears. *Journal of Quantitative Linguistics*, 2(3):217–237, 1995.

[Lin *et al.*, 2003] W. Lin, R. Yangarber, and R. Grishman. Bootstrapped learning of semantic classes. In *ICML Workshop on The Continuum from Labeled to Unlabeled Data*, 2003.

[Magnini *et al.*, 2002] B. Magnini, M. Negri, R. Prevete, and H. Tanev. Is it the right answer? exploiting web redundancy for answer validation. In *ACL*, 2002.

[Milch *et al.*, 2004] B. Milch, B. Marthi, and S. Russell. BLOG: Relational modeling with unknown objects. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*, 2004.

[Riloff and Jones, 1999] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, 1999.

[Skounakis and Craven, 2003] M. Skounakis and M. Craven. Evidence combination in biomedical natural-language processing. In *BIOKDD*, 2003.

[Turney, 2001] P. D. Turney. Mining the Web for synonyms: PMI–IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, 2167:491–502, 2001.

A Probabilistic Framework for Recognizing Intention in Information Graphics*

∗**Stephanie Elzer,** ∗∗**Sandra Carberry,** ∗ ∗ ∗**Ingrid Zukerman,** ∗∗**Daniel Chester,**
∗ ∗ ∗∗**Nancy Green,** ∗∗**Seniz Demir**
∗Dept. of Computer Science, Millersville University, Millersville, PA 17551
∗∗Dept. of Computer Science, University of Delaware, Newark, DE 19716
∗ ∗ ∗School of Comp. Science and Software Engrg, Monash University, Clayton, Victoria 3800 Australia
∗ ∗ ∗∗Dept. of Mathematical Sciences, Univ. of North Carolina at Greensboro, Greensboro, NC 27402

Abstract

This paper extends language understanding and plan inference to information graphics. We identify the kinds of communicative signals that appear in information graphics, describe how we utilize them in a Bayesian network that hypothesizes the graphic's intended message, and discuss the performance of our implemented system. This work is part of a larger project aimed at making information graphics accessible to individuals with sight impairments.

1 Introduction

For individuals who have serious sight impairments, documents that contain information graphics (bar charts, line graphs, etc.) pose challenging problems. Although devices have been developed for conveying information graphics in alternative mediums such as musical tones or tactile images, these approaches have serious limitations. For example, they often require expensive equipment, as well as requiring the user to construct a "mental map" of the graphic, which is difficult for congenitally blind users who do not have the personal knowledge to assist them in the interpretation of the image [Kennel, 1996]. The underlying hypothesis of our work is that alternative access to what the graphic looks like is not enough — the user should be provided with the message and knowledge that one would gain from viewing the graphic in order to enable effective and efficient use of this information resource.

We are developing an interactive natural language system which infers the intended message of an information graphic, provides a summary that includes the intended message along with notable features of the graphic, and then responds to follow-up questions from the user. Figure 1 shows the architecture of our system. The visual extraction module (VEM) analyzes the graphic and provides an XML representation of the graphic to the caption tagging module (CTM) which extracts information from the caption and passes the augmented XML representation to the intention recognition module (IRM). The IRM is responsible for recognizing the intended message of the information graphic and sending it to the content planning module (CPM), which will augment the intended message with related interesting features of the graphic. The message organization module (MOM) will then organize the most salient propositions into a coherent summary, which will be rendered in natural language and conveyed to the user via speech synthesis. The followup question module (FQM) will allow the user to interactively seek additional information about the graphic.

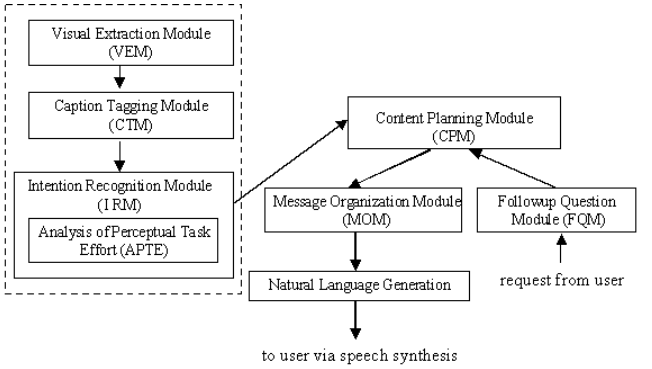


Figure 1: System Architecture

This paper identifies the communicative signals that appear in information graphics and discusses how we exploit them in a Bayesian network that hypothesizes a graphic's intended message. It concludes with several illustrative examples and an evaluation of our system's performance, which provides strong evidence of the effectiveness of our methodology. To our knowledge, our work is the first to extend plan recognition to the domain of information graphics.

2 Approach

Language research has posited that a speaker or writer executes a speech act whose intended meaning he expects the listener to be able to deduce, and that the listener identifies the intended meaning by reasoning about the observed signals and the mutual beliefs of author and interpreter [Grice, 1969]. However, as [Clark, 1996] noted, language is more than just words. It is any "signal" (or lack of signal when one is expected), where a signal is a deliberate action that is intended

*Authors can be reached via email as follows: elzer@cs.millersville.edu, {carberry, chester, demir}@cis.udel.edu, Ingrid.Zukerman@infotech.monash.edu.au, nlgreen@uncg.edu.

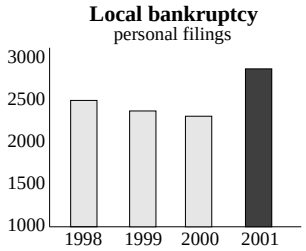


Figure 2: Graphic from a 2001 Local Newspaper

to convey a message. Although some information graphics are only intended to display data values,[1] the overwhelming majority of information graphics from popular media sources appear to have some underlying goal or intended message, such as the graphic in Figure 2 whose communicative goal is ostensibly to convey the sharp increase in local bankruptcies in 2001 compared with the previously decreasing trend. Applying Clark's view of language to information graphics, it is reasonable to presume that the author of an information graphic expects the viewer to deduce the message that he intended to convey by reasoning about the graphic itself, the salience of entities in the graphic, and mutual beliefs.

Beginning with the seminal work of [Perrault and Allen, 1980], researchers have applied plan inference techniques to a variety of problems associated with understanding utterances, particularly within dialogue. Given domain knowledge in the form of operators that decompose goals into a sequence of subgoals, along with evidence in the form of an observed action (such as an utterance), a plan inference system can chain backwards on the plan operators to deduce one or more high-level goals that might have led the agent to perform the observed action as part of an overall plan for achieving his goal(s). The high-level communicative goals in the plan capture the utterance's intended meaning.

In their work on intelligent multimedia generation, the AutoBrief group proposed that speech act theory can be extended to the generation of graphical presentations [Kerpedjiev and Roth, 2000]. When designing an information graphic, the designer has one or more high-level communicative goals. He constructs a graphic that he believes will enable the viewer to perform certain perceptual and cognitive tasks which, along with other knowledge, will enable the viewer to recognize the graphic's intended message [Kerpedjiev and Roth, 2000]. By *perceptual tasks* we mean tasks that can be performed by simply viewing the graphic, such as finding the top of a bar in a bar chart; by *cognitive tasks* we mean tasks that are done via mental computations, such as computing the difference between two numbers.

In our research, we extend plan inference techniques to inferring intention from information graphics. Our plan operators capture knowledge about how the graphic designer's goal of conveying a message can be achieved via the viewer performing certain perceptual and cognitive tasks, as well as knowledge about how perceptual and cognitive tasks decompose into sets of simpler tasks. Figure 3 displays a plan operator consisting of:

- **Goal:** the goal that the operator achieves
- **Data Requirements:** requirements which the data must satisfy in order for the operator to be applicable in a graphic planning paradigm
- **Display Constraints:** features that constrain how the information graphic is eventually constructed if this operator is part of the final plan
- **Body:** lower-level subgoals that must be accomplished in order to achieve the overall goal of the operator

For plan recognition, the plan operators facilitate chaining backwards from perceptual tasks that can be performed on a given graphic, to hypothesize candidate goals and plans. The display constraints are used in reverse, that is, to eliminate operators from consideration (i.e., if the graphic does not satisfy the constraints on the display, then the operator could not have been part of a plan that produced the graphic). The data requirements are used to instantiate parameters in the operator (i.e., the data must have had certain characteristics for the operator to have been included in the designer's plan, and these limit how the operator's arguments can be instantiated).

Following the work of [Charniak and Goldman, 1993] and others, we capture plan inference in a probabilistic framework. We use our plan operators, along with the evidence that we glean from the information graphic itself, to construct a Bayesian network which allows us to reason about the likelihood of various candidate plans.

3 Network Structure

Although we believe that our findings are extendible to other kinds of information graphics, our current work focuses on bar charts. Our analysis of simple bar charts[2] has shown that the intended message can be classified into one of 12 high-level categories; examples of such categories are:

- GetRank: *Viewer to believe that <entity-1> is ranked <rank> among the entities in the graphic*
- Change-trend: *Viewer to believe that there is a <slope-1> trend from <param1> to <param2> and a significantly different <slope-2> trend from <param3> to <param4>*
- Relative-difference: *Viewer to believe that the value of element <param1> is <comparison> the value of element <param2>, where <comparison> is greater-than, less-than, or equal-to.*

The top-level node in our network captures the likelihood of all of the possible categories of high-level intentions underlying a graphic. The entry with the highest probability in this node represents the category most likely to represent the graphic designer's primary intention or communicative goal for the graph. Each individual category of high-level goal is represented as a child of this top-level node. We refer to the process of replacing one or more of the parameters of a goal or perceptual task with specific elements or entities of the graphic as *instantiating* that goal or task. The network nodes

[1] [Yu *et al.*, 2002] developed a pattern recognition algorithm for summarizing interesting features of automatically generated graphics of time-series data from a gas turbine engine.

[2] We are not yet considering composite or grouped bar charts.

Goal: RankFromBar($<viewer>$, $<ds>$, $<g>$, $<b_i>$, $<v_i>$, $<rank>$)
Gloss: Viewer to believe from graphic $<g>$ that the element of dataset $<ds>$ depicted as bar $<b_i>$ with the value $<v_i>$ for primary key $<att_1>$ has a value for $<att_2>$ that is $<rank>$ among the values of $<att_2>$ associated with elements displayed on graphic $<g>$
Data Requirements:
1. $<att_1>$ is the primary key attribute for dataset $<ds>$
2. $<att_2>$ is a dependent attribute for dataset $<ds>$
3. The values of $<att_2>$ have a natural ordering along a quantitative scale
4. The value of $<att_2>$ for $<v_i>$ has rank $<rank>$ among the elements of $<ds>$

Display Constraints:
1. Graph $<g>$ is a bar chart
2. For each value of $<att1>$ in dataset $<ds>$, the value of $<att1>$ and associated value of $<att2>$ is displayed via a bar on graph $<g>$

Body:
1. PerceiveRank: Viewer can perform the perceptual task of finding the $<rank>$ relative to $<att_2>$ of bar $<b_i>$ in graph <g>
2. GetLabel: Viewer can perform the perceptual task of finding the value $<v_i>$ for $<att_1>$ where $<v_i>$ corresponds with the bar $<b_i>$ on graph $<g>$

Figure 3: Operator for finding the rank of a particular bar

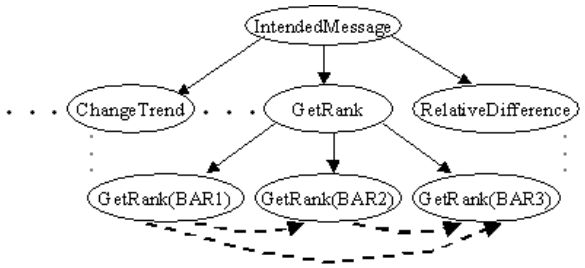


Figure 4: Top levels of Belief Network

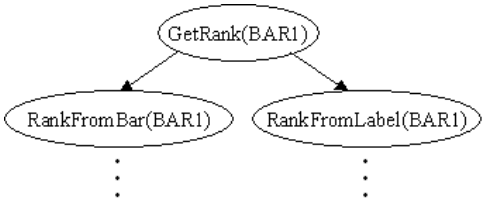


Figure 5: Alternative Ways of Achieving the Same Goal

in the top two levels are not instantiated. Specific instantiations appear in the network as children of the nodes representing the high-level intentions. For example, the GetRank node is shown with several children in Figure 4, illustrating several of its possible instantiations — namely, finding the rank of the first, second, or third bar in the graphic. (For readability, only the instantiation of the bar parameter $< b_i >$ in each GetRank node is shown in Figure 4.) These are alternative instantiations and so inhibitory links [Huber *et al.*, 1994] are used to capture their mutual exclusivity. The child node of GetRank with the highest probability represents the instantiation of that goal most likely to be part of the graphic designer's intended plan. Therefore, if the entry for GetRank has the highest probability in the IntendedMessage root node, our system selects the instantiated child node of GetRank with the highest probability, and produces the instantiated version of GetRank as its hypothesis about the intended message of the graphic.

If there are multiple ways for a goal to be achieved, these are captured as children of the instantiated goal node. For example, a viewer might come to believe that the United States ranks third in GDP per capita in one of two ways: 1) he might seek out the bar for the U.S. and perceive that it is the third highest bar in the graph, or 2) he might notice the bar representing the U.S. in the graphic (perhaps it is a different color), and perceive that that bar is the third highest in the graph and that it represents the GDP per capita for the U.S. These possibilities are captured respectively as RankFromLabel and RankFromBar in Figure 5.

The nodes in the top levels of the network, representing the high-level intentions and the alternative ways of achieving them (Figure 4), exist in every network and are added at the beginning of the network construction process. Nodes representing perceptual tasks (the lowest-level nodes in the network) are then added, and upward chaining via the plan operators adds higher level goals until a link is established to one of the top-level goal nodes shown in Figure 4. However, automatically adding to the network all of the possible instantiations of all possible perceptual tasks rapidly becomes infeasible due to the overwhelming size of the resultant network and practical constraints on memory. Thus, instead of constructing the network a priori to include every possible instantiation of each task, the network is built dynamically (using the API for Netica [Norsys, 2005]) when the system is presented with a new information graphic, and nodes are added to the network only as suggested by communicative signals in the graphic, as discussed in the next two sections.

4 Communicative Signals

In plan recognition systems involving dialogue, the communicative signals are naturally centered around the utterances, and the inference process uses signals such as the surface form of the utterance, the focus of attention in the dialogue, etc. Following AutoBrief [Kerpedjiev and Roth, 2000] [Green *et al.*, 2004], we contend that when constructing a graphic, the designer made certain design decisions in order to make "important" tasks (the ones that the viewer is intended to perform in getting the graphic's message) as *easy* or as *salient* as possible. The realization of these design decisions in the information graphic serve as communicative signals. The rest of this section identifies the kinds of com-

municative signals that appear in information graphics, and Section 5 discusses how these signals are used in our plan inference framework.

4.1 Effort as a Communicative Signal

The design choices made by the graphic designer facilitate some perceptual tasks more than others. For example, if the viewer is intended to compare the height of two bars, placing the two bars beside each other in the graphic will facilitate the comparison task. Thus the relative effort required for a perceptual task serves as a communicative signal about what tasks the designer expects the viewer to perform.

4.2 Salience as a Communicative Signal

Our working hypothesis is that if the graphic designer goes to the effort of employing attention-getting devices to make certain elements of the graphic particularly salient, then the salient elements serve as communicative signals — i.e., the designer probably intends for them to be part of the intended message of the graphic.

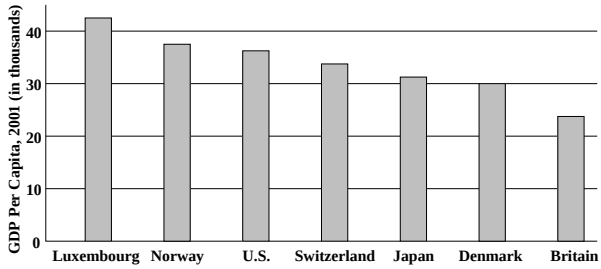


Figure 6: Information Graphic Example[3]

Highlighting (via a difference in color or shading) is one way of signalling salience. Suppose that the bar for the U.S. was the only red bar in the graph shown in Figure 6. This would suggest that this bar is particularly relevant to the intended message of the graphic. Annotating individual elements with their exact values also signals salience. This is not the case when all of the elements are displayed with their exact values, since this is a general design feature of the graphic and does not draw attention to a specific subset of elements. Other salient entities include any element that is significantly taller than all of the other elements in the graphic and the most recent date on a time-line, since the viewer will certainly notice the height of a bar that is taller than all of the others, and will naturally be interested in what has occurred most recently. Although no specific action is required on the part of the graphic designer to make these elements salient, we posit that it is mutually believed by both designer and viewer that such elements will be salient to the viewer.

Captions by themselves are often too general [Corio and Lapalme, 1999] or ill-formed to rely solely on them to identify what the graphic is intended to convey. However, we have found that captions often contain useful communicative signals [Elzer *et al.*, 2005]. First, nouns in captions can function the same as highlighting; if the noun matches a label in the graphic, then it signals the salience of the graphic element associated with that label. Certain verbs and adjectives in captions also serve as communicative signals. In this case, the verbs and adjectives signal the salience of certain high-level tasks. For example, the adjective *highest* in the caption "Highest in GDP" suggests that the viewer should find the maximum in the graphic, whereas the verb *beats* in the caption "U.S. Beats Japan" indicates that the viewer should note the relative difference in value between the U.S. and Japan. [Elzer *et al.*, 2005] presents our work on exploiting captions in understanding information graphics.

[3]This is based on a U.S. News & World Report graphic in our corpus.

5 Exploiting the Communicative Signals

The communicative signals gleaned from an information graphic can be exploited for two purposes: 1) to select perceptual task nodes for insertion in the network, and 2) as evidence that will influence the hypothesis about the graphic designer's intentions.

As discussed in Section 3, the Bayesian network must be restricted to those perceptual tasks suggested by the graphic. We use two kinds of communicative signals for this purpose: perceptual effort and salient elements. We developed a set of rules, based on research by cognitive scientists such as [Lohse, 1993], for estimating the required perceptual effort for a given perceptual task in a given graphic (eyetracking experiments validating these rules are detailed in [Elzer *et al.*, 2004]). This set of rules is used to identify the set of instantiated perceptual tasks in a graphic that are easiest relative to others, and nodes representing these tasks are added to the network. In some graphics, these are the only communicative signals available. However, if there are salient elements in the graphic (such as a highlighted bar or a bar whose label appears as a noun in the caption), this salience signals that the element may play a role in the plan for achieving the intended message of the graphic. Therefore, we instantiate each possible perceptual task with the salient elements, add them to the network, and chain upward as described in Section 3 until a link is established to the top-level goal node. As nodes are added via upward chaining, their subgoals (as captured in the plan operators) are also added, so that perceptual tasks required to achieve these subgoals are included in the network even if they were not among the easiest or salient tasks.

A Bayesian network also needs to explicitly record evidence that should influence the credibility of different hypotheses. Perceptual task effort and salient elements provide evidence about the intended message. For each perceptual task node in the network, evidence nodes are added as children of the perceptual task node, as shown in Figure 7 – in our diagrams, nodes representing evidence extracted from the graphic are shown with dashed outlines. For example, the *Effort* evidence node in Figure 7 captures the relative effort required for the perceptual task that is its parent, the *Highlighting* node captures whether a parameter in the perceptual task is instantiated with a highlighted element in the graphic, and the *NounInCaption* node captures whether a parameter in the perceptual task is instantiated with a bar whose label matches a noun in the caption.

Certain verbs and adjectives in the caption are also com-

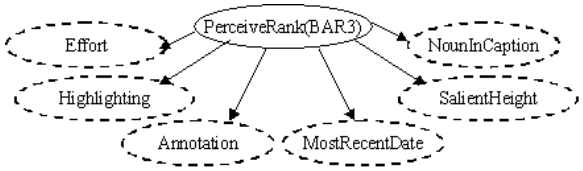


Figure 7: Perceptual Task Node with Evidence Nodes

municative signals that provide evidence about the intended message of an information graphic. Using Wordnet, we developed classes of verbs and adjectives where members of each class perform in a similar way within the domain of information graphic captions. (In the case of adjectives derived from verbs, such as the adjective *growing* in the caption *A Growing Biotech Market*, the adjective is treated as a verb using its root form.) For example, the class for the verb *rise* includes verbs such as *increase*, *grow*, *expand*, *climb*, and *soar*. Verb class and adjective class evidence nodes record which verb and/or adjective class (if any) appears in the caption; they are inserted as children of the top-level node as shown in Figure 8, since this type of evidence suggests a particular category of high-level intention.

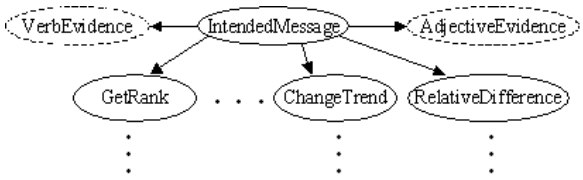


Figure 8: Top-Level Node with Verb and Adjective Evidence

6 Gathering the Probabilities

In a Bayesian network, conditional probability tables capture the probability of each of the values of a child node given the value of its parent(s). In contrast to many other probabilistic plan recognition models where it is difficult to empirically determine the probabilities, the probabilities for our belief network have been obtained through a corpus analysis.

We automated the construction of a spreadsheet containing information from each graphic needed to compute the necessary probabilities. Each graphic's XML representation was analyzed to determine for each possible instantiation of each perceptual task:

- the relative effort (categorized as easy, medium, hard, or impossible) as estimated by our effort estimation rules discussed in Section 5
- which parameters refer to elements that were salient in the graphic and the kind of salience (highlighted bars, annotated bars, bars with labels that appear as nouns in the caption, etc.)

In addition, the occurrence in the caption of a verb or adjective class that suggested a particular intention was entered into the spreadsheet.

Two coders had previously identified the intention of each graphic in the corpus. We applied our plan operators to construct a plan (constrained by what appeared in the graphic) for achieving the posited intention of each graphic; the goal-subgoal relationships, including the low-level perceptual and cognitive tasks, were entered into the spreadsheet.

PerceiveRank(<bar>)	**InPlan**	**NotInPlan**
Only <bar> annotated	24.99	2.3
<bar> and others annotated	0.01	0.9
only others annotated	0.01	19.5
no bars annotated	74.99	77.3

Figure 9: A Sample Conditional Probability Table

Spreadsheet formulas were then constructed to compute all of the required conditional probability tables. Examples of the needed probability tables include: 1) the conditional probability of a particular perceptual task being easy, medium, hard or impossible given that the perceptual task is (or is not) part of the plan, 2) the conditional probability, for each class of verb, that the verb class appears in the caption given that the communicative goal of the graphic does (or does not) fall into a particular category of intention, and 3) the conditional probability of a bar being annotated given that recognizing the intended message entails (or does not entail) performing a particular perceptual task involving that bar. Figure 9 shows this latter conditional probability table.

7 System Performance

7.1 Illustrative Examples

The following examples illustrate how different kinds of evidence impact our system's hypothesis. The examples are based on the bar chart in Figure 6. The XML representation of each example was produced by the Visual Extraction Module (VEM), augmented by the Caption Tagging Module (CTM), and then processed by the Intention Recognition Module (IRM) (see Figure 1). Given the graphic as it appears in Figure 6, our system hypothesizes that the graphic is intended to convey the relative rank in GDP of different countries and assigns this intention a likelihood of 87%. Other possibilities also have some probability assigned to them. For example, the intention of conveying that Luxembourg has the highest GDP is assigned a probability of 12.4% because the bars are in sorted order according to height, thus making it relatively easy for the viewer to recognize the maximum, and because finding the entity in the graphic with the maximum value is a fairly common intention (occurring approximately 22.7% of the time in our corpus). However, there is no other evidence suggesting that the bar representing the maximum value is salient (such as that bar being highlighted, or "Luxembourg" being mentioned in the caption), so the system hypothesizes that the viewer is intended to notice the relative rank of all of the countries listed.

Suppose, however, that the bar representing the U.S. was darker than the other bars. Now, the hypothesis that the intended message of the graphic is to convey that the rank of the U.S. is third is assigned a probability of 99.5%. The fact that the bar is highlighted provides strong evidence that it plays a role in the intended message of the graphic. Elements of the graphic could also be made salient in other ways, such as through annotations. Suppose that the bar representing the U.S. was still darker than the other bars, but that the bars

representing the U.S. and Japan (and only those bars) were annotated with their exact values. Here the evidence still suggests the salience of the U.S., but also suggests that Japan is salient. The fact that both bars are now salient will provide evidence against intentions involving only the U.S. and will favor hypotheses involving both bars. Thus it is not surprising that the system hypothesizes the intention of the graphic to be finding the relative difference (and the degree of that difference) between the GDP of the U.S. and Japan and assigns it a likelihood of 87.3%. Now consider a significant variation of the graphic design. Suppose, again, that the bar representing the U.S. was darker than the other bars. But now suppose that the bars were sorted by the alphabetical order of their labels, rather than by descending order of their height. The perceptual task of determining the rank of the U.S. is now so difficult to perform that the system assigns only a negligible probability to the task of getting the rank of the U.S.

7.2 Overall Evaluation and Future Work

To evaluate our system, we used leave-one-out cross validation. We performed a series of experiments in which each graphic in a corpus of 110 graphics is selected once as the test graphic. The data pertaining to the test graphic was removed from calculation of the probabilities used in the conditional probability tables. The test graphic was then presented to our intention recognition module, and the Bayesian network was constructed with probability tables acquired from the remaining graphics. The system was judged to fail if either its top-rated hypothesis did not match the intended message that was assigned to the graphic by the coders or the probability rating of the system's top-rated hypothesis did not exceed 50%. Overall success was then computed by averaging together the results of the whole series of experiments. Our overall accuracy was 79.1%. As a baseline, our most commonly occurring intention (at 23.6%) was a rising trend. Our success rate provides strong evidence of the success of our methodology.

Most of the errors in our current evaluation are caused by sparseness of the data; for example, if we only have a single graphic using a verb like "stabilize" to indicate a stable trend, the leave-one-out validation will not have any evidence linking the verb class to the intention, and we may get an incorrect result when evaluating the graphic. We strongly believe that as we expand our corpus, our success rate will improve. We currently limit ourselves to recognizing what appears to be the primary communicative intention of an information graphic; in the future we will also consider secondary intentions. We will also extend our work to other kinds of information graphics such as line graphs and pie charts, and to complex graphics, such as composite bar charts.

8 Conclusion

This paper has presented a probabilistic framework that is the first to extend plan inference to the problem of recognizing the communicative message of an information graphic. We have identified the communicative signals that appear in information graphics and exploited them as evidence in a Bayesian belief network. The evaluation of our implemented system's performance demonstrates the effectiveness of our methodology, which offers promise as a means of providing access via natural language to information graphics for individuals with sight impairments.

References

[Charniak and Goldman, 1993] E. Charniak and R. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, Nov. 1993.

[Clark, 1996] H. Clark. *Using Language*. Cambridge University Press, 1996.

[Corio and Lapalme, 1999] M. Corio and G. Lapalme. Generation of texts for information graphics. *Proc. European Workshop on Natural Language Generation*, 49–58, 1999.

[Elzer *et al.*, 2004] S. Elzer, S. Carberry, N. Green, and J. Hoffman. Incorporating perceptual task effort into the recognition of intention in information graphics. In *Proceedings of the 3rd Int'l Conference on Diagrams*, LNAI 2980, pages 255–270, March 2004.

[Elzer *et al.*, 2005] S. Elzer, S. Carberry, D. Chester, S. Demir, N. Green, I. Zukerman, and K. Trnka. Exploring and exploiting the limited utility of captions in recognizing intention in information graphics. In *Proceedings of ACL*, June 2005, to appear.

[Green *et al.*, 2004] Nancy Green, Giuseppe Carenini, Stephan Kerpedjiev, Joe Mattis, Johanna Moore, and Steven Roth. Autobrief: an experimental system for the automatic generation of briefings in integrated text and information graphics. *International Journal of Human-Computer Studies*, 61(1):32–70, 2004.

[Grice, 1969] H. P. Grice. Utterer's meaning and intentions. *Philosophical Review*, p. 147–177, 1969.

[Huber *et al.*, 1994] M. Huber, E. Durfee, and M. Wellman. The automated mapping of plans for plan recognition. In *Proc. of the Conference on Uncertainty in Artificial Intelligence*, July 1994.

[Kennel, 1996] A. Kennel. Audiograf: A diagram-reader for the blind. In *Second Annual ACM Conference on Assistive Technologies*, p. 51–56, 1996.

[Kerpedjiev and Roth, 2000] S. Kerpedjiev and S. Roth. Mapping communicative goals into conceptual tasks to generate graphics in discourse. In *Intelligent User Interfaces*, p. 157–164, 2000.

[Lohse, 1993] G. Lohse. A cognitive model for understanding graphical perception. *Human-Computer Interaction*, 8:353–388, 1993.

[Norsys, 2005] Norsys Software Corp.: Netica. *http://www.norsys.com/netica.html*.

[Perrault and Allen, 1980] R. Perrault and J. Allen. A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, 6(3-4):167–182, 1980.

[Yu *et al.*, 2002] J. Yu, J. Hunter, E. Reiter, and S. Sripada. Recognising visual patterns to communicate gas turbine time-series data. In A. Macintosh, R. Ellis, and F. Coenen, editors, *Proceedings of ES2002*, p. 105–118, 2002.

Feature Generation for Text Categorization Using World Knowledge

Evgeniy Gabrilovich and **Shaul Markovitch**
{gabr,shaulm}@cs.technion.ac.il
Computer Science Department, Technion, 32000 Haifa, Israel

Abstract

We enhance machine learning algorithms for text categorization with generated features based on domain-specific and common-sense knowledge. This knowledge is represented using publicly available ontologies that contain hundreds of thousands of concepts, such as the Open Directory; these ontologies are further enriched by several orders of magnitude through controlled Web crawling. Prior to text categorization, a feature generator analyzes the documents and maps them onto appropriate ontology concepts, which in turn induce a set of generated features that augment the standard bag of words. Feature generation is accomplished through contextual analysis of document text, implicitly performing word sense disambiguation. Coupled with the ability to generalize concepts using the ontology, this approach addresses the two main problems of natural language processing—synonymy and polysemy. Categorizing documents with the aid of knowledge-based features leverages information that cannot be deduced from the documents alone. Experimental results confirm improved performance, breaking through the plateau previously reached in the field.

1 Introduction

The state of the art systems for text categorization use induction algorithms in conjunction with word-based features ("bag of words"). After a decade of improvements, the performance of the best document categorization systems became more or less similar, and it appears as though a plateau has been reached, as neither system is considerably superior to others, and improvements are becoming evolutionary [Sebastiani, 2002].

The bag of words (BOW) approach is inherently limited, as it can only use pieces of information that are explicitly mentioned in the documents, and even that provided the same vocabulary is consistently used. Specifically, this approach has no access to the wealth of world knowledge possessed by humans, and is easily puzzled by facts and terms not mentioned in the training set.

To illustrate the limitations of the BOW approach, consider document #15264 in Reuters-21578, which is one of the most frequently used datasets in text categorization research. This document discusses a joint mining venture by a consortium of companies, and belongs to the category "copper". However, the document only briefly mentions that the aim of this venture is mining copper; instead, this fairly long document mainly talks about the mutual share holdings of the companies involved (Teck Corporation, Cominco, and Lornex Mining), as well as discusses other mining activities of the consortium. Consequently, three very different text classifiers we used (SVM, KNN and C4.5) failed to classify the document correctly. This comes as no surprise—"copper" is a fairly small category, and neither of these companies, nor the location of the venture (Highland Valley in British Columbia) are ever mentioned in the training set for this category. The failure of the bag of words approach is therefore unavoidable, as it cannot reason about the important components of the story.

We argue that this needs not be the case. When a Reuters editor originally handled this document, she most likely knew quite a lot about the business of these companies, and easily assigned the document to the category "copper". It is this kind of knowledge that we would like machine learning algorithms to have access to.

To date, quite a few attempts have been made to deviate from the orthodox bag of words paradigm, usually with limited success. In particular, representations based on phrases [Dumais *et al.*, 1998; Fuernkranz *et al.*, 2000], named entities [Kumaran and Allan, 2004], and term clustering [Lewis and Croft, 1990; Bekkerman, 2003] have been explored. In the above example, however, none of these techniques could overcome the underlying problem—lack of world knowledge.

In order to solve this problem and break through the existing performance barrier, a fundamentally new approach is apparently necessary. One possible solution is to completely depart from the paradigm of induction algorithms in an attempt to perform deep understanding of the document text. Yet, considering the current state of natural language processing systems, this does not seem to be a viable option (at least for the time being).

We propose an alternative solution that capitalizes on the power of existing induction techniques while enriching the language of representation, namely, exploring new feature spaces. Prior to text categorization, we employ a *feature generator* that uses common-sense and domain-specific knowledge to enrich the bag of words with new, more informative features. Feature generation is performed completely automatically, using machine-readable hierarchical repositories of knowledge such as the Open Directory Project (ODP), Yahoo! Web Directory, and the Wikipedia encyclopedia.

In this paper we use the ODP as a source of background knowledge. Thus, in the above example the feature generator "knows" that the companies mentioned are in the mining

business, and that Highland Valley happens to host a copper mine. This information is available in Web pages that discuss the companies and their operations, and are cataloged in corresponding ODP categories such as *Mining_and_Drilling* and *Metals*. Similarly, Web pages about Highland Valley are cataloged under *British_Columbia*. To amass this information, we crawl the URLs cataloged in the ODP, thus effectively multiplying the amount of knowledge available many times over. Armed with this knowledge, the feature generator constructs new features that denote these ODP categories, and adds them to the bag of words. The augmented feature space provides text classifiers with a cornucopia of additional information. Indeed, our implementation of the proposed methodology classifies this document correctly.

Feature generation (FG) techniques were found useful in a variety of machine learning tasks [Markovitch and Rosenstein, 2002; Fawcett, 1993; Matheus, 1991]. These techniques search for new features that describe the target concept better than the ones supplied with the training instances. A number of feature generation algorithms were proposed [Pagallo and Haussler, 1990; Matheus and Rendell, 1989; Hu and Kibler, 1996; Murphy and Pazzani, 1991], which led to significant improvements in performance over a range of classification tasks. However, even though feature generation is an established research area in machine learning, only a few works applied it to text processing [Kudenko and Hirsh, 1998; Mikheev, 1999; Cohen, 2000]. With the exception of a few studies using WordNet [Scott, 1998; Urena-Lopez *et al.*, 2001], none of them attempted to leverage repositories of world knowledge.

The contributions of this paper are threefold. First, we propose a framework and a collection of algorithms that perform feature generation based on very large-scale repositories of human knowledge. Second, we propose a novel kind of contextual analysis performed during feature generation, which views the document text as a sequence of local contexts, and performs implicit word sense disambiguation. Finally, we describe a way to further enhance existing knowledge bases by several orders of magnitude by crawling the World Wide Web. As we show in Section 3, our approach allows to break the performance barrier currently reached by the best text categorization systems.

2 Feature Generation Methodology

The proposed methodology allows principled and uniform integration of external knowledge to construct new features. In the preprocessing step we use knowledge repositories to build a feature generator. Applying the feature generator to the documents produces a set of generated features. These features undergo feature selection, and the most discriminative ones are added to the bag of words. Finally, we use traditional text categorization techniques to learn a text categorizer in the augmented feature space.

Suitable knowledge repositories satisfy the following requirements:

1. The repository contains a collection of *concepts* organized in a hierarchical tree structure, where edges represent the "is-a" relationship. Using a hierarchical ontology allows us to perform powerful generalizations.
2. There is a collection of texts associated with each concept. The feature generator uses these texts to learn the definition and scope of the concept, in order to be able to assign it to relevant documents.

We currently use the ODP as our knowledge base, however, our methodology is general enough to facilitate other sources of common-sense and domain-specific knowledge that satisfy the above assumptions. The Open Directory comprises a hierarchy of approximately 600,000 categories that catalog over 4,000,000 Web sites, each represented by a URL, a title, and a brief summary of its contents. The project constitutes an ongoing effort promoted by over 65,000 volunteer editors around the globe, and is arguably the largest publicly available Web directory. Being the result of *pro bono* work, the Open Directory has its share of drawbacks, such as non-uniform coverage, duplicate subtrees in different branches of the hierarchy, and sometimes biased coverage due to peculiar views of the editors in charge. Nonetheless, ODP embeds a colossal amount of human knowledge in a wide variety of areas, covering even most specific scientific and technical concepts. In what follows, we refer to the ODP category nodes as *concepts*, to avoid possible confusion with the term *categories*, which is usually reserved for the labels assigned to documents in text categorization.

2.1 Building a Feature Generator

We start with a preprocessing step, performed once for all future text categorization tasks. We induce a hierarchical text classifier that maps pieces of text onto relevant ODP concepts, which later serve as generated features. The resulting classifier is called a *feature generator* according to its true purpose in our scheme, as opposed to the text categorizer (or classifier) that we build ultimately. The feature generator represents ODP concepts as vectors of their most characteristic words, which we call *attributes* (reserving the term *features* to denote the properties of documents in text categorization). We now explain how the attribute vectors are built.

We use the textual descriptions of ODP nodes and their URLs as training examples for learning the feature generator. Although these descriptions alone constitute a sizeable amount of information, we devised a way to increase the volume of training data by several orders of magnitude. We do so by crawling the Web sites pointed at by all cataloged URLs, and obtain a small representative sample of each site. Pooling together the samples of all sites associated with an ODP node gives us a wealth of additional information about it.

Texts harvested from the WWW are often plagued with noise, and without adequate noise reduction crawled data may do more harm than good. To remedy the situation, we perform *attribute selection* for each ODP node; this can be done using any standard feature selection technique such as information gain. For example, consider the top 10 attributes selected for the ODP concepts *Science—science, research, scientific, biology, laboratory, analysis, university, theory, study, scientist*, and *Artificial_Intelligence—neural, artificial, algorithm, intelligence, AAAI, Bayesian, probability, IEEE, cognitive, inference*. Additional noise reduction is achieved by pruning nodes having too few URLs or situated too deep in

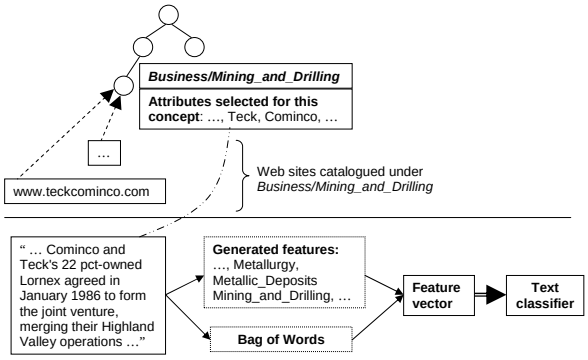


Figure 1: Feature generation example

the tree (and thus representing overly specific concepts), assigning their textual content to their parents.

In our current implementation, the feature generator works as a nearest neighbor classifier—it compares its input text to (the attribute vectors of) all ODP nodes, and returns the desired number of best-matching ones. The generator also performs *generalization* of concepts, and constructs features based on the classified concepts *per se* as well as their ancestors in the hierarchy.

Let us revisit the example from Section 1. While building the feature generator, our system crawls the Web sites cataloged under mining-related ODP concepts such as *Business/Mining_and_Drilling*, *Science/Technology/Mining* and *Business/Industrial_Goods_and_Services/Materials/Metals*. These include `www.teckcominco.com` and `www.mining-surplus.com` that belong to the (now merged) Teck Cominco company. Due to the company's prominence, it is mentioned frequently in the Web sites we have crawled, and consequently the words "Teck" and "Cominco" are included in the set of attributes selected to represent the above concepts. Figure 1 illustrates the process of feature generation for this example.

2.2 Contextual Feature Generation

Traditionally, feature generation uses the basic features supplied with the training instances to construct more sophisticated features. In the case of text processing, however, applying this approach to the bag of words leads to losing the important information about word ordering. Therefore, we argue that feature generation becomes much more powerful when it operates on the raw document text. But should the generator always analyze the whole document as a single unit, similarly to regular text classifiers?

We believe that considering the entire document may often be misleading, as its text can be too diverse to be readily mapped to the right set of concepts, while notions mentioned only briefly may be overlooked. Instead, we propose to partition the document into a series of non-overlapping segments (called *contexts*), and then generate features at this finer level. Each context is classified into a number of concepts in the knowledge base, and pooling these concepts together results in *multi-faceted* classification for the document. This way, the resulting set of concepts represents the various aspects or sub-topics covered by the document.

Potential candidates for such contexts are simple sequences of words, or more linguistically motivated chunks such as sentences or paragraphs. The optimal resolution for document segmentation can be determined automatically using a validation set. We propose a more principled *multi-resolution* approach that simultaneously partitions the document at several levels of linguistic abstraction (windows of words, sentences, paragraphs, up to taking the entire document as one big chunk), and performs feature generation at each of these levels. We rely on the subsequent *feature selection* step to eliminate extraneous features, preserving only those that genuinely characterize the document.

In fact, the proposed approach tackles the two most important problems in natural language processing, namely, *synonymy* and *polysemy*. Classifying individual contexts implicitly performs *word sense disambiguation*, and thus resolves word polysemy to some degree. A context that contains one or more polysemous words is mapped to the concepts that correspond to the sense *shared* by the context words. Thus, the correct sense of each word is determined with the help of its neighbors. At the same time, enriching document representation with high-level concepts and their generalizations addresses the problem of synonymy, as the enhanced representation can easily recognize that two (or more) documents actually talk about related issues, even though they do so using different vocabularies.

Let us again revisit our running example. During feature generation, document #15264 is segmented into a sequence of contexts, which are then mapped to mining-related ODP concepts (e.g., *Business/Mining_and_Drilling*). These concepts, as well as their ancestors in the hierarchy, give rise to a set of generated features that augment the bag of words (see Figure 1). Observe that the training documents for the category "copper" underwent a similar processing when a text classifier was induced. Consequently, features based on these concepts *were selected* during feature selection thanks to their high predictive capacity. It is due to these features that the document is now categorized correctly, while without feature generation it consistently caused BOW classifiers to err.

2.3 Feature Selection

Using support vector machines in conjunction with bag of words, Joachims [1998] found that SVMs are very robust even in the presence of numerous features, and further observed that the multitude of features are indeed useful for text categorization. These findings were corroborated in more recent studies [Brank *et al.*, 2002; Bekkerman, 2003] that observed either no improvement or even small degradation of SVM performance after feature selection.[1] Consequently, many later works using SVMs did not apply feature selection at all [Leopold and Kindermann, 2002; Lewis *et al.*, 2004].

This situation changes drastically as we augment the bag of words with generated features. First, nearly any technique for automatic feature generation can easily generate huge numbers of features, which will likely aggravate the "curse of dimensionality". Furthermore, it is feature selection that allows the feature generator not to be a perfect classifier. When at

[1] Gabrilovich and Markovitch [2004] described a class of problems where feature selection from the bag of words actually improves SVM performance.

least some of the concepts assigned to the document are correct, feature selection can identify them and seamlessly eliminate the spurious ones.

3 Empirical Evaluation

We implemented the proposed methodology using an ODP snapshot as of April 2004.

3.1 Implementation Details

After pruning the *Top/World* branch that contains non-English material, we obtained a hierarchy of over 400,000 concepts and 2,800,000 URLs. Applying our methodology to a knowledge base of this scale required an enormous engineering effort. Textual descriptions of the concepts and URLs amounted to 436 Mb of text. In order to increase the amount of information for training the feature generator, we further populated the ODP hierarchy by crawling all of its URLs, and taking the first 10 pages (in the BFS order) encountered at each site. This operation yielded 425 Gb worth of HTML files. After eliminating all the markup and truncating overly long files at 50 Kb, we ended up with 70 Gb of additional textual data. Compared to the original 436 Mb of text supplied with the hierarchy, we obtained over a 150-fold increase in the amount of data. After removing stop words and rare words (occurring in less than 5 documents) and stemming the remaining ones, we obtained 20,700,000 distinct terms that were used to represent ODP nodes as attribute vectors. Up to 1000 most informative attributes were selected for each ODP node using the Document Frequency criterion (other commonly used feature selection techniques, such as Information Gain, χ^2 and Odds Ratio, yielded slightly inferior results). We used the *multi-resolution* approach for feature generation, classifying document contexts at the level of words, sentences, paragraphs, and finally the entire document.Features were generated from the 10 best-matching ODP concepts for each context.

3.2 Experimental Methodology

The following test collections were used:

1. Reuters-21578 [Reuters, 1997]. Following common practice, we used the ModApte split (9603 training, 3299 testing documents) and two category sets, 10 largest categories and 90 categories with at least one training and testing example.

2. Reuters Corpus Volume I (RCV1) [Lewis *et al.*, 2004] has over 800,000 documents, and presents a new challenge for text categorization. To speedup experimentation, we used a subset of the corpus with 17808 training documents (dated 20–27/08/96) and 5341 testing ones (28–31/08/96). Following Brank *et al.* [2002], we used 16 Topic and 16 Industry categories that constitute a representative sample of the full groups of 103 and 354 categories, respectively. We also randomly sampled the Topic and Industry categories into several sets of 10 categories each (Table 1 shows 3 category sets in each group with the highest improvement in categorization performance).[2]

3. 20 Newsgroups (20NG) [Lang, 1995] is a well-balanced dataset of 20 categories containing 1000 documents each.

4. Movie Reviews (Movies) [Pang *et al.*, 2002] defines a sentiment classification task, where reviews express either positive or negative opinion about the movies. The dataset has 1400 documents in two categories (positive/negative).

We used support vector machines[3] as our learning algorithm to build text categorizers, since prior studies found SVMs to have the best performance for text categorization [Dumais *et al.*, 1998; Yang and Liu, 1999]. Following established practice, we use the precision-recall Break-Even Point (BEP) to measure text categorization performance. For the two Reuters datasets we report both micro- and macro-averaged BEP, since their categories differ in size significantly. Micro-averaged BEP operates at the document level and is primarily affected by categorization performance on larger categories. On the other hand, macro-averaged BEP averages results for individual categories, and thus small categories with few training examples have large impact on the overall performance. For both Reuters datasets we used a fixed data split, and consequently used macro sign test (S-test) [Yang and Liu, 1999] to assess the statistical significance of differences in classifier performance. For 20NG and Movies we performed 4-fold cross-validation, and used paired t-test to assess the significance.

3.3 The Effect of Feature Generation

We first demonstrate that the performance of basic text categorization in our implementation (column "Baseline" in Table 1) is consistent with other published studies (all using SVM). On Reuters-21578, Dumais *et al.* [1998] achieved micro-BEP of 0.920 for 10 categories and 0.870 for all categories. On 20NG, Bekkerman [2003] obtained BEP of 0.856. Pang *et al.* [2002] obtained accuracy of 0.829 on Movies. The minor variations in performance are due to differences in data preprocessing used in different systems; for example, for the Movies dataset we worked with raw HTML files rather than with the official tokenized version, in order to recover sentence and paragraph structure for contextual analysis. For RCV1, direct comparison with published results is more difficult, as we limited the category sets and the date span of documents to speedup experimentation.

Table 1 shows the results of using feature generation with significant improvements ($p < 0.05$) shown in bold. For both Reuters datasets, we consistently observed larger improvements in macro-averaged BEP, which is dominated by categorization effectiveness on small categories. This goes in line with our expectations that the contribution of external knowledge should be especially prominent for categories with few training examples. As can be readily seen, categorization performance was improved for all datasets, with notable improvements of up to 25.4% for Reuters RCV1 and 3.6% for Movies. Given the performance plateau currently reached by the best text categorizers, these results clearly demonstrate the advantage of knowledge-based feature generation.

3.4 Actual Examples Under a Magnifying Glass

Thanks to feature generation our system correctly classifies the running example document #15264. Let us consider additional testing examples from Reuters-21578 that are incor-

[2] Full definition of the category sets we used is available at `http://www.cs.technion.ac.il/~gabr/ijcai2005-appendix.html`.

[3] SVM^{light} implementation [Joachims, 1998].

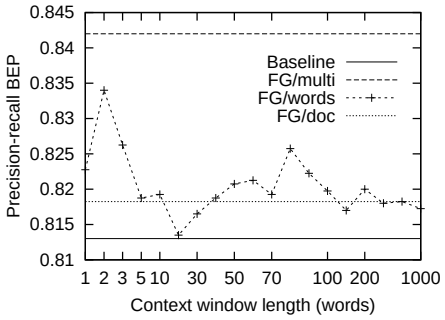


Figure 2: Varying context length (Movies)

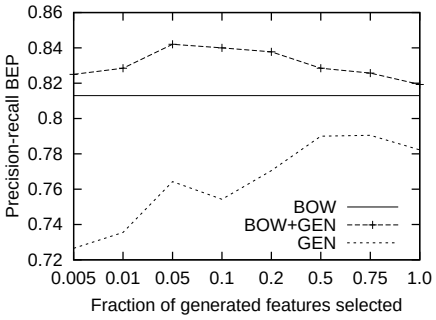


Figure 3: Feature selection (Movies)

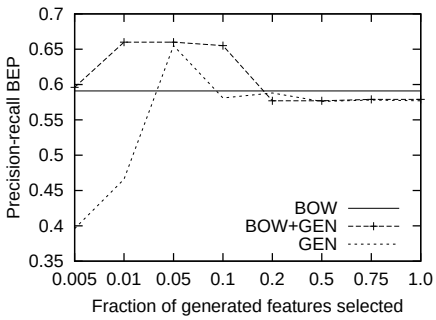


Figure 4: Feature selection (RCV1/Topic-16)

Dataset	Baseline		Feature generation		Improvement vs. baseline	
	micro BEP	macro BEP	micro BEP	macro BEP	micro BEP	macro BEP
Reuters-21578						
10 categories	0.925	0.874	0.930	0.884	+0.5%	+1.1%
90 categories	0.877	0.602	0.880	0.614	+0.3%	+2.0%
RCV1						
Industry-16	0.642	0.595	0.648	**0.613**	+0.9%	**+3.0%**
Industry-10A	0.421	0.335	**0.457**	**0.420**	**+8.6%**	**+25.4%**
Industry-10B	0.489	0.528	**0.530**	**0.560**	**+8.4%**	**+6.1%**
Industry-10C	0.443	0.414	**0.468**	**0.463**	**+5.6%**	**+11.8%**
Topic-16	0.836	0.591	0.840	**0.660**	+0.5%	**+11.7%**
Topic-10A	0.796	0.587	0.806	**0.695**	+1.3%	**+18.4%**
Topic-10B	0.716	0.618	0.731	**0.651**	+2.1%	**+5.3%**
Topic-10C	0.719	0.616	0.727	**0.646**	+1.1%	**+4.9%**
20NG	0.854		**0.858**		**+0.5%**	
Movies	0.813		**0.842**		**+3.6%**	

Table 1: Text categorization with and without feature generation

rectly categorized by the BOW classifier. Document #16143 belongs to the category "money-fx" (money/foreign exchange) and discusses the devaluation of the Kenyan shilling. Even though "money-fx" is one of the 10 largest categories, the word "shilling" does not occur in its training documents even once. However, the feature generator easily recognizes it as a kind of currency, and produces features such as *Recreation/Collecting/Paper_Money* and *Recreation/Collecting/Coins/World_Coins*. These high-level features were also constructed for many training examples, and consequently the document is now classified correctly.

Similarly, document #18748 discusses Italy's balance of payments and belongs to the category "trade", while the word "trade" itself does not occur in this short document. However, when the feature generator considers document contexts discussing Italian deficit as reported by the Bank of Italy, it correctly maps them to concepts such as *Society/Government/Finance*, *Society/Issues/Economic/International/Trade*, *Business/International_Business_and_Trade*. Due to these features, which were also generated for training documents in this category, the document is now categorized correctly.

3.5 The Effect of Contextual Analysis

We now explore the various possibilities to define document contexts for feature generation. Figure 2 shows how text categorization performance on the Movies dataset changes for various contexts. The x-axis measures context length in words, and the *FG/words* curve corresponds to applying the feature generator to the context of that size. With these word-level contexts, maximum performance is achieved when using pairs of words (x=2). The *FG/doc* line shows the result of using the entire document as a single context. In this case, the results are somewhat better than without feature generation (*Baseline*), but are still inferior to the more fine-grained word-level contexts (*FG/words*). However, the best performance by far is achieved when using the multi-resolution approach (*FG/multi*), in which we use a series of linguistically motivated chunks of text, starting with individual words, and then generating features from sentences, paragraphs, and finally the entire document.

3.6 The Utility of Feature Selection

Under the experimental settings defined in Section 3.1, feature generation constructs approximately 4–5 times as many features as are in the bag of words. We conducted two experiments to understand the effect of feature selection in conjunction with feature generation.

Since earlier studies found that most BOW features are indeed useful for SVM text categorization (Section 2.3), in our first experiment we only apply feature selection to the generated features, and use the selected ones to augment the (entire) bag of words. In Figures 3 and 4, the *BOW* line depicts the baseline performance without generated features, while the *BOW+GEN* curve shows the performance of the bag of words augmented with progressively larger fractions of generated features (sorted by information gain). For both datasets, the performance peaks when only a small fraction of the generated features are used, while retaining more generated features has a noticeable detrimental effect. Similar phenomena have been observed for other datasets; we omit the results owing to lack of space.

Our second experiment was set up to examine the performance of the generated features alone, without the bag of words (*GEN* curve in Figures 3 and 4). For Movies, discarding the BOW features hurts the performance somewhat, but the decrease is far less significant than what could be expected—using only the generated features we lose less than 3% in BEP compared with the BOW baseline. For 20NG, a similar experiment sacrifices about 10% off the BOW performance, as this dataset is known to have a very diversified vocabulary, for which many studies found feature selection to be particularly harmful. However, the situation is reversed for both Reuters datasets. For Reuters-21578, the generated features alone yield a 0.3% improvement in micro- and

macro-BEP for 10 categories, while for 90 categories they only lose 0.3% in micro-BEP and 3.5% in macro-BEP compared with the bag of words. For RCV1/Industry-16, disposing of the bag of words hurts the BEP by 1–3%. Surprisingly, for RCV1/Topic-16 (Figure 4) the generated features *per se* command a 10.8% improvement in macro-BEP, rivalling the performance of *BOW+GEN* that only gains another 1% improvement (Table 1). We interpret these findings as a further reinforcement of the quality of representation due to the generated features.

4 Conclusions and Future Work

We proposed a feature generation methodology for text categorization. In order to render machine learning algorithms with common-sense and domain-specific knowledge possessed by humans, we use large hierarchical knowledge bases to build a *feature generator*. The latter analyzes documents prior to text categorization, and augments the conventional bag of words representation with relevant concepts from the knowledge base. The enriched representation contains information that cannot be deduced from the document text alone.

We further described the multi-resolution analysis that examines the document text at several levels of linguistic abstraction, and performs feature generation at each level. Considering polysemous words in their native context implicitly performs word sense disambiguation, and allows the feature generator to cope with word synonymy and polysemy.

Empirical evaluation definitively confirmed that knowledge-based feature generation brings text categorization to a new level of performance. Interestingly, the sheer breadth and depth of the ODP, further boosted by crawling the URLs cataloged in the directory, brought about improvements both in regular text categorization as well as in the (non-topical) sentiment classification task.

We believe that this research only scratches the surface of what can be achieved with knowledge-rich features. In our future work, we plan to investigate new algorithms for mapping document contexts onto hierarchy concepts, as well as new techniques for selecting attributes that are most characteristic of every concept. We intend to apply focused crawling to only collect relevant web pages when crawling cataloged URLs. In addition to the ODP, we also plan to make use of domain-specific hierarchical knowledge bases, such as the Medical Subject Headings (MeSH). Finally, we conjecture that knowledge-based feature generation will also be useful for other information retrieval tasks beyond text categorization, and we intend to investigate this in our future work.

Acknowledgments

We thank Lev Finkelstein and Alex Gontmakher for many a helpful discussion. This research was partially supported by Technion's Counter-Terrorism Competition and by the MUSCLE Network of Excellence.

References

[Bekkerman, 2003] R. Bekkerman. Distributional clustering of words for text categorization. Master's thesis, Technion, 2003.

[Brank *et al.*, 2002] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Interaction of feature selection methods and linear classification models. In *Text Learning Workshop, ICML*, 2002.

[Cohen, 2000] W. Cohen. Automatically extracting features for concept learning from the web. In *ICML'00*, 2000.

[Dumais *et al.*, 1998] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM'98*, 1998.

[Fawcett, 1993] T. Fawcett. *Feature Discovery for Problem Solving Systems*. PhD thesis, UMass, May 1993.

[Fuernkranz *et al.*, 2000] J. Fuernkranz, T. Mitchell, and E. Riloff. A case study in using linguistic phrases for text categorization on the WWW. *Learning for Text Categorization*. AAAI Press, 2000.

[Gabrilovich and Markovitch, 2004] E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In *ICML'04*, pages 321–328, 2004.

[Hu and Kibler, 1996] Y. Hu and D. Kibler. A wrapper approach for constructive induction. In *AAAI-96*, 1996.

[Joachims, 1998] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML'98*, pages 137–142, 1998.

[Kudenko and Hirsh, 1998] D. Kudenko and H. Hirsh. Feature generation for sequence categorization. In *AAAI'98*, 1998.

[Kumaran and Allan, 2004] G. Kumaran and J. Allan. Text classification and named entities for new event detection. *SIGIR*, 2004.

[Lang, 1995] K. Lang. Newsweeder: Learning to filter netnews. In *ICML'95*, pages 331–339, 1995.

[Leopold and Kindermann, 2002] E. Leopold and J. Kindermann. Text categorization with support vector machines: How to represent texts in input space. *Machine Learning*, 46, 2002.

[Lewis and Croft, 1990] D. Lewis and W. Croft. Term clustering of syntactic phrases. In *SIGIR'90*, 1990.

[Lewis *et al.*, 2004] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[Markovitch and Rosenstein, 2002] S. Markovitch and D. Rosenstein. Feature generation using general constructor functions. *Machine Learning*, 49:59–98, 2002.

[Matheus and Rendell, 1989] C. Matheus and L. Rendell. Constructive induction on decision trees. *11th Int'l Conf. on AI*, 1989.

[Matheus, 1991] C. Matheus. The need for constructive induction. In *8th Int'l Workshop on Machine Learning*, 1991.

[Mikheev, 1999] A. Mikheev. Feature lattices and maximum entropy models. *Information Retrieval*, 1999.

[Murphy and Pazzani, 1991] P. Murphy and M. Pazzani. ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. In *ICML'91*, 1991.

[Pagallo and Haussler, 1990] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *ICML'90*, 1990.

[Pang *et al.*, 2002] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *EMNLP'02*, 2002.

[Reuters, 1997] *Reuters-21578 text categorization test collection*. `daviddlewis.com/resources/testcollections/reuters21578`.

[Scott, 1998] S. Scott. Feature engineering for a symbolic approach to text classification. Master's thesis, U. Ottawa, 1998.

[Sebastiani, 2002] F. Sebastiani. Machine learning in automated text categorization. *ACM Comp. Surveys*, 34(1), 2002.

[Urena-Lopez *et al.*, 2001] L. Urena-Lopez, M. Buenaga, and J. Gomez. Integrating linguistic resources in TC through WSD. *Computers and the Humanities*, 35, 2001.

[Yang and Liu, 1999] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR'99*, pages 42–49, 1999.

Learning Strategies for Open-Domain Natural Language Question Answering

Eugene Grois **David C. Wilkins**
Department of Computer Science and Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
{e-grois, dcw}@uiuc.edu

Abstract

We present an approach to automatically learning strategies for natural language question answering from examples composed of textual sources, questions, and answers. Our approach formulates QA as a problem of first order inference over a suitably expressive, learned representation. This framework draws on prior work in learning action and problem-solving strategies, as well as relational learning methods. We describe the design of a system implementing this model in the framework of natural language question answering for story comprehension. Finally, we compare our approach to three prior systems, and present experimental results demonstrating the efficacy of our model.

1 Introduction

This paper presents an approach to automatically learning strategies for natural language question answering from examples composed of textual sources, questions, and answers. Our approach is focused on one specific type of text-based question answering known as *story comprehension*. Most TREC-style QA systems are designed to extract an answer from a document contained in a fairly large general collection [Voorhees, 2003]. They tend to follow a generic architecture, such as the one suggested by [Hirschman and Gaizauskas, 2001], that includes components for document pre-processing and analysis, candidate passage selection, answer extraction, and response generation. Story comprehension requires a similar approach, but involves answering questions from a single narrative document. An important challenge in text-based question answering in general is posed by the syntactic and semantic variability of question and answer forms, which makes it difficult to establish a match between the question and answer candidate. This problem is particularly acute in the case of story comprehension due to the rarity of information restatement in the single document.

Several recent systems have specifically addressed the task of story comprehension. The Deep Read reading comprehension system [Hirschman *et al.*, 1999] uses a statistical bag-of-words approach, matching the question with the lexically most similar sentence in the story. Quarc [Riloff and Thelen, 2000] utilizes manually generated rules that selects a sentence deemed to contain the answer based on a combination of syntactic similarity and semantic correspondence (i.e., semantic categories of nouns). The Brown University statistical language processing class project systems [Charniak, *et al.*, 2000] combine the use of manually generated rules with statistical techniques such as bag-of-words and bag-of-verb matching, as well as deeper semantic analysis of nouns. As a rule, these three systems are effective at identifying the sentence containing the correct answer as long as the answer is explicit and contained entirely in that sentence. They find it difficult, however, to deal with semantic alternations of even moderate complexity. They also do not address situations where answers are split across multiple sentences, or those requiring complex inference.

Our framework, called QABLe (Question-Answering Behavior Learner), draws on prior work in learning action and problem-solving strategies [Tadepalli and Natarajan, 1996; Khardon, 1999]. We represent textual sources as sets of features in a sparse domain, and treat the QA task as behavior in a stochastic, partially observable world. QA strategies are learned as sequences of transformation rules capable of deriving certain types of answers from particular text-question combinations. The transformation rules are generated by instantiating primitive domain operators in specific feature contexts. A process of reinforcement learning [Kaebling, *et al.*, 1996] is used to select and promote effective transformation rules. We rely on recent work in attribute-efficient relational learning [Khardon *et al.*, 1999; Cumby and Roth, 2000; Even-Zohar and Roth, 2000] to acquire natural representations of the underlying domain features. These representations are learned in the course of interacting with the domain, and encode the features at the levels of abstraction that are found to be conducive to successful behavior. This selection effect is

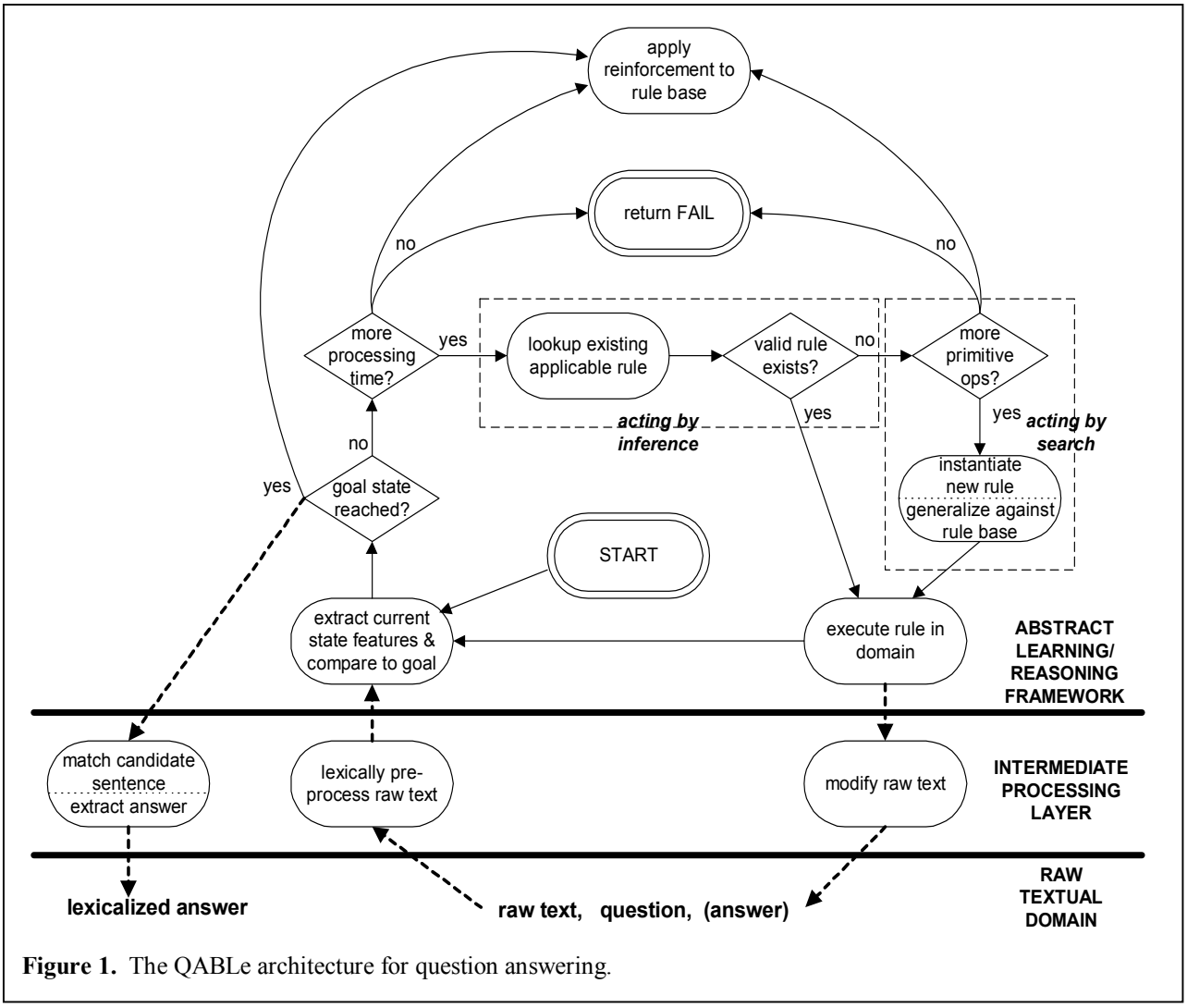


Figure 1. The QABLe architecture for question answering.

achieved by a fusion of abstraction space generalization [Sacerdoti, 1974; Knoblock, 1992] and reinforcement learning elements.

The rest of this paper is organized as follows. Section 2 presents the details of the QABLe framework. In section 3 we describe preliminary experimental results which indicate promise for our approach. In section 4 we summarize and draw conclusions.

2 QABLe – Learning to Answer Questions

2.1 Overview

Figure 1 shows a diagram of the QABLe framework. The bottom-most layer is the natural language textual domain. It represents raw textual sources, questions, and answers. The intermediate layer consists of processing modules that translate between the raw textual domain and the top-most layer, an abstract representation used to reason and learn.

This framework is used both for learning to answer questions and for the actual QA task. While learning, the system is provided with a set of training instances, each consisting of a textual narrative, a question, and a corresponding answer. During the performance phase, only the narrative and question are given.

At the lexical level, an answer to a question is generated by applying a series of *transformation rules* to the text of the narrative. These transformation rules augment the original text with one or more additional sentences, such that one of these explicitly contains the answer, *and* matches the form of the question.

On the abstract level, this is essentially a process of searching for a path through problem space that transforms the world state, as described by the textual source and question, into a world state containing an appropriate answer. This process is made efficient by learning answer-generation strategies. These strategies store procedural knowledge regarding the way in which answers are derived from text, and suggest appropriate transformation rules at each step in the answer-generation process. Strategies (and the procedural knowledge stored therein) are acquired by

explaining (or deducing) correct answers from training examples. The framework's ability to answer questions is tested only with respect to the kinds of documents it has seen during training, the kinds of questions it has practiced answering, and its interface to the world (domain sensors and operators).

In the next two sections we discuss lexical pre-processing, and the representation of features and relations over them in the QABLe framework. In section 2.4 we look at the structure of transformation rules and describe how they are instantiated. In section 2.5, we build on this information and describe details of how strategies are learned and utilized to generate answers. In section 2.6 we explain how candidate answers are matched to the question, and extracted.

2.2 Lexical Pre-Processing

Several levels of syntactic and semantic processing are required in order to generate structures that facilitate higher order analysis. We currently use MontyTagger 1.2, an off-the-shelf POS tagger based on [Brill, 1995], for POS tagging. At the next tier, we utilize a Named Entity (NE) tagger for proper nouns a semantic category classifier for nouns and noun phrases, and a co-reference resolver (that is limited to pronominal anaphora). Our taxonomy of semantic categories is derived from the list of unique beginners for WordNet nouns [Fellbaum, 1998]. We also have a parallel stage that identifies phrase types. Table 1 gives a list of phrase types currently in use, together with the categories of questions each phrase type can answer. In the near future, we plan to utilize a link parser to boost phrase-type tagging accuracy. For questions, we have a classifier that identifies the semantic category of information requested by the question. Currently, this taxonomy is identical to that of semantic categories. However, in the future, it may be expanded to accommodate a wider range of queries. A separate module reformulates questions into statement form for later matching with answer-containing phrases.

2.3 Representing the Question-Answering Domain

In this section we explain how features are extracted from raw textual input and tags which are generated by pre-processing modules.

Instantiate Rule
Given:
- set of primitive operators
- current state specification
- goal specification

1. select primitive operator to instantiate
2. bind active state variables & goal spec to existentially quantified condition variables
3. execute action in domain
4. update expected effect of new rule according to change in state variable values

Figure 2. Procedure for instantiating transformation

Phrase Type	Comments
SUBJ	"Who" and nominal "What" questions
VERB	event "What" questions
DIR-OBJ	"Who" and nominal "What" questions
INDIR-OBJ	"Who" and nominal "What" questions
ELAB-SUBJ	descriptive "What" questions (eg. what kind)
ELAB-VERB-TIME	
ELAB-VERB-PLACE	
ELAB-VERB-MANNER	
ELAB-VERB-CAUSE	"Why" question
ELAB-VERB-INTENTION	"Why" as well as "What for" question
ELAB-VERB-OTHER	smooth handling of undefined verb phrase types
ELAB-DIR-OBJ	descriptive "What" questions (eg. what kind)
ELAB-INDIR-OBJ	descriptive "What" questions (eg. what kind)
VERB-COMPL	WHERE/WHEN/HOW questions concerning state or status

Table 1. Phrase types used by QABLe framework.

A lexical sentence is represented as a sequence of words $\langle w_1, w_2, \ldots, w_n \rangle$, where *word*($w_i$, word) binds a particular word to its position in the sentence. The k^{th} sentence in a passage is given a unique designation s_k. Several simple functions capture the syntax of the sentence. The sentence *Main* (e.g., main verb) is the controlling element of the sentence, and is recognized by *main*(w_m, s_k). Parts of speech are recognized by the function *pos*, as in *pos*(w_i, NN) and *pos*(w_i, VBD). The relative syntactic ordering of words is captured by the function *before*(w_i, w_j). It can be applied recursively, as *before*(w_i, *before*(w_j, w_k)) to generate the entire sentence starting with an arbitrary word, usually the sentence *Main*. Thus for each word w_i in the sentence, $inSentence(w_i, s_i) \Rightarrow main(w_m, s_k) \wedge (before(w_i, w_m) \vee before(w_m, w_i))$. A consecutive sequence of words is a *phrase entity* or simply *entity*. It is given the designation e_x and declared by a binding function, such as *entity*(e_x, *NE*) for a named entity, and *entity*(e_x, *NP*) for a syntactic group of type *noun phrase*. Each phrase entity is identified by its *head*, as *head*(w_h, e_x), and we say that the phrase head controls the entity. A phrase entity is defined as $head(w_h, e_x) \wedge inPhrase(w_i, e_x) \wedge \ldots \wedge inPhrase(w_j, e_x)$.

We also wish to represent higher-order relations such as functional roles and semantic categories. Functional dependency between pairs of words is encoded as, for example, *subj*(w_i, w_j) and *aux*(w_j, w_k). Functional groups are represented just like phrase entities. Each is assigned a designation r_x, declared for example, as *func_role*(r_x, *SUBJ*), and defined in terms of its head and members (which may be individual words or composite entities). Semantic

categories are similarly defined over the set of words and syntactic phrase entities – for example, *sem_cat*(c_x, *PERSON*) ∧ *head*(w_h, c_x) ∧ *pos*(w_h, NNP) ∧ *word*(w_h, "John").

Semantically, sentences are treated as events defined by their verbs. A multi-sentential passage is represented by tying the member sentences together with relations over their verbs. We declare two such relations – *seq* and *cause*. The *seq* relation between two sentences, *seq*(s_i, s_j) ⇒ *prior*(*main*(s_i), *main*(s_j)), is defined as the sequential ordering in time of the corresponding events. The *cause* relation *cause*(s_i, s_j) ⇒ *cdep*(*main*(s_i), *main*(s_j)) is defined such that the second event is causally dependent on the first.

2.4 Primitive Operators and Transformation Rules

The system, in general, starts out with no procedural knowledge of the domain (*i.e.*, no transformation rules). However, it is equipped with 9 primitive operators that define basic actions in the domain. Primitive operators are existentially quantified. They have no activation condition, but only an *existence condition* – the minimal binding condition for the operator to be applicable in a given state. A primitive operator has the form $C^E \rightarrow \hat{A}$, where C^E is the existence condition and $\hat{A}$ is an action implemented in the domain. An example primitive operator is

primitive-op-1 : $\exists\, w_x, w_y \rightarrow$ add-word-after-word(w_y, w_x)

Other primitive operators delete words or manipulate entire phrases. Figure 3 lists all nine primitive operators. Note that primitive operators act directly on the syntax of the domain. In particular, they manipulate words and phrases. A primitive operator bound to a state in the domain constitutes a transformation rule. The procedure for instantiating transformation rules using primitive operators is given in Figure 2. The result of this procedure is a universally quantified rule having the form $C \wedge G^R \rightarrow A$. A may represent either the name of an action in the world or an internal predicate. C represents the necessary condition for rule activation in the form of a conjunction over the *relevant* attributes of the world state. G^R represents the expected effect of the action. G^R is inductively acquired from prior applications of the rule. For example,

$$before(w_1, w_2) \wedge before(w_2, w_3) \wedge G^R(before(w_3, w_4)) \rightarrow add-word-after-word(w_4, w_3)$$

indicates that when the phrase "w_1 w_2 w_3" is found in the text, this operator is expected to attach w_4 to the end, generating the phrase "w_1 w_2 w_3 w_4". For the rule to be effective in a given state, G^R must match all or part of the system's goal specification in that state.

1. $\exists\, w_x, w_y \rightarrow$ add-word-after-word(w_y, w_x)
2. $\exists\, w_x, w_y \rightarrow$ add-word-before-word(w_y, w_x)
3. $\exists\, w_x \rightarrow$ delete-word(w_x)
4. $\exists\, w_x, p_y \rightarrow$ add-word-after-phrase(p_y, w_x)
5. $\exists\, w_x, p_y \rightarrow$ add-word-before-phrase(p_y, w_x)
6. $\exists\, p_x, w_y \rightarrow$ add-phrase-after-word(w_y, p_x)
7. $\exists\, p_x, w_y \rightarrow$ add-phrase-before-word(w_y, p_x)
8. $\exists\, p_x, p_y \rightarrow$ add-phrase-after-phrase(p_y, p_x)
9. $\exists\, p_x, p_y \rightarrow$ add-phrase-before-phrase(p_y, p_x)

Figure 3. Primitive operators used to instantiate transformation rules.

An instantiated rule is assigned a *rank* composed of:

- priority rating (p)
- level of experience with rule (f)
- confidence in current parameter bindings (c)

The first component, priority rating, is an inductively acquired measure of the rule's performance on previous instances. The second component modulates the priority rating with respects to a frequency of use measure. The third component captures any uncertainty inherent in the underlying features serving as parameters to the rule. The rank of a rule is computed by the following function:

$$rank = p \times c \times (1 + \log(1 + f))$$

Each time a new rule is added to the rule base, an attempt is made to combine it with similar existing rules to produce more general rules having a wider relevance and applicability.

Given a rule $c_a \wedge c_b \wedge g_x^R \wedge g_y^R \rightarrow A_1$ covering a set of example instances E_1 and another rule $c_b \wedge c_c \wedge g_y^R \wedge g_z^R \rightarrow A_2$ covering a set of examples E_2, we add a more general rule $c_b \wedge g_y^R \rightarrow A_3$ to the strategy. The new rule A_3 is consistent with E_1 and E_2. In addition it will bind to any state where the literal c_b is active. Therefore the hypothesis represented by the triggering condition is likely an overgeneralization of the target concept. This means that rule A_3 may bind in some states erroneously. However, since all rules that can bind in a state compete to fire in that state, if there is a better rule, then A_3 will be preempted and will not fire.

2.5 Generating Answers

Returning to Figure 1, we note that at the abstract level the process of answer generation begins with the extraction of features active in the current state. These features represent low-level textual attributes and the relations over them described in section 2.3.

Immediately upon reading the current state, the system checks to see if this is a *goal state*. A goal state is a state whose corresponding textual domain representation contains

an explicit answer in the right form to match the questions. In the abstract representation, we say that in this state all of the goal constraints are satisfied.

If the current state is indeed a goal state, no further inference is required. The inference process terminates and the actual answer is identified by the matching technique described in section 2.6 and extracted.

If the current state is not a goal state and more processing time is available, QABLe passes the state to the Inference Engine (IE). This module stores strategies in the form of decision lists of rules. For a given state, each strategy may recommend at most one rule to execute. For each strategy this is the first rule in its decision list to fire. The IE selects the rule among these with the highest relative rank, and recommends it as the next transformation rule to be applied to the current state.

If a valid rule exists it is executed in the domain. This modifies the concrete textual layer. At this point, the pre-processing and feature extraction stages are invoked, a new current state is produced, and the inference cycle begins anew.

If a valid rule cannot be recommend by the IE, QABLe passes the current state to the Search Engine (SE). The SE uses the current state and its set of primitive operators to instantiate a new rule, as described in section 2.4. This rule is then executed in the domain, and another iteration of the process begins.

If no more primitive operators remain to be applied to the current state, the SE cannot instantiate a new rule. At this point, search for the goal state cannot proceed, processing terminates, and QABLe returns failure.

When the system is in the training phase and the SE instantiates a new rule, that rule is generalized against the existing rule base. This procedure attempts to create more general rules that can be applied to unseen example instances.

Once the inference/search process terminates (successfully or not), a reinforcement learning algorithm is applied to the entire rule search-inference tree. Specifically, rules on the solution path receive positive reward, and rules that fired, but are not on the solution path receive negative reinforcement.

2.6 Candidate Answer Matching and Extraction

As discussed in the previous section, when a goal state is generated in the abstract representation, this corresponds to a textual domain representation that contains an explicit answer in the right form to match the questions. Such a candidate answer may be present in the original text, or may be generated by the inference/search process. In either case, the answer-containing sentence must be found, and the actual answer extracted. This is accomplished by the Answer Matching and Extraction procedure.

The first step in this procedure is to reformulate the question into a statement form. This results in a sentence containing an empty slot for the information being queried. For example, "How far is the drive to Chicago?" becomes "The drive to Chicago is _____." Recall further that QABLe's pre-processing stage analyzes text with respect to various syntactic and semantic types. In addition to supporting abstract feature generation, these tags can be used to analyze text on a lexical level. Thus, the question above is marked up as [ELAB-VERB <quantity-distance> (WRB How) (RB far)] [VERB (VBZ is)] [SUBJ <action> (DT the) (NN drive)] [VERB-COMPL (TO to) (NNP <place> Chicago)]. Once reformulated into statement form, this becomes [ELAB-VERB <quantity-distance> _____] [VERB (VBZ is)] [SUBJ <action> (DT the) (NN drive)] [VERB-COMPL (TO to) (NNP <place> Chicago)]. The goal now is to find a sentence whose syntactic and semantic analysis matches that of the reformulated question's as closely as possible. Thus, for example the text may contain the sentence "The drive to Chicago is 2 hours" with the corresponding analysis [SUBJ <action> (DT the) (NN drive)] [VERB-COMPL (TO to) (NNP <place> Chicago)] [VERB (VBZ is)] [VERB-COMPL <quantity-time> (CD 2) (NNS hours)]. Notice that all of the elements of this candidate answer match the corresponding elements of the question, with the exception of the semantic category of the ELAB-VERB phrase. This is likely not the answer we are seeking. The text may contain a second sentence "The drive to Chicago is 130 miles", that analyses as [SUBJ <action> (DT the) (NN drive)] [VERB-COMPL (TO to) (NNP <place> Chicago)] [VERB (VBZ is)] [VERB-COMPL <quantity-distance> (CD 130) (NNS miles)]. In this case, all of the elements match their counterparts in the reformulated question. Thus, the second sentence can be matched as the correct answer with high confidence.

3 Experimental Evaluation

3.1 Experimental Setup

We evaluate our approach to open-domain natural language question answering on the Remedia corpus. This is a collection of 115 children's stories provided by Remedia Publications for reading comprehension. The comprehension of each story is tested by answering five *who*, *what*, *where*, and *why* questions.

The Remedia Corpus was initially used to evaluate the Deep Read reading comprehension system [Hirschman *et al.*, 1999], and later also other systems, including Quarc [Riloff and Thelen, 2000] and the Brown University statistical language processing class project [Charniak, *et al.*, 2000].

The corpus includes two answer keys. The first answer key contains annotations indicating the story sentence that is lexically closest to the answer found in the published answer key (AutSent). The second answer key contains sentences that a human judged to best answer each question (HumSent). Examination of the two keys shows the latter to be more reliable. We trained and tested using the HumSent answers. We also compare our results to the HumSent results of prior systems. In the Remedia corpus, approximately 10% of the questions lack an answer. Following prior work, only questions with annotated answers were considered.

System	who	what	when	where	why	Overall
Deep Read	48%	38%	37%	39%	21%	36%
Quarc	41%	28%	55%	47%	28%	40%
Brown	57%	32%	32%	50%	22%	41%
QABLe-N/L	48%	35%	52%	43%	28%	41%
QABLe-L	56%	41%	56%	45%	35%	47%
QABLe-L+	59%	43%	56%	46%	36%	48%

Table 2. Comparison of QA accuracy by question type.

System	total # rules learned	total # rules on all solution paths	average # rules on solution path per correct answer
QABLe-L	3,463	426	3.02
QABLe-L+	16,681	411	2.85

Table 3. Analysis of transformation rule learning and use.

We divided the Remedia corpus into a set of 55 tests used for development, and 60 tests used to evaluate our model, employing the same partition scheme as followed by the prior work mentioned above. With five questions being supplied with each test, this breakdown provided 275 example instances for training, and 300 example instances to test with. However, due to the heavy reliance of our model on learning, many more training examples were necessary. We widened the training set by adding story-question-answer sets obtained from several online sources[1]. All of these additional stories are supplied with questions, but few have answers. To complete the training set, a human generated reasonable answers to the questions. With the extended corpus, QABLe was trained on 262 stories with 3-5 questions each, corresponding to 1000 example instances.

3.2 Discussion of Results

Table 2 compares the performance of different versions of QABLe with those reported by the three systems described above. We wish to discern the particular contribution of transformation rule learning in the QABLe model, as well as the value of expanding the training set. Thus, the QABLe-N/L results indicate the accuracy of answers returned by the QA matching and extraction algorithm described in section 2.6 only. This algorithm is similar to prior answer extraction techniques, and provides a baseline for our experiments. The QABLe-L results include answers returned by the full QABLe framework, including the utilization of learned transformation rules, but trained only on the limited training portion of the Remedia corpus. The QABLe-L+ results are for the version trained on the expanded training set.

As expected, the accuracy of QABLe-N/L is comparable to those of the earlier systems. The Remedia-only training set version, QABLe-L, shows an improvement over both the baseline QABLe, and most of the prior system results. This is due to its expanded ability to deal with semantic alternations in the narrative by finding and learning transformation rules that reformulate the alternations into a lexical form matching that of the question.

The results of QABLe-L+, trained on the expanded training set, are for the most part noticeably better than those of QABLe-L. This is because training on more example instances leads to wider domain coverage through the acquisition of more transformation rules. Table 3 gives a break-down of rule learning and use for the two learning versions of QABLe. The first column is the total number of rules learned by each system version. The second column is the number of rules that ended up being successfully used in generating an answer. The third column gives the average number of rules each system needed to answer an answer (where a correct answer was generated). Note that QABLe-L+ used fewer rules on average to generate more correct answers than QABLe-L. This is because QABLe-L+ had more opportunities to refine its policy controlling rule firing through reinforcement and generalization.

Note that the learning versions of QABLe do significantly better than the QABLe-N/L and all the prior systems on *why*-type questions. This is because many of these questions require an inference step, or the combination of information spanning multiple sentences. QABLe-L and QABLe-L+ are able to successfully learn transformation rules to deal with a subset of these cases.

4 Conclusion

In this paper we present an approach to automatically learn strategies for natural language questions answering from examples composed of textual sources, questions, and corresponding answers. The strategies thus acquired are composed of ranked lists transformation rules that when

[1] See for example, http://www.edhelper.com/ReadingComprehension.htm and http://www.miguelmllop.com/practice/intermediate/readingcomprehension/readingcompindex.htm

applied to an initial state consisting of an unseen text and question, can derive the required answer. We compare our approach to three prior systems, and present experimental results demonstrating the efficacy of our model. In particular, we show that our approach is effective given a sufficiently large training corpus, and reasonably accurate pre-processing of raw input data.

References

[Brill, 1995] E. Brill. Transformation-based error driven learning and natural language processing: A case study in part of speech tagging. In *Computational Linguistics*, 21(4):543-565, 1995.

[Charniak, *et al.*, 2000] Charniak, Y. Altun, R. de Salvo Braz, B. Garrett, M. Kosmala, T. Moscovich, L. Pang, C. Pyo, Y. Sun, W. Wy, Z. Yang, S. Zeller, and L. Zorn. Reading comprehension programs in a statistical-language-processing class. *ANLP/NAACL-00*, 2000.

[Cumby and Roth, 2000] C. Cumby and D. Roth. Relational representations that facilitate learning. *KR-00*, pp. 425-434, 2000.

[Even-Zohar and Roth, 2000] Y. Even-Zohar and D. Roth. A classification approach to word prediction. *NAACL-00*, pp. 124-131, 2000.

[Fellbaum, 1998] C. Fellbaum (ed.) *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[Hirschman and Gaizauskas, 2001] L. Hirschman and R. Gaizauskas. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275-300, 2001.

[Hirschman *et al.*, 1999] L. Hirschman, M. Light, and J. Burger. Deep Read: A reading comprehension system. *ACL-99*, 1999.

[Kaebling, *et al.*, 1996] L. P. Kaebling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *J. Artif. Intel. Research*, 4:237-285, 1996.

[Khardon *et al.*, 1999] R. Khardon, D. Roth, and L. G. Valiant. Relational learning for nlp using linear threshold elements, *IJCAI-99*, 1999.

[Khardon, 1999] R. Khardon. Learning to take action. *Machine Learning* 35(1), 1999.

[Knoblock, 1992] C. Knoblock. Automatic generation of abstraction for planning. *Artificial Intelligence*, 68(2):243-302, 1992.

[Riloff and Thelen, 2000] E. Riloff and M. Thelen. A rule-based question answering system for reading comprehension tests. *ANLP/NAACL-2000*, 2000.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115-135, 1974.

[Tadepalli and Natarajan, 1996] P. Tadepalli and B. Natarajan. A formal framework for speedup learning from problems and solutions. *J. Artif. Intel. Research*, 4:445-475, 1996.

[Voorhees, 2003] E. M. Voorhees Overview of the TREC 2003 question answering track. *TREC-12*, 2003.

Shallow Semantics for Relation Extraction

Sanda Harabagiu, Cosmin Adrian Bejan and **Paul Morărescu**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688, USA
sanda,ady,paul@hlt.utdallas.edu

Abstract

This paper presents a new method for extracting meaningful relations from unstructured natural language sources. The method is based on information made available by shallow semantic parsers. Semantic information was used (1) to enhance a dependency tree kernel; and (2) to build semantic dependency structures used for enhanced relation extraction for several semantic classifiers. In our experiments the quality of the extracted relations surpassed the results of kernel-based models employing only semantic class information.

1 Introduction

With the advent of the Internet, more and more information is available electronically. Most of the time, information on the Internet is unstructured, generated in textual form. One way of automatically identifying information of interest from the vast Internet resources is by recognizing relevant entities and meaningful relations they share. Examples of entities are PERSONS, ORGANIZATIONS and LOCATIONS. Examples of relations are ROLE, NEAR and SOCIAL.

In the 90s, the Message Understanding Conferences (MUCs) and the TIPSTER programs gave great impetus to research in information extraction (IE). The systems that participated in the MUCs have been quite successful at recognizing relevant entities, reaching near-human precision with over 90% accuracy. More recently, the Automatic Content Extraction (ACE) program focused on identifying not only relevant entities, but also meaningful relations between them. If success in recognizing entities with high precision was attributed to the usage of finite-state transducers, in the last three years the dominant successful technique for extracting relations was based on kernel methods.

Kernel methods are non-parametric density estimation techniques that compute a *kernel function* to measure the similarity between data instances. [8] introduced the formalization of relation extraction in terms of tree kernels, which are kernels that take advantage of syntactic trees. [1] extended the work by using dependency trees that describe the grammatical relations between the words of a sentence. Furthermore, each word of a dependency tree is augmented with lexico-semantic information, including semantic information available from the WordNet database [2]. The average precision of relation extraction using dependency trees was reported in [1] to be 67.5%. Since meaningful relations between relevant entities are of semantic nature, we argue that additional semantic resources should be used for extracting relations from texts. In this work, we were interested in investigating the contribution of two shallow semantic parsing techniques to the quality of relation extraction.

We explored two main resources: *PropBank* and *FrameNet*. Proposition Bank or PropBank is a one million word corpus annotated with predicate-argument structures. The corpus consists of the Penn Treebank 2 Wall Street Journal texts (*www.cis.upenn.edu/∼treebank*). The PropBank annotations were performed at University of Pennsylvania (*www.cis.upenn.edu/∼ace*). To date PropBank has addressed only predicates lexicalized by verbs, proceeding from the most to the least common verbs while annotating verb predicates in the corpus. The FrameNet project (*www.icsi.berkeley.edu/∼framenet*) produced a lexico-semantic resource encoding a set of *frames*, which represent schematic representations of situations characterized by a set of *target words*, or lexicalized predicates, which can be verbs, nouns or adjectives. In each frame, various participants and conceptual roles are related by case-roles or theta-roles which are called *frame elements* or FEs. FEs are local to each frame, some are quite general while others are specific to a small family of lexical items. FrameNet annotations were performed on a corpus of over three million words. Recently, semantic parsers using PropBank and FrameNet have started to become available. In each sentence, verbal or nominal predicates are discovered in relation to their arguments or FEs. Our investigation shows that predicate-arguments structures and semantic frames discovered by shallow semantic parsers play an important role in discovering extraction relations. This is due to the fact that arguments of extracted relations belong to arguments of predicates or to FEs.

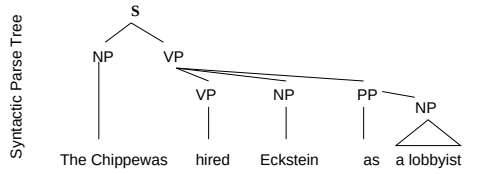
To investigate the role of semantic information for relation extraction we have used two shallow semantic parsers, one trained on PropBank and one on FrameNet. We used the semantic information identified by the parsers in two ways. First it was used to enhance the features of dependency kernels. Second, it was used to generate a new representation, called semantic dependency structure. The results of the ex-

periments indicate that not only the precision of kernel methods is improved, but also relation extraction based on shallow semantics outperforms kernel-based methods.

The remainder of this paper is organized as follows. Section 2 describes the semantic parsers. Section 3 details the kernel methods and their enhancement with semantic information. In Section 4 we show how relation extraction based on semantic information is produced. Section 5 details the experimental results and Section 6 summarizes our conclusions.

2 Shallow Semantic Parsing

Shallow semantic information represented by predicates and their arguments, or frames and their FEs, can be identified in text sentences by semantic parsers. The idea of automatically identifying and labeling shallow semantic information was pioneered by [3]. Semantic parsers operate on the output of a syntactic parser. When using the PropBank information, the semantic parser (1) identifies each verbal predicate and (2) labels its arguments. The expected arguments of a predicate are numbered sequentially from Arg0 to Arg5. Additionally, the arguments may include functional tags from Treebank, e.g. ArgM-DIR indicates a directional, ArgM-LOC indicates a locative and ArgM-TMP stands for a temporal. When using the FrameNet information, the semantic parser (1) identifies the target words; (2) disambiguates the semantic frame for each target word; and (3) labels the FEs that relate to the target word based on the frame definition. For example, in FrameNet the EMPLOYMENT frame has the FEs: Employee, Employer, Purpose, Compensation, Manner, Place, Position, Time, Field, Duration and Task.



Semantic Parser (PropBank)	**Task 1:**	[The Chippewas]	hired	[Eckstein]	[as a lobbyist]
	Task 2:	*Arg0*	*PREDICATE*	*Arg1*	*ArgM*
Semantic Parser (FrameNet)	**Task 1:**	[The Chippewas]	hired	[Eckstein]	[as a lobbyist]
	Task 2:	*Employer*	*TARGET*	*Employee*	*Position*

Figure 1: Sentence with labeled semantic roles.

Figure 1 illustrates the output of both semantic parsers when processing a sentence. The parsing consists of two tasks: (1) identifying the parse tree constituents corresponding to arguments of each predicate or FEs of each frame; and (2) recognizing the role corresponding to each argument or FE. Each task is cast as a separate classifier. For example, task 1 identifies the two NPs and the PP as arguments and FEs respectively. The second classifier assigns the specific roles: Arg0, Arg1 and ArgM given the predicate "hired" and FEs *Employer*, *Employee* and *Position* given the target word "hired". In the case of semantic parsing parsed on FrameNet, a third task of finding the frame corresponding to the sentence is cast as a third classification problem.

− PHRASE TYPE (pt): This feature indicates the syntactic type of the phrase labeled as a frame element, e.g. NP for Employer in Figure 1.
− PARSE TREE PATH (path): This feature contains the path in the parse tree between the predicate phrase and the target word, expressed as a sequence of nonterminal labels linked by direction symbols (up or down), e.g. NP ↑S ↓VP ↓VP for Employer in Figure 1.
− POSITION (pos) − Indicates if the constituent appears before or after the predicate in the sentence.
− VOICE (voice) − This feature distinguishes between active or passive voice for the predicate phrase.
− HEAD WORD (hw) − This feature contains the head word of the evaluated phrase. Case and morphological information are preserved.
− GOVERNING CATEGORY (gov) − This feature applies to noun phrases only, and it indicates if the NP is dominated by a sentence phrase (typical for subject arguments with active−voice predicates), or by a verb phrase (typical for object arguments).
− TARGET WORD − In our implementation this feature consists of two components: (1) WORD: the word itself with the case and morphological information preserved; and (2) LEMMA which represents the target normalized to lower case and infinitive form for the verbs or singular for nouns.

Figure 2: Feature Set 1 (FS1).

− CONTENT WORD (cw) − Lexicalized feature that selects an informative word from the constituent, different from the head word.
PART OF SPEECH OF HEAD WORD (hPos) − The part of speech tag of the head word.
PART OF SPEECH OF CONTENT WORD (cPos) −The part of speech tag of the content word.
NAMED ENTITY CLASS OF CONTENT WORD (cNE) − The class of the named entity that includes the content word
BOOLEAN NAMED ENTITY FLAGS − A feature set comprising: − neOrganization: set to 1 if an organization is recognized in the phrase − neLocation: set to 1 a location is recognized in the phrase − nePerson: set to 1 if a person name is recognized in the phrase − neMoney: set to 1 if a currency expression is recognized in the phrase − nePercent: set to 1 if a percentage expression is recognized in the phrase − neTime: set to 1 if a time of day expression is recognized in the phrase − neDate: set to 1 if a date temporal expression is recognized in the phrase

Figure 3: Feature Set 2 (FS2).

Several classifiers were previously tried for this problem: decision trees [7], and Support Vector Machines (SVMs) [6]. Based on the results of the CoNLL and Senseval-3 evaluations[1], we selected an implementation based on SVMs, using the SVMlight package [2]. For the semantic parser based on PropBank, we combined feature sets: FS1 (illustrated in Figure 2) introduced in [3], FS2 (illustrated in Figure 3) introduced in [7], FS3 (illustrated in Figure 4) introduced in [6]. For the semantic parser based on FrameNet we have also added the feature set FS4 (illustrated in Figure 5). The se-

[1]The Conference on Natural Language Learning (CoNLL) (http://cnts.uia.ac.be/signll/shared.html) has focused in 2004 on the problem of semantic role labeling, using PropBank data. Senseval-3 (www.senseval.org/senseval3) had a semantic role labeling task, using FrameNet data.

[2]http://svmlight.joachims.org/

mantic parser using FrameNet also required the disambiguation of the frame. To disambiguate the frame, we used the BayesNet algorithm implemented in the Weka learning package [3]. The features that were used are (1) the target word; (2) the part-of-speech of the target word; (3) the phrase type of all the FEs (e.g. NP, VP, PP); and (4) the *grammatical function*. The grammatical function is defined in Figure 4. To learn the grammatical function, we again used the SVM with the features FS1, FS2, the features `Human` and `Target-Type` from FS3 and only the feature `PP-PREP` from FS4.

HUMAN: This feature indicates whether the syntactic phrase is either (1) a personal pronoun or (2) a hyponym of sense 1 of PERSON in WordNet
SUPPORT_VERBS that are recognized for adjective or noun target words have the role of predicate for the FEs. For example, if the target = clever, in the sentence "Smith is very clever, but he's no Einstein", the FE = Smith is an argument of the support verb 'is' rather than of the target word. The values of this feature are either (1) The POS of the head of the VP containing the target word or (2) NULL if the target word does not belong to a VP
TARGET−TYPE: the lexical class of the target word, e.g. VERB, NOUN or ADJECTIVE
LIST_CONSTITUENT (FEs): This feature represents a list of the syntactic consituents covering each FE of the frame recognized in a sentence. For the example illustrated in Figure 1, the list is: [NP, NP, PP]
Grammatical Function: This feature indicates whether the FE is: − an External Argument (Ext) − an Object (Obj) − a Complement (Comp) − a Modifier (Mod) − Head noun modified by attributive adjective (Head) − Genitive determiner (Gen) − Appositive (Appos)
LIST_Grammatical_Function: This feature represents a list of the grammatical functions of the FEs recognized in the sentence.
NUMBER_FEs: This feature indicates how many FEs were recognized in each sentence.
FRAME_NAME: This feature indicates the name of the semantic frame for which FEs are labeled
COVERAGE: This feature indicates whether there is a syntactic structure in the parse tree that perfectly covers the FE
CORE: This feature indicates whether the FE is one that instantiates a conceptually necessary participant of a frame. For example, in the REVENGE frame, Punishment is a core element. The values of this feature are: (1) core; (2) peripheral and (3) extrathemathic. FEs that mark notions such as Time, Place, Manner and Degree are peripheral. Extratematic FEs situate an event against a backdrop of another event, by evoking a larger frame for which the target event fills a role.
SUB_CORPUS: In FrameNet, sentences are annotated with the name of the subcorpus they belong to. For example, for a verb target word, V−swh represents a subcorpus in which the trget word is a predicate to a FE included in a relative clause headed by a wh−word.

Figure 4: Feature Set 3 (FS3).

3 Dependency Tree Kernels

In [1] the relation extraction problem was cast as a classification problem based on kernels that operate on dependency trees. Kernels measure the similarity between two instances of a relation. If $\mathbf{X}$ is the instance space, a kernel function is a mapping K:$\mathbf{X}\times\mathbf{X}\rightarrow[0,\infty)$ such that given two instances

[3] *http://www.cs.waikato.ac.nz/∼ml*

PARSE TREE PATH WITH UNIQUE DELIMITER − This feature removes the direction in the path, e.g. VBN−VP−ADVP
PARTIAL PATH − This feature uses only the path from the constituent to the lowest common ancestor of the predicate and the constituent
FIRST WORD − First word covered by constituent
FIRST POS − POS of first word covered by constituent
LAST WORD − Last word covered by the constituent
LAST POS − POS of last word covered by the constituent
LEFT CONSTITUENT − left sibling constituent label
LEFT HEAD − Left sibling head word
LEFT POS HEAD − Left sibling POS of head word
RIGHT CONSTITUENT − right sibling constituent label
RIGHT HEAD − Right sibling head word
RIGHT POS HEAD − Right sibling POS of head word
PP PREP − If constituent is labeled PP get first word in PP
DISTANCE − Distance in the tree from constituent to the target word

Figure 5: Feature Set 4 (FS4).

x and y, $K(x,y) = \sum_i \phi_i(x)\phi_i(y) = \phi(x)\cdot\phi(y)$, where $\phi_i(x)$ is some feature function over the instance x. The instances can be represented in several ways. First, each sentence where a relation of interest occurs can be viewed as a list of words. Thus, the similarity between two instances represented in this way is computed as the number of common words between the two instance sentences. All words from instances x and y are indexed and $\phi_i(x)$ is the number of times instance x contains the word referenced by i. Such a kernel is known as *bag-of-words kernel*. When sentences are represented as strings of words, *string kernels*, count the number of common subsequences in the two strings and weight their matches by their length. Thus $\phi_i(x)$ is the number of times string x contains the subsequence referenced by i. If the instances are represented by syntactic trees, more complex kernels are needed. A class of kernels, called *convolution kernels*, was proposed to handle such instance representations. Convolution kernels measure the similarity between two structured instances by summing the similarity of their substructures. Thus, given all possible substructures in instances x and y, $\phi_i(x)$ counts not only the number of times the substructure referenced by i is matched into x, but also how many times it is matched into any of its substructures.

Given a training set T=$\{x_1,\ldots x_N\}$, kernel methods compute the *Gram* matrix G such that G_{ij}=$K(x_i,x_j)$. G enables a classifier to find a hyperplane which separates instances of different classes. G enables classifiers to find a separating hyperplane that separates positive and negative examples. When a new instance y needs to be classified, y is projected into the feature space defined by the kernel function. Classification consists of determining on which side of the separating hyperplane y lies. *Support Vector Machines*(SVMs) formulate the task of finding the separating hyperplane as a solution to a quadratic programming problem.Therefore, following the solution proposed by [1], they are used for classifying relation instances in texts.

To measure the similarity between two instances of the

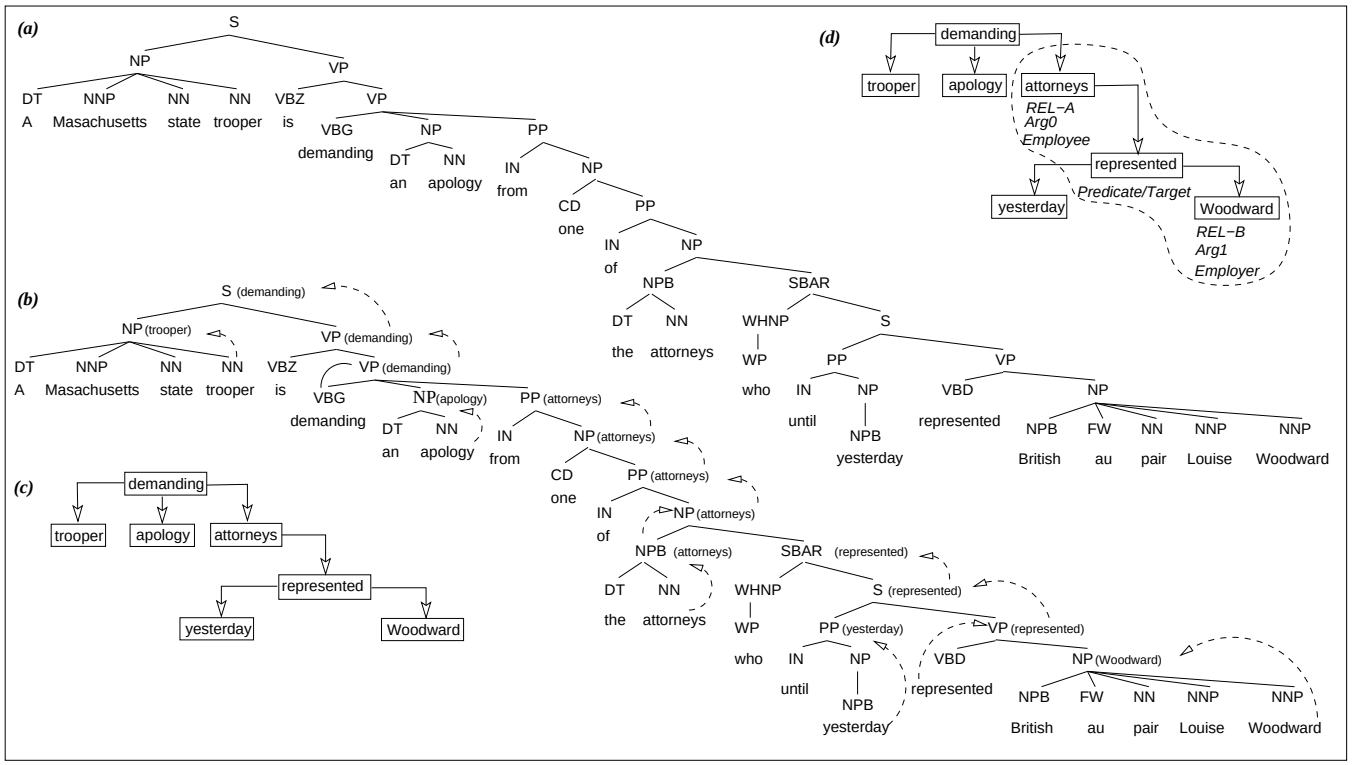


Figure 6: (a) Syntactic parse; (b) Head word propagation; (c) Dependency tree; (d) Semantic dependency tree.

same extraction relation both [1] and [8] relied on kernels that operate on trees expressing syntactic information. To build such a tree we have used the Collins syntactic parser. When using this parser, for each constituent in the parse tree, we also have access to a *dependency* model that enables the mapping of parse trees into sets of binary relations between the head-word of each component and its sibling words. For example, Figure 6(a) describes the parse tree of a sentence. For each possible constituent in the parse tree, rules first described in [4] identify the head-child and propagate the head word to the parent. Figure 6(b) illustrates the propagation for the parse tree illustrated in Figure 6(a). When the propagation is over, head-modifier relations are extracted, generating a dependency structure. Figure 6(c) illustrates the dependency structure of the sentence that was analyzed syntactically in Figure 6(a). The nodes of this dependency structure were augmented with features, to enable the calculation of the kernel. Figure 7 lists the features that were assigned to each node in the dependency tree. Two sets of features were used: F1, the features proposed in [1] and F2, a new set of features that we have added. Feature set F1 contains: (1) the word; (2) the part-of-speech (POS) (24 values); (3) a generalized POS (5 values); (4) the tag of the nonterminal from the parse tree having the feature word as a head; (5) the entity type, as defined by the ACE guidelines; (6) the entity level (e.g. name); (7) the relation argument (e.g. REL-A); and (8) a WordNet hypernym. The new set of features F2 contains: (1) the predicate argument number provided by the semantic parser when using PropBank; (2) the predicate target for that argument; (3) the FE provided by the semantic parser when using FrameNet; (4) the target word for the frame; (5) the grammatical function; (6) the Frame; (7) the WordNet domain; (8) the WordNet concept that expressed the type of relation in which the word may belong within the domain; (9) a set of other related WordNet concepts (e.g. direct hypernym); (10) the ProbBank concepts that most frequently occur as arguments for the predicate; (11) the FrameNet concepts that most frequently occur in FEs for the frame. The features and their values for the node *"attorneys"* are listed in Figure 7.

Feature Set F1	Example	Feature Set F2	Example
1 Word	attorneys	1 Predicate−argument number	Arg0
2 Part−of−speech	NN	2 Predicate Target	represent
3 General−POS	noun	3 FE	Employee
4 Syntactic chunk tag	NP	4 Target word	represent
5 Entity−type	PERSON	5 Gramatical function	Ext
6 Entity−level	nominal	6 Frame	Employment
7 Relation−argument	REL−A	7 WordNet Domain	jurisprudence
8 WordNet hypernym	Individual	8 WordNet Relation Concept	lawyer−client
		9 WordNet Semantic Concepts	professional,paralegal
		10 PropBank(WN) Concepts	banker, lawyer, share
		11 FrameNet(WN) Concepts	relation, men

Figure 7: Sets of features assigned to each node in the dependency tree.

The features are used by a tree kernel function $K(T_1, T_2)$ that returns a similarity score in the range $(0, 1)$. We preferred the more general version of the kernel introduced in [1] to the kernel described by [8]. This kernel is based on two functions defined on the features of tree nodes: a matching function $m(t_i, t_j) \in \{0, 1\}$ and a similarity function $s(t_i, t_j) \in (0, \infty)$. The feature vector of a tree node $\phi(t_i) = \{v_1, \ldots v_d\}$ consists of two possibly overlapping subsets $\phi_m(t_i) \subseteq \phi(t_i)$ and $\phi_s(t_i) \subseteq \phi(t_i)$. As in [1], $\phi_m(t_i)$

are used by the matching function and $\phi_s(t_i)$ are used by the similarity function. The two functions are defined by:

$$m(t_i, t_j) = \begin{cases} 1 & \text{if } \phi_m(t_i) = \phi_m(t_j) \\ 0 & \text{otherwise} \end{cases}$$

and

$$s(t_i, t_j) = \sum_{v_q \in \phi_s(t_i)} \sum_{v_r \in \phi_s(t_j)} C(v_q, v_r)$$

where $C(v_q, v_r)$ is some compatibility function between the two feature values. For example, in the simplest case where

$$C(v_q, v_r) = \begin{cases} 1 & \text{if } v_q = v_r \\ 0 & \text{otherwise} \end{cases}$$

$s(t_i, t_j)$ returns the number of feature values in common between the feature vectors $\phi_s(t_i)$ and $\phi_s(t_j)$. For two dependency trees T_1 and T_2 with root nodes r_1 and r_2 the tree kernel is defined as:

$$K(T_1, T_2) = \begin{cases} 0 & \text{if } m(r_1, r_2)\text{=0} \\ s(r_1, r_2) + K_c(r_1[c], r_2[c]) & \text{otherwise} \end{cases}$$

where K_c is a kernel function over the children of the nodes r_1 and r_2. Let **a** and **b** be sequences of indices such that **a** is a sequence $a_1 \leq a_2 \leq \ldots \leq a_n$ and likewise for **b**. Let $d(a) = a_n - a_1 + 1$ and $l(a)$ be the length of **a**. Then for every $t_i \in T_1$ and $t_j \in T_2$, we have

$$K_c(t_i[c], t_j[c]) = \sum_{a,b,l(a)=l(b)} \lambda^{d(a)} \lambda^{d(b)} K(t_i[a], t_j[b])$$

where $\lambda \in (0, 1)$ represents a decay factor that penalizes matching subsequences that are spread out within the child sequences. The definition of K_c, the kernel function over children, assumes that the matching function used in the definition of the tree kernel $K(t_i[a], t_j[b])$ operates not only on single nodes, but also on node sequences $t_i[a]$ or $t_j[b]$. If all the nodes in the sequence are matched, $m(t_i[a], t_j[b]) = 1$. For each matching pair of nodes (a_i, b_j) is a matching subsequence, we accumulate the result of the similarity function $s(a_i, b_j)$ and then recursively search for matching subsequences of *their* children.

As in [1], we implemented two types of tree kernels: a contiguous kernel and a sparse kernel. A *contiguous* kernel only matches children subsequences that are uninterrupted by non-matching nodes. Therefore $d(a) = l(a)$. A *sparse* tree kernel, by contrast, allows non-matching nodes within matching subsequences.

4 Relation Extraction

When analyzing the dependency kernels, we noticed that only few nodes bear semantic information derived by the semantic parsers. We also noticed that these nodes are clustered together in the dependency tree. For example, Figure 6(d) illustrates the cluster of nodes from the dependency tree that contains semantic information. Instead of using the entire dependency tree to compute similarities, we selected sub-trees that contain nodes having values for the features from set F2 (illustrated in Figure 7). Typically these nodes correspond to target predicates and their arguments or FEs. This allowed us to compare trees of the form $SDT(R_1)$ [“$attorneys$”→“$represented$”←“$Woodward$”] and $SDT(R_2)$[“$intern$”→“$dismissed$”←“$lawyer$”]. We called such trees *semantic dependency trees* since they are characterized by semantic features present in the nodes of dependency trees. Semantic dependency trees (SDTs) are binary trees containing three nodes: a verbal predicate that is the root of the tree; and two children nodes, each an argument of the predicate. To measure the similarity of two SDTs we built a very simple kernel:

$$K(T_1, T_2) = \begin{cases} 0 & \text{if } m(c(T_1), c(T_2))\text{=0} \\ S(r(T_1), r(T_2)) + S_p(T_1, T_2) & \text{otherwise} \end{cases}$$

where the matching function is performed only on the children. The matching features that were used comprised ϕ_m^1=$\{General\text{-}POS, Entity\text{-}Type, Relation\text{-}argument\}$, ϕ_m^2=$\{FE\}$ and ϕ_m^3=$\{Predicate\text{-}argument\ number\}$. $m(c(T_1), c(T_2))$=1 if there is a combination of the pair of arguments that has the same matching features. For example, in the case of $SDT(R_1)$ and $SDT(R_2)$, the combination is {(“$attorneys$”, “$lawyer$”) and (“$intern$”, “$Woodward$”)} when using $\phi_m^1 \cup \phi_m^2$, since both attorney and lawyer are nouns, Persons, REL-A and they are both covered by the FE *Employee* in their respective frames. To measure the similarity between the verbal predicates, the function $S(r(T_1), r(T_2))$ measures the semantic compatibility of the predicates. The features used for measuring similarity of predicates are ϕ_S^P=$\{Frame, Predicate\ Target, WordNet\ Domain\}$. The compatibility measures assigned to each feature are: 1 if the same predicate target, 0.9 if both predicates are targets of the same frame, 0.7 if both predicates are targets of frames from the same event structure, 0.5 if both predicates belong to the same WordNet domain and 0.3 if the predicates are not covered in FrameNet, but have the same arguments in PropBank as other predicates from the same frame. The predicate similarity brings forward semantic frames that characterize a type of extraction relation. For example, in the case of Role_Client relation, such frames were (a) Employment_end and its subframes from the event structure; Employment_Start; and (b) Commerce_Sell or (c) Commerce_Buy.

The $S_p(T_1, T_2)$ similarity focuses on the combination of FEs or predicate-arguments that are identified for the children of the SDTs. The similarity features that are used are {FE, Predicate-argument number, Grammatical Function, WordNet Domain, WordNet Relation Concept}. For example, when the similarity $S_p(SDT(R_1), SDT(R_2))$ is computed, since we have an $\{Employer\_Employee\}$ relation between the FEs in both SDTs, the confidence assigned based on identical FE-FE relation is 1. If we have identical Predicate-argument numbers, the confidence is 0.6. For identical WordNet domains, 0.4 and for the same WordNet relation concept, we assign the confidence 0.7.

One limitation of the SDTs stems from the fact that this formalism cannot capture extraction relations that are expressed in the same noun phrase, e.g. “our customers”, “his urologist” or “George’s high school”. To recognize such relations we consider that they relate to some arguments of “unspecified” predicates. In the case when NPs contain pronouns, they are resolved by a successful coreference resolution algorithm [5], and the pronoun is substituted by a pair (referent,

CATEGORY). For example, the pronoun "our" from "our customers" is replaced by ("IBM", ORGANIZATION)

By measuring the semantic similarity between the pair (NP_head, Category) and any pair of arguments of a predicate from the training corpus, a plausible predicate can be found. For the noun phrase "our customers", the $SDT[\text{"}shop\text{"} \leftarrow \text{"}billed\text{"} \rightarrow \text{"}customers\text{"}]$ was deemed the most relevant, assigning the predicate "billed" to the two arguments: ORGANIZATION and "customers". The semantic similarity was measured by enhancing the WordNet, similarity measure is publicly available resource (*www.d.umn.edu/∼tpederse/Pubs/AAAI04PedersenT.pdf*), to handle semantic classes identified for entities, e.g. ORGANIZATION, DISEASE.

5 Experimental results

To evaluate the role of shallow semantics provided by semantic parsers on relation extraction we have used the Automatic Content Extraction (ACE) corpus available from LDC (LDC2003T11). The data consists of 422 annotated text documents gathered from various newspapers and broadcasts. Five entity types have been annotated (PERSON, ORGANIZATION, GEO-POLITICAL ENTITY, LOCATION, FACILITY) along with the 24 types of relations. The relations are listed in Figure 8.

At_Located	At_Residence	Role_Affiliate	Role_Founder
Role_Staff	Near_Relative−loc	Role_Client	Soc_Associate
Role_Member	Role_Citizen	Role_Owner	Soc_Spouse
Role_Mgmt	Soc_Professional	Role_Other	Soc_Sibling
Part_Part−of	Part_Subsidiary	Soc_Relative	Part_Other
At_Based−in	Soc_Parent	Soc_Personal	Soc_Grandparent

Figure 8: Extraction relations evaluated in ACE.

We implemented the same five kernels as [1]: K_0=sparse kernel, K_1=contiguous kernel, K_2=bag-of-words kernel and K_3=K_0+K_2 and K_4=K_1+K_2 and used first only the feature set F1 from Figure 7 and then both feature sets F1 and F2. The comparison of the kernel performance of the two experiments is listed in Figure 9.

Kernel	Avg. Precision		Avg. Recall		Avg. F−score	
	Features		Features		Features	
	F1	F1+F2	F1	F1+F2	F1	F1+F2
K0	52.3	55.8	19.7	34.6	28.62	42.71
K1	55.1	61.4	20.4	32.3	29.78	42.33
K2	38.2	48.7	7.2	28.5	12.12	35.96
K3	54.8	61.3	18.5	40.7	27.66	48.92
K4	60.5	72.2	20.3	44.5	30.4	55.06

Figure 9: Kernel performance comparison.

We used each kernel within an SVM (we augmented the SVMlight implementation to include our kernels). We choose to train on all 24 relations, not only on the first 5-high level relation types as was done in [1]. The results indicate that on average, for K_4, the best performing kernel, we obtained an increase of 24.66% in the F-score when features provided by the semantic parsers were added. When relying on SDTs, the average Precision that was obtained was 89.3%, the recall was 76.4%, thus an F-score of 82.35%, when using the same data as [1]. Figure 10 illustrates F-score obtained when using several other machine learning techniques available in the Weka package. In Figure 10 P indicates results for relations involving a predicate, NP indicates results for relations within an NP and C indicates the combined results. The results show that frame semantics produce an enhancement of 53.24% over previous state-of-art results in relation extraction. Furthermore, they show that semantic representations such as frames or predicate-argument structures have a wider impact on classification performance than the classification technique.

	Bayes Nets	Naive Bayes	AdaBoost	Bagging	Stacking	ID3	J48	Random Forest	Random Tree	SVM	Kernels on SDTs
P	77.97	79.10	62.14	82.48	62.14	73.45	83.05	82.48	79.09	81.35	82.17
NP	79.32	82.52	63.10	85.22	63.71	72.35	84.81	83.10	80.63	83.92	83.85
C	78.43	80.24	62.46	83.39	62.66	72.71	83.64	82.69	79.60	82.21	82.73

Figure 10: Results of relation extraction.

In the ACE data 61.71% of the training/testing data could be cast into SDTs. The semantic similarity between arguments of a relation within the same NP and arguments present in SDTs allowed the extraction with an average F1-score of 78.41%. The quality of the extraction results depend on the quality of the semantic parsers, that obtained F-scores of over 90% in recent SENSEVAL evaluations.

6 Conclusions

In this paper we have introduced a new dependency structure that relies on semantic information provided by shallow semantic parsers. This structure enabled the extraction of relevant relations with better performance than previous state-of-the-art kernel methods. Furthermore, the semantic features enabled similarly good results to be obtained with a few other learning algorithms. We also used compatibility functions that made use of semantic knowledge. This framework could be extended to allow processing of idiomatic predicates, e.g. [PERSON "lobbying on behalf of"" ORGANIZATION], and combined predications.

References

[1] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the ACL-2004*, 2004.

[2] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press., 1998.

[3] Daniel Gildea and Daniel Jurasky. Automatic labeling of semantic roles. *Computational Linguistic*, 28(3):496–530, 2002.

[4] F. Jelineck, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, and S. Roukos. Decision tree parsing using a hidden derivational model. In *Proceedings of the HLT Workshop-1994*.

[5] X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. A Mention-Synchronous Coreference Resolution Algorithm Based On the Bell Tree. In *Proceedings of the ACL-2004*, 2004.

[6] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. Support Vector Learning for Semantic Argument Classifi cation. *Journal of Machine Learning Research*, 2004.

[7] M. Surdeanu, S. M. Harabagiu, J. Williams, and J. Aarseth. Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the ACL-2003*, 2003.

[8] D. Zelenko, C. Aone, and A. Richardella. Kernel Methods for Relation Extraction. In *Proceedings of the EMNLP-2002*, pages 71–78, 2002.

Semantic Argument Classification Exploiting Argument Interdependence

Zheng Ping Jiang [1] and **Jia Li** [2] and **Hwee Tou Ng** [1,2]
1. Singapore-MIT Alliance, E4-04-10, 4 Engineering Drive 3, Singapore 117576
2. Department of Computer Science, National University of Singapore,
3 Science Drive 2, Singapore 117543
{jiangzp, lijia1, nght}@comp.nus.edu.sg

Abstract

This paper describes our research on automatic semantic argument classification, using the PropBank data [Kingsbury *et al.*, 2002]. Previous research employed features that were based either on a full parse or shallow parse of a sentence. These features were mostly based on an individual semantic argument and the relation between the predicate and a semantic argument, but they did not capture the interdependence among all arguments of a predicate. In this paper, we propose the use of the neighboring semantic arguments of a predicate as additional features in determining the class of the current semantic argument. Our experimental results show significant improvement in the accuracy of semantic argument classification after exploiting argument interdependence. Argument classification accuracy on the standard Section 23 test set improves to 90.50%, representing a relative error reduction of 18%.

1 Introduction

Deriving the semantic representation of a sentence is important in many natural language processing tasks, such as information extraction and question answering. The recent availability of semantically annotated corpora, such as FrameNet [Baker *et al.*, 1998][1] and PropBank [Kingsbury *et al.*, 2002][2], prompted research in automatically producing the semantic representations of English sentences. For PropBank, the semantic representation annotated is in the form of semantic roles, such as $ARG0$, $ARG1$, etc. of each predicate in a sentence, and the process of determining these semantic roles is known as semantic role labeling.

Most previous research treats the semantic role labeling task as a classification problem, and divides the task into two subtasks: *semantic argument identification* and *semantic argument classification*. Semantic argument identification involves classifying each syntactic element in a sentence into either a semantic argument or a non-argument. A syntactic element can be a word in a sentence, a chunk in a shallow parse, or a constituent node in a full parse tree. Semantic argument classification involves classifying each semantic argument identified into a specific semantic role, such as $ARG0$, $ARG1$, etc.

Various features based on either a shallow parse or full parse of a sentence have been proposed. These features are then used by some machine learning algorithm to build a classifier for semantic argument classification. While these features capture the syntactic environment of the semantic argument currently being classified, they overlook the semantic context of the argument.

We propose a notion of *semantic context*, which consists of the already identified or role-classified semantic arguments in the context of an argument that is currently being classified. Semantic context features are defined as features extracted from the neighboring semantic arguments, and used in classifying the current semantic argument. The use of these features explicitly captures the interdependence among the semantic arguments of a predicate.

In this paper, we focus on the semantic argument classification task using PropBank[Kingsbury *et al.*, 2002]. Our semantic context features are derived from full parse trees. Our experimental results show significant improvement in the accuracy of semantic argument classification after adding the semantic context features, when compared to using only features of the current semantic argument.

The rest of this paper is organized as follows: Section 2 explains in detail the application of semantic context features to semantic argument classification; Section 3 gives a specific example to illustrate how semantic context features can improve semantic argument classification; Section 4 presents our experimental results; Section 5 reviews related work; and Section 6 concludes the paper.

2 Semantic Argument Classification with Semantic Context Features

Similar to [Pradhan *et al.*, 2005], we divide the task of semantic role labeling into two sequential subtasks: semantic argument identification and semantic argument classification, and treat each subtask as a separate classification problem. Our investigation focuses on semantic argument classification with the assumption that the semantic arguments have been correctly identified by the semantic argument identifica-

[1] http://framenet.icsi.berkeley.edu/
[2] http://www.cis.upenn.edu/~ace

Feature	Description
Sentence level features	
predicate (Pr)	predicate lemma in the predicate-argument structure
voice (Vo)	grammatical voice of the predicate, either active or passive
subcat (Sc)	grammar rule that expands the predicate's parent node in the parse tree
Argument-specifi c features	
phrase type (Pt)	syntactic category of the argument constituent
head word (Hw)	head word of the argument constituent
Argument-predicate relational features	
path (Pa)	syntactic path through the parse tree from the argument constituent to the predicate node
position (Po)	relative position of the argument consitituent with respect to the predicate node, either left or right

Table 1: Baseline features

tion module in the first step.

2.1 Baseline Features

By treating the semantic argument classification task as a classification problem using machine learning, one of the most important steps in building an accurate classifier is the choice of appropriate features. Most features used in prior research, such as [Gildea and Jurafsky, 2002; Pradhan *et al.*, 2005; Bejan *et al.*, 2004], can be categorized into three types:

- *Sentence level features*, such as predicate, voice, and predicate subcategorization.
- *Argument-specific features*, such as phrase type, head word, content word, head word's part of speech, and named entity class of the argument.
- *Argument-predicate relational features*, such as an argument's position with respect to the predicate, and its syntactic path to the predicate.

The above features attempt to capture the syntactic environment of the semantic argument currently being classified. They are entirely determined by the underlying shallow parse or full parse of the sentence. They carry no information about the semantic arguments that have already been identified or classified. As such, the classification of each semantic argument is done independently from other semantic arguments. It assumes that the semantic arguments of the same predicate do not influence each other.

We use the baseline features in [Pradhan *et al.*, 2005] as our baseline. These features are explained in Table 1.

2.2 Semantic Context Features

For a semantic argument, its semantic context features are defined as the features of its neighboring semantic arguments. The combination of the features of the current semantic argument and its neighboring semantic arguments encodes the interdependence among the semantic arguments.

To illustrate, the parse tree in Figure 1 is annotated with the semantic arguments of the predicate "added". These semantic arguments are $ARG1$, $ARG2$, $ARG4$, and $ARGM\text{-}ADV$.

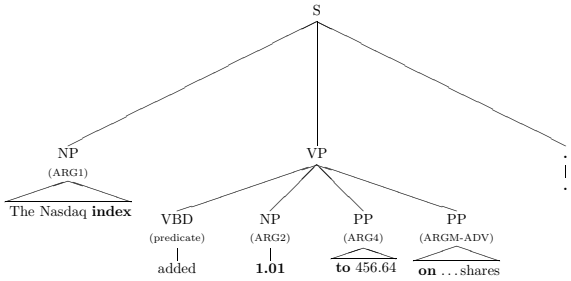


Figure 1: Semantically labeled parse tree, from dev set 00

Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	ARG2
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	ARG4
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	ARGM-ADV

Table 2: Semantic context features based on Figure 1

Each row of Table 2 contains the baseline features (as defined in Table 1) extracted for each semantic argument.

In Figure 1, suppose the semantic argument identification module has correctly identified the constituents "The Nasdaq index", "1.01", "to 456.64", and "on . . . shares" as semantic arguments of the predicate "added". Assume further that the semantic argument "The Nasdaq index" has been assigned the semantic role $ARG1$. Now consider the task of assigning a semantic role to the semantic argument "1.01". Without using semantic context features, the baseline features used for the semantic argument "1.01" appear in row 2 of Table 2.

If we augment with the semantic context features of all neighboring arguments, then the features Pt, Hw, Pa, and Po of the neighboring arguments $ARG1$, $ARG4$, and $ARGM\text{-}ADV$ are added as additional features. Also, since "The Nasdaq index" has been assigned a semantic role, its argument class $ARG1$ is also used as an additional semantic context feature Ar.

2.3 Various Ways of Incorporating Semantic Context Features

There are many possible subsets of the semantic context features in Table 2 that one can choose to add to the baseline features of the current semantic argument being classified. In addition, since the semantic role labels of arguments that are already classified (like $ARG1$ in the previous example) are available as additional context features, the order in which we process the semantic arguments during semantic argument classification can affect the accuracy of the semantic role labels assigned. A good classifier should incorporate an informative set of semantic context features, as well as adopt an argument ordering that helps to give the best overall accuracy for semantic argument classification.

We use context feature acronyms with subscripts to denote particular types of context features at specific locations relative to the current argument being classified. For example, Hw_1 refers to the head word feature of the argument imme-

Feature	Description	Examples
Pt_i	the syntactic category of the ith context semantic argument	Pt_{-1}=NP ; Pt_1=PP
Hw_i	the head word of the ith context semantic argument	Hw_{-1}=index ; Hw_1=to
Pa_i	the path of the ith context semantic argument	Pa_{-1}=NP∧S_VP_VBD ; Pa_1=PP∧VP_VBD
Po_i	the position of the ith context semantic argument	Po_{-1}=L ; Po_1=R
Ar_i	the semantic role of the ith previous semantic argument	Ar_{-1}=ARG1

Table 3: Examples of semantic context features. The current semantic argument being classified is "1.01" as shown in Figure 1.

diately to the right of the current argument being classified. More examples are given in Table 3. We also use set notation $\{-i..i\}$ to denote the set of context features with subscript index $j \in \{-i, \ldots, i\}$. For example, $Hw_{\{-1..1\}}$ denotes the context features Hw_{-1} and Hw_1.

Adding All Types of Context Features

We could choose to add all types of context features within a certain window size. To illustrate with the example in Figure 1, suppose the current semantic argument being classified is "1.01". If we add all types of context features from the set $\{-1..1\}$ (i.e., within a window size of one), then the additional context features added are those in Table 4 that are darkly shaded. The baseline features for the current argument "1.01" is lightly shaded in Table 4.

Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	*
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	

Table 4: Adding all types of context features from the set $\{-1..1\}$

Adding a Single Type of Context Features

Alternatively, we could add only a specific type of context features within a certain window size, since it may be the case that certain types of context features are more effective than others. To illustrate again with the same example. If we add the head word features $Hw_{\{-2..2\}}$ (i.e., within a window size of two), then the additional context features added are those in Table 5 that are darkly shaded. The baseline features for the current argument "1.01" is lightly shaded in Table 5.

Different Argument Ordering

So far we have implicitly assumed that the semantic arguments are processed in a *linear ordering*, i.e., according to the natural textual order in which the semantic arguments appear in a sentence. In PropBank, the semantic arguments of a single predicate in a sentence do not overlap, so this ordering

Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
				nil			
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	*
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	
				nil			

Table 5: Adding one type of context features $Hw_{\{-2..2\}}$

is well-defined. Using a linear ordering, the semantic arguments in Figure 1 are processed in the order $ARG1$, $ARG2$, $ARG4$, and $ARGM\text{-}ADV$.

Experiments reported in [Lim *et al.*, 2004] indicate that semantic arguments spanned by the parent of the predicate node (the immediate clause) can be more accurately classified. Hence, if we process these semantic arguments first, their semantic role labels Ar_i may be more accurately determined and so may help in the subsequent classification of the remaining semantic arguments. Inspired by [Lim *et al.*, 2004], we view a parse tree as containing subtrees of increasing size, with each subtree rooted at each successive ancestor node of the predicate node. In *subtree ordering*, the semantic arguments spanned by a smaller subtree are processed first before the arguments of the next larger subtree. Using a subtree ordering, the semantic arguments in Figure 1 are processed in the order $ARG2$, $ARG4$, $ARGM\text{-}ADV$, and $ARG1$.

3 An Illustrating Example of the Utility of Semantic Context Features

In this section, we shall motivate the utility of semantic context features by considering the example in Figure 1. The predicate "add" has three rolesets defined in PropBank, as shown in Table 6. Each roleset corresponds to a different argument structure. Depending on the particular roleset that a specific instance of "add" belongs to, different argument classes are assigned to its semantic arguments.

Roleset	Arguments
add.01	ARG0(speaker), ARG1(utterance)
add.02	ARG0(adder), ARG1(thing being added), ARG2(thing being added to)
add.03	ARG1(logical subject, patient, thing rising / gaining / being added to), ARG2(amount risen), ARG4(end point), ARGM-LOC(medium)

Table 6: The rolesets of "add", as defined in PropBank

The roleset for "add" in our example is "add.03", where the first argument to the left of "added", "The Nasdaq index", is the "logical subject, patient, thing rising / gaining / being added to". This contrasts with the second roleset "add.02", such as "added" in the sentence "Judge Curry added an additional $55 million to the commission's calculations.", which is an illustrating example used in PropBank's description of the roleset "add.02".

During classification of the semantic arguments "1.01" and "to 456.64" in this example, using only the baseline fea-

tures of an individual semantic argument results in "1.01" being misclassified as $ARG1$ and "to 456.64" being misclassified as $ARG2$, when their correct semantic argument classes should be $ARG2$ and $ARG4$ respectively. Since the two rolesets "add.02" and "add.03" have similar argument structures (add something to something), and since "add.02" occurs more frequently than "add.03", the semantic argument classification module has assigned the semantic argument classes of the roleset "add.02" to "1.01" and "to 456.64", which is incorrect.

However, our experiments in section 4 show that a classifier augmented with the context features $Hw_{\{-2..2\}}$ assigns the correct semantic argument classes to "1.01" and "to 456.64". This is because in the sentence "index added something to something", that "index" is the head word of the first argument gives a strong clue that the correct roleset is "add.03", since "index" cannot fill the adder (agent) role of "add.02". In fact, occurrence counts from PropBank section 02-21 indicate that of the three occurrences of "index" as the head word of the first argument of predicate "add", all three appear in the roleset "add.03". This example illustrates the importance of neighboring semantic arguments in the context of the current semantic argument in determining its correct semantic argument class.

4 Experimental Results

Our semantic argument classification module uses support vector learning and is implemented with yamcha[3] and tinysvm[4] [Kudo and Matsumoto, 2001]. Polynomial kernel with degree 2 is used in a one-vs-all classifier. The cost per unit violation of the margin $C = 1$, and the tolerance of the termination criterion $e = 0.001$.

The training, development, and test data sections follow the conventional split of Section 02-21, 00, and 23 of PropBank release I respectively. In Section 4.1 to Section 4.4, we present the accuracy scores when evaluated on development section 00. In Section 4.5, we present the accuracy scores on test section 23.

The semantic argument classification accuracy when evaluated on section 00 using baseline features is 88.16%, assuming that semantic argument identification is done correctly (i.e., "gold argument identification"). In Section 4.1 to Section 4.4, accuracy scores that show a statistically significant improvement (χ^2 test with $p = 0.05$) over the baseline score of 88.16% are marked by "*". The highest score of each row in the score tables is boldfaced.

4.1 Results of Linear Ordering

Results Using All Types of Context Features

Table 7 contains accuracy scores after augmenting baseline features with all types of semantic context features. Ar feature is only present within context feature sets that contain neighboring arguments to the left of the current argument. We observe that the accuracy scores have a significant improvement from the baseline if context features on both sides of the current argument are used together. Context features to the left of the current argument are not effective when used alone.

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
all	89.27*	**89.32***	88.91	88.52	88.20
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
all	**87.88**	87.41	87.14	86.84	86.70
feature	$\{0..1\}$	$\{0..2\}$	$\{0..3\}$	$\{0..4\}$	$\{0..5\}$
all	88.74	**88.82**	88.48	88.32	88.08

Table 7: Accuracy based on adding all types of semantic context features, with increasing window size, assuming correct argument identification and linear ordering of processing arguments

Results Using a Single Type of Context Features

Accuracy scores of semantic argument classification after augmenting baseline features with a single type of context features are shown in Table 8. The best improvement results from the use of head word features Hw. The use of features Ar reduces accuracy. This is likely due to semantic argument classes not being accurately determined, and errors committed in a position in linear ordering might affect classification accuracy at subsequent positions. However, a better argument ordering may improve the effectiveness of Ar.

4.2 Results of Subtree Ordering

We repeat the experiments of the last subsection, but this time with the use of subtree ordering. The accuracy scores are given in Table 9 and Table 10. The most prominent difference from the results of linear ordering is the better accuracy scores with the use of the Ar features, as shown in Table 10 compared with Table 8. The improvement observed is consistent with the findings of [Lim *et al.*, 2004], that the argument class of the semantic arguments spanned by the parent of the predicate node can be more accurately determined.

4.3 More Experiments with the Ar Feature

Unlike other semantic context features that are completely determined once argument identification is completed, the Ar feature is dynamically determined during argument classification. It may be possible to improve the overall accuracy of

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
Pt	89.01*	**89.06***	88.89	88.81	88.55
Hw	89.49*	**89.82***	89.56*	89.42*	89.32*
Pa	**89.17***	89.04*	88.89	88.66	88.55
Po	88.63	**88.74**	88.58	88.54	88.24
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
Ar	**87.98**	87.72	87.73	87.78	87.75

Table 8: Accuracy based on adding a single type of semantic context features, with increasing window size, assuming correct argument identification and linear ordering of processing arguments

[3] http://chasen.org/~taku/software/yamcha/

[4] http://chasen.org/~taku/software/TinySVM/

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
all	**88.99***	88.96*	88.62	88.43	88.33
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
all	**88.92**	88.83	88.87	88.66	88.40
feature	$\{0..1\}$	$\{0..2\}$	$\{0..3\}$	$\{0..4\}$	$\{0..5\}$
all	**88.41**	88.38	88.08	87.80	87.63

Table 9: Accuracy based on adding all types of semantic context features, with increasing window size, assuming correct argument identification and subtree ordering of processing arguments

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
Pt	**89.03***	88.98*	88.83	88.77	88.46
Hw	89.29*	**89.60***	89.29*	89.20*	89.08*
Pa	89.25*	**89.25***	89.14*	88.95	88.74
Po	88.77	**88.96***	88.72	88.37	88.21
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
Ar	88.85	88.84	**88.90**	88.87	88.83

Table 10: Accuracy based on adding a single type of semantic context features, with increasing window size, assuming correct argument identification and subtree ordering of processing arguments

assigning argument classes to all the semantic arguments of a predicate if we use a beam search algorithm to keep the best k combinations of argument classes assigned to all semantic arguments processed so far.

Ar Feature with Gold Semantic Argument Class

To determine the maximum accuracy improvement possible with the use of the Ar feature, we assume that the argument classes of all previous semantic arguments have been accurately determined and used as semantic context features. The experimental results are presented in Table 11, titled "linear gold" and "subtree gold" respectively for linear and subtree ordering.

Ar Feature with Beam Search

Tinysvm's output scores are converted to confidence scores between 0 and 1, by applying the sigmoid function $y = \frac{1}{1+e^{-x}}$. We implemented a beam search algorithm to find the overall best sequence of argument classes to be assigned to all semantic arguments. At each iteration, the beam search algorithm processes one semantic argument and determines the top 3 argument classes with the highest confidence scores for each semantic argument. It then finds and keeps the best k (k = 10 in our implementation) argument class sequences overall. Each argument class sequence is assigned a confidence score, which is the product of the confidence scores of all arguments in the sequence. Finally, the sequence of argument classes for all semantic arguments with the highest score is chosen. Experimental results are presented under "linear beam" and "subtree beam" in Table 11, and show improvements after using beam search with Ar feature for subtree ordering.

	Accuracy of increasing context window size				
Ar with	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
linear	**87.98**	87.72	87.73	87.78	87.75
linear beam	**88.83**	88.46	88.64	88.66	88.67
linear gold	89.37*	89.54*	89.59*	**89.60***	89.58*
subtree	88.85	88.84	**88.90**	88.87	88.83
subtree beam	89.13*	**89.20***	89.18*	89.19*	89.20*
subtree gold	89.87*	89.84*	90.09*	90.07*	**90.11***

Table 11: More experiments with the Ar feature, using beam search and gold semantic argument class history

4.4 Combining Multiple Semantic Context Features

The best accuracy score we have obtained so far is 89.82%, given by $Hw_{-2..2}$ in Table 8. This score is a statistically significant improvement over the baseline score of 88.16% obtained with only baseline features. We want to further leverage on the other types of semantic context features. Error analysis of experiments in Table 8 and 10 on development section 00 shows that classifiers using different semantic context features make different classification mistakes. Thus we would like to explore the combination of multiple semantic context features. This provides a basis for combining multiple semantic context features. Here we more carefully select the context features than simply using all available features as in Table 7 and 9.

A Single Classifier with Multiple Context Features

The best context features from each row of Table 8, $Pt_{\{-2..2\}}$, $Hw_{\{-2..2\}}$, $Pa_{\{-1..1\}}$, $Po_{\{-2..2\}}$, and $Ar_{\{-1..0\}}$ are combined to form a new classifier based on linear ordering. Similarly $Pt_{\{-1..1\}}$, $Hw_{\{-2..2\}}$, $Pa_{\{-2..2\}}$, $Po_{\{-2..2\}}$, and $Ar\{-3..0\}$ from Table 10 are combined to form a new classifier based on subtree ordering. Evaluation on development section 00 gives accuracy scores of 89.54% and 89.19%, for linear and subtree ordering, respectively. The lack of accuracy improvement over that of $Hw_{-2..2}$ shows that such a simple combination might accumulate the errors introduced by each additional context feature.

Voting with Multiple Classifiers

Instead of building a new classifier with multiple semantic context features, one can combine the classification results of multiple classifiers and hope to achieve better accuracy through consensus and mutual error correction. The voting process combines k different classifiers each belonging to one row in Table 8 or Table 10. Currently only classifiers based on the same ordering are combined. Classifiers are chosen from different context feature types, but can be of the same context feature window size. For a particular semantic argument A, each candidate argument class will accumulate its confidence score assigned by all the voting classifiers. The candidate argument class with the highest aggregate score will be output as the argument class for A. Exhaustive experiments are carried out with $k = 2, 3, 4, 5$. On the development section 00, the best voting classifier in linear ordering is $\{Pt_{\{-1..1\}}, Hw_{\{-4..4\}}, Pa_{\{-2..2\}}, Ar_{\{-1..0\}}\}$, with accuracy 90.32%. The best for subtree ordering is

Linear ordering		Subtree ordering	
Feature	Accuracy	Feature	Accuracy
$Pt_{\{-2..2\}}$	89.26*	$Pt_{\{-1..1\}}$	88.88
$Hw_{\{-2..2\}}$	89.92*	$Hw_{\{-2..2\}}$	89.66*
$Pa_{\{-1..1\}}$	89.20*	$Pa_{\{-2..2\}}$	89.34*
$Po_{\{-2..2\}}$	88.96	$Po_{\{-2..2\}}$	89.20*
$Ar_{\{-1..0\}}$	88.14	$Ar_{\{-3..0\}}$	88.88
$Ar^{beam}_{\{-1..0\}}$	88.80	$Ar^{beam}_{\{-2..0\}}$	89.13*
$All_{\{-2..2\}}$	89.61*	$All_{\{-1..1\}}$	89.33*
$Vote_{linear}$	90.15*	$Vote_{subtree}$	90.50*

Table 12: Semantic argument classification accuracy on test section 23. Baseline accuracy is 88.41%.

$\{Hw_{\{-2..2\}}, Pa_{\{-1..1\}}, Ar_{\{-4..0\}}\}$ with accuracy 90.62%. We denote the two voting classifiers as $Vote_{linear}$ and $Vote_{subtree}$ respectively.

Note that the accuracy score of 90.62% achieved by $Vote_{subtree}$ is a statistically significant improvement over the best score we have obtained without voting, 89.82% of $Hw_{\{-2..2\}}$ in Table 8.

4.5 Testing on Section 23

Based on the experiments on development section 00, we choose the best classifiers and apply them for semantic argument classification on test section 23. The baseline classification accuracy for section 23 is 88.41%, with the use of only baseline features. Subtree voting improves the accuracy to 90.50%, representing a relative error reduction of 18%. The detailed scores are in Table 12.

5 Related Work

To overcome the inadequate assumption of semantic argument independence during classification, [Punyakanok *et al.*, 2004] employed integer programming to impose global constraints in semantic role labeling. These global constraints can be viewed as an introduction of semantic context knowledge, e.g., no argument classes should be duplicated. However, these constraints are rule-based and may not always be applicable due to exceptions. Our proposed approach can be viewed as inducing the constraints via statistical learning.

The methods proposed in [Hacioglu *et al.*, 2004; Kouchnir, 2004; Park *et al.*, 2004] used features such as the phrase type and head word of neighboring syntactic chunks. Experiments in [Pradhan *et al.*, 2005; Fleischman *et al.*, 2003; Kwon *et al.*, 2004] used the semantic role labels of previous semantic arguments as dynamic context features and showed improved accuracy. This demonstrates the utility of limited semantic context features. The semantic context features proposed in this paper capture more extensively the interdependence among arguments.

Our proposed subtree ordering can be viewed as a generalization of the two-level incremental approach of [Lim *et al.*, 2004] in processing arguments, in which a sentence is divided into *immediate clause* and *upper clauses*.

6 Conclusion

In this paper, we successfully used additional features of the neighboring semantic arguments in determining the class of the current semantic argument. Our experimental results have shown significant improvement in the accuracy of semantic argument classification after exploiting argument interdependence. Argument classification accuracy on the standard Section 23 test set yields a relative error reduction of 18%.

Acknowledgements

We thank Taku Kudo and Sameer Pradhan for their help in answering our questions related to the work in this paper.

References

[Baker *et al.*, 1998] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet Project. *Proceedings of COLING*, 1998.

[Bejan *et al.*, 2004] C. A. Bejan, A. Moschitti, P. Morarescu, G. Nicolae, and S. Harabagiu. Semantic Parsing Based on FrameNet. *Proceedings of SENSEVAL-3*, 2004.

[Fleischman *et al.*, 2003] M. Fleischman, N. Kwon, and E. Hovy. Maximum Entropy Models for FrameNet Classification. *Proceedings of EMNLP*, 2003.

[Gildea and Jurafsky, 2002] D. Gildea and D. Jurafsky. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 2002.

[Hacioglu *et al.*, 2004] K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. Semantic Role Labeling by Tagging Syntactic Chunks. *Proceedings of CoNLL*, 2004.

[Kingsbury *et al.*, 2002] P. Kingsbury, M. Palmer, and M. Marcus. Adding Semantic annotation to the Penn Treebank. *Proceedings of HLT*, 2002.

[Kouchnir, 2004] B. Kouchnir. A Memory-based Approach for Semantic Role Labeling. *Proceedings of CoNLL*, 2004.

[Kudo and Matsumoto, 2001] T. Kudo and Y. Matsumoto. Chunking with Support Vector Machines. *Proceedings of NAACL*, 2001.

[Kwon *et al.*, 2004] N. Kwon, M. Fleischman, and E. Hovy. FrameNet-based Semantic Parsing using Maximum Entropy Models. *Proceedings of COLING*, 2004.

[Lim *et al.*, 2004] J.-H. Lim, Y.-S. Hwang, S.-Y. Park, and H.-C. Rim. Semantic Role Labeling using Maximum Entropy Model. *Proceedings of CoNLL*, 2004.

[Park *et al.*, 2004] K.-M. Park, Y.-S. Hwang, and H.-C. Rim. Two-Phase Semantic Role Labeling based on Support Vector Machines. *Proceedings of CoNLL*, 2004.

[Pradhan *et al.*, 2005] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. Support Vector Learning for Semantic Argument Classification. *To appear in Machine Learning*, 2005.

[Punyakanok *et al.*, 2004] V. Punyakanok, D. Roth, W.-T. Yih, and D. Zimak. Semantic Role Labeling via Integer Linear Programming Inference. *Proceedings of COLING*, 2004.

Automatic Text-to-Scene Conversion in the Traffic Accident Domain

Richard Johansson **Anders Berglund** **Magnus Danielsson** **Pierre Nugues**
LUCAS, Department of Computer Science, Lund University
Box 118
SE-221 00 Lund, Sweden
{richard, pierre}@cs.lth.se, d98ab@efd.lth.se
magnus.danielsson2@comhem.se

Abstract

In this paper, we describe a system that automatically converts narratives into 3D scenes. The texts, written in Swedish, describe road accidents. One of the program's key features is that it animates the generated scene using temporal relations between the events. We believe that this system is the first text-to-scene converter that is not restricted to invented narratives.

The system consists of three modules: natural language interpretation based on information extraction (IE) methods, a planning module that produces a geometric description of the accident, and finally a visualization module that renders the geometric description as animated graphics.

An evaluation of the system was carried out in two steps: First, we used standard IE scoring methods to evaluate the language interpretation. The results are on the same level as for similar systems tested previously. Secondly, we performed a small user study to evaluate the quality of the visualization. The results validate our choice of methods, and since this is the first evaluation of a text-to-scene conversion system, they also provide a baseline for further studies.

1 Introduction

For a machine, text-to-scene conversion consists in synthesizing a 2D or 3D geometric description from a text and in displaying it. Ideally, a text-to-scene converter would recreate mental images we form when we read a text. This represents a demanding task involving semantic and cognitive capabilities and to many people seems both a far off and surreal fantasy. However, there have been a small number of systems that provided insights into the feasibility of it while at the same time showing significant limitations [Adorni *et al.*, 1984; Coyne and Sproat, 2001; Arens *et al.*, 2002]. First, all the systems are restricted to very simple narratives, typically invented by the authors themselves. Furthermore, none of the authors report details on the text corpus they used or any precise description of the results. Another significant point is that these systems all focus on spatial relations, while ignoring the temporal dimension completely. Most of them are limited to recreating static scenes.

In this paper, we describe a new version of the Carsim system [Johansson *et al.*, 2004; Dupuy *et al.*, 2001], which is a text-to-scene converter that handles real texts and that we evaluated using quantitative methods. The program generates 3D graphics from traffic accident reports generally collected from web sites of Swedish newspapers. One of its key features is that it takes time and temporal relations between events into account to animate the synthesized scene.

The structure of this article is as follows: Section 2 describes the Carsim system and the application domain. Section 3 details the implementation of the natural language interpretation module. Then, in Section 4, we turn to the spatial and temporal reasoning that is needed to construct the visualization. The evaluation is described in Section 5. Finally, we discuss the results and their implications in Section 6.

2 The Carsim System

Narratives of a car accidents often make use of space descriptions, movements, and directions that are sometimes difficult to grasp for readers. We believe that forming consistent mental images is necessary to understand them properly. However, some people have difficulties in imagining situations and may need visual aids pre-designed by professional analysts.

Carsim is a computer program[1] that addresses this need. It is intended to be a helpful tool that can enable people to imagine a traffic situation and understand the course of events properly. The program analyzes texts describing car accidents and visualizes them in a 3D environment.

To generate a 3D scene, Carsim combines natural language processing components and a visualizer. The language processing module adopts an information extraction strategy and includes machine learning methods to solve coreference, classify predicate/argument structures, and order events temporally. However, as real texts suffer from underspecification and rarely contain a detailed geometric description of the actions, information extraction alone is insufficient to convert narratives into images automatically. To handle this, Carsim

[1] An online demonstration of the system is available at http://www.lucas.lth.se/lt.

infers implicit information about the environment and the involved entities from key phrases in the text, knowledge about typical traffic situations, and properties of the involved entities. The program uses a visualization planner that applies spatial and temporal reasoning to "imagine" the entities and actions described in the text, and that tries to find the simplest configuration that fits the description.

2.1 A Corpus of Traffic Accident Descriptions

Carsim has been developed using authentic texts. As a development set, we collected approximately 200 reports of road accidents from various Swedish newspapers. The task of analyzing the news reports is made more complex by their variability in style and length. The size of the texts ranges from a couple of sentences to more than a page. The amount of details is overwhelming in some reports, while in others, most of the information is implicit. The complexity of the accidents described ranges from simple crashes with only one car to multiple collisions with several participating vehicles and complex movements. Although our work has concentrated on the press clippings, we also have access to accident reports from the STRADA database (Swedish TRaffic Accident Data Acquisition) of Vägverket, the Swedish traffic authority.

The next text is an excerpt from our test corpus. This report is an example of a press wire describing an accident.

> *En bussolycka i södra Afghanistan krävde på torsdagen 20 dödsoffer. Ytterligare 39 personer skadades i olyckan som inträffade tidigt på torsdagsmorgonen två mil norr om staden Kandahar. Bussen var på väg från Kandahar mot huvudstaden Kabul när den under en omkörning körde av vägbanan och voltade, meddelade general Salim Khan, biträdande polischef i Kandahar. Läget för några av de skadade uppgavs som kritiskt.*
>
> TT-AFP & Dagens Nyheter, July 8, 2004

> A bus accident in southern Afghanistan last Thursday claimed 20 victims. Additionally, 39 people were injured in the accident, which occurred early Thursday morning twenty kilometers north of the city Kandahar. The bus was on its way from Kandahar towards the capital Kabul when it left the road while overtaking and overturned, said general Salim Khan, assistant head of police in Kandahar. The state of some of the injured was said to be critical.
>
> The text above, our translation.

2.2 Architecture of the Carsim System

We use a division into modules where each one addresses one step of the conversion process (see Figure 1).

- A *natural language processing* module that interprets the text to produce an intermediate symbolic representation.
- A *spatio-temporal planning and inference* module that produces a full geometric description given the symbolic representation.
- A *graphical* module that renders the geometric description as graphics.

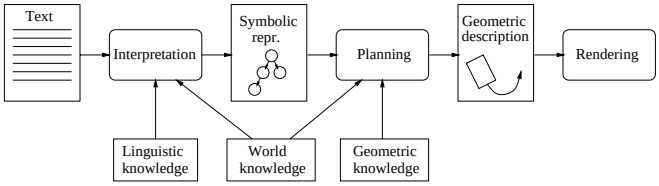


Figure 1: System architecture.

We use the intermediate representation as a bridge between texts and geometry. This is made necessary because the information expressed by most reports has usually little affinity with a geometric description. Exact and explicit accounts of the world and its physical properties are rarely present. In addition, our vocabulary is finite and discrete, while the set of geometric descriptions is infinite and continuous.

Once the NLP module has interpreted and converted a text, the planner maps the resulting symbolic representation of the world, the entities, and behaviors, onto a complete and unambiguous geometric description in a Euclidean space.

Certain facts are never explicitly stated, but are assumed by the author to be known to the reader. This includes linguistic knowledge, world knowledge (such as traffic regulations and typical behaviors), and geometric knowledge (such as typical sizes of vehicles). The language processing and planning modules take this knowledge into account in order to produce a credible geometric description that can be visualized by the renderer.

2.3 The Symbolic Representation

The symbolic representation has to manage the following trade-off. In order to be able to describe a scene, it must contain enough information to make it feasible to produce a consistent geometric description, acceptable to the user. On the other hand, the representation has to be close to ways human beings describe things to capture information in the texts.

We used four concept categories that we ordered in an inheritance hierarchy. The representation is implemented using Minsky-style ("object-oriented") frames, which means that the objects in the representation consist of a number of predefined attribute/values slots. This ontology was designed with assistance of traffic safety experts. It consists of:

- *Objects*. These are typically the physical entities that are mentioned in the text, but we might also need to present abstract or oneiric entities as symbols in the scene. Each object has a type that is selected from a predefined, finite set. `Car` and `Truck` are examples of object types.
- *Events*. They correspond intuitively to an activity that goes on during a period in time and here to the possible object behaviors. We represent events as entities with a type from a predefined set. `Overturn` and `Impact` are examples.
- *Relations and Quantities*. The objects and the events need to be described and related to each other. The most obvious examples of such information are *spatial* information about objects and *temporal* information about events. We should be able to express not only exact

quantities, but also qualitative information (by which we mean that only certain fundamental distinctions are made). `Behind`, `FromLeft`, and `During` are examples of spatial and temporal relations.

- *Environment*. The environment of the accident is important for the visualization to be understandable. Significant environmental parameters include light, weather, road conditions, and type of environment (such as rural or urban). Another important parameter is topography, but we have set it aside since we have no convenient way to express this qualitatively.

3 Natural Language Interpretation

We use information extraction techniques to interpret the text. This is justified by the symbolic representation, which is restricted to a limited set of types and the fact that only a part of the meaning of the text needs to be presented visually. The IE module consists of a sequence of components (Figure 2). The first components carry out a shallow parse: POS tagging, NP chunking, complex word recognition, and clause segmentation. This is followed by a cascade of semantic markup components: named entity recognition, temporal expression detection, object markup and coreference, and predicate argument detection. Finally, the marked-up structures are interpreted to yield the resulting symbolic representation of the accident. The development of the IE module has been made more complex by the fact that few tools or annotated corpora are available for Swedish. The only significant external tool we have used is the Granska POS tagger [Carlberger and Kann, 1999].

3.1 Entity Detection and Coreference

A correct detection of the involved entities is crucial for the graphical presentation to be understandable. We first search the likely participants among the noun phrases in the text by checking them against a dictionary. We then apply a coreference solver to link the groups that refer to identical entities. This results in a set of equivalence classes referring to entities that are likely to be participants in the accident.

The coreference solver uses a hand-written filter in conjunction with a statistical system based on decision trees [Danielsson, 2005]. The filter first tests each antecedent-anaphor pair using 12 grammatical and semantic features to prevent unlikely coreference. The statistical system then uses 20 features to classify pairs of noun groups as coreferring or not. These features are lexical, grammatical, and semantic. The trees were induced from a set of hand-annotated examples using the ID3 algorithm and a method inspired by Soon *et al.* [2001]. We implemented a novel *feature transfer* mechanism that propagates and continuously changes the values of semantic features in the coreference chains during clustering. This means that the coreferring markables inherit semantic properties from each other. Feature transfer, as well as domain-specific semantic features, proved to have a significant impact on the performance.

3.2 Domain Events

In order to produce a symbolic representation of the accident, we need to recreate the course of events. We find the events using a two-step procedure. First, we identify and mark up text fragments that describe events, and locate and classify their arguments. Secondly, we interpret the fragments in order to produce event objects as well as the involved participants, spatial and temporal relations, and information about the environment. This two-step procedure is similar to other work that uses predicate-argument structures for IE, for example [Surdeanu *et al.*, 2003].

We classify the arguments of each predicate (assign them a semantic role) using a statistical system, which was trained on about 900 hand-annotated examples. Following Gildea and Jurafsky [2002], there has been a relative consensus on the features that the classifier should use. We use a slightly different set; since we have no full parser, there are no features that refer to the parse tree. Also, since the system is domain-specific, we have introduced a *semantic type* feature that takes the following values: dynamic object, static object, human, place, time, cause, or speed.

Similarly to the method described by Gildea and Jurafsky [2002], the classifier chooses the role that maximizes the estimated probability of a role given the values of the target, head, and semantic type attributes:

$$\hat{P}(r|t, head, sem) = \frac{C(r, t, head, sem)}{C(t, head, sem)}.$$

If a particular combination of target, head, and semantic type is not found in the training set, the classifier uses a back-off strategy, taking the other attributes into account. In addition, we tried other classification methods (ID3 with gain ratio, SVMs) without any significant improvement.

When the system has located the references to domain events in the text, it can interpret them; that is, we map the text fragments onto entities in the symbolic representation. This stage makes significant use of world knowledge, for example to handle relationships such as metonymy and ownership.

Since it is common that events are mentioned more than once in the text, we need to remove the duplicates in order not to introduce unnecessary animations. Event coreference is a task that can be treated using similar methods as those we used for object coreference. However, event coreference is a simpler problem since the range of candidates is narrowed by the involved participants and the event type. To get a minimal description of the course of events, we have found that it is sufficient to unify as many events as possible, taking event types and participants into account.

Finally, we fill in the information that is lacking due to mistakes or underspecification using default and heuristic rules. Thus, we have a complete description of the events and the participants.

3.3 Temporal Ordering of Events

Since we produce an animation rather than just a static image, we take time into account and we find the temporal relations between the events. Although the planner alone can infer a probable course of events given the positions of the participants, and some orderings are deducible by means of simple ad-hoc rules that place effects after causes (such as a fire after a collision), we have opted for a general approach.

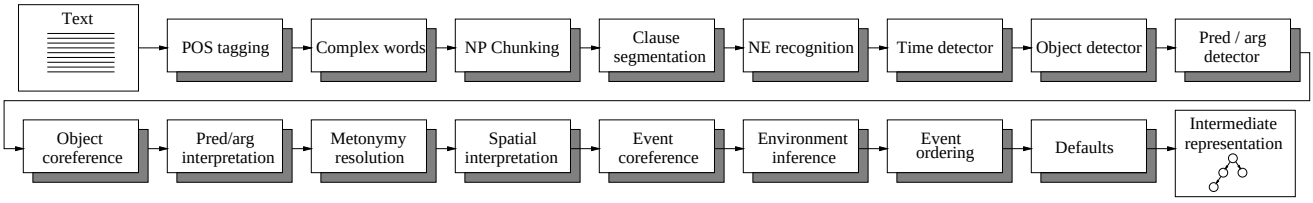


Figure 2: Architecture of the language interpretation module.

We developed a component based on the generic TimeML framework [Pustejovsky *et al.*, 2002]. We first create an ordering of all events in the text (where all verbs, and a small set of nouns, are considered to refer to events) by generating temporal links (orderings) between the words that denote the events. The links are generated by a hybrid system that consists of a statistical system based on decision trees and a small set of hand-written heuristics.

The statistical system considers events that are close to each other in the text, and that are not separated by explicit temporal expressions. It was trained on a set of hand-annotated examples, which consists of 476 events and 1,162 temporal relations. The decision trees were produced using the C4.5 tool [Quinlan, 1993] and make use of the following information:

- *Tense*, *aspect*, and *grammatical construct* of the verb groups that denote the events.
- *Temporal signals* between the words. This is a TimeML term for temporal connectives and prepositions such as "when", "after", and "during".
- *Distance* between the words, measured in tokens, sentences, and in punctuation signs.

The range of possible output values is the following subset of Allen's relations: *simultaneous*, *after*, *before*, *is_included*, *includes*, and *none*.

After the decision trees have been applied, we remove conflicting temporal links using probability estimates derived from C4.5. Finally, we extract the temporal relations between the events that are relevant to Carsim.

3.4 Inferring the Environment

The environment is important for the graphical presentation to be credible. We use traditional IE techniques, such as domain-relevant patterns, to find explicit descriptions of the environment.

As noted by the WordsEye team [Sproat, 2001], the environment of a scene may often be obvious to a reader even though it is not described in the text. In order to capture this information, we try to infer it using prepositional phrases that occur in the description of the events. We then use a Naïve Bayesian classifier to guess the environment.

4 Planning the Animation

We use a planning system to create the animation out of the extracted information. It first determines a set of constraints that the animation needs to fulfill. Then, it goes on to find the initial directions and positions. Finally, it uses a search algorithm to find the trajectory layout. Since we use no backtracking, this separation into steps introduces a risk of bad choices. However, it reduces the computation load and proved sufficient for the texts we considered, enabling an interactive generation of 3D scenes and a better user experience.

4.1 Finding the Constraints

The constraints on the animation are created using the detected events and the spatial and temporal relations combined with the implicit knowledge about the world. The events are expressed as conjunctions of primitive predicates about the objects and their behavior in time. For example, if we state that there is an `Overtake` event where O_1 overtakes O_2, this is translated into the following proposition:

$$\exists t_1, t_2. MovesSideways(O_1, Left, t_1) \\ \wedge Passes(O_1, O_2, t_2) \wedge t_1 < t_2$$

In addition, other constraints are implied by the events and our knowledge of the world. For example, if O_1 overtakes O_2, we add the constraints that O_1 is initially positioned behind O_2, and that O_1 has the same initial direction as O_2. Other constraints are added due to the non-presence of events, such as

$$NoCollide(O_1, O_2) \equiv \neg \exists t. Collides(O_1, O_2, t)$$

if there is no mentioned collision between O_1 and O_2.

4.2 Finding Initial Directions and Positions

We use constraint propagation techniques to infer initial directions and positions for all the involved objects. We first set those directions and positions that are stated explicitly. Each time a direction is uniquely determined, it is set and this change propagates to the sets of available choices of directions for other objects, whose directions have been stated in relation to the first one. When the direction can't be determined uniquely for any object, we pick one object and set its direction. This goes on until the initial directions have been inferred for all objects.

4.3 Finding the Trajectories

After the constraints have been found, we use the IDA* search method to find a trajectory layout that is as simple as possible while violating no constraints. As heuristic function, we use the number of violated constraints multiplied by a constant in order to keep the heuristic admissible.

The most complicated accident in our development contains 8 events, which results in 15 constraints during search,

and needs 6 modifications of the trajectories to arrive at a trajectory layout that violates no constraints. This solution is found in a few seconds. Most accidents can be described using only a few constraints.

At times, no solution is found within reasonable time. Typically, this happens when the IE module has produced incorrect results. In this case, the planner backs off. First, it relaxes some of the temporal constraints (for example: $Simultaneous$ constraints are replaced by $NearTime$). Next, all temporal constraints are removed.

5 Evaluation

We evaluated the components of the system, first by measuring the quality of the extracted information using standard IE evaluation methods, then by performing a user study to determine the overall perception of the complete system. For both evaluations, we used 50 previously unseen texts, which had been collected from newspaper sources on the web. The size of the texts ranged from 36 to 541 tokens.

5.1 Evaluation of the Information Extraction Module

For the IE module, three aspects were evaluated: object detection, event detection, and temporal ordering of the events. Table 1 shows the precision and recall figures.

	P	R	$F_{\beta=1}$
Objects	0.96	0.86	0.91
Events	0.86	0.85	0.85
Temporal relations (correct events)	0.73	0.55	0.62
Temporal relations (all events)	0.61	0.51	0.55

Table 1: Statistics for the IE module on the test set.

The evaluations of object and event detection were rather straightforward. A road object was considered to be correctly detected if a corresponding object was either mentioned in or implicit from the text, and the type of the object was correct. The same criteria applied to the detection of events, but here we also added the criterion that the actor (and victim, where it applies) must be correct.

Evaluating the quality of the temporal orderings proved to be less straightforward. First, to make it feasible to compare the graph of orderings to the correct graph, it must be converted to some normal form. For this, we used the transitive closure (that is, we made all implicit links explicit). The transitive closure has some drawbacks – for example, one small mistake may cause a large impact on the precision and recall measures if many events are involved. However, we found no other obvious method for normalizing the temporal graphs.

A second problem to resolve was that of how to evaluate temporal orderings when the events are not all correctly detected. This difficulty arises when the event coreference resolution fails and multiple instances of the same event are reported. Because of this complication, we measured the link precision and recall for two cases: first, only those links that connect properly detected events; secondly, all links. We then compared the results.

The results of the event detection is comparable to those reported in previously published work. [Surdeanu *et al.*, 2003] reports an F-measure of 0.83 in the Market Change domain for a system that uses similar IE methods.[2] Although our system has a different domain, a different language, and different resources (their system is based on PropBank), the figures are roughly similar. The somewhat easier task of detecting the objects results in higher figures, demonstrating that the method chosen works satisfactorily for the task at hand.

We believe that the figures for the temporal relations will prove competitive. We are not aware of any previous result in automatic detection of temporal relations in IE. The figures also show that the difference between the methods of evaluation is relatively small, suggesting that both methods are useful when evaluating temporal orderings.

5.2 User Study to Evaluate the Visualization

Four users were shown the animations of subsets of the 50 test texts. Figure 3 shows an example corresponding to the text from Subsection 2.1. The users graded the quality of animations using the following scores: 0 for wrong, 1 for "more or less" correct, and 2 for perfect. The average score was 0.91. The number of texts that had an average score of 2 was 14 (28 percent), and the number of texts with an average score of at least 1 was 28 (56 percent). These figures demonstrate that the chosen strategy is viable, especially in a restricted context like the traffic accident domain. However, interpretation of the figures is difficult since there are no previously published results. In any case, they provide a baseline for further studies, possibly in another domain.

To determine whether the small size of our test group introduced a risk of invalid results, we calculated the standard deviation of annotations[3], and we obtained the value of 0.45. Replacing all annotations with random values from the same probability distribution resulted in a standard deviation of 0.83 on average. In addition, the pairwise correlation of the annotations is 0.75. This suggests that the agreement among annotators is enough for the figures to be relevant.

During discussions with users, we had a number of unexpected opinions about the visualizations. One important example of this is the quantity of implicit information they infer from reading the texts. For example, given a short description of a crash in an urban environment, one user imagined a collision of two moving vehicles at an intersection, while another user interpreted it as a collision between a moving and a parked car.

This user response shows that the task of imagining a situation is difficult for humans as well as for machines. Furthermore, while some users have suggested that we improve the realism (for example, the physical behavior of the objects), discussions generally made it clear that the semi-realistic graphics that we use (see Figure 3) may suggest to the user

[2] We have assumed that the Templettes that they use roughly can be identified with events.

[3] We calculated this using the formula $\sqrt{\frac{\sum (x_{ij}-\dot{x}_i)^2}{\sum (n_i-1)}}$, where x_{ij} is the score assigned by annotator j on text i, $\dot{x}_i$ the average score on text i, and n_i the number of annotators on text i.

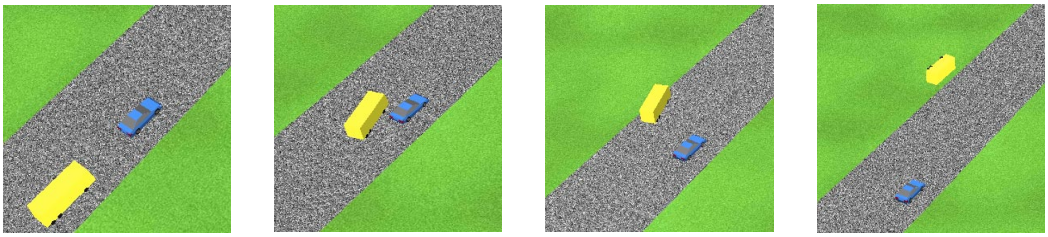


Figure 3: Screenshots from the animation of the text above.

that the system knows more than it actually does. Since the system visualizes symbolic information, it may actually be more appropriate to present the graphics in a more "abstract" manner that reflects this better, for example via symbolic signs in the scene.

6 Conclusion and Perspectives

We have presented an architecture and a strategy based on information extraction and symbolic visualization that enables to convert real texts into 3D scenes. As far as we know, Carsim is the only text-to-scene conversion system that has been developed and tested using non-invented narratives. It is also unique in the sense that it produces animated graphics by taking takes temporal relations between events into account.

We have provided the first quantitative evaluation of a text-to-scene conversion system, which shows promising results that validate our choice of methods and set a baseline for future improvements.

In the future, we would like to extend the prototype system to deeper levels of semantic information. While the current prototype uses no external knowledge, we would like to investigate whether it is possible to integrate additional knowledge sources in order to make the visualization more realistic and understandable. Two important examples of this are geographical and meteorological information, which could be helpful in improving the realism and in creating a more accurate reconstruction of the circumstances and the environment. Another topic that has been prominent in our discussions with traffic safety experts is how to reconcile different narratives that describe the same accident.

Acknowledgements

The first author wishes to thank Margaret Newman-Nowicka for her numerous suggestions for improvement of this paper.

This work is partly supported by grant number 2002-02380 from the Språkteknologi program of Vinnova, the Swedish Agency of Innovation Systems.

References

[Adorni *et al.*, 1984] Giovanni Adorni, Mauro Di Manzo, and Fausto Giunchiglia. Natural language driven image generation. In *Proceedings of COLING 84*, pages 495–500, Stanford, California, 1984.

[Arens *et al.*, 2002] Michael Arens, Artur Ottlik, and Hans-Hellmut Nagel. Natural language texts for a cognitive vision system. In Frank van Harmelen, editor, *ECAI2002, Proceedings of the 15th European Conference on Artificial Intelligence*, Lyon, July 21-26 2002.

[Carlberger and Kann, 1999] Johan Carlberger and Viggo Kann. Implementing an efficient part-of-speech tagger. *Software Practice and Experience*, 29:815–832, 1999.

[Coyne and Sproat, 2001] Bob Coyne and Richard Sproat. WordsEye: An automatic text-to-scene conversion system. In *Proceedings of the Siggraph Conference*, Los Angeles, 2001.

[Danielsson, 2005] Magnus Danielsson. Maskininlärningsbaserad koreferensbestämning för nominalfraser applicerat på svenska texter. Master's thesis, Lunds Universitet, 2005.

[Dupuy *et al.*, 2001] Sylvain Dupuy, Arjan Egges, Vincent Legendre, and Pierre Nugues. Generating a 3D simulation of a car accident from a written description in natural language: The Carsim system. In *ACL2001: Workshop on Temporal and Spatial Information Processing*, pages 1–8, Toulouse, July 7 2001.

[Gildea and Jurafsky, 2002] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[Johansson *et al.*, 2004] Richard Johansson, David Williams, Anders Berglund, and Pierre Nugues. Carsim: A System to Visualize Written Road Accident Reports as Animated 3D Scenes. In *ACL2004: Second Workshop on Text Meaning and Interpretation*, pages 57–64, Barcelona, July 25-26 2004.

[Pustejovsky *et al.*, 2002] James Pustejovsky, Roser Saurí, Andrea Setzer, Rob Gaizauskas, and Bob Ingria. TimeML Annotation Guidelines. Technical report, 2002.

[Quinlan, 1993] John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffman, 1993.

[Soon *et al.*, 2001] Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001.

[Sproat, 2001] Richard Sproat. Inferring the environment in a text-to-scene conversion system. In *Proceedings of the K-CAP'01*, October 22-23 2001.

[Surdeanu *et al.*, 2003] Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of the ACL*, Sapporo, Japan, 2003.

Redundancy-free Island Parsing of Word Graphs

Bernd Kiefer
Deutsches Forschungszentrum für Künstliche Intelligenz
Stuhlsatzenhausweg 3, 66119 Saarbrücken,
kiefer@dfki.de

Abstract

Island parsing is a bidirectional parsing strategy mostly used in speech analysis, as well as in applications where robustness is highly relevant and/or processing resources are limited. Although there exists an efficient redundancy-free island parsing algorithm for string input, it has not yet been applied to word graph input, an application which is central for speech analysis systems. This paper describes how the established algorithm can be generalized from string input to word graphs, increasing its flexibility by integrating the selection of island seeds into the search process inherent to parsing.

1 Introduction

Island parsing is a parsing strategy for context free grammars, mostly used in speech applications ([Ageno, 2003], [Gallwitz *et al.*, 1998], [Thanopoulos *et al.*, 1997], [Mecklenburg *et al.*, 1995], [Brietzmann, 1992]). It is a *bidirectional* strategy, in that incomplete parse items, which encode partially filled right hand sides of a context free rule, may extend in both directions. Furthermore, parsing starts at some highly ranked input items called *seeds* and tries to explore the "islands of certainty" first.

Since island parsing starts building all possible derivations from every seed in both directions, provisions must be taken so as to avoid multiple computation of identical subderivations, which would lead to spurious ambiguities and reduced efficiency. To my knowledge, there is only one description of an efficient fully redundancy free algorithm for island parsing, contained in an article comparing different bidirectional parsing approaches for context-free grammars, namely [Satta and Stock, 1994].

This algorithm splits up the chart into consecutive regions such that the region borders correspond to chart nodes and every region contains exactly one seed item. Derived chart items lying entirely within a region have been constructed starting at the seed in a manner similar to Earley parsing, which guarantees that they are build in a unique way. Items crossing region borders are a possible source of redundancy, because the same derivation could be build starting either at the right or the left seed (see figure 2 for an illustration). By fixing the expansion direction of those items when they combine for the first time, duplicate derivations are avoided.

This paper extends the original algorithm in two aspects that make it more feasible for speech applications.

Firstly, the original algorithm deals only with string input. Because many speech applications require the direct analysis of word graphs, it is desirable to extend the method to word graph input. Word graphs are acyclic directed graphs of input items with exactly one source node and one sink node (a node with in-degree resp. out-degree zero). They encode ambiguous input using possibly overlapping sub-paths, which lead to more complicated input configurations and require a modification of the original algorithm. The word graph in figure 1, for example, can not be split into more than one region because every border at an inner node would cross an input item, thus inhibiting the use of more than one seed.

A modified algorithm must be able to deal with these configurations without losing efficiency, since speech applications are typically time critical and often have limited space resources. Therefore, care has been taken to preserve the efficiency and the redundancy avoidance of the original.

Secondly, instead of picking all seeds in advance, the modified version integrates their selection into the search inherent to the parsing process. Since a lower number of seeds may result in faster parsing, it is advantageous to be able to base seed selection also on information created *during* parsing, namely, on already constructed items and their quality.

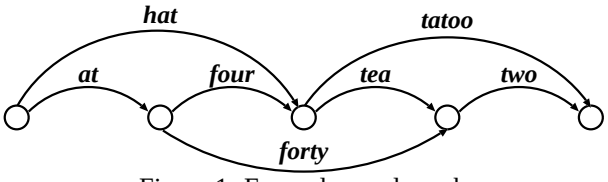


Figure 1: Example word graph

Selecting a seed becomes one of the actions the parser can take, like the combination or creation of other parse items. This provides full flexibility in the design of the search strategy, which in resource-limited applications can have a big impact on the quality of the, possibly partial, results. Picking all seeds in advance is then just one of the possible options.

Furthermore, dynamic selection of seeds is performed such that all sub-paths of the word graph will be be properly explored to arrive at a complete solution, which might not be

the case if seeds were picked disadvantageously. If, for example, the seeds in figure 1 were the items labeled *hat* and *tea*, it might happen that the sub-path containing *forty* would not be considered.

The modified algorithm does not fix regions for the whole word graph in advance. The resulting chart configurations are rather such that every path through the word graph has its own regions, and the borders of the different paths can, but need not, coincide. Since similar properties hold for the chart items with respect to the per-path regions as they do for the string version, the modified algorithm is still correct, complete and free of redundancy.

2 The Original Algorithm

Because [Satta and Stock, 1994] aim at describing bidirectional context-free parsing in more generality, the formulation of the island parsing algorithm itself is somewhat complicated and its implementation is not obvious at first glance. To facilitate the description of the modifications, the original algorithm is presented first, albeit in an alternative, but equivalent form. Similar notation is used as far as possible to stress the connection between the two formulations. In many respects, it is an ordinary chart parsing algorithm, that is only complicated by the bookkeeping necessary to avoid redundant computations. After introducing the notation, the algorithm is presented as a set of pseudo-code procedures.

The algorithm uses a context free grammar $G = (\Sigma, N, P, S)$, where Σ and N are finite sets of *terminal* and *nonterminal* symbols, respectively, P is the set of *rules* $r := D_r \rightarrow Z_{r,1} \cdots Z_{r,\pi_r}$, where $D_r \in N$ and each $Z_{r,i} \in \Sigma \cup N$. π_r is the number of symbols on the right hand side of rule r. $S \in N$ is the start symbol of the grammar. The grammar must not contain empty rules, i.e, rules of the form $A \rightarrow \epsilon$. The input is a string of L terminal symbols $a_1, \ldots, a_L$. The algorithm uses a chart of size $L+1$: a two-dimensional array $t_{i,j}$, $i, j \in \{0, \ldots, L\}$, where each $t_{i,j}$ contains a set of two kinds of items: complete and incomplete items.

A *complete* chart item is a triple (NT, i, j) with $NT \in \Sigma \cup N$ being the terminal or nonterminal category, i and j the index of the start and end node of the item, respectively.

For *incomplete* items, we introduce symbols $I_r^{k,l}$ that represent dotted items (partial derivations) of a rule $r \in P$:

$$I_r^{k,l} \equiv D_r \rightarrow Z_{r,1} \cdots Z_{r,k} \bullet Z_{r,k+1} \cdots Z_{r,l} \bullet Z_{r,l+1} \cdots Z_{r,\pi_r}$$

with $k, l \in \{0, \ldots, \pi_r\}$ and $k \leq l$. Let I_G be the set of all symbols $I_r^{k,l}$ for the dotted items of grammar G. Analogous to complete items, incomplete items are triples $(I_r^{k,l}, i, j)$, where $k > 0$ or $l < \pi_r$, or both.

The island seeds are represented by a set of indices $\mathcal{S} = \{i_1, \ldots, i_q\}$ of the corresponding input symbols. For the island parsing algorithm, the chart is divided into regions, such that every region contains exactly one seed. The indices of the region borders are named p_k, with $p_0 = 0$, $p_q = L$ and $i_k \leq p_k < i_{k+1}, k \in \{1, \ldots, q-1\}$. The region between a seed and its left border is called *right substring*, because items in this region are built in a right-to-left top-down fashion. Analogously, there is a *left substring* to the right of a seed, where items are built from left-to-right, respectively.

The algorithm is started by adding all tuples $(a_i, i-1, i)$ to $t_{i-1,i}$ and then calling *add_complete* for all of them. The program terminates whenever a derivation from the start symbol to the input string was found and the **exit** statement in *add_new* was reached, or else, if there are no more items to add, in which case the string is rejected.

Although items can potentially combine with other items at both sides, their expansion direction is restricted *dynamically* to avoid redundant computation of sub-derivations. These restrictions are implemented using two additional two-dimensional $L+1 \times L+1$ arrays *block_left* and *block_right*, that contain symbols of $N \cup \Sigma \cup I_G$. If, for example, $A \in$ *block_right*(i, j), the item $(A, i, j) \in t_{i,j}$ cannot combine with any item adjacent to its right.

To illustrate the algorithm, figure 2 shows an example chart that will be referred to throughout the next paragraphs. Complete and incomplete items are represented by solid respectively dashed arcs, bearing symbols from N, Σ, and I_G as labels. Input items a_1 and a_5 are the seeds, the border between them is at $p_1 = 2$. The little blocks at the end of the arcs depict the values of *block_left* and *block_right*, respectively.

The blocking of a complete item is based on its relation to the seeds. Items dominating a seed, i.e., items whose yield contains at least one seed, are blocked on both sides and will only be extended by the projection step in the procedure *add_complete* below. This is the case for all seed items, but also for the item $(C, 0, 2)$ which projects to the incomplete item $(S \rightarrow \bullet\, C \bullet B\, a_5\, a_6, 0, 2)$. Complete items in a right substring, like the items labeled B, A, or a_3, are blocked at the left side and can therefore only combine with active items to their right. Thus, items in a right substring will be built right-to-left starting at the seed. Complete items in a left substring are treated analogously.

An unusual feature of this algorithm is that two incomplete items can be combined (see the second and fourth **for** loop in procedure *add_incomplete*), while other chart parsing algorithms only allow the combination of an incomplete with a complete item. At the borders, these are the only possible combinations, since all complete items have been blocked.

When incomplete items are created, they can at first extend in both directions, except for those where one of the dots is at its outermost position. Incomplete items are blocked when combined with another item for the first time. If they combine to the right, they will be forced to combine to the right from that time on by blocking them at the left side, and vice versa.

In figure 2, incomplete items $(S \rightarrow \bullet\, C \bullet B\, a_5\, a_6, 0, 2)$ and $(S \rightarrow C \bullet B\, a_5\, a_6 \bullet, 2, 6)$ have been combined to the complete item $(S, 0, 6)$ and were then blocked at the left resp. right side, with no effect because of the dot positions. Alternatively, $(S \rightarrow \bullet C \bullet B a_5 a_6, 0, 2)$ and $(S \rightarrow C \bullet B a_5 \bullet a_6, 2, 5)$ could have been combined. The second item would have been blocked at the right side (instead of left as in the figure) and $(S \rightarrow C \bullet B a_5 a_6 \bullet, 2, 6)$ could not have been built. Instead, the resulting incomplete item $(S \rightarrow \bullet C B a_5 \bullet a_6, 0, 5)$ would combine with the input item $(a_6, 5, 6)$ to produce $(S, 0, 6)$.

This mechanism synchronizes incomplete items, especially those that cross region borders, which guarantees that items whose yield contains more than one seed are built in exactly one manner.

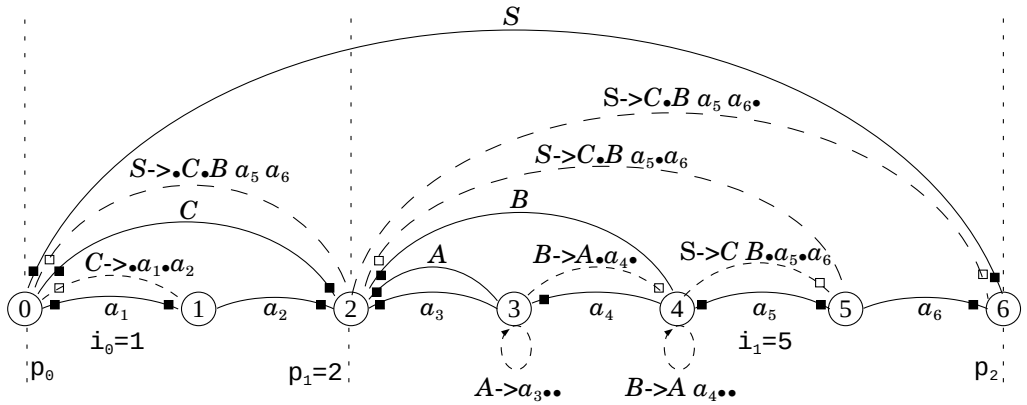


Figure 2: A decorated parse chart generated by the original algorithm. $a_1, \ldots, a_6$ are the terminal (input) items, A, B, C and S the nonterminals, p_0, p_1, p_2 the region borders, i_0 and i_1 the seed indices

```
proc add_complete(NT, i, j) ≡
  if i < i_h ≤ j for some h then
    /* seed dominating: block both sides */
    block_right(i, j) := block_right(i, j) ∪ {NT}
    block_left(i, j) := block_left(i, j) ∪ {NT}
    /* project step: add D_r → ... Z_{r,k} • NT • Z_{r,k+2} ... */
    for I_r^{k,k+1} ∈ I_G with Z_{r,k+1} = NT do
      add_new(I_r^{k,k+1}, i, j)
  else
    if p_{h-1} < i < j < i_h for some h then
      /* right substring : block complete item left */
      block_left(i, j) := block_left(i, j) ∪ {NT}
      /* combine with D_r → ... NT • Z_{r,k+1} ... • ... */
      for (I_r^{k,l}, j, m) ∈ t_{j,m}
          with Z_{r,k} = NT ∧ I_r^{k,l} ∉ block_left(j, m) do
        block_right(j, m) := block_right(j, m) ∪ {I_r^{k,l}}
        add_new(I_r^{k-1,l}, i, m)
    else /* i_h ≤ i < j ≤ p_{h+1} for some h */
      /* left substring : block complete item right */
      block_right(i, j) = block_right(i, j) ∪ {NT}
      /* combine with D_r → ... • ... Z_{r,l} • NT ... */
      for (I_r^{k,l}, m, i) ∈ t_{m,i}
          with Z_{r,l+1} = NT ∧ I_r^{k,l} ∉ block_right(m, i) do
        block_left(m, i) := block_left(m, i) ∪ {I_r^{k,l}}
        add_new(I_r^{k,l+1}, m, j)
end

proc add_incomplete(I_r^{k,l}, i, j) ≡
  /* I_r^{k,l} : D_r → ··· Z_{r,k} • Z_{r,k+1} ··· Z_{r,l} • Z_{r,l+1} ··· */
  if p_{h-1} ≤ i < i_h for some h then l-predict(I_r^{k,l}, i)
  if i_h ≤ j ≤ p_h for some h then r-predict(I_r^{k,l}, j)
  if k > 0 ∧ I_r^{k,l} ∉ block_left(i, j) then
    /* combine to the left with complete items */
    for (Z_{r,k}, m, i) ∈ t_{m,i} with Z_{r,k} ∉ block_right(m, i) do
      block_right(i, j) := block_right(i, j) ∪ {I_r^{k,l}}
      add_new(I_r^{k-1,l}, m, j)
    /* combine to the left with incomplete items */
    for (I_r^{n,k}, m, i) ∈ t_{m,i} with I_r^{n,k} ∉ block_right(m, i) do
      /* block both incomplete items appropriately */
      block_left(m, i) := block_left(m, i) ∪ {I_r^{n,k}}
      block_right(i, j) := block_right(i, j) ∪ {I_r^{k,l}}
      add_new(I_r^{n,l}, m, j)
  if l < π_r ∧ I_r^{k,l} ∉ block_right(i, j) then
    /* combine to the right with complete items */
    for (Z_{r,l+1}, j, m) ∈ t_{j,m} with Z_{r,l+1} ∉ block_left(j, m) do
      block_left(i, j) := block_left(i, j) ∪ {I_r^{k,l}}
      add_new(I_r^{k,l+1}, i, m)
    /* combine to the right with incomplete items */
    for (I_r^{l,n}, j, m) ∈ t_{j,m} with I_r^{l,n} ∉ block_left(j, m) do
      block_left(i, j) := block_left(i, j) ∪ {I_r^{k,l}}
      block_right(j, m) := block_right(j, m) ∪ {I_r^{l,n}}
      add_new(I_r^{k,n}, i, m)
end

proc add_new(X, i, j) ≡
  if X = I_r^{0,π_r} then
    if D_r = S ∧ i = 0 ∧ j = L then exit(accept)
    if (D_r, i, j) ∉ t_{i,j} then
      t_{i,j} := t_{i,j} ∪ {(D_r, i, j)}
      add_complete(D_r, i, j)
  else
    if (X, i, j) ∉ t_{i,j} then
      t_{i,j} := t_{i,j} ∪ {(X, i, j)}
      add_incomplete(X, i, j)
end

proc l-predict(I_r^{k,l}, i) ≡
  if Z_{r,k} ∈ N ∧ Z_{r,k} ∉ predict_left(i) then
    predict_left(i) := predict_left(i) ∪ {Z_{r,k}}
    for I_u^{π_u,π_u} with D_u = Z_{r,k} do
      add_incomplete(I_u^{π_u,π_u}, i, i)
      l-predict(I_u^{π_u,π_u}, i)
end

proc r-predict(I_r^{k,l}, i) ≡
  if Z_{r,l+1} ∈ N ∧ Z_{r,l+1} ∉ predict_right(i) then
    predict_right(i) := predict_right(i) ∪ {Z_{r,l+1}}
    for I_u^{0,0} with D_u = Z_{r,l+1} do
      add_incomplete(I_u^{0,0}, i, i)
      r-predict(I_u^{0,0}, i)
end
```

The procedures *l-predict* and *r-predict* recursively generate top down predictions for an incomplete item, to both sides, if the item is dominating a seed, to the left, if it is in a right substring, and to the right otherwise. They keep track of the predictions generated so far using two arrays of length $L + 1$,

storing the nonterminals for which left or right predictions have been introduced at a specific chart node.

The loops at chart node 3 and 4 in figure 2 have been generated by *l-predict*. Items that lie completely in right or left substrings stem from these top down predictions, like the item labeled with $B \rightarrow A \bullet a_4 \bullet$ or the complete item with label B.

[Satta and Stock, 1994] give a more formal description of the algorithm, including an invariant describing its behaviour.

3 Modified Algorithm

In the modified algorithm, instead of fixing seed and border indices in advance, every chart item is assigned a *state*, which is one of right substring, left substring or seed dominating (*right*, *left* and *seed* in the pseudo-code, respectively). Additionally, complete items with a terminal category, i.e., input items, can have neutral state, in fact, they are given this state during initialization.

Because the search strategy of the parser shall be adaptable, a priority is assigned to every item, which is used in connection with a priority queue (an *agenda*) to expand the best items first. The assignment of priority values is omitted here for the sake of clarity.

During initialization, all input items are added to the chart, their state is set to neutral and they are added to the priority queue. Parsing then continues by taking the highest ranked item from the priority queue and expanding it. A seed is selected when a neutral terminal item is retrieved from the queue. Its state is updated to *seed dominating*, i.e., the item itself becomes a seed. This puts seed selection on a level with the expansion of items, simplifying the implementation of a search strategy, owing to uniformity.

If terminal items are neutral when they are combined with another item in the first or third **for** loop of the modified *add_incomplete* procedure, they change state accordingly, either to *left* or *right*, depending on whether the incomplete item grew to the left, in which case the item is now member of a right substring, or vice versa. When such a terminal item is retrieved from the priority queue later during parsing, its state is already set and it does not become a seed.

Any other complete or incomplete combined items inherit their state from their daughters: if at least one of the daughters is seed dominating, the new item becomes seed dominating too, otherwise all daughters are members of the same substring, and the new item gets assigned the same state.

All conditionals that use the seed and border indices in the original algorithm are replaced by conditionals checking the state of the items. As a consequence, the seed and border indices are no longer needed.

Instead of a string with L elements, the parser gets a word graph as input. A word graph is an acyclic directed graph W of terminal items (a, i, j) with exactly one source and one sink node (nodes with in-degree resp. out-degree zero). The start and end node indices of the input items are typically in topological order, so that the source node gets index zero and the sink the maximal end node index of all input items, which in the modified version becomes the value of L.

Parsing stops when either a complete derivation was found or the priority queue becomes empty, which means that the word graph must be rejected. Since all input items were added to the priority queue in the beginning, it is also guaranteed that every sub-path of the word graph has been processed properly if parsing should stop with a failure. Every input item will then have a non-neutral state, which means that it at least took part in some of the derivations.

The procedures *l-predict* and *r-predict* are the same as in the original algorithm, and are omitted here.

proc *add_complete*(NT, i, j) $\equiv$
  if *state*$(NT, i, j) =$ *seed* **then**
    /* project step: add $D_r \rightarrow \ldots Z_{r,k} \bullet NT \bullet Z_{r,k+2} \ldots$ */
    for $I_r^{k,k+1} \in I_G$ *with* $Z_{r,k+1} = NT$ **do**
      add_new$(I_r^{k,k+1}, i, j,$ *seed*, *seed*$)$
  elsif *state*$(NT, i, j) =$ *right* **then**
    /* combine with $D_r \rightarrow \ldots NT \bullet Z_{r,k+1} \ldots \bullet \ldots$ */
    for $(I_r^{k,l}, j, m) \in t_{j,m}$
        with $Z_{r,k} = NT \wedge I_r^{k,l} \notin$ *block_left*(j, m) **do**
      add_new$(I_r^{k-1,l}, i, m,$ *right*, *state*$(I_r^{k,l}, j, m))$
      block_right$(j, m) :=$ *block_right*$(j, m) \cup \{I_r^{k,l}\}$
  elsif *state*$(NT, i, j) =$ *left* **then**
    /* combine with $D_r \rightarrow \ldots \bullet \ldots Z_{r,l} \bullet NT \ldots$ */
    for $(I_r^{k,l}, m, i) \in t_{m,i}$
        with $Z_{r,l+1} = NT \wedge I_r^{k,l} \notin$ *block_right*(m, i) **do**
      add_new$(I_r^{k,l+1}, m, j,$ *left*, *state*$(I_r^{k,l}, m, i))$
      block_left$(m, i) :=$ *block_left*$(m, i) \cup \{I_r^{k,l}\}$
end

proc *add_incomplete*$(I_r^{k,l}, i, j)$ $\equiv$
  /* $I_r^{k,l} : D_r \rightarrow \cdots Z_{r,k} \bullet Z_{r,k+1} \cdots Z_{r,l} \bullet Z_{r,l+1} \cdots$ */
  if *state*$(I_r^{k,l}, i, j) \in \{$*seed*, *right*$\}$ **then** *l-predict*$(I_r^{k,l}, i)$
  if *state*$(I_r^{k,l}, i, j) \in \{$*seed*, *left*$\}$ **then** *r-predict*$(I_r^{k,l}, j)$
  if $k > 0 \wedge I_r^{k,l} \notin$ *block_left*(i, j) **then**
    /* combine to the left with complete items */
    for $(Z_{r,k}, m, i) \in t_{m,i}$
        with state$(Z_{r,k}, m, i) \in \{$*right*, *neutral*$\}$ **do**
      if *state*$(Z_{r,k}, m, i) =$ *neutral*
        then *state*$(Z_{r,k}, m, i) :=$ *right*
      add_new$(I_r^{k-1,l}, m, j,$ *state*$(Z_{r,k}, m, i),$ *state*$(I_r^{k,l}, i, j))$
      block_right$(i, j) :=$ *block_right*$(i, j) \cup \{I_r^{k,l}\}$
    /* combine to the left with incomplete items */
    for $(I_r^{n,k}, m, i) \in t_{m,i}$ *with* $I_r^{n,k} \notin$ *block_right*(m, i) **do**
      add_new$(I_r^{n,l}, m, j,$ *state*$(I_r^{n,k}, m, i),$ *state*$(I_r^{k,l}, i, j))$
      block_left$(m, i) :=$ *block_left*$(m, i) \cup \{I_r^{n,k}\}$
      block_right$(i, j) :=$ *block_right*$(i, j) \cup \{I_r^{k,l}\}$
  if $l < \pi_r \wedge I_r^{k,l} \notin$ *block_right*(i, j) **then**
    /* combine to the right with complete items */
    for $(Z_{r,l+1}, j, m) \in t_{j,m}$
        with state$(Z_{r,l+1}, j, m) \in \{$*left*, *neutral*$\}$ **do**
      if *state*$(Z_{r,l+1}, j, m) =$ *neutral*
        then *state*$(Z_{r,l+1}, j, m) :=$ *left*
      add_new$(I_r^{k,l+1}, i, m,$ *state*$(I_r^{k,l}, i, j),$ *state*$(Z_{r,l+1}, j, m))$
      block_left$(i, j) :=$ *block_left*$(i, j) \cup \{I_r^{k,l}\}$
    /* combine to the right with incomplete items */
    for $(I_r^{l,n}, j, m) \in t_{j,m} \wedge \neg$*block_left*$(I_r^{l,n}, j, m)$ **do**
      add_new$(I_r^{k,n}, i, m,$ *state*$(I_r^{k,l}, i, j),$ *state*$(I_r^{l,n}, j, m))$
      block_left$(i, j) :=$ *block_left*$(i, j) \cup \{I_r^{k,l}\}$
      block_right$(j, m) :=$ *block_right*$(j, m) \cup \{I_r^{l,n}\}$
end

```
proc add_new(X, i, j, state1, state2) ≡
  if X = I_r^{0,π_r} then X := D_r
  if X = S ∧ i = 0 ∧ j = L then exit(accept)
  if (X, i, j) ∉ t_{i,j} then
    if state1 = seed ∨ state2 = seed then
      state(X, i, j) := seed
    else
      if state1 = right
        then state(X, i, j) := right
      else state(X, i, j) := left
    t_{i,j} := t_{i,j} ∪ {(X, i, j)}
    push((X, i, j), p_queue)
end

proc main(W) ≡
  for (a, i, j) ∈ W do
      t_{i,j} := t_{i,j} ∪ {(a, i, j)}
      state(a, i, j) := neutral
      push((a, i, j), p_queue)
  while ¬empty(p_queue) do
        (X, i, j) := pop_max(p_queue)
        if X ∈ Σ ∧ state(X, i, j) = neutral then
          state(X, i, j) := seed
        if X ∈ N ∪ Σ then add_complete(X, i, j)
        else add_incomplete(X, i, j)
  exit(reject)
end
```

4 Correctness of the modified algorithm

If W contains only string input and priorities are set appropriately to select the right seeds, the modified algorithm works like the original. This is achieved by using the maximal priority value for all seed input items, the minimal value for all other input items, and priority values strictly between these values for all other items. Thus, seeds will be considered first and appropriately marked, and other input items will be considered after all possible combinations have been tried in some order, which is consistent with the original.

It remains to be shown that in case of true word graph input, the algorithm will still be correct and redundancy-free. New situations arise from the fact that there are parallel subpaths of neutral input items to previously treated regions of the chart, and new, possibly derived items can now interact with existing ones created from previous expansions.

The argumentation will be based on the respective properties of the original, which can not be shown here. It is clear that sub-derivations with equal span and item label may be produced because of the ambiguity in the word graph input. These items are not redundant because they have different yields. The type of redundancy that must be avoided is the multiple creation of *identical* items with both identical derivation and yield, which could be produced by an incorrect implementation of the island algorithm due to bidirectionality and multiple seeds. Redundancy can therefore only occur relative to a path through the word graph.[1]

Although there are six cases in total to be considered (three each for left and right substrings), the treatment of left and right substrings will be completely analogous, so we will content ourselves with the discussion of the former.

[1]For an in-depth description of the redundancy problem on strings and formal proofs, see [Satta and Stock, 1994]

4.1 A new left substring ends in a right substring

In this situation, which is depicted in fig. 3, node j behaves like a new border node between seed s_0 and s_2. Because of the completeness of the original algorithm, all possible derivations compatible with the seed s_0 must be available at node j, although some of them may be blocked. Assume we lose a complete derivation because of an indispensable incomplete item that is blocked on the left side (like the item labeled A in fig. 3). If this is the case, there must be ancestor items of A whose creation caused the blocking. One of these ancestors, ultimately the one that ends in the sink node[2], is available for combination at node j, which is a contradiction.

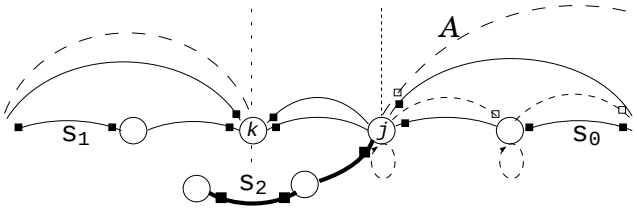


Figure 3: A new left substring ends in a right substring. The thick arcs are items that are expanded later in the parsing process. k is the old border node.

As was argued above, redundant items can only occur relative to a path of input items through the word graph. Since the same synchronization of items was used for the path through s_2 and s_0 with border j as in the string method, the chart must be free of redundancy for this path too.

A special case of the configuration described in this section is given when the new sub-path hits the old border node (e.g., node k in fig. 3). In this case, it is obvious that all, and only the correct derivations will be created.

4.2 A new left substring ends in a left substring

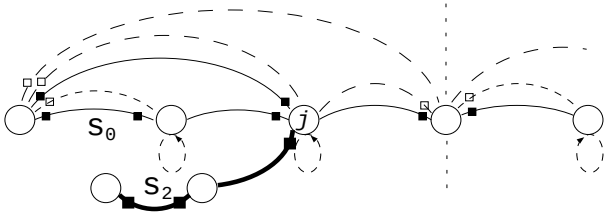


Figure 4: A new left substring ends in a left substring

All complete items starting at node j in figure 4 are available to the new sub-path. Every derivation starting at j that is compatible with the new seed s_2 but not with the old one (s_0) will be constructed by the appropriate predictions and expansions, and since the predict methods keep track of which nonterminals have already been predicted, no work is duplicated and thus, no redundancy is produced. For blocked incomplete items, the same argumentation as in 4.1 applies, which guarantees completeness.

[2]Incomplete items ending in the sink node can not be blocked at the left side because there is no item to the right they can combine with.

4.3 An alternative left substring path overruns a seed

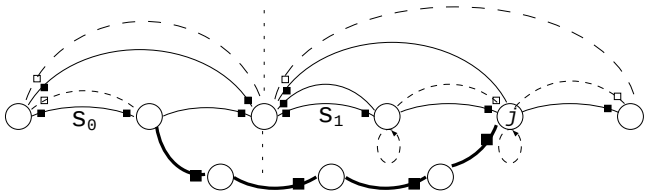


Figure 5: A seed is overrun by an alternative left substring

This situation, which is shown in figure 5, is almost the same as in section 4.2, except that the new items ending in node j do not come from a new seed on the parallel path, but from an alternative path from seed s_0. Therefore, the same argumentation applies as in 4.2 above.

4.4 Worst Case Complexity

The number of symbols $I_r^{k,l}$ is bounded by $\pi_G^2|P|$ where π_G is the maximal length of a rule $r \in P$. On the chart, and on the agenda, can not be more than $n^2(|G| + \pi_G^2|P|)$ items, where $n \leq |W|$ is the highest chart index and $|G| = |N \cup \Sigma|$.

The function *add_complete* is called at most $n^2|G|$ times, and the second and third **for** loops in this function are executed at most $n\pi_G^2|P|$ times, while the first is executed at most $\pi_G|P|$ times, which makes this function $O(n^3|G|\pi_G^2|P|)$.

The function *add_incomplete* is called at most $n^2\pi_G^2|P|$ times. The second and fourth **for** loops, where active items are combined, can be executed at most $n\pi_G|P|$ times, which makes the whole function $O(n^3\pi_G^3|P|)$, while the first and third are executed at most $n|G|$ times.

These two functions clearly dominate the prediction functions, which makes the whole algorithm $O(n^3|G|\pi_G^2|P|)$ or $O(n^3\pi_G^3|P|)$, whichever is dominant.

5 Conclusion and further considerations

An efficient island parsing algorithm for string input was generalized to make it more feasible for the use in speech applications. The new version deals with word graphs as input without losing the beneficial properties of the original. It also integrates the selection of seeds into the parser's search process, which, in addition to more uniformity, provides the user with more flexibility in the design of the search strategy.

The data structures for blocking and keeping the state of an item can be implemented as bit vectors, which produces minimal space and time overhead for all the blocking and state conditionals.

The modified algorithm has been implemented for context free grammars with annotated feature structures. This implementation also provides pluggable search strategies to facilitate experimentation.

From the point of view of the search strategy, the atomic action of the modified algorithm (one *parsing task*) is the expansion of an item, like, for example, in [Caraballo and Charniak, 1998]. To be able to define a more fine grained strategy, the parser could be changed such that the tasks are instead combination of two items, projection and prediction, or a subset of the three ([Kay, 1986], [Erbach, 1991]).

The price to pay for the increased flexibility is a larger agenda, maybe prohibitively large, if the word graphs are big and/or the grammar is highly ambiguous. The changes to the algorithm are obvious, and it will depend on the specific task, whether the more elaborate search strategy will achieve better results or improved parsing efficiency.

Acknowledgments

I am very indebted to Giorgio Satta for his help in fully understanding the bits and pieces of the original algorithm, for the discussions and for his patience. I also want to thank Berthold Crysmann and Melanie Siegel for their help in preparing this paper and the anonymous reviewers for their constructive comments. This research was supported by the German Ministry for Education and Research under grant no. 01 IM D01, to the project SmartWeb.

References

[Ageno, 2003] A. Ageno. *An Island-Driven Parsing System*. PhD thesis, Universitat Politècnica de Catalunya, 2003.

[Brietzmann, 1992] A. Brietzmann. "Reif für die Insel". Syntaktische Analyse natürlich gesprochener Sprache durch bidirektionales Chart-Parsing. In H. Mangold, editor, *Sprachliche Mensch-Maschine-Kommunikation*. Oldenbourg, München; Wien, 1992.

[Caraballo and Charniak, 1998] S. Caraballo and E. Charniak. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298, 1998.

[Erbach, 1991] G. Erbach. An environment for experimentation with parsing strategies. In *Proc. of the 12th Int. Conf. on AI*, pages 931–936, 1991.

[Gallwitz *et al.*, 1998] F. Gallwitz, M. Aretoulaki, M. Boros, J. Haas, S. Harbeck, R. Huber, H. Niemann, and E. Nöth. The Erlangen Spoken Dialogue System EVAR: A State–of–the–Art Information Retrieval System. In *Proc. of ISSD 98)*, pages 19–26, Sydney, Australia, 1998.

[Kay, 1986] Martin Kay. Algorithm schemata and data structures in syntactic processing. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*, pages 35–70. Kaufmann, Los Altos, CA, 1986.

[Mecklenburg *et al.*, 1995] K. Mecklenburg, P. Heisterkamp, and G. Hanrieder. A robust parser for continuous spoken language using prolog. In *Proc. of NLULP 95*, pages 127–141, Lisbon, Portugal, 1995.

[Satta and Stock, 1994] G. Satta and O. Stock. Bidirectional context-free grammar parsing for natural language processing. *Artifical Intelligence*, 69:123–164, 1994.

[Thanopoulos *et al.*, 1997] A. Thanopoulos, N. Fakotakis, and G. Kokkinakis. Linguistic processor for a spoken dialogue system based on island parsing techniques. In *Proc. of 5th Eurospeech*, volume 4, pages 2259–2262, 1997.

Automatic Evaluation of Text Coherence: Models and Representations

Mirella Lapata
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, UK
mlap@inf.ed.ac.uk

Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
G-468, the Stata Center
Cambridge, MA 14853
regina@csail.mit.edu

Abstract

This paper investigates the automatic evaluation of text coherence for machine-generated texts. We introduce a fully-automatic, linguistically rich model of local coherence that correlates with human judgments. Our modeling approach relies on shallow text properties and is relatively inexpensive. We present experimental results that assess the predictive power of various discourse representations proposed in the linguistic literature. Our results demonstrate that certain models capture complementary aspects of coherence and thus can be combined to improve performance.

1 Introduction

The use of automatic methods for evaluating machine-generated text is quickly becoming mainstream in machine translation, text generation, and summarization. These methods target various dimensions of text quality, ranging from sentence grammaticality to content selection [Bangalore *et al.*, 2000; Papineni *et al.*, 2002; Hovy and Lin, 2003]. However, a fundamental aspect of text structure — its *coherence* — is still evaluated manually.

Coherence is a property of well-written texts that makes them easier to read and understand than a sequence of randomly strung sentences. Although the same information can be organized in multiple ways to create a coherent text, some forms of text organization will be indisputably judged incoherent. The automatically-generated summary in Table 1 is an example of a badly organized text: the presence of thematically unrelated sentences and non-canonical time sequences makes it nearly impossible to comprehend. We want to develop an automatic method that can distinguish such a summary from a text with no coherence violations.

This paper focuses on *local coherence*, which captures text organization at the level of sentence to sentence transitions, and is undoubtedly necessary for global coherence. This topic has received much attention in the linguistic literature [Grosz *et al.*, 1995; Morris and Hirst, 1991; Halliday and Hasan, 1976] and a variety of qualitative models have been proposed. The emphasis of our work is on *quantitative* models of local coherence that can be efficiently computed from raw text, and readily utilized for automatic evaluation. Our approach relies on shallow text properties that can be easily identified and quantified without recourse to elaborate world knowledge and handcrafted rules.

Newspapers reported Wednesday that three top Libyan officials have been tried and jailed in the Lockerbie case. Secretary-General Kofi Annan said Wednesday that he may travel to Libya next week in hopes of closing a deal. The sanctions were imposed to force Libyan leader Moammar Gadhafi to turn the men over. Louis Farrakhan, the leader of a U.S. Muslim group, met with Gadhafi and congratulated him on his recovery from a hip injury.

Table 1: A low-coherence summary

The coherence models described in this paper fall into two broad classes that capture orthogonal dimensions of entity distribution in discourse. The first class incorporates *syntactic* aspects of text coherence and characterizes how mentions of the same entity in different syntactic positions are spread across adjacent sentences. Inspired by Centering Theory [Grosz *et al.*, 1995], our algorithm abstracts a collection of coherent texts into a set of entity transition sequences, and defines a probabilistic model over their distribution. Given a new text, the model evaluates its coherence by computing the probability of its entity transitions according to the training data.

The second, *semantic*, class of models quantifies local coherence as the degree of connectivity across text sentences. This approach is motivated by the findings of Halliday and Hasan [1976] who emphasized the role of lexical cohesion in text coherence. The key parameter of these models is the definition of semantic relatedness (as a proxy for connectivity) among text entities. We explore a wealth of similarity measures, ranging from distributional to taxonomy-based, to find an optimal lexico-semantic representation for the coherence assessment task.

We employ our models to evaluate the coherence of multidocument summaries produced by systems that participated in the Document Understanding Conference. We acquire coherence ratings for a large collection of machine-generated summaries by eliciting judgments from human subjects, and examine the extent to which the predictions of various models correlate with human intuitions. Our experiments demonstrate that while several models exhibit a statistically significant agreement with human ratings, a model that fuses the syntactic and the semantic views yields substantial improvement over any single model. Our contributions are twofold:

Modeling We present a fully automatic, linguistically rich model of local coherence that correlates with human judgments. This model is the first attempt, to our knowledge, to automatically evaluate the local coherence of machine-generated texts.

Linguistic Analysis We present experimental results that assess the predictive power of various knowledge sources and discourse representations proposed in the linguistic literature. In particular, we show that certain models capture complementary aspects of coherence and thus can be combined to improve performance.

2 Related Work

Most of the work on automatic coherence evaluation has been done in the context of automatic essay scoring. Higgins *et al.* [2004] develop a system that assesses global aspects of coherence in student essays. They use a manually annotated corpus of essays to learn which types of discourse segments can cause breakdowns in coherence. Other approaches focus on local coherence. Miltsakaki and Kukich [2004] manually annotate a corpus of student essays with entity transition information, and show that the distribution of transition types correlates with human grades. Foltz *et al.* [1998] propose a model of local coherence that presupposes no manual coding. A text is considered coherent if it exhibits a high degree of meaning overlap between adjacent sentences. They employ a vector-based representation of lexical meaning and assess semantic relatedness by measuring the distance between sentence pairs. Foltz *et al.* report that the model correlates reliably with human judgments and can be used to analyze discourse structure. The success of this approach motivates our research on semantic association models of coherence.

Previous work has primarily focused on human authored texts and has typically utilized a single source of information for coherence assesment. In contrast, we concentrate on machine generated texts and assess which knowledge sources are appropriate for measuring local coherence. We introduce novel models but also assess whether previously proposed ones (e.g., Foltz *et al.* [1998]) generalize to automatically generated texts. As a byproduct of our main investigation, we also examine whether humans can reliably rate texts in terms of coherence, thus undertaking a large-scale judgment elicitation study.

3 Models of Local Coherence

In this section we introduce two classes of coherence models, and describe how they can be used in automatic evaluation. We motivate their construction, present a corresponding discourse representation, and an inference procedure.

3.1 The Syntactic View

Linguistic Motivation Centering theory [Grosz *et al.*, 1995; Walker *et al.*, 1998] is one of the most influential frameworks for modeling local coherence. Fundamental in Centering's study of discourse is the way entities are introduced and discussed. The theory asserts that discourse segments in which successive utterances mention the same entities are *more coherent* than discourse segments in which multiple entities are discussed. Coherence analysis revolves around *patterns of local entity transitions* which specify how the focus of discourse changes from sentence to sentence. The key assumption is that certain types of entity transitions are likely to appear in locally coherent discourse. Centering also establishes constraints on the linguistic realization of focus, suggesting that focused entities are likely to occupy prominent syntactic positions such as subject or object.

1. [Former Chilean dictator Augusto Pinochet]$_{\mathbf{O}}$, was arrested in [London]$_{\mathbf{X}}$ on [14 October]$_{\mathbf{X}}$ 1998.
2. [Pinochet]$_{\mathbf{S}}$, 82, was recovering from [surgery]$_{\mathbf{X}}$.
3. [The arrest]$_{\mathbf{S}}$ was in [response]$_{\mathbf{X}}$ to [an extradition warrant]$_{\mathbf{X}}$ served by [a Spanish judge]$_{\mathbf{S}}$.
4. [Pinochet]$_{\mathbf{O}}$ was charged with murdering [thousands]$_{\mathbf{O}}$, including many [Spaniards]$_{\mathbf{O}}$.
5. [Pinochet]$_{\mathbf{S}}$ is awaiting [a hearing]$_{\mathbf{O}}$, [his fate]$_{\mathbf{X}}$ in [the balance]$_{\mathbf{X}}$.
6. [American scholars]$_{\mathbf{S}}$ applauded the [arrest]$_{\mathbf{O}}$.

Table 2: Summary augmented with syntactic annotations for entity grid computation.

	Dictator	Augusto	Pinochet	London	October	Surgery	Arrest	Extradition	Warrant	Judge	Thousands	Spaniards	Hearing	Fate	Balance	Scholars	
1	**O**	**O**	**O**	**X**	**X**	–	–	–	–	–	–	–	–	–	–	–	1
2	–	–	**S**	–	–	**X**	–	–	–	–	–	–	–	–	–	–	2
3	–	–	–	–	–	–	**S**	**X**	**X**	**S**	–	–	–	–	–	–	3
4	–	–	**O**	–	–	–	–	–	–	–	**O**	**O**	–	–	–	–	4
5	–	–	**S**	–	–	–	–	–	–	–	–	–	**O**	**X**	**X**	–	5
6	–	–	–	–	–	–	**O**	–	–	–	–	–	–	–	–	**S**	6

Table 3: An entity grid

Discourse representation To expose entity transition patterns characteristic of coherent texts, we represent a text by an *entity grid*. The grid's columns correspond to discourse entities, while the rows correspond to utterances (see Table 3). We follow Miltsakaki and Kukich [2004] in assuming that an utterance is a traditional sentence (i.e., a main clause with accompanying subordinate and adjunct clauses).

Grid columns record an entity's presence or absence in a sequence of sentences $(S_1, \ldots, S_n)$. More specifically, each grid cell represents the role r_{ij} of entity e_j in a given sentence S_i. Grammatical roles reflect whether an entity is a subject (**s**), object (**o**), neither (**x**) or simply absent (**–**). Table 3 illustrates an entity grid constructed for the text in Table 2. Since the text contains six sentences, the grid columns are of length six. As an example consider the grid column for the entity *arrest*, **[– – s – – o]**. It records that *arrest* is present in sentence 3 as a subject and in sentence 6 as an object, but is absent from the rest of the sentences.

Ideally, each entity in the grid should represent an equivalence class of coreferent nouns (e.g., *Former Chilean dictator Augusto Pinochet* and *Pinochet* refer to the same entity). However, the automatic construction of such an entity grid requires a robust coreference tool, able to accurately process texts with coherence violations. Since coreference tools are typically trained on coherent texts, this requirement is hard to satisfy. Instead, we employ a simplified representation: each noun in a text corresponds to a different entity in the grid. The

simplification allows us to capture noun-coreference, albeit in a shallow manner — only exact repetitions are considered coreferent. In practice, this means that named entities and compound nouns will be treated as denoting more than one entity. For instance, the NP *Former Chilean dictator Augusto Pinochet* will be mapped to three entities: *dictator*, *Augusto*, and *Pinochet*. We further assume that each noun within an NP bears the same grammatical role as the NP head. Thus, all three nouns in the above NP will be labeled as objects. When a noun is attested more than once with a different grammatical role in the same sentence, we default to the role with the highest grammatical ranking (i.e., $\mathbf{s} > \mathbf{o} > \mathbf{x}$).

Entity grids can be straightforwardly computed provided that an accurate parser is available. In the experiments reported throughout this paper we employed Collins' [1998] state-of-the-art statistical parser to identify discourse entities and their grammatical roles. Entities involved in passive constructions were identified using a small set of patterns and their corresponding deep grammatical role was entered in the grid (see the grid cell **o** for *Pinochet*, Sentence 1, Table 3).

Inference A fundamental assumption underlying our inference mechanism is that the distribution of entities in coherent texts exhibit certain regularities reflected in the topology of grid columns. Grids of coherent texts are likely to have some dense columns (i.e., columns with just a few gaps such as *Pinochet* in Table 3) and many sparse columns which will consist mostly of gaps (see *London, judge* in Table 3). One would further expect that entities corresponding to dense columns are more often subjects or objects. These characteristics will be less pronounced in low-coherence texts.

We define the coherence of a text T $(S_1,\ldots,S_n)$ with entities $e_1 \ldots e_m$ as a joint probability distribution that governs how entities are distributed across document sentences:

$$P_{coherence}(T) = P(e_1 \ldots e_m; S_1 \ldots S_n) \tag{1}$$

We further assume (somewhat simplistically) that an entity is selected into a document independently of other entities:

$$P_{coherence}(T) \approx \prod_{j=1}^{m} P(e_j; S_1 \ldots S_n) \tag{2}$$

To give a concrete example, we will rate the coherence of the text in Table 3 by multiplying together the probabilities of the entities *Dictator*, *Augusto*, *Pinochet*, etc.

We define $P(e_j; S_1 \ldots S_n)$ as a probability distribution over transition sequences for entity e_j across all n sentences of text T. $P(e_j; S_1 \ldots S_n)$ is estimated from grid columns, as observed in a corpus of coherent texts:

$$\begin{aligned} P(e_j; S_1 \ldots S_n) &= P(r_{1,j} \ldots r_{n,j}) \\ &= \prod_{i=1}^{n} P(r_{i,j} | r_{1,j} \ldots r_{(i-1),j}) \\ &\approx \prod_{i=1}^{n} P(r_{i,j} | r_{(i-h),j} \ldots r_{(i-1),j}) \end{aligned} \tag{3}$$

where $r_{i,j}$ represents the grammatical role for entity e_j in sentence i (see the grid columns in Table 3). The estimates for $P(r_{i,j} | r_{1,j} \ldots r_{(i-1),j})$ are obtained using the standard Markov assumption of independence, where h is the history size. Assuming a first-order Markov model, $P(Pinochet; S_1 \ldots S_6)$ will be estimated by multiplying together $P(\mathbf{o})$, $P(\mathbf{s}|\mathbf{o})$, $P(-|\mathbf{s})$, $P(\mathbf{o}|-)$, $P(\mathbf{s}|\mathbf{o})$, and $P(-|\mathbf{s})$ (see the column for *Pinochet* in Table 3).

To compare texts with variable lengths and entities, the probabilities for individual columns are normalized by column length (n) and the probability of the entire text is normalized by the number of columns (m):

$$P_{coherence}(T) \approx \frac{1}{m \cdot n} \sum_{j=1}^{m} \sum_{i=1}^{n} \log P(r_{i,j} | r_{(i-h),j} \ldots r_{(i-1),j}) \tag{4}$$

Once the model is trained on a corpus of coherent texts, it can be used to assess the coherence of unseen texts. Texts that are given a high probability $P_{coherence}(T)$ will be deemed more coherent than those given low probability.

Notice that the model is unlexicalized, i.e., the estimation of $P(e_j; S_1 \ldots S_n)$ is not entity-specific. In practice, this means that entities with the same column topology (e.g., *London*, *October* in Table 3) will be given the same probability.

3.2 The Semantic View

Linguistic motivation An important factor in text comprehension is the degree to which sentences and phrases are linked together. The observation dates back to Halliday and Hasan [1976] who stressed the role of lexical cohesion in text coherence. A number of linguistic devices — entity repetition, synonymy, hyponymy, and meronymy — are considered to contribute to the "continuity of lexical meaning" observed in coherent text. Morris and Hirst [1991] represent lexical cohesion via *lexical chains*, i.e., sequences of related words spanning a topical text unit, thus formalizing the intuition that coherent units will have a high concentration of dense chains. They argue that the distribution of lexical chains is a surface indicator of the structure of coherent discourse.

Discourse representation The key premise behind lexical chains is that coherent texts will contain a high number of semantically related words. This allows for a particularly simple representation of discourse that does not take account of syntactic structure or even word order within a sentence. Each sentence is thus represented as a bag of words. These can be all words in the document (excluding function words) or selected grammatical categories (e.g., verbs or nouns). For our models, we assume that each sentence is represented by a set of nouns.

Central to this representation is the ability to measure semantic similarity. Different coherence models can be therefore defined according to the chosen similarity measure.

Inference Measuring local coherence amounts to quantifying the degree of semantic relatedness between sentences. Specifically, the coherence of text T is measured by taking the mean of all individual transitions.

$$coherence(T) = \frac{\sum_{i=1}^{n-1} sim(S_i, S_{i+1})}{n-1} \tag{5}$$

where $sim(S_i, S_{i+1})$ is a measure of similarity between sentences S_i and S_{i+1}.

We have experimented with three broad classes of models which employ *word-based*, *distributional*, and *taxonomy-based similarity measures*.

In its simplest form, semantic similarity can be operationalized in terms of word overlap:

$$sim(S_1, S_2) = \frac{2|words(S_1) \cap words(S_2)|}{(|words(S_1)| + |words(S_2)|)} \tag{6}$$

where $words(S_i)$ is the set of words in sentence i. The main drawback of this measure is that it will indicate low-coherence for sentence pairs that have no words in common, even though they may be semantically related.

Measures of distributional similarity, however, go beyond mere repetition. Words are considered similar if they occur within similar contexts. The semantic properties of words are captured in a multi-dimensional space by vectors that are constructed from large bodies of text by observing the distributional patterns of co-occurrence with their neighboring words. For modeling coherence, we want to be able to compare the similarity of sentences rather than words (see (4)). Our computation of sentence similarity follows the method described in Foltz [1998]: each sentence is represented by the mean (centroid) of the vectors of its words, and the similarity between two sentences is determined by the cosine of their means.

$$\begin{aligned} sim(S_1,S_2) &= \cos(\mu(\vec{S_1}),\mu(\vec{S_2})) \\ &= \frac{\sum_{j=1}^{n} \mu_j(\vec{S_1})\mu_j(\vec{S_2})}{\sqrt{\sum_{j=1}^{n}(\mu_j(\vec{S_1}))^2}\sqrt{\sum_{j=1}^{n}(\mu_j(\vec{S_2}))^2}} \end{aligned} \quad (7)$$

where $\mu(\vec{S_i}) = \frac{1}{|S_i|}\sum_{\vec{w}\in S_i}\vec{w}$, and $\vec{w}$ is the vector for word w.

An alternative to inducing word similarity relationships from co-occurrence statistics in a corpus, is to employ a manually crafted resource such as WordNet [Fellbaum, 1998]. WordNet-based similarity measures have been shown to correlate reliably with human similarity judgments and have been used in a variety of applications ranging from the detection of malapropisms to word sense disambiguation (see Budanitsky and Hirst [2001] for an extensive survey). We employed five measures commonly cited in the literature. Two of these measures [Hirst and St-Onge, 1998; Lesk, 1986] define similarity solely in terms of the taxonomy, whereas the other three are based on information theory [Resnik, 1995; Jiang and Conrath, 1997; Lin, 1998] and combine taxonomic information with corpus counts. By considering a broad range of these measures, we should be able to distill which ones are best suited for coherence assessment.

All WordNet-based measures compute a similarity score at the sense level. We define the similarity of two sentences S_1 and S_2 as:

$$sim(S_1,S_2) = \frac{\sum_{\substack{w_1\in S_1 \\ w_2\in S_2}} \operatorname{argmax}_{\substack{c_1\in senses(w_1) \\ c_2\in senses(w_2)}} sim(c_1,c_2)}{|S_1||S_2|} \quad (8)$$

where $|S_i|$ is the number of words in sentence i. Since the appropriate senses for words w_1 and w_2 are not known, our measure selects the senses which maximize $sim(c_1,c_2)$.

4 Collecting Coherence Judgments

To comparatively evaluate the coherence models introduced above, we first needed to establish an independent measure of coherence by eliciting judgments from human participants.

Materials and Design For optimizing and testing our models, we created a corpus representative of coherent and incoherent texts. More specifically, we elicited coherence judgments for multi-document summaries produced by systems and human writers for the Document Understanding Conference (DUC, 2003). Automatically generated summaries are prone to coherence violations [Mani, 2001] — sentences are often extracted out of context and concatenated to form a text — and are therefore a good starting point for coherence analysis.

We randomly selected 16 input document clusters[1] and included in our materials six summaries corresponding to each cluster. One summary was authored by a human, whereas five were produced by automatic summarization systems that participated in DUC. Thus the set of materials contained $6 \cdot 16 = 96$ summaries. From these, six summaries were reserved as a development set, whereas the remaining 90 summaries were used for testing (see Section 5).

Procedure and Subjects Participants first saw a set of instructions that explained the task, defined local coherence, and provided several examples. Then the summaries were presented; a new random order of presentation was generated for each participant. Participants were asked to use a seven point scale to rate how coherent the summaries were without having seen the source texts. The study was conducted remotely over the Internet and was completed by 177 unpaid volunteers (approximately 23 per summary), all native speakers of English.

The judgments were averaged to provide a single rating per summary and all further analyses were conducted on means. We report results on inter-subject agreement in Section 5.2.

5 Experiments

All our models were evaluated on the DUC summaries for which we elicited coherence judgments. We used correlation analysis to assess the degree of linear relationship between human ratings and coherence as estimated by our models. In this section we provide an overview of the parameters we explored, and present and discuss our results.

5.1 Parameter Estimation

Since the models we evaluated have different training requirements (e.g., some must be trained on a large corpus and others are not corpus specific), it was not possible to have a uniform training corpus for all models. Here, we give an overview of the data requirements for our models and explain how these were met in our experiments.

For the entity-based models, we need corpus data to select an appropriate history size and to estimate the probability of various entity transitions. History size was adjusted using the development corpus described above. Entity transition probabilities were estimated from a different corpus: we used 88 human summaries that were produced by DUC analysts. We assumed that human authored summaries were coherent and therefore appropriate for obtaining reliable estimates. The grid columns were augmented with the start and end symbols, increasing the size of the grid cell categories to six. We estimated the probabilities of individual columns using n-gram models (see (3)) of variable length (i.e., 2–4) smoothed with Witten Bell discounting.

[1] Summaries with ungrammatical sentences were excluded from our materials, to avoid eliciting judgments on text properties that were not coherence-related.

	Humans	Egrid	Overlap	LSA	HStO	Lesk	JCon	Lin
Egrid	.246*							
Overlap	.120	−.341**						
LSA	.230*	.042	.013					
HStO	.322**	.071	.093	.037				
Lesk	.125	.227	−.032	.098	.380**			
JCon	−.290**	−.392**	.485**	.035	.625**	.270*		
Lin	.173	.074	−.107	.053	.776**	.421**	.526**	
Resnik	.207	−.003	.052	−.063	.746**	.410**	.606**	.809**
*$p < .05$ (2-tailed) **$p < .01$ (2-tailed)								

Table 4: Correlation between human ratings and coherence models, measured by Pearson coefficient. Stars indicate the level of statistical significance.

Some of the models based on semantic relatedness provide a coherence score without presupposing access to corpus data. These models are based on the measures proposed by Hirst and St-Onge [1998], Lesk [1986] and on word overlap. Since they require no prior training, they were directly computed on a lemmatized version of the test set. Other semantic association models rely on corpus data to either automatically construct representations of lexical meaning [Foltz *et al.*, 1998] or to populate WordNet's representations with frequency counts [Resnik, 1995; Jiang and Conrath, 1997; Lin, 1998]. We used a lemmatized version of the North American News Text Corpus for this purpose. The corpus contains 350 million words of text taken from a variety of news sources and is therefore appropriate for modeling the coherence of news summaries. Vector-based representations were created using a term-document matrix. We used singular value decomposition [Berry *et al.*, 1994] to reduce the vector space to 100 dimensions and thus obtained a semantic space similar to Latent Semantic Analysis (LSA, Foltz *et al.* [1998]).

5.2 Results

The evaluation of our coherence models was driven by two questions: (a) What is the contribution of various linguistic sources to coherence modeling? and (b) Are the two modeling frameworks complementary or redundant?

Upper bound We first assessed human agreement on coherence judgments. Inter-subject agreement gives an upper bound for the task and allows us to interpret model performance in relation to human performance. To measure agreement, we employed leave-one-out resampling [Weiss and Kulikowski, 1991]. The technique correlates the ratings of each participant with the mean ratings of all other participants. The average agreement was $r = .768$.

Model Performance Table 4 displays the results of correlation analysis on 58 summaries[2] from our test set. Significant correlations are observed for the entity grid model (Egrid), the vector-based model (LSA), and two WordNet-based models, employing Hirst and St-Onge's [1998] (HStO) and Jiang and Conrath's [1997] (JCon) similarity measures.

The significant correlation of the Egrid model with human judgments empirically validates Centering's claims about the importance of entity transitions in establishing local coherence. The performance of the LSA model confirms previous claims in the literature [Foltz *et al.*, 1998] that distributional similarity is a significant predictor of text coherence. The fact that JCon correlates[3] with human judgments is not surprising either; its performance has been superior to other WordNet-based measures on two tasks that are particularly relevant for the problem considered here: modeling human similarity judgments and the automatic detection of malapropisms, a phenomenon that often results in locally incoherent discourses (see Budanitsky and Hirst [2001] for details). It is worth considering why HStO performs better than JCon. In quantifying semantic similarity, HStO uses all available relations in WordNet (e.g., antonymy, meronymy, hyponymy) and is therefore better-suited to capture cohesive ties arising from semantically related words than JCon which exploits solely hyponymy. These results further suggest that the chosen similarity measure plays an important role in modeling coherence.

The correlations obtained by our models are substantially lower when compared with the inter-subject agreement of .768. Our results indicate that there is no single method which captures all aspects of local coherence, although HStO yields the highest correlation coefficient in absolute terms. It is worth noting that Egrid is competitive with LSA and the semantic association models based on WordNet, even though it is unlexicalized. We conjecture that Egrid makes up for the lack of lexicalization by taking syntactic information into account and having a more global perspective of discourse. This is achieved by explicitly modeling entity transitions spanning more than two consecutive sentences.

As far as model intercorrelations are concerned, note that Egrid is not significantly correlated with the LSA model, thus indicating that the two models capture complementary coherence properties. Interestingly, LSA displays no correlation with the WordNet-based models. Although both types of models rely on semantic relatedness, they employ distinct representations of lexical meaning (word co-occurrence-based vs. taxonomy-based). Expectedly, the WordNet-based coherence models are all intercorrelated (see Table 4).

Model Combination An obvious question is whether a better fit with the experimental data can be obtained via model combination. A standard way to integrate different predictors (i.e., models) is multiple linear regression. Since several of

[2] The rest of the test data —32 summaries —was used for evaluating our combined model.

[3] The correlation in negative, because JCon is a measure of dissimilarity; the coeffi cient therefore has the opposite sign in comparison with the other WordNet-based measures.

the models were intercorrelated, we performed stepwise regression (using forward selection) to determine the best set of predictors for coherence. The best-fitting combined model (obtained on the set of 58 summaries) included five variables: Egrid, Overlap, LSA, HStO, and Lesk.

Next we tested the combined model's performance on 32 unseen summaries, the output of two systems different from those used for assessing the contribution of the individual models. More specifically, we obtained a coherence score for each unseen summary using the combined model and then compared the estimated values and the human ratings. The comparison yielded a correlation coefficient of .522 ($p < .01$) outperforming any single model. This is a linguistically richer model which integrates several interrelated aspects of coherence (both syntactic and semantic) such as repetition (Overlap), syntactic prominence (Egrid), and semantic association (LSA, HStO, Lesk).

6 Conclusions

In this paper we have compared and contrasted two main frameworks for representing and measuring text coherence. Our syntactic framework operationalizes Centering's notion of local coherence using entity grids. We argued that this representation is particularly suited for uncovering entity transition types typical of coherent and incoherent texts and introduced a model that quantitatively delivers this assessment. Our semantic framework capitalized on the notion of similarity between sentences. We experimented with a variety of similarity measures employing different representations of lexical meaning: word-based, distributional, and taxonomy-based. Our experiments revealed that the two modeling approaches are complementary: our best model retains aspects of entity coherence as well as semantic relatedness.

The modeling approach taken in this paper relies on shallow text properties and is relatively inexpensive (assuming access to a taxonomy and a parser). This makes the proposed models particularly attractive for the automatic evaluation of machine generated summaries. An important future direction lies in the development of a lexicalized version of the entity-grid model that combines the benefits of grid column topology and semantic similarity. Further investigations on different languages, text types, and genres will test the generality and portability of our models. We will also examine whether additional linguistic knowledge (e.g., coreference resolution, causality) will result in improved performance.

Acknowledgments

The authors acknowledge the support of EPSRC (Lapata; grant GR/T04540/01) and the National Science Foundation (Barzilay; CAREER grant IIS-0448168). Thanks to Eli Barzilay, Frank Keller, Smaranda Muresan, Kevin Simler, Caroline Sporleder, Chao Wang, and Bonnie Webber for helpful comments and suggestions.

References

[Bangalore *et al.*, 2000] Srinivas Bangalore, Owen Rambow, and Steven Whittaker. Evaluation metrics for generation. In *Proceedings of the INLG*, pages 1–8, 2000.

[Berry *et al.*, 1994] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1994.

[Budanitsky and Hirst, 2001] Alexander Budanitsky and Graeme Hirst. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Proceedings of ACL Worskhop on WordNet and Other Lexical Resources*, pages 29–34, 2001.

[Collins, 1998] Michael Collins. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1998.

[Fellbaum, 1998] Christiane Fellbaum, editor. *WordNet: An Electronic Database*. MIT Press, 1998.

[Foltz *et al.*, 1998] Peter W. Foltz, Walter Kintsch, and Thomas K. Landauer. Textual coherence using latent semantic analysis. *Discourse Processes*, 25(2&3):285–307, 1998.

[Grosz *et al.*, 1995] Barbara Grosz, Aravind K. Joshi, and Scott Weinstein. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225, 1995.

[Halliday and Hasan, 1976] M. A. K. Halliday and Ruqaiya Hasan. *Cohesion in English*. Longman, 1976.

[Higgins *et al.*, 2004] Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. Evaluating multiple aspects of coherence in student essays. In *Proceedings of the NAACL*, pages 185-192, 2004.

[Hirst and St-Onge, 1998] Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database and Some of its Applications*, pages 305–332. MIT Press, 1998.

[Hovy and Lin, 2003] Eduard Hovy and Chin-Yew Lin. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the HLT/NAACL*, pages 71–78, 2003.

[Jiang and Conrath, 1997] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING*, 1997.

[Lesk, 1986] Michael Lesk. Automatic sense disambiguation: How to tell a pine cone from an ice cream cone. In *Proceedings of the 1986 Special Interest Group in Documentation*, pages 24–26, 1986.

[Lin, 1998] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the ICML*, pages 296–304, 1998.

[Mani, 2001] Inderjeet Mani. *Automatic Summarization*. John Benjamins Pub Co, 2001.

[Miltsakaki and Kukich, 2004] Eleni Miltsakaki and Karen Kukich. Evaluation of text coherence for electronic essay scoring systems. natural language engineering. *Natural Language Engineering*, 10(1):25–55, 2004.

[Morris and Hirst, 1991] Jane Morris and Graeme Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 1(17):21–43, 1991.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLUE: a method for automatic evaluation of machine translation. In *Proceedings of the ACL*, pages 311–318, 2002.

[Resnik, 1995] Philip Resnik. Using information content to evaluate semantic similarity. In *Proceedings of 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.

[Walker *et al.*, 1998] Marilyn A. Walker, Arvind K. Joshi, and Ellen F. Prince, editors. *Centering Theory in Discourse*. Clarendon Press, Oxford, 1998.

[Weiss and Kulikowski, 1991] Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann, 1991.

Semantic annotation of unstructured and ungrammatical text

Matthew Michelson and Craig A. Knoblock
University of Southern California
Information Sciences Institute,
4676 Admiralty Way
Marina del Rey, CA 90292 USA
{michelso,knoblock}@isi.edu

Abstract

There are vast amounts of free text on the internet that are neither grammatical nor formally structured, such as item descriptions on Ebay or internet classifieds like Craig's list. These sources of data, called "posts," are full of useful information for agents scouring the Semantic Web, but they lack the semantic annotation to make them searchable. Annotating these posts is difficult since the text generally exhibits little formal grammar and the structure of the posts varies. However, by leveraging collections of known entities and their common attributes, called "reference sets," we can annotate these posts despite their lack of grammar and structure. To use this reference data, we align a post to a member of the reference set, and then exploit this matched member during information extraction. We compare this extraction approach to more traditional information extraction methods that rely on structural and grammatical characteristics, and we show that our approach outperforms traditional methods on this type of data.

1 Introduction

The Semantic Web will revolutionize the use of the internet, but the idea faces some major challenges. First, construction of the Semantic Web requires a lot of extra markup on documents, but this work should not be forced upon everyday users. Second, there is a lot of information that would be more useful if it were annotated for the Semantic Web, but the nature of the data makes it difficult to do so. Examples of this type of data are the text of EBay posts, internet classifieds like Craig's list, bulletin boards such as Bidding For Travel,[1] or even the summary text below the hyperlinks returned after querying Google. We call each piece of text from these sources a "post." It would be beneficial to add semantic annotation to such posts, like that shown in Figure 1, but the annotation task should carry no burden to human users.

Information extraction (IE) can be used to extract and semantically annotate pieces of some text. However, IE on posts is especially difficult because the data is neither structured enough to use wrapper technologies such as Stalker [Muslea *et al.*, 2001] nor grammatical enough to exploit Natural Language Processing (NLP) techniques such as those used in Whisk [Soderland, 1999].

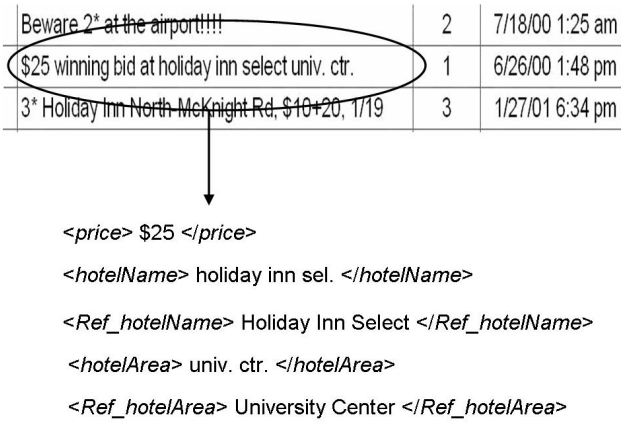


Figure 1: A post from Bidding For Travel

This lack of grammar and structure can be overcome by adding knowledge to IE. This extra knowledge consists of collections of known entities and their common attributes, which we call "reference sets." A reference set can be an online set of reference documents, such as the CIA World Fact Book. It can also be an online (or offline) database, such as the Comics Price Guide.[2] With the Semantic Web we envision building reference sets from the numerous ontologies that already exist. Continuing with our hotel example from Figure 1, assume there is an ontology of U.S. hotels, and from it we build a reference set with the following attributes: city, state, star rating, hotel name, area name, etc.

To use reference sets for semantic annotation we exploit the reference set to determine which, if any, of the attributes appear in the post. To do this, we first determine which member of the reference set best matches the post. We call this the record linkage step. Then we exploit the attributes of this

[1] www.biddingfortravel.com

[2] www.comicspriceguide.com

reference set member for the information extraction step by identifying and labeling attributes from the post that match those from the matching member of the reference set. We annotate the post in this manner.

For instance, the circled hotel post in Figure 1 matches the reference set member with the hotel name of "Holiday Inn Select" and the hotel area of "University Center." Using this match we label the tokens "univ. ctr." of the post as the "Hotel Area," since they match the hotel area attribute of the matching reference set record. In this manner we annotate all of the attributes in the post that match those of the reference set. Figure 2 illustrates our approach on the example post from Figure 1. For purposes of exposition, the reference set shown only has 2 attributes: name and area.

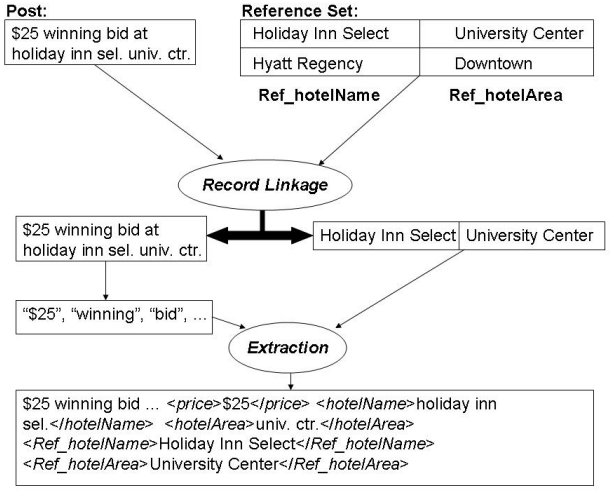


Figure 2: Annotation Algorithm

In addition to annotating attributes in the post from the reference set, we also annotate attributes that are identifiable, but not easily represented in reference sets. Examples of such attributes include prices or dates. This is shown by the price of Figure 1. Also, we include annotation for the attributes of the matching reference member. (These are called "Ref_hotel..." in Figure 1.) Since attribute values differ across posts, these reference member attributes provide a set of normalized values for querying. Also, the reference set attributes provide a simple visual validation to the user that the IE step identified things correctly. Lastly, by including attributes from the matching reference member, we can provide values for attributes that were not included in the post. In our example post of Figure 1, the user did not include a star rating. By including the reference set member's attribute for this star rating, we add useful annotation for information that was not present initially.

This paper describes our algorithm for semantic annotation using reference sets. Section 2 describes the record linkage step and Section 3 describes the extraction step. Section 4 presents experimental evaluation, and section 5 presents discussion of results. Section 6 presents related research, and section 7 describes our conclusions.

2 Aligning Posts to a Reference Set

To correctly parse the attributes from the post, we need to first decide what those attributes are. To aid in this process, we match the post to a member of the reference set.

Since it is infeasible to compare the post to all members of the reference set, we construct a set of candidate matches in a process known as "blocking". Many blocking methods have been proposed in the record linkage community (see [Baxter *et al.*, 2003] for a recent survey of some), but the basic idea is to cluster candidates together around a blocking key. In our work, a candidate for a post is any member of the reference set that shares some n-gram with that post. The choice of algorithm here is independent of the overall alignment algorithm.

Next we find the candidate that best matches the post. That is, we must align one data source's record (the post) to a record from the other data source (the reference set candidates). This alignment is called record linkage [Fellegi and Sunter, 1969].

However, our record linkage problem differs and is not well studied. Traditional record linkage matches a record from one data source to a record from another data source by relating their respective, decomposed attributes. Yet the attributes of the posts are embedded within a single piece of text. We must match this text to the reference set, which is already decomposed into attributes and which does not have the extraneous tokens present in the post. With this type of matching traditional record linkage approaches do not apply.

Instead, we create a vector of scores, V_{RL}, for each candidate. V_{RL} is composed of similarity scores between the post and each attribute of the reference set. We call these scores *RL_scores*. V_{RL} also includes *RL_scores* between the post and all of the attributes concatenated together. In the example reference set from Figure 2, the schema has 2 attributes *<Hotel Name, Hotel Area>*. If we assume the current candidate is <"Hyatt", "PIT Airport">. Then we define V_{RL} as:

V_{RL}=<*RL_scores(post,"Hyatt")*,
RL_scores(post,"PIT Airport"),
RL_scores(post,"Hyatt PIT Airport")>

Each *RL_scores(post,attribute)* is itself a vector, composed of three other vectors:

RL_scores(post,attribute)=<*token_scores(post,attribute)*,
edit_scores(post,attribute),
other_scores(post,attribute)>

The three vectors that compose *RL_scores* represent different similarity types. The vector *token_scores* is the set of token level similarity scores between the post and the attribute, including Jensen-Shannon distance (both with a Dirichlet prior and a Jelenik-Mercer mixture model) and Jaccard similarity. The vector *edit_scores* consists of the following edit distance functions: Smith-Waterman distance, Levenstein distance, Jaro-Winkler similarity and character level Jaccard similarity. (All of the scores in *token_scores* and *edit_scores* are defined in [Cohen *et al.*, 2003].) Lastly, the vector *other_scores* consists of scores that did not fit into either of the other categories, specifically the Soundex score between the post and the attribute and the Porter stemmer score between the two.

Using *RL_scores* of just the reference attributes themselves gives a notion of the field similarity between the post and the match candidate. The *RL_scores* that uses the concatenation of the reference attributes gives an idea of the record level match. Since the post is not yet broken into attributes, this is how we determine these field and record level similarities. We could use only the concatenated scores since all we desire is a record level match. However, it is possible for different records in the reference set to have the same record level score, but different scores for the attributes. If one of these records had a higher score on a more discriminative attribute, we would like to capture that.

After all of the candidates are scored, we then rescore each V_{RL}. For each element of V_{RL}, the candidates with the maximum value at that index map this element to 1. The rest of the candidates map this element to 0.

For example, assume we have 2 candidates, with the vectors $V1_{RL}$ and $V2_{RL}$:

$V1_{RL}$ = <.999,1.2,...,0.45,0.22>
$V2_{RL}$ = <.888,0.0,...,0.65,0.22>

After rescoring they become:

$V1_{RL}$ = <1,1,...,0,1>
$V2_{RL}$ = <0,0,...,1,1>

The rescoring helps to determine the best possible candidate match for the post. Since there might be a few candidates with similarly close values, and only one of them is a best match, the rescoring separates out the best candidate as much as possible.

After rescoring, we pass each V_{RL} to a Support Vector Machine (SVM) [Joachims, 1999] trained to label them as matches or non-matches. When the match for a post is found, the attributes of the matching reference set member are added as annotation to the post.

3 Extracting Data from Posts

To exploit the reference set for extraction we use the attributes of the best match from the reference set as a basis for identifying similar attributes in the post.

To begin the extraction process, we break the post into tokens. In our example from Figure 2, the post becomes the set of tokens, {"$25", "winning", "bid",...}. Each of these tokens is then scored against each attribute of the record from the reference set that was deemed the match. This scoring consists of building a vector of scores which we call V_{IE}. Although similar to V_{RL}, the V_{IE} vector does not include the vector *token_scores* because we are comparing single tokens of the post. Instead, we include a vector *common_scores*. The *common_scores* vector includes user defined functions such as regular expressions, which help identify the attributes that are not present in the reference set, such as price or date. So, V_{IE} consists of the vector *common_scores* and an *IE_scores* vector between each token and each attribute from the reference set.

Our *RL_scores* vector from the record linkage step,

RL_scores(post,attribute)=<token_scores(post,attribute),
edit_scores(post,attribute),
other_scores(post,attribute)>

becomes,

IE_scores(token,attribute)=<edit_scores(token,attribute),
other_scores(token,attribute)>

So, using our example from Figure 2, we match the post to the reference set member {"Holiday Inn Select","University Center"}. So, if we generate the V_{IE} for the post token "univ." it would look like:

V_{IE} = <*common_scores("univ."),*
IE_scores("univ.","Holiday Inn Select"),
IE_scores("univ.","University Center")>

Since each V_{IE} is not a member of a cluster where the winner takes all, there is no binary rescoring.

Each V_{IE} is then passed to a multiclass SVM [Tsochantaridis *et al.*, 2004] trained to give it a class label, such as hotel name, hotel area, or price. Intuitively, similar attribute types should have a similar V_{IE}. We expect that hotel names will generally have high scores against the reference set attribute of hotel names, and small scores against the other attributes, and the vector will reflect this.

The SVM learns that any vector that does not look like anything else should be labeled as "junk", which can then be ignored. This is an important idea because without the benefits of a reference set this would be an extraordinarily difficult task. If features such as capitalization and token location were used, who is to say "Great Deal" is not a car name? Also, many traditional IE systems that work in this unique domain of ungrammatical, unstructured text, such as addresses and bibliographies, assume that each token of the text must be classified as something, an assumption that cannot be made when users are entering text.

However, by treating each token in isolation there is a chance that a junk token will be mislabeled. For example, a junk token might have enough letters to be labeled as a *hotel area*. Then when we extract the *hotel area* from the post, it will have a noisy token. Therefore, labeling each token individually gives an approximation of the data to be extracted.

To improve extraction, we can exploit the power of the reference set by comparing the whole extracted field to its analogue reference set attribute. Thus, once all of the tokens from a post are processed and we have whole attributes labeled, we take each attribute and compare it to the corresponding one from the reference set. Then we can remove the tokens that introduce noise in the extracted attribute.

To do this, we first get two baseline scores between the extracted attribute and the reference set attribute. One is a Jaccard similarity, which demonstrates token level similarity. However, since there are many misspellings and such, we also need an edit-distance based similarity metric. For this we use the Jaro-Winkler metric. These baselines give us an idea of how accurately we performed on the approximate extraction. Using our post from Figure 2, assume the phrase "holiday inn sel. univ. ctr." was "holiday inn sel. ***in*** univ. ctr.". In this case, we might extract "holiday inn sel. in" as the *hotel name*. In isolation, the token "in" could be the "inn" of a hotel name. Comparing this extracted hotel name to the reference attribute, "Holiday Inn Select," we get a Jaccard similarity of 0.4 and a Jaro-Winkler score of 0.87.

Next, we go through the extracted attribute, removing one token at a time and calculating the new Jaccard and Jaro-Winkler similarities. If both of these new scores are higher than the baseline, then that token is a candidate for removal. Once all of the tokens are processed in this way, the candidate for removal that has the highest scores is removed, and we repeat the whole process. The process ends when there are no more tokens that yield improved scores when they are removed. In our example, we find that "in" is a removal candidate since it yields a Jaccard score of 0.5 and a Jaro-Winkler score of 0.92. Since it has the highest scores after the iteration, it is removed. Then we see that removing any of the remaining tokens does not provide improved scores, so the process ends.

Aside from increasing the accuracy of extraction, this approach has the added benefit of disambiguation. For instance, a token might be both a *hotel name* and a *hotel area*, but not both at the same time? As an example, "airport" could be part of a *hotel name* or a *hotel area*. In this case, we could label it as both, and the above approach would remove the token from the attribute that it is not. However, our implementation currently assigns only one label per token, so we did not test this disambiguation technique.

Thus, the whole extraction process takes a token of the text, creates the V_{IE} and passes this to the SVM which generates a label for the token. Then each field is cleaned up and we add the annotation to the post. This produces the output shown at the end of Figure 2.

4 Results

To validate our approach, we implemented our algorithm in a system named Phoebus and tested the technique in two domains: hotel postings and comic books.

In the *hotel* domain, we attempt to parse the hotel name, hotel area, star rating of the hotel, price and dates booked from the Bidding For Travel website. This site is a forum where users share successful bids for Priceline. We limited our experiment to posts about hotels in Sacramento, San Diego and Pittsburgh. As a reference set, we use the Bidding For Travel hotel guides. These guides are special posts that list all of the hotels that have ever been posted about in a given area. These posts provide the hotel name, hotel area and the star rating, which are used as the reference set attributes.

The *comic* domain uses posts from EBay about comic books for sale searched by keyword "Incredible Hulk" and "Fantastic Four". Our goal is to parse the title, issue number, price, condition, publisher, publication year and the description from each post. (Note: the description is a few word description commonly associated with a comic book, such as *1st appearance the Rhino*.) As a reference set for this domain, we used the Comics Price Guide[3] for lists of all of the Incredible Hulk and Fantastic Four comics, as well as a list of all possible comic book conditions. In this case, the reference set included the attributes title, issue number, description, condition and publisher.

Experimentally, we split the posts in each domain into 2 folds, one for training and one for testing, where the training fold was 30% of the total posts, and the testing fold was the remaining 70%. We ran 10 trials and report the average results for these 10 trials.

[3]http://www.comicspriceguide.com/

4.1 Alignment Results

Our approach hinges on exploiting reference sets, so the alignment step should perform well. We report the results of the alignment step in Table 1. According to the usual record linkage statistics we define:

$$Precision = \frac{\#CorrectMatches}{\#TotalMatchesMade}$$

$$Recall = \frac{\#CorrectMatches}{\#PossibleMatches}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precison + Recall}$$

We compare our record linkage approach to WHIRL [Cohen, 2000]. WHIRL is record linkage system that does soft joins across tables by computing vector-based cosine similarities between the attributes. All other record linkage systems require matching based on decomposed fields, so WHIRL served as a benchmark because it does not have this requirement. As input to WHIRL, one table was the test set of posts (70% of the posts) and the other table was the reference set with the attributes concatenated together. As in our record linkage step, using the concatenation of attributes can best mirror finding a record level match. This was also done because joining across each reference set attribute separately leaves no way to combine the matches for these queries. For example, would we only count the reference set members that score highest for every attribute as matches?

We ran a similarity join across these tables, which produced a list of matches, ordered by descending similarity score. For each post that had matches from the join, the reference set member with the highest similarity score was called its match. These results are also reported in Table 1. Since Phoebus is able to represent both an attribute level and record level similarity in its score, using more than just token based cosine similarity, it outperformed WHIRL.

	Prec.	Recall	F-measure
Hotel			
Phoebus	93.60	91.79	**92.68**
WHIRL	83.52	83.61	83.13
Comic			
Phoebus	93.24	84.48	**88.64**
WHIRL	73.89	81.63	77.57

Table 1: Record linkage results

4.2 Extraction Results

We also performed experiments to validate our approach to extraction. Specifically, we compare our technique to two other IE methods as baselines.

One baseline is Simple Tagger from the MALLET [McCallum, 2002] suite of text processing tools. Simple Tagger

is an implementation of Conditional Random Fields (CRF). CRFs have been effectively used in IE. As an example, one algorithm combines information extraction and coreference resolution using CRFs [Wellner *et al.*, 2004].

We also present our results versus Amilcare [Ciravegna, 2001], which uses shallow Natural Language Processing for information extraction. It has been empirically shown that Amilcare does much better in extraction versus other symbolic systems [Ciravegna, 2001]. It presents a good benchmark versus NLP based systems, which we expect will not do well on our domains. For the tests, we supplied our reference data as gazetteers to Amilcare.

Our IE technique includes scores that we deemed "common" scores, which are used to help identify attributes not in the reference set. We make these clear for each domain, since they are the only domain specific scores we include. For the hotel domain, our common scores are *matchPriceRegex* and *matchDateRegex*, which give a positive score if a token matches a price or date regular expression, and 0 otherwise. For the comic domain, we use *matchPriceRegex* and *matchYearRegex*.

We present our results using Precision, Recall and F-Measure as defined above. Tables 2 and 3 show the results of correctly labeling the tokens within the posts with the correct attribute label for the Hotel and Comic domains, respectively. Attributes in *italics* are attributes that exist in the reference set. The column Freq shows the average number of tokens that have the associated label.

Table 4 shows the results reported for all possible tokens, which is a weighted average, since some attribute types are more frequent than others. Also included in Table 4 are "field level" summary results. Field level results regard a piece of extracted information as correctly labeled only if all of the tokens that should be present are, and there are no extraneous tokens. In this sense, it is a harsh, all or nothing metric, but it is a good measure of how useful a technique would really be.

We tested the F-Measures for statistical significance using a two-tailed paired t-test with α=0.05. In Table 2 the only F-Measure difference that was not significant was the *Star* attribute between Phoebus and Simple Tagger. In Table 3 the F-Measures for Price were not significant between Phoebus and Simple Tagger and between Phoebus and Amilcare. In Table 4 all differences in F-Measure proved statistically significant.

Phoebus outperforms the other systems on almost all attributes, and for all summary results. There were 3 attributes where Phoebus was outperformed, and two of these warrant remarks. (We ignore hotel name since it was so similar.) On Comic titles, Phoebus had a much lower recall because it was unable to extract parts of titles that were not in the reference set. Consider the post, "The incredible hulk and Wolverine #1 Wendigo". In this post, Phoebus extracts "The incredible hulk," but the actual title is "The incredible hulk and Wolverine." In this case, the limited reference set hindered Phoebus, but this could be corrected by including more reference set data, such as other comic book price guides.

The Comic description is the other attribute where Simple Tagger outperformed Phoebus. Simple Tagger learned that for the most part, there is an internal structure to descriptions, such that they are almost never broken up in the middle. For instance, many descriptions go from the token "1st" to a few words after, with nothing interrupting them in the middle. Simple Tagger, then, would label all of these tokens as a description. However, lots of times it labeled too many. This way, it had a very high recall for description, by labeling so much data, but it suffered in other categories by labeling things such as conditions as descriptions too.

Phoebus had the highest level of precision for Comic descriptions, but it had a very low recall because it ignored many of the description tokens. Part of this problem stemmed from classifying tokens individually. Since many of the description tokens were difficult to classify from a single token perspective, they were ignored as junk.

Hotel					
		Prec.	Recall	F-Measure	Freq
Area	Phoebus	89.25	87.5	**88.28**	809.7
	Simple Tagger	92.28	81.24	86.39	
	Amilcare	74.20	78.16	76.04	
Date	Phoebus	87.45	90.62	**88.99**	751.9
	Simple Tagger	70.23	81.58	75.47	
	Amilcare	93.27	81.74	86.94	
Name	Phoebus	94.23	91.85	93.02	1873.9
	Simple Tagger	93.28	93.82	**93.54**	
	Amilcare	83.61	90.49	86.90	
Price	Phoebus	98.68	92.58	**95.53**	850.1
	Simple Tagger	75.93	85.93	80.61	
	Amilcare	89.66	82.68	85.86	
Star	Phoebus	97.94	96.61	**97.84**	766.4
	Simple Tagger	97.16	97.52	97.34	
	Amilcare	96.50	92.26	94.27	

Table 2: Extraction results: Hotel domain

Comic					
		Prec.	Recall	F-Measure	Freq
Condition	Phoebus	91.80	84.56	**88.01**	410.3
	Simple Tagger	78.11	77.76	77.80	
	Amilcare	79.18	67.74	72.80	
Descript.	Phoebus	69.21	51.50	59.00	504.0
	Simple Tagger	62.25	79.85	**69.86**	
	Amilcare	55.14	58.46	56.39	
Issue	Phoebus	93.73	86.18	**89.79**	669.9
	Simple Tagger	86.97	85.99	86.43	
	Amilcare	88.58	77.68	82.67	
Price	Phoebus	80.00	60.27	**68.46**	10.7
	Simple Tagger	84.44	44.24	55.77	
	Amilcare	60.0	34.75	43.54	
Publisher	Phoebus	83.81	95.08	**89.07**	61.1
	Simple Tagger	88.54	78.31	82.83	
	Amilcare	90.82	70.48	79.73	
Title	Phoebus	97.06	89.90	93.34	1191.1
	Simple Tagger	97.54	96.63	**97.07**	
	Amilcare	96.32	93.77	94.98	
Year	Phoebus	98.81	77.60	**84.92**	120.9
	Simple Tagger	87.07	51.05	64.24	
	Amilcare	86.82	72.47	78.79	

Table 3: Extraction results: Comic domain

5 Discussion

One drawback when using supervised learning systems is the cost to label training data. However, our entire algorithm gen-

	Hotel					
	Token level				Field level	
	Prec.	Recall	F-Mes.	Prec.	Recall	F-Mes.
Phoebus	93.60	91.79	**92.68**	87.44	85.59	**86.51**
Simple Tagger	86.49	89.13	87.79	79.19	77.23	78.20
Amilcare	86.12	86.14	86.11	85.04	78.94	81.88
	Comic					
	Token level				Field level	
	Prec.	Recall	F-Mes.	Prec.	Recall	F-Mes.
Phoebus	93.24	84.48	**88.64**	81.73	80.84	**81.28**
Simple Tagger	84.41	86.04	85.43	78.05	74.02	75.98
Amilcare	87.66	81.22	84.29	90.40	72.56	80.50

Table 4: Summary extraction results

eralizes well and can perform well with little training data. Table 5 presents summary, token level results for extraction. Here we trained Phoebus on 10% of the posts, and then tested on the other 90%. In both domains, the results are similar to those of the experimental set up, which used 30% of the data for training.

	Prec.	Recall	F-measure
Hotel (30%)	93.60	91.79	92.68
Hotel (10%)	93.66	90.93	92.27
Comic (30%)	93.24	84.48	88.64
Comic (10%)	91.41	83.63	87.34

Table 5: Phoebus: Trained on 10% of the data

Since we return the reference set attributes as annotation, we need to examine whether or not this annotation is valid. To do this, we need a way to link the results from performing record linkage to extraction results. Note we can consider the fact that a correct match during record linkage is the same as correctly identifying those attributes from the reference set in the post, at the field level. In our ongoing example from Figure 2, when we match the post to the reference member with a hotel name of "Holiday Inn Select", it is like extracting this hotel name from the post. Thus, we can consider our record linkage results to be field level extraction results for the attributes in the reference set.

Using the reference set attributes as annotation has some interesting implications. For instance, the attributes from the reference set provide a normalized platform of values for querying the data. Also, returning reference set attributes for types not found in the post provides information that might have been missing previously. As an example, a post might leave out the star rating. Yet, now we have one from the reference set record upon which to query. Another interesting implication arises as a solution for the cases where extraction is hard. None of the systems extracted the comic description well. However, by including the reference set description, we consider the record linkage results as how effectively we extracted (and thus labeled), a description. This yields an improvement of over 20% for precision, and almost 10% for recall.

It may seem that using the reference set attributes for annotation solves the problem, but this is not the case. For one thing, we want to see the actual values entered for different attributes. Also, there are cases when the extraction results outperform the record linkage results as seen with the hotel name and star rating. This happens because even if a post is matched to an incorrect member of the reference set, that incorrect member is most likely very close to the correct match, and so it can be used to correctly extract much of the information. For example, there might be hotels with the same star rating and hotel name, but with a different area. If this area is not included in the post or included in a convoluted way the record linkage step might not get a correct match. However, the reference set member could still be used to extract the hotel name and star rating.

Lastly, the more discriminative information that we can pass to the SVM, the better it will perform. Assume that all we wanted to do is extract price and date from hotel posts, and we would get the rest of the annotation from the reference set. We would still want to train the SVM to extract the attributes of the reference set, because it would then know that by classifying a certain piece of information as a certain attribute, it is not another piece of information. That is to say, it is just as important to recognize what a token is not. For example, consider a reference set hotel named "The $41 Inn". Then when post matches this reference set member, and we see the token "$41", we know that it is most likely a hotel name. If we did not train to extract all of the attributes, this would be classified as a price, since the SVM has no notion of a hotel name.

Extraction on all of the attributes also helps the system to classify (and ignore) tokens that are "junk". Labeling something as junk is much more descriptive if it is labeled junk out of many possible class labels that could share lexical characteristics. This helps to improve the extraction results on items that are not in the reference set, and we see this in our results.

On the topic of reference sets, it is important to note that the algorithm is not tied to a single reference set. The algorithm extends to include multiple reference sets by iterating the process for each reference set used.

Consider the following two cases. If we want to extract conference names and cities but we only use one reference set, it would have to contain the power set of cities crossed with conference names. However, if we have two reference sets, one for each attribute, we can run the algorithm once with the conference name data, and once with a reference set of cities.

The next interesting case happens when a post contains more than one of the same attribute. For example, we want to extract two cities from some post. If we use one reference set, then it would include the cross product of all cities. However, we can use a single reference set of city names if we slightly modify the algorithm. We make a first pass with the city reference set. During this pass, the record linkage match will either be one of the cities that matches best, or a tie between them. In the case of a tie, we just choose the first match. Using this reference city we then extract the city from the post, and remove it from the post. Then we simply run the process again, which will catch the second city, using the same, single reference set. This could be repeated as many times as needed.

6 Related Work

Our work is motivated by the goal that the cost of annotating documents for the Semantic Web should be free, that is, automatic and invisible to users [Hendler, 2001]. Many researchers have followed this path, attempting to automatically mark up documents for the Semantic Web, as we propose here [Vargas-Vera *et al.*, 2002; Handschuh *et al.*, 2002; Cimiano *et al.*, 2004; Dingli *et al.*, 2003]. However, these systems rely on lexical information, such as part-of-speech tagging or shallow Natural Language Processing to do their extraction/annotation (e.g. Amilcare [Ciravegna, 2001]). This is not an option when the data is ungrammatical, like our post data. In a similar vein, there are systems such as ADEL [Lerman *et al.*, 2004] which rely on the structure to identify and annotate records in web pages. Again, the failure of our data to exhibit structure makes this approach inappropriate. So, while there is a fair amount of work in automatic labeling, there is not much emphasis on techniques that could do this on text that is unstructured and ungrammatical.

While the idea of record linkage is not new [Fellegi and Sunter, 1969] and is well studied even now [Bilenko and Mooney, 2003], most of the focus for this work matches one set of records to another set of records based on their decomposed attributes. There is little work on matching data sets where one record is a single string composed of the other data set's attributes to match on, as in our case. The WHIRL system [Cohen, 2000] allows for record linkage without decomposed attributes, but as shown in Section 4.1 we outperform WHIRL by exploiting a larger set of features to represent both a field and record level similarity.

Using the reference set's attributes as normalized values is similar to the idea of data cleaning. However, most of the data cleaning algorithms assume that there are tuple-to-tuple transformations [Lee *et al.*, 1999; Chaudhuri *et al.*, 2003]. That is, there is some function that maps the attributes of one tuple to the attributes of another. This approach would not work on our data, where all of the attributes are embedded within the post, which maps to a set of attributes from the reference set.

Information extraction can semantically annotate data, which is why we chose to compare our technique to other IE approaches, such as the Simple Tagger Conditional Random Field [McCallum, 2002]. Other IE approaches, such as Datamold [Borkar *et al.*, 2001] and CRAM [Agichtein and Ganti, 2004], segment whole records (like bibliographies) into attributes. However, both of these systems require that every token of a record receive a label, which is not possible with posts that are filled with irrelevant tokens. CRAM is also similar in its use of reference sets for extraction. However, they assume that the reference set members already match the data for extraction, while we do this record linkage automatically. Another IE approach similar to ours performs named entity recognition using a dictionary component [Cohen and Sarawagi, 2004]. However, this technique requires that entire segments have the same class label, while our technique can handle the case where an attribute is broken up in the middle by another attribute, say a hotel name interrupted by a hotel area.

7 Conclusion

In this paper we presented an algorithm for semantically annotating text that is ungrammatical and unstructured. This technique provides much more utility to data sources that are full of information, but cannot support structured queries. Using this approach, Ebay agents could monitor the auctions looking for the best deals, or a user could find the average price of a four star hotel in San Diego. This approach to semantic annotation is necessary as we transition into the Semantic Web, where information needs annotation for software systems to use it, but users are unwilling to provide the required annotation.

In the future, we we would like to link this technique with a mediator [Thakkar *et al.*, 2004] framework for automatically acquiring reference sets. This is similar to automatically incorporating secondary sources for record linkage [Michalowski *et al.*, 2005]. How to automatically formulate a query to retrieve the correct domain reference set is a direction of future research. Also, our current implementation only gives one class label per token. Ideally we would give a token all possible labels, and then remove the extraneous tokens when we clean up the attributes, as described in Section 3.

8 Acknowledgements

We would like to thank William Cohen, Andrew McCallum and Fabio Ciravegna for allowing public use of their systems and code. Also, thanks to Kristina Lerman and Snehal Thakkar for their comments.

This research is based upon work supported in part by the National Science Foundation under Award No. IIS-0324955, in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010, in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-1-0504, in part by the Air Force Office of Scientific Research under grant number FA9550-04-1-0105, and in part by the United States Air Force under contract number F49620-02-C-0103.

The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

References

[Agichtein and Ganti, 2004] Eugene Agichtein and Venkatesh Ganti. Mining reference tables for automatic text segmentation. In *the Proceedings of the 10th ACM Int'l Conf. on Knowledge Discovery and Data Mining*, Seattle, Washington, August 2004. ACM Press.

[Baxter *et al.*, 2003] Rohan Baxter, Peter Christen, and Tim Churches. A comparison of fast blocking methods for

record linkage. In *Proceedings of the ACM Workshop on Data Cleaning, Record Linkage, and Object Identification*, 2003.

[Bilenko and Mooney, 2003] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the 9th ACM Int'l Conf. on Knowledge Discovery and Data Mining*, pages 39–48, Washington, DC, August 2003. ACM Press.

[Borkar *et al.*, 2001] Vinayak Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. Automatic segmentation of text into structured records. In *Proceedings of ACM SIGMOD*, 2001.

[Chaudhuri *et al.*, 2003] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, and Rajeev Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proceedings of ACM SIGMOD*, pages 313–324. ACM Press, 2003.

[Cimiano *et al.*, 2004] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *Proceedings of the 13th international conference on World Wide Web*, pages 462–471. ACM Press, 2004.

[Ciravegna, 2001] Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

[Cohen and Sarawagi, 2004] William Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *Proceedings of the 10th ACM Int'l Conf. Knowledge Discovery and Data Mining*, Seattle, Washington, August 2004. ACM Press.

[Cohen *et al.*, 2003] William W. Cohen, Pradeep Ravikumar, and Stephen E. Feinberg. A comparison of string metrics for matching names and records. In *Proceedings of the Workshop on Data Cleaning and Object Consolidation*, 2003.

[Cohen, 2000] William W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):288–321, 2000.

[Dingli *et al.*, 2003] Alexiei Dingli, Fabio Ciravegna, and Yorick Wilks. Automatic semantic annotation using unsupervised information extraction and integration. In *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation*, 2003.

[Fellegi and Sunter, 1969] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[Handschuh *et al.*, 2002] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-cream - semi-automatic creation of metadata. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*. Springer Verlag, 2002.

[Hendler, 2001] James Hendler. Agents and the semantic web. *IEEE Intelligent Systems*, 16(2):30–37, 2001.

[Joachims, 1999] Thorsten Joachims. *Advances in Kernel Methods - Support Vector Learning*, chapter 11: Making large-Scale SVM Learning Practical. MIT-Press, 1999.

[Lee *et al.*, 1999] Mong-Li Lee, Tok Wang Ling, Hongjun Lu, and Yee Teng Ko. Cleansing data for mining and warehousing. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications*, pages 751–760. Springer-Verlag, 1999.

[Lerman *et al.*, 2004] Kristina Lerman, Cenk Gazen, Steven Minton, and Craig A. Knoblock. Populating the semantic web. In *Proceedings of the Workshop on Advances in Text Extraction and Mining*, 2004.

[McCallum, 2002] Andrew McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[Michalowski *et al.*, 2005] Martin Michalowski, Snehal Thakkar, and Craig A. Knoblock. Automatically utilizing secondary sources to align information across sources. In *AI Magazine, Special Issue on Semantic Integration*, volume 26, pages 33–45. 2005.

[Muslea *et al.*, 2001] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.

[Soderland, 1999] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

[Thakkar *et al.*, 2004] Snehal Thakkar, Jose Luis Ambite, and Craig A. Knoblock. A data integration approach to automatically composing and optimizing web services. In *Proceedings of the ICAPS Workshop on Planning and Scheduling for Web and Grid Services*, Whistler, BC, Canada, 2004.

[Tsochantaridis *et al.*, 2004] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, August 2004.

[Vargas-Vera *et al.*, 2002] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt, and Fabio Ciravegna. Mnm: Ontology driven semi-automatic and automatic support for semantic markup. In *Proceedings of the 13th International Conference on Knowledge Engineering and Management*, 2002.

[Wellner *et al.*, 2004] Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 2004.

Temporal Context Representation and Reasoning

Dan Moldovan, Christine Clark, Sanda Harabagiu
Language Computer Corporation
Richardson, Texas, 75080
{moldovan, christine, sanda}@languagecomputer.com

Abstract

This paper demonstrates how a model for temporal context reasoning can be implemented. The approach is to detect temporally related events in natural language text and convert the events into an enriched logical representation. Reasoning is provided by a first order logic theorem prover adapted to text. Results show that temporal context reasoning boosts the performance of a Question Answering system.

1 Introduction

Representing and reasoning about time-dependent information is important for applications ranging from databases, planning, natural language processing, and others. For example, temporal reasoning is essential in question answering to successfully addressing a time sensitive and dynamic world. Questions requiring temporal relations are not currently solvable in most state of the art question answering systems. Of the factoid TREC questions from 2001-2004 approximately 8.7% require temporal context reasoning. A temporal context scopes changing world views and serves as a tool for disambiguating time dependent events or intervals. Questions such as *Who was the president of South Korea when Bill Clinton was President of the US?* need temporal event detection and context based reasoning.

While there has been much theoretical research and modeling of context reasoning in the temporal domain, few systems have reported an evaluation of end to end temporal context enhanced question answering. This paper presents such a system. We report on (1) temporal context indexing for passage retrieval, (2) a fast and robust representation of time and annotation of time events, (3) the implementation of a machine learning based temporal event recognition module, (4) a model for a knowledge representation that maps natural language context into first order logic Suggested Upper Merged Ontology(SUMO)[Niles and Pease, 2001] predicates, and (5) a temporal context reasoning engine.

2 Approach

Several logic or semantic standard approaches to represent time (or events in time) have been defined. The most common ones include absolute dating schemes, relative event ordering or duration based approaches. Decisions about the nature of the event (instantaneous or extended) have to be made. The problem with many discrete schemes for natural language applications (such as Mani and Wilson, 2000) is the lack of absolute timestamps for many events. Propagation approaches to assign absolute time stamps to event clauses for sequence recognition have been implemented [Filatova and Hovy, 2001]. However, besides the recognition problems (Filatova and Hovy report 52% accuracy) these approaches do not capture temporal relations that are based on duration of events.

Relative event orderings and duration relations that hold true at all times should not be labeled with absolute timestamps as in: *People are born before they die*. This holds true also for event orderings commonly found in user manuals, like IT support information or cookbooks as in: *What process do I need to terminate after a failed query?*

[Allen, 1991] demonstrates a constraint propagation approach based on a set of possible interval relations that could capture this, but the recognition of these events and their relations is still an open research area.

Our approach uses a hybrid of the above mentioned methods. Passage retrieval makes use of a descrete time scale in the form of temporal indexing so as to capture all the relevant candidate passages that may not have absolute time specified in the document text. For the time-event model, high coverage signal words and phrases that specify how temporal elements should be related are used. The chosen signals are based on the annotations in TimeML used in the TERQAS project [Pustejovsky et al., 2002] and are mined for and annotated in a sampling of articles from the TREC collection. The signals are then classified by the type of temporal relation they express, such as overlapping, inclusion, ordering, and more. This data is used for training the automatic temporal event detection module that recognizes 7 distinct temporal event interval relations on the question and its answer candidates. The detected temporal events enrich a logical form representation of the question and answer candidates with

SUMO predicates and functions. Based on these temporal constraints, a context resolution engine operates as an answer candidate filter.

3 Passage Retrieval with Temporal Context

One common class of time dependent questions targets facts that are rooted in an absolute time, such as: *Who led the NHL in playoff goals in June 1998?* To ensure that all passages relevant to a temporally constrained question are retrieved it is necessary to index the temporal contexts in a document. The approach is to scan each document in a collection for its time stamp as well as any underspecified or relative references to time. A date resolution module processes all underspecified and relative dates to accurately anchor these temporal references in a calendar year. The context field consists of a *(year, month, day, hour, minute, second)* tuple, where any member in the tuple is optional. For a document with time, *T(1998, 06, 15, X, X, X)*, the following temporal context entries are computed and displayed in Table 1:

Temporal Reference	Resolved Date
last year	T(1999,X,X,X,X,X)
next week	T(1998,06,16-22,X,X,X)
in a week	T(1998,06,22,X,X,X)
today	T(1998,06,15,X,X,X)

Table 1: Resolved Temporal Context

The resolved references as well as the document time stamp is indexed and made searchable for time dependent queries. For the question above the query issued to the index is:
(_human) AND (lead) AND (nhl OR (national AND hockey AND league)3) AND (playoff) AND (goal) AND T(1998, 06, X, X,X,X,X)

For questions involving a time range, the query is translated into a conjunction of time operators. As an example, *What Chinese Dynasty was during 1412-1431?* translates to: *(_organization) AND (chinese) AND (dynasty) AND (T(1412,X,X,X,X,X) OR T(1413,X,X,X,X,X) OR ... OR T(1431,X,X,X,X,X))*

Internally the search engine computes the allowed range and searches for dates indexed in that range. The temporal context for the question is enforced by searching the index for the context detected in the question and applying document dates to all passages retrieved that do not have an explicit context within a window of 3 sentences. These resolved dates are passed along to the answer processing module which invokes the signal based temporal event recognizer and the context resolution modules to verify that the candidate answers do meet the temporal constraints in the question

4 Data Annotation

For an initial analysis of the signal words, the TimeML [Pustejovsky et al., 2000] TimeBank4 data functioned as a source of time signal expressions to be used as seeding material in the TREC 2002 corpus. The signal list served as a means to bootstrap a sample of the TREC collection for annotation purposes. The goal of the annotation was to provide data for a machine learning algorithm aimed at disambiguating signal words, detecting time events, and determining the boundaries of the time event.

Category (example)	Interpretation	% Dist
Sequence (before, after)	E1 happened in full before E2	12.7
Containment (in, of)	E1 is contained by E2	30.2
Overlap (at,as,on)	An interval exists that is contained by E1 and E2	28.6
Right Open Interval (from, since)	E1 is the left boundary of E2, right is undefined	6.1
Left Open Interval (to, until)	E2 is the right boundary of E1, left is undefined	4.5
Closed Interval (for, all)	E1 lasts for the duration of E2	6.3
Absolute Ordering (first, last)	E1 has an ordering relative to E1	11.6

Table 2: Signal Annotation Categories and their Distribution

The requirements of the annotation scheme used to mark up 1000 sentences from NYT, 150 from APW, and 150 sentences from XIN were robustness, simplicity, and sufficient granularity of the time events.

Time events are defined as related intervals, which are bound by a signal expression indicating the relation of the time events. Additionally, each signal can modify up to two events and can determine one of the following relations: *sequence, containment, overlap, open interval (left or right), duration and absolute ordering.* An event is interpreted as an interval bound by signal words or phrases and can be noun phrases *(the bombing)* or verb groups *(participated)*.

Since the current model operates at the sentence level, a marker is added to signals that need more information to resolve its arguments such as implicit context or missing discourse. Table 2 lists the categories of signal words, their natural language interpretation, and their distribution across a sampling of the TREC collection.
Example annotations based on the signal categories:
Which country <E1 id=889> declared </E1 id=889> independence <S id=889 class=contain> in </S id=889><E2 id=889> 1776 </E2 id=889>?
<S id=358 class=sequence> After </S id=358> quickly <E1 id=358> decapitating </E1 id=358> the bird, Susan <E2 id=358> scalded </E2 id=358> the carcass.

5 Detection of Temporally Ordered Events

Automated discovery of temporally ordered events requires detecting a temporal triple **(S,E1,E2)** which consists of a time dependent signal word **(S)** and its corresponding temporal event arguments, **(E1)** and **(E2)**. Detection of a temporal triple is complicated by two issues:
(1) **Disambiguation of signal words:** Not all signal words are unambiguously classified as time indicators.
(a) *He stood **before** the judge* vs *He proofread the*

*manuscripts **before** mailing it to the publisher*
(b) *He woke up **at** 10:00* vs *He looked through the window **at** the rising sun*
(2) **Attaching events to signal words**: Although some temporal events are found near their signal, many signals occur with their temporal events underspecified (non-local).
(a) *The problem was resolved **after** 4 hours of intensive maintenance but **before** anybody was harmed.*

In the above example the signal word ***after*** easily attaches to *resolved* and *4 hours*, while the second signal word, ***before*** has a non-local E1 reference (also *resolved*).

To address these issues machine learning is employed in two stages, first to recognize and disambiguate signals, and second to discover and attach temporal events to their signals. The input to the learner is the data annotated. The predictive classifiers that result from learning are used to automatically detect temporal triples (signals with their attached temporal events) in natural language text.

Signal Disambiguation

In the first stage of the machine learning process, a signal's surface form is classified either as a temporal **(signal)** or non-temporal occurrence, with signals classified further as to how they order their events, see Table 2.

A chunk parse tree is examined for text segments that match the surface form of a signal which are placed in their own chunk if necessary. A set of features, listed in Table 3, is then computed and used to classify each of these chunks. The hyperterm feature can be described as a broadly applicable feature space constructed from SUMO. Each SUMO class, instance, or relation is a unique string, here called a term. All terms are collected and connected to their hyperterm via the SUMO axioms *(subclass) (subrelation)* and *(instance)*. The result is a directed acyclic graph connecting all 1020 terms to their hyperterms and rooted at the SUMO class *Entity*. This hyperterm tree along with the WordNet to SUMO mappings [Niles and Pease, 2003] is prepared offline. Using this resource to generate the hyperterm feature consists of computing the union of the hyperterms for all WordNet concepts (senses) for each word in the chunk.

The intuition behind using SUMO is: (1) Coarse grained distinctions appear to be better represented in SUMO hypterterm tree than in WordNet, that is, a single term "Process" appears to capture the idea of eventness easily where several WordNet hypernyms would be required to capture the same distinction. (2) All open class words are united in a single hyperterm tree whereas in WordNet verbs have shallow hypernymy and Adjective and Adverbs have no hypernymy. This allows for i) better generalization and similarity across chunks that have words not seen during training since unseen words will still share a significant set of hyperterm features and for ii) unified treatment of all four open lexical classes in the software. The resulting decision will be one of 8 choices corresponding to "No signal" and the 7 types of signals listed in Table 2.

Temporal Event Attachment

Once a signal has been identified, its associated events (E1 and E2) must be attached. This task is achieved with a local attachment algorithm followed by a signal chaining attachment algorithm.

Feature Description	Origin
Signal surface form	[Hindle and Rooth, 1993]
Chunk's phrase tag	[Kudo and Matsumoto, 2001]
Chunk's hyperterms	[new]
+/- 1 chunk's phrase tag	[Kudo and Matsumoto, 2001]
+/- 1 chunk's hyperterms	[new]
+/- 2 chunk's phrase tag	[Kudo and Matsumoto, 2001]
+/- 2 chunk's hyperterms	[new]

Table 3: Signal and Local Attachment Features and their Origin

The *Local Attachment Algorithm* proceeds from left to right evaluating each unattached chunk to its nearest signal by generating features for classification. The decision associated with the features is one of *Skip* (do not attach), *E1* (attach as E1), or *E2* (attach as E2). The features used and their origin are listed in Table 4.

The *Signal Chaining Attachment Algorithm* enables signals that are missing an attachment to pick up an attachment from another signal, called link signals. The algorithm proceeds left to right evaluating a signal's missing argument against another signal's existing argument. The decision is either *Skip*, *E1*, or *E2* wherein the latter two result in an attachment. The algorithm iterates until no attachments are made. Although this algorithm uses the features in Table 3, their contribution is low and so motivated the additional features listed in Table 4.

Feature Description	Origin
Missing signal argument (E1,E2)	[new]
Signal class	[mod. from [Allen, 1991]]
Link signal surface form	[Hindle and Rooth, 1993]
Link signal argument (E1,E2)	[new]
Link signal class	[mod. from [Allen, 1991]]
Parsing direction (left,right)	standard

Table 4: Signal Chaining Features and their Origin

Results

Each classifier was built using the LIBSVM [1] implementation of Support Vector Machines, with the radial basis kernel, $\exp(-\gamma\|u-v\|^2)$, and included performing a grid search to find the best model parameters. Measurements were taken by randomizing the annotated sentences and evaluating each tenth of the data by training on the other 90% (10-fold cross-validation). The results measure the performance of the module as a whole. Measuring individual decisions of the classifiers are not instructive due to the dominance of *NoSignal* decisions during disambiguation and *Skip* decisions during

[1] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

attachment parsing.

Task	F-measure
Signal detection	71.4%
Attachment	49.1%

Table 5: Automated Detection Results

The attachment results are affected negatively by parsed chunk boundaries not matching the temporal event text segment marked by the annotator, which results in an annotated attachment being scored as incorrect. An attachment will also score as incorrect if it is not the correct argument of the signal, or if the signal was not classified correctly. No partial credit is given. In the current data set 41.8% of the attachment annotations fail to align with a chunk boundary. This indicates that temporal event boundaries should not always rely on the parse chunk boundaries. An attachment will also score as incorrect if it is not the correct argument of the signal, or if the signal was not classified correctly. The scoring program is strict in that a temporally ordered event is correct only when the signal has been classified correctly and both event arguments have been detected and assigned in the proper order.
Based on the results of our experiments reported in Table 5, with 1300 sentences from the TREC corpus, it seems apparent that more training data is needed. In order to verify that, the method and algorithm used for the current implementation is sound and scalable, another experiment was conducted using the same methodology over half the corpus (650 sentences). The results of that experiment posted an F-measure of 52.7% for signal detection and 36.1% for attachment. Comparing that to the original results it is clear that the technique outlined in this section scales well and will benefit from the ongoing effort to annotate more training data.

6 A Knowledge Representation with Temporal Context

The output of the Temporal Event Detection, described in section 4, is transferred to the SUMO enhanced logical form module. The function of this module is to translate the natural language candidate sentences, marked up with time event chunks, to a temporally enhanced first order logic assertion. The input time event chunks are labeled with the class of the signal from the list in Table 2, along with the parsed chunks identified as the time event arguments to the signal.

The Knowledge Representation for the temporally enhanced logic form is layered on top of the Logical Representation proposed in [Rus and Moldovan, 2002], where the first order logic structure is derived from the syntactic parse of the text and each token in the text is converted to a predicate. From this structure temporally related SUMO predicates are generated based on hand coded interpretation rules for the signal classes. The purpose of the interpretation rules is to define an algorithm for assigning a signal word to a SUMO predicate and defining the manner in which the slots for the predicate are determined. Table 6 enumerates the signal classes, and the SUMO predicate corresponding to the interpretation rule.

Signal Class	SUMO Logic
<S sequence, E1, E2>	earlier(E1,E2)
<S contain, E1, E2>	during(E1,E2)
<S overlap, E1, E2>	overlapsTemporally(E1,E2)
<S open_right, E1, E2>	meetsTemporally(E1,E2)
<S open_left, E1, E2>	meetsTemporally(E2,E1)
<S closed, E1, E2>	duration(E1,E2)

Table 6: Signal Class to SUMO Mapping

Once the SUMO predicate is chosen, the arguments to the predicate are the event argument ids from the heads of the chunks passed as attachments to the signal expression. Since all temporal SUMO predicates operate on time intervals, absolute times stated in the text are translated into a pair of time point predicates to specify the begin and end of the interval. A detailed example follows for the text: *That compares with about 531,000 arrests in 1993 before Operation Gatekeeper started.*

The temporal event recognizer disambiguates **in** and **before** as temporal signals, and classifies (1) **in** as a contain interval signal, and (2) **before** as a sequence interval signal. The *Local Attachment and Signal Chaining* algorithms then determine the time event arguments for each signal. The output triples are *(S1:contain=in, E1=arrests, E2=1993)* and *(S2:before=before, E1=arrests, E2=started)*. Once the mapping for the temporally ordered events and the absolute time events are complete, the SUMO predicates are generated. Predicates that were derived from signal words replace the signals in the logical form.
... 531_000_NN(x2) & arrest_NN(x2) & 1993_NN(x3) & Operation_NN(x4) & Gatekeeper_NN(x5) & nn_NNC(x6,x4,x5) & start_VB(e2,x6,x10) & ***earlier(WhenFn(x2), WhenFn(e2)) & Time(BeginFn(x11), 1993, 1, 1, 0, 0, 0) & Time(EndFn(x11), 1993, 12, 31, 23, 59, 59) & during(WhenFn(x3), x11) & during(WhenFn(x2), x3)***

7 Temporally Enhanced Inference

The temporal context resolution module employs a first order logic resolution style theorem prover adapted for natural language text processing [Moldovan et al, 2003]. The Set of Support (SOS) is the search strategy employed by the prover to partition the axioms into those that have support and those that are considered auxiliary [Wos, 1988]. This strategy restricts the search such that a new clause is inferred if and only if one of its parent clauses come from the SOS. The inference rule sets are based on hypperresolution, paramdodulation, and demodulation. Hyperresolution is an inference rule that does multiple steps in one, paramodulation introduces the notion of equality substitution for the demodulators required on the temporal axioms. Since temporal reasoning is intensive and may require long proofs, hyperresolution and paramodulation inference rules are necessary as they allow for a more compact and efficient proof by condensing multiple steps into one.

The inputs to the context resolution module include:
(1) The context relevant pieces of the natural language questions and candidate answers converted to a SUMO enhanced

first order logical form [Rus and Moldovan, 2002]. The question predicates are negated to invoke a proof by contradiction
(2) A set of empirically derived world knowledge axioms. Ex: EVIDENCE IS SUPPORT FOR A CLAIM
evidence_NN(x2) -> support_NN(x2) & for_IN(x2,x3) & claim_NN(x3)
(3) Linguistic transformation rules derived from the input parse of the question and its candidate answers.
Ex: THE LEADER OF X, LEADS X.
_human_NE(x1) & leader_NN(x1) & of_IN(x1,x2) <-> lead_VB(e1,x1,x2)
(4) WordNet derived axioms such as lexical chains[Moldovan and Novischi, 2002] and glosses. Ex: TO START SOMETHING IS SEMANTICALLY RELATED TO ESTABLISHING SOMETHING
start_VB(e1,x1,x2) -> establish_VB(e1,x1,x2)
(5) A SUMO Knowledge Base of temporal reasoning axioms that consists of axioms for a representation of time points and time intervals, Allen primitives, and temporal functions.
Ex: DURING IS A TRANSITIVE ALLEN PRIMITIVE
during(TIME1, TIME2) & during(TIME2, TIME3) -> during(TIME1, TIME3)

Axioms in set (1) are loaded into the Set of Support. The predicates representing the temporally relevant information in the question are *anded* together, negated, and universally quantified, so as to invoke a proof by contradiction. The temporal predicates for the candidate answer are also *anded* together, existentially quantified and loaded into the SOS. The axioms in sets (2) - (3) are loaded into the Usable list to assist in the inference process between the question and the answer.

The output of the temporal context module comprises filtered and re-ranked answers that are scored based on their semantic and temporal similarity to the question. The inference engine will continue seeking a proof until any of the *TIME* predicates are unsatisfied. Partial satisfiability is permitted for other predicates in the question via a backoff algorithm integrated in the inference engine [Moldovan et al, 2003].

The following example illustrates how a temporal context resolution module is used to verify the temporal constraints of a question over a candidate answer:
Question: *What country controlled Syria in 1930?*
Question Logical Form:
_country_AT(x2) & control_VB(e1,x2,x1) & Syria_NN(x1) & overlapsTemporally_CTMP(e1,x3) & Time_CTMP(BeginFn(x3),1930,1,1,0,0,0) & Time_CTMP(EndFn(x3),1930,12,31,23,59,59)
Question Axiom:
-(exists e1 x1 x2 x3 (_country_AT(x2) & control_VB(e1,x2,x1) & Syria_NN(x1) & overlapsTemporally_CTMP(e1,x3) & Time_CTMP(BeginFn(x3),1930,1,1,0,0,0) & Time_CTMP(EndFn(x3),1930,12,31,23,59,59)))
Candidate Answer 5:
France has a different relationship with Syria, partly because it was once a French-mandated territory, from 1920-1946
Answer Logical Form:
France_NN(x1) & _country_NE(x1) & have_VB(e1,x1,x2) & different_JJ(x2) & relationship_NN(x2) & with_IN(x2,x6) & Syria_NN(x6) & partly_RB(e1) & because_IN(e1,e3) & once_RB(e3) & French_NN(x5) & mandate_VB(e3,x5,x6) & territory_NN(x6) & during_CTMP(x6,x9) & Time_CTMP(BeginFn(x9),1920,1,1,0,0,0) & Time_CTMP(EndFn(x9),1946,12,31,23,59,59)
Answer Axiom:
exists e1 e2 x1 x2 x3 x4 x6 x7 x8 x9 (France_NN(x1) & _country_NE(x1) & have_VB(e1,x1,x2) & different_JJ(x2) & relationship_NN(x2) & with_IN(x2,x6) & Syria_NN(x6) & partly_RB(e1) & because_IN(e1,e3) & once_RB(e3) & French_NN(x5) & mandate_VB(e3,x5,x6) & territory_NN(x6) & during_CTMP(x6,x9) & Time_CTMP(BeginFn(x9),1920,1,1,0,0,0) & Time_CTMP(EndFn(x9),1946,12,31,23,59,59))
Linguistic Axiom:
all x1 (French_NN(x1) → France_NN(x1))
Lexical Chain Axiom:
all e1 x1 x2 (mandate_VB(e1,x1,x2) → control_VB(e1,x1,x2))
SUMO Axioms:
Two time intervals overlapsTemporally if and only if each one is a temporalPart of the other:
all I1 I2 (ISA(I1, TimeInterval) & ISA(I2, TimeInterval) & overlapsTemporally_CTMP(I1, I2) ↔ (exists I3 (ISA(I3, TimeInterval) & temporalPart_CTMP(I3, I1) & temporalPart_CTMP(I3, I2))))
Interval1 is temporalPart of Interval2 if Interval1 is during Interval2:
all T1 T2 (ISA(T1, TimeInterval) & ISA(T2, TimeInterval) & during_CTMP(T1, T2) → temporalPart_CTMP(T1, T2)).
Interval1 is during Interval2 if Interval1 starts after Interval2 and Interval1 ends before Interval2:
all I1 I2 ((before_CTMP(EndFn(I1), EndFn(I2)) & before_CTMP(BeginFn(I2), BeginFn(I1))) → during_CTMP(I1,I2)).
A mapping of question predicates to the inferred answer predicates is provided in Table 7:

Question Predicates	Answer Predicates
_country_AT(x2)	French_NN(x5) → France_NN(x5)
control_VB(e1,x2,x1)	mandate_VB(e3,x5,x6)
Syria_NN(x1)	Syria_NN(x6)
overlapsTemporally_CTMP(e1, x3) & Time_CTMP(BeginFn(x3), 1930, 1, 1, 0, 0, 0) & Time_CTMP(EndFn(x3), 1930, 12, 31, 23, 59, 59)	during_CTMP(x6, x9) & Time_CTMP(BeginFn(x9), 1920, 1, 1, 0, 0, 0) & Time_CTMP(EndFn(x9), 1946, 12, 31, 23, 59, 59)

Table 7: Question to Answer Predicate Resolution

Proofs for answers which do not have the correct context will fail, resulting in the candidate answer being pruned from the answer list. This in turn promotes all candidates below. As an example, the context resolution module will fail to find a proof for the first candidate answer returned by the answer processing module (without temporal reasoning), whose document date is 1998:
The United States did not punish Israel when it occupied territories of some Arab countries such as Palestine and Syria, and refused to comply with relevant U.N.resolutions on the Middle East issues.

Since the document date of the candidate does not meet the temporal constraints of the question, the incorrect answer is filtered from the list and correct answer, candidate 5 is rightly promoted to position 1.

8 Question Answering Results

A set of 200 time dependent questions was used as a benchmark for measuring the contribution of temporal context to a state of the art question answering system. The benchmark was created primarily of questions that required temporal reasoning and so were previously unanswerable. Further, they were derived from a 300 MB sub-collection of the TREC corpus that included a sample of XIN, APW, and NYT articles. Table 8 summarizes the results of the system at the exact and sentence answer level with and without temporal context. The first two columns list QA results for the percent of correct answers in the first position and the Mean Reciprocal Rank, without temporal context. The second two columns list the corresponding results for temporally enhanced QA. The final column gives the percent increase for answers in the first position.

answer type	no context		context		increase in
	first	MRR	first	MRR	first
exact	16.4%	.191	20.0%	.259	22.0%
sentence	29.1%	.328	37.1%	.381	30.0%

Table 8: QA Results

9 Discussion

A means to represent and reason about context is crucial for the performance of a question answering system over a time sensitive corpus. Advantages include: retrieving answer passages with relative or underspecified dates, discarding temporally inaccurate answers, and capturing the ordering of events via signal words. Temporal inference is expensive, however, and should be ameliorated by development of a special purpose reasoner. Further, the performance of signal and event detection should be improved by the ongoing effort to annotate training data, as demonstrated by the scaling experiments. Despite this, the results presented in this paper have shown that significant improvement in open domain question answering can in fact be achieved by integrating temporal context reasoning.

10 Acknowledgements

This work was supported in part by the ARDA AQUAINT program. Special thanks to Bob Hauser, Arthur Dexter, and Jens Stephan for their invaluable research, implementation, and annotation efforts.

References

[Allen, 1991] James Allen. *Time and Time Again: The Many Ways to Represent Time*. International Journal of intelligent Systems, July 1991, pp341-355.

[Filatova and Hovy, 2001] Elena Filatova and Eduard Hovy. 2001. *Assigning Time-Stamps to Event-Clauses*. Proceedings of ACL Workshop on Temporal and Spatial Reasoning. Toulouse, France.

[Hindle and Rooth, 1993] Donald Hindle and Mats Rooth 1993 *Structural ambiguity and lexical relations*. Computational Linguistics, 19(2):313-330

[Kudo and Matsumoto, 2001] Taku Kudo and Yuji Matsumoto 2001 *Chunking with Support Vector Machines.* Proceedings of the NAACL, pages 192-199.

[Mani and Wilson, 2000] Inderjeet Mani and George Wilson. 2000. *Robust Temporal Processing of News.* Proceedings of the 38th Annual Meeting of the ACL. Honkong, China

[Moldovan and Novischi, 2002] D. Moldovan and A. Novischi 2002 *Lexical Chains for Question Answering.* Proceedings of COLING 2002, pages 674-680.

[Moldovan et al, 2003] D. Moldovan, C. Clark, S. Harabagiu, S. Maiorano 2003 *COGEX: A Logic Prover for Question Answering.* Proceedings of the Human Language Technology and North American Chapter of the Association for Computational Linguistics 2003, pages 87-93.

[Niles and Pease, 2001] Ian Niles and Adam Pease. 2001. *Towards a Standard Upper Ontology*. Proceedings of the 2nd International Conference on Formal Ontology in Information Systems. Ogunquit, Maine, October 2001

[Niles and Pease, 2003] Ian Niles and Adam Pease. 2003. *Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology*. Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE '03). Las Vegas, Nevada, June 23-26, 2003

[Pustejovsky et al., 2000] Pustejovsky, Castano, Ingria, Sauri, Gaizauskas, Setzer, Katz. 2000. *TimeML: Robust Specification of Event and Temporal Expressions in Text*. Fifth International Workshop on Computational Semantics

[Pustejovsky et al., 2002] James Pustejovsky. 2002 *TERQAS: Time and Event Recognition for Question Answering Systems*. An ARDA Workshop on Advanced Question Answering Technology, January-July 2002.

[Rus and Moldovan, 2002] V Rus and D. Moldovan. 2002 *A High Performance Logic Form Transformation.* International Journal on Artificial Intelligence Tools, Vol. 11, No. 3 (2002) 437-454

[Wos, 1988] Larry Wos. 1998. *Automated Reasoning, 33 Basic Research Problems*. Prentice Hall, 1988.

A Machine Learning Approach to Identification and Resolution of *One*-Anaphora

Hwee Tou Ng[*], **Yu Zhou**[*], **Robert Dale**[†] **and Mary Gardiner**[†]
[*]Department of Computer Science, National University of Singapore, Singapore
[†]Centre for Language Technology, Macquarie University, Sydney, Australia
{nght,zhouyu}@comp.nus.edu.sg, {rdale,gardiner}@ics.mq.edu.au

Abstract

We present a machine learning approach to identifying and resolving *one*-anaphora. In this approach, the system first learns to distinguish different uses of instances of the word *one*; in the second stage, the antecedents of those instances of *one* that are classified as anaphoric are then determined. We evaluated our approach on written texts drawn from the informative domains of the British National Corpus (BNC), and achieved encouraging results. To our knowledge, this is the first learning-based system for the identification and resolution of *one*-anaphora.

1 Introduction

The word *one* is a frequently used word in English: in the 100-million-word British National Corpus (BNC), it accounts for 0.26% of all the words, ranking as the 110th most common word, and the 14th most common pronominal form, more frequent than the pronoun *us*.[1] Not all of these instances of *one* are anaphoric; the most common use is as a number (as in *one book*), and there are a number of other uses. However, any natural language processing (NLP) system which tries to process anaphora in unrestricted text will need to be able to determine whether a particular instance of *one* is being used anaphorically, and when this is the case, what the antecedent of the anaphor is.

In brief, a *one*-anaphor is an anaphoric noun phrase (NP) headed by the word *one*, as in the following example:

(1) Her greater sympathy for *the Atlantic connection* over *the European one* is not widely shared by colleagues.

One-anaphora is sometimes referred to as "identity-of-sense anaphora", in contrast to the more common pronominal "identity-of-reference" anaphora [Hankamer and Sag, 1976]. In processing a pronominal reference, we are generally looking for an antecedent noun phrase that refers to the same entity as the anaphoric form; however, *one*-anaphoric forms are generally used to refer to something of the same *kind* as something mentioned before. Hence, in order to interpret an instance of *one*-anaphora, we must identify the antecedent textual material that provides the semantic content alluded to by the use of *one*. In Example (1), correctly interpreting *the European one* requires inferring that it has a relationship to the noun phrase *the Atlantic connection*. The range of semantic relationships that can hold between an *one*-anaphoric form and its antecedent is broad and complex; for present purposes, we are interested in identifying the antecedent noun phrase that contains the relevant semantic content.

We designed two systems that are applied consecutively to identify and resolve *one*-anaphora. In the case of Example (1), our *one*-expression classifier first determines that *the European one* is in fact a *one*-anaphor; then our *one*-anaphora resolver would identify the noun phrase *the Atlantic connection* as the antecedent of the *one*-anaphor. In this paper, an antecedent is defined as the noun phrase from which the head sense of the *one*-anaphor can be determined.

The pervasiveness of anaphoric reference in general means that anaphora resolution is recognized as an important subtask in natural language processing; *one*-anaphora resolution, however, has been relatively neglected. As noted, the frequency of occurrence of the word *one* means that any real NLP system cannot just ignore it; however, the knowledge sources (typically, features of the linguistic context) used in state-of-the-art noun phrase coreference resolution systems (e.g., [Soon *et al.*, 2001; Ng and Cardie, 2002]) are not applicable to *one*-anaphora. Consequently, *one*-anaphora resolution is a task that requires special attention.

The rest of this paper is organized as follows. We first introduce the various uses of the word *one* in English and the different types of antecedents of *one*-anaphora. Then we give an overview of how our *one*-expression classifier and *one*-anaphora resolver are applied consecutively to identify and resolve *one*-anaphora, followed by two separate sections giving a detailed description and evaluation of the two systems. In the final two sections, we describe related work and conclude.

2 Classes of *One*-Expressions

In this section, we introduce the various uses of *one* in English and the different types of antecedents of *one*-anaphora, along with some statistics of their distribution in the BNC.

[1]These figures are based on Adam Kilgarriff's frequency lists at http://www.itri.brighton.ac.uk/∼Adam.Kilgarriff/bnc-readme.html.

2.1 Uses of *One* and Corpus Annotation

The taxonomy of uses of the word *one* adopted in this section is based on existing theories of the various uses of *one* [Halliday and Hasan, 1976; Webber, 1979; Dahl, 1985; Luperfoy, 1991] and our own study of *one*-expressions. We divided uses of *one* into six classes: Numeric, Partitive, Anaphoric, Generic, Idiomatic, and Unclassifiable.

1. **Numeric** *One***:** Modifies a head noun to indicate singularity as in Example (2); this is the only adjectival use of *one*.

(2) John has *one* blue T-shirt.

2. **Partitive** *One***:** Selects an individual from a set. It is followed by an *of*-prepositional phrase headed by a plural noun or pronoun as in Example (3).

(3) A special exhibition of books for children forms *one* of the centrepieces of the 41st annual Frankfurt Book Fair.

3. **Anaphoric** *One***:** Relates a set of properties to the set of properties mentioned by the antecedent. There are three types of *one*-anaphors, distinguished by the type of their antecedents. First, the antecedent may be a kind, as in the *noisy cameras* of Example (4); second, the antecedent may be a set of entities, as in the set of *two World Bank men* of Example (5); and third, the antecedent may refer to a single instance, as in *this book* of Example (6).

(4) I have an aversion to *noisy cameras*, and this *one* rings several decibels before it's done with winding on the film.

(5) *The two World Bank men*, *one* German and *one* British, strode across the tarmac.

(6) Would you like *this book*? Yes, I would like that *one*.

4. **Generic** *One***:** A pronominal use that refers to a generic person or to the speaker of a sentence; often used in subject position followed by a modal verb and a main verb that takes an animate subject.

(7) *One* must think a little deeper to discover the underlying social roots of the problem.

5. **Idiomatic** *One***:** Conventionalized uses whose semantics appear not to be based on general use, but rather on idiomatic patterns, as in Example (8).

(8) It would be perfect to have a loved *one* accompany me in the whole trip.

6. **Unclassifiable** *One***:** Inevitably, there are instances which are difficult to classify as any of the above, as in Example (9).

(9) Cursed be every one who curses you.

For the present study, we randomly selected 1,577 *one* expressions from the BNC [2], and manually annotated these with the six classes above. The distribution of each class in the annotated corpus is shown in Table 1, and it mirrors the distribution of *one* expressions in naturally occurring text.

[2]These *one*-expressions are from written text in the informative domains of BNC. Spoken text and written text in the imaginative domain (i.e., texts which are fictional or which are generally perceived as literary or creative) are not considered because they contain a large amount of dialog, which makes *one*-anaphora resolution a harder task. This restriction follows the genre of MUC texts [MUC-7, 1997], a widely used data set in noun phrase coreference research, which also only included written newspaper texts.

Class	Frequency	%
Numeric	739	46.9
Partitive	399	25.3
Anaphoric	240	15.2
Generic	167	10.6
Unclassifiable	25	1.6
Idiomatic	7	0.4
Total	1,577	100.0

Table 1: Distribution of uses of *one* in the annotated corpus.

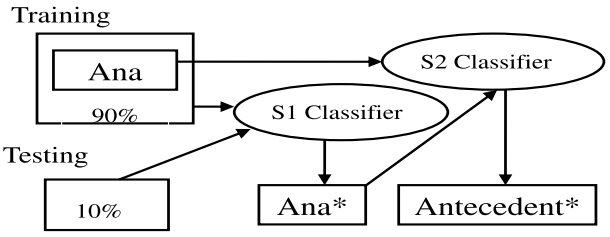


Figure 1: How training and test data are used.

2.2 Annotating Antecedents of *One*-Anaphora

We annotated the 240 examples of *one*-anaphors in our corpus a second time, marking the antecedent in each case. Not every *one*-anaphor has an explicit noun phrase antecedent as shown in the previous anaphoric examples; in such cases, the reader has to infer the nature of the antecedent from the available text. We label those cases where there is no explicit antecedent as "*one*-anaphors with implicit antecedents". The remaining *one*-anaphors are "*one*-anaphors with explicit antecedents". Of the 240 *one*-anaphors, 98.3% had explicit antecedents.

3 An Overview of Our Approach

Before we describe our two machine learning systems for *one*-expression classification and *one*-anaphora resolution, we first provide an overview of how the two systems are applied consecutively to accomplish the overall task of identifying and resolving *one*-anaphora. As an example, we use one trial in the 10-fold cross validation to illustrate the process. Here, the identification of *one*-anaphora is Step 1 (S1), and the resolution of *one*-anaphora is Step 2 (S2).

As shown in Figure 1, a gold standard corpus is divided into two sets, containing 90% and 10% of the data respectively. The larger set is used to train the S1 classifier. In the S1 training data, there is a set of anaphoric examples, Ana. This set is used to train the S2 classifier.

The smaller set of 10% of the examples is used for testing. First, the test examples are passed to the S1 classifier, which identifies a set of anaphoric examples, Ana*. This set may contain errors in identification (a *one*-anaphor not classified as anaphoric, or a non-anaphoric example classified as

anaphoric). Ana* is in turn passed to the S2 classifier, which outputs the antecedents it finds, denoted as Antecedent*. When calculating accuracy, Ana* and Antecedent* are used to compare with the gold standard annotation to measure both the S1 accuracy and the overall accuracy of the combined systems.

4 *One*-Expression Classification

Our *one*-expression classifier classifies a given *one*-expression into one of the six classes listed in Table 1.

4.1 The Features

We devised a set of features that is useful in determining which of the six classes a given *one*-expression belongs to. We focus on the classes Numeric, Partitive, Anaphoric, and Generic, since 98% of the instances are from these classes.

The Numeric use of *one* is the simplest case, since it is the only adjectival use in English. It can be readily identified by the part-of-speech (POS) tag of the word *one*. Partitive use of *one* can be identified by checking the syntactic context (i.e., [*one* [*of* NP_{pl}]]).

Discrimination between the Anaphoric and Generic classes is more complex. We used the three features *isSubj*, *isAnimateVerb*, and *isModalVerb* to identify instances of the Generic class, in line with the observations mentioned in the definition of Generic *One* . We also noticed that the relative position of *one* in its host NP and the category of the word immediately preceding *one* both give strong hints as to the function of *one* in its host NP. Therefore, we added two more features, *positionInNP* and $W_{-1}POS$, to assist in classification.

We experimented with a total of 7 features as described below. Each instance in the training and test data set is thus represented as a feature vector of seven values. The feature values are acquired by running the Charniak Parser [Charniak, 2000] over the corpus, in combination with information about verbs of cognition from WordNet [Fellbaum, 1998]:

1. $\mathbf{W_0POS}$ is the POS tag of *one* assigned by the Charniak Parser. Its possible values are *CD*, *NN*, *NNP*, *PRP*.
2. **isOfPlural** checks whether *one* is followed by *of* prepositional phrase with a plural head noun/pronoun. Its possible values are *plural*, *notPlural*, or *NA* when the word *one* is not followed by a PP headed by *of*. For example:
   (a) *plural*: *One* of the patients survived.
   (b) *notPlural*: The problem was the unusual *one* of a warmish, wet spring.
3. **isSubj** checks whether the word *one* is in subject position. This feature value is inferred from the parse tree structure; its possible values are *true* or *false*.
4. **isAnimateVerb** checks whether the lemmatized verb following subject *one* is a verb of cognition according to WordNet: some examples are *think*, *judge*, *analyze*, and *doubt*. The possible values for this feature are *true*, *false*, or *NA* when *one* is not the subject. Example (7) would get *true* for this feature.
5. **isModalVerb** checks whether the verb phrase following a subject *one* contains a modal verb (i.e., a word with MD POS tag, such as *must*). The possible values for this feature are *true*, *false*, or *NA* when *one* is not the subject. Example (7) would get *true* for this feature.
6. **positionInNP** indicates the position of the word *one* in its host NP. It has four possible values: *SingleOne* when the NP only consists of the word *one*; *Leftmost* when the NP has multiple words and *one* is the leftmost word; *Rightmost* when the NP has multiple words and *one* is the rightmost word; and *Middle* otherwise. For example:
   a) *SingleOne*: [*One*] might suppose that ...
   b) *Leftmost*: I was concerned with [*one* thing] only.
   c) *Rightmost*: [the European *one*]
   d) *Middle*: on [the *one* hand]
7. $\mathbf{W_{-1}POS}$ is the POS tag of the word immediately preceding *one*. Its possible values are the 45 POS tags in Penn TreeBank tagset and *NA* when *one* is the first word of the sentence.

	Num in data set	Num identified	Num correct	R	P	F1
NUM	739	730	693	93.8	94.9	94.3
PAR	399	412	390	97.7	94.7	96.2
ANA	240	281	192	80.0	68.3	73.7
GEN	167	152	127	76.0	83.6	79.6
Sub Total	1,545	1,575	1,402	90.7	89.0	89.8
UNC	25	2	0	0	0	—
IDIO	7	0	0	0	—	—
Total	1,577	1,577	1,402	88.9	88.9	88.9

Table 2: Accuracy of *one*-expression classification.

The learning algorithm used in our *one*-expression classification system is C4.5 [Quinlan, 1993]. This is a commonly used decision tree learning algorithm and may be considered as a baseline against which other learning algorithms can be compared.

4.2 Evaluation and Error Analysis

Evaluation

Table 2 gives the 10-fold cross validation results for the *one*-expression classifier for each of the 6 classes.

The accuracy of identifying *one*-anaphora is 73.7%, which is not as high as the Numeric, Partitive, and Generic classes. The remaining two classes, Idiomatic and Unclassifiable, are poorly discriminated; when counted in, these pull down the overall classification accuracy from 89.8% to 88.9%.

Error Analysis

Table 3 provides a matrix of the number of misclassifications in each class. Row 4 and Column 4, highlighted, show that the major confusion with the Anaphoric class comes from the Generic and Numeric classes, with Partitive making a smaller contribution to the erroneous classifications.

Our classifier's performance is highly impacted by the accuracy of POS tagging and parsing: failures here caused most of the confusion with Numeric and Partitive. For example, *one* in (4) is classified as Numeric because *rings* is wrongly

classified as=>*	NUM*	PAR*	**ANA***	GEN*	UNC*	IDIO*
NUM	na	2	**35**	8	1	0
PAR	3	na	**6**	0	0	0
ANA	**14**	**18**	**na**	**15**	**1**	**0**
GEN	9	0	**31**	na	0	0
UNC	10	1	**12**	2	na	0
IDIO	1	1	**5**	0	0	na

Table 3: *One*-expression classifier error matrix.

tagged as NNS; and Partitive *one* in the phrase *one or more of them* is classified as Anaphoric because it is wrongly parsed as *[[one] or [more of [them]]]*.

The confusion with the Generic class is mainly caused by occurrences of anaphoric *one* as subject and generic *one* in non-subject position. Such cases can also be confusing for human readers: in a sentence like *One is usually shunting around in the yard*, we might think we have a generic use, but the *one* may refer to a previously mentioned *locomotive*.

5 *One*-Anaphora Resolution

Our *one*-anaphor resolver attempts to identify the NP in the preceding linguistic context that provides the semantic content required for interpretation of a *one*-anaphor; we refer to this NP as the antecedent NP.

5.1 Experimental Data

We trained and tested a *one*-anaphor resolution classifier using a set of positive/negative antecedent-anaphor pairs. A pair is positive when the candidate antecedent in this pair is the actual antecedent of the anaphor; otherwise, it is negative.

Training Data

In each trial of the 10-fold cross validation, the actual *one*-anaphora instances in the gold standard training corpus of Step 1 are used to create Step 2 training instances in the corresponding trial as shown in Figure 1.

Creation of training instances consisted of three steps. First, each sentence containing *one* and its three preceding sentences [3] were processed by RM NP chunker [Ramshaw and Marcus, 1995] to carry out NP chunking. Second, each pair of an anaphor and its actual antecedent were used to create a positive training instance. Lastly, to generate negative training instances, anaphors were paired with each of the NPs that appeared between the anaphor and its real antecedent.

In our experiment, we further adjusted the ratio of positive to negative instances in the training data by controlling the number of negative instances randomly picked from the whole set of negative instances. We decided to set the ratio at 1:1, which introduces no preference for either one of the two assignments to the classifier. This procedure produced a set of 472 antecedent-anaphor pairs in total, of which 236 (50%) were positive instances. As already noted, each trial used roughly 90% of this set as training data.

[3] If the sentence containing one appeared close to the beginning of the text and there were less than three preceding sentences, we used all available preceding sentences.

Test Data

In each trial, instances of *one* identified as being anaphoric in the test data of Step 1 were used to create Step 2 test instances in the corresponding trial as shown in Figure 1.

Creation of test instances consisted of two steps. The first step is the same as that for the training data; in the second step, every base NP preceding the instance of *one* is a potential antecedent, so each of these NPs was paired with *one*.

When doing testing, the *one*-anaphora resolution algorithm starts from the immediately preceding base NP and proceeds backward in the reverse order of the NPs in the context until there is no remaining NP to test, or an antecedent is found (i.e., the classifier returns true).

5.2 The Features

To decide whether the anaphor in a given antecedent-anaphor pair refers to the candidate antecedent, we need features that show a preference for good candidates.

As with pronominal anaphora, an intuitively appealing feature to use is whether the candidate antecedent NP is in focus [Sidner, 1981; Vieira and Poesio, 2000]. In anaphora resolution, two commonly used features for approximating the notion of focus are syntactic role and recency: a candidate that fills a salient syntactic role such as subject, or one that is very recent, is often the focus of the discourse. We used four features to measure this type of information: *AnteIsSubj*, *AnteInRelClause*, *AnteIsNearestNP*, and *bothInPP*. The feature *bothInPP* allows us to take syntactic parallelism into consideration. The POS tag of the head word of the candidate antecedent is also a good feature in filtering out improper candidates: a proper noun should not be the antecedent of a *one*-anaphor [Dahl, 1985].

We used a total of 5 features as described below. Their values are acquired from the Charniak Parser output of the training/test data set.

1. **hwPOSofAnte** is the POS tag of the head word of the candidate antecedent. Its possible values are the set of POS tags of head words that occur in our training/test data. In this paper, the head word of the candidate antecedent is defined as the rightmost noun, or the rightmost word if no noun is found, in the base NP candidate antecedent.

2. **bothInPP** checks whether the candidate antecedent and the *one*-anaphor are both in prepositional phrases (PP), and identifies the types of PPs they are in. It has five possible values: *NA* when the candidate antecedent is not in PP; *OnlyAnteInPP* when the candidate antecedent is in PP, but the *one*-anaphor is not in PP; *SharePP* when the candidate antecedent and the *one*-anaphor are in the same PP; *CommonPreposition* when both candidate antecedent and *one*-anaphor are in different PPs, but the prepositions of the two PPs are the same; and *DifferentPreposition* when both candidate antecedent and *one*-anaphor are in different PPs with different prepositions.

3. **AnteIsSubj** checks whether the candidate antecedent is in subject position. Its possible values are *true* or *false*.

4. **AnteIsNearestNP** checks whether the candidate antecedent is the nearest NP preceding the *one*-anaphor. Its possible values are *true* or *false*.
5. **AnteInRelClause** checks whether the candidate antecedent is in a relative clause. Its possible values are *true* or *false*.

Again, the learning algorithm used in our one-anaphora resolution engine is C4.5.

5.3 Evaluation and Error Analysis

Evaluation

In order to evaluate the overall performance of our approach to *one*-anaphora identification and resolution, we conducted a 10-fold cross validation of the *one*-anaphora resolver, where each trial is performed based on the result of our *one*-expression classifier in Step 1. The overall recall, precision, and F-measure are presented in Table 4, together with the accuracy of *one*-anaphora identification from Step 1.

The "S2 correct" in Table 4 Column 9 is the sum of two values. The first value is the number of hits in correctly identifying the explicit antecedent of a *one*-anaphor. A hit of this type occurs when a *one*-anaphor with an explicit antecedent is correctly identified as anaphoric in Step 1 and the actual antecedent is found in Step 2. The second value is the number of hits returning no antecedent for a *one*-anaphor when it has no annotated antecedent. A hit of this type occurs when a *one*-anaphor without an explicit antecedent is correctly identified as anaphoric in Step 1, and none of the candidate antecedents is accepted in Step 2.

The overall recall is the sum in "S2 correct" divided by the total number of *one*-anaphors with or without explicit antecedent in the data set (Column 1); the overall precision is the sum in "S2 correct" divided by the total number of *one*-anaphors with or without explicit antecedent identified in Step 1 (Column 3). The F-measure finally achieved is 45.7%.

We compared two baseline heuristics with our accuracy: nearestNP and nearestSubj, which always assign the nearest NP or Subject preceding the *one*-anaphor as its antecedent. The two baseline accuracies are calculated by applying the two heuristics on the set of anaphoric *ones* identified in Step 1; nearestNP (nearestSubj) heuristics achieved 28.8% (23.8%) accuracy, considerably lower than the overall accuracy of our system.

Error Analysis and Future Improvement

Since we perform *one*-anaphora resolution on the output of our *one*-anaphora identification system, the 45.7% overall accuracy is a combination of both Step 1 and Step 2 performance. Errors introduced in Step 1 were never remedied in Step 2, and they directly affected both overall recall and precision. In other words, given a perfect Step 2 classifier to work on the current *one*-anaphora identification output, the highest overall F-measure achievable is 73.7%. Therefore, any further improvement in Step 1 performance would significantly improve the overall accuracy. This could be done by adding features to improve the performance of Step 1, or adding a remedy strategy in later processing.

Our system finds it difficult to locate actual antecedents that are far away from their *one*-anaphors: preference is wrongly given to closer and more salient candidates. In Example (10), *the director* has a strong syntactic preference and it is considered before *moustache*, so the system wrongly returns *the director* even before checking the actual antecedent.

(10) He has this ridiculous *moustache*. Ken Russell, the director, insisted I grew one of my own, rather than wear a false *one*, so that I looked completely convincing.

We expect that such mistakes could be corrected by using semantic features.

5.4 The Contribution of the Features

To evaluate the relative contribution of the various knowledge sources to the overall accuracy of *one*-anaphora identification and resolution, we ran a series of leave-one-out classifiers, where we first used all Step 1 features and disabled one Step 2 feature at a time; then we used all Step 2 features and disabled one Step 1 feature at a time. The contribution of the features measured in terms of the overall F-measure is shown in Table 4 Column 12.

The critical features which cause a substantial reduction of overall F-measure when disabled are Step 1 features *positionInNP*, *isOfPlural*, and W_0POS, as well as Step 2 features *hwPOSofAnte* and *bothInPP*. The remaining features only have a small impact on the overall F-measure.

6 Related Work

The literature on *one*-anaphora is small; we cited the most significant works in the area in Section 2. Most of the existing literature is more concerned with describing the phenomenon than in determining how it might be handled automatically.

There is, of course, an extensive literature on computational techniques for resolving pronominal anaphora, going back to at least the 1970s. Of the more recent research in the area, important work is that of Lappin and Leass [1994] and Kennedy and Boguraev [1996], who provided heuristics that could be used to determine the antecedents of pronominal forms. Soon *et al.* [2001] and Ng and Cardie [2002] used a machine learning approach for coreference resolution. However, most of the linguistic features used in the work on pronominal anaphora are not applicable to the *one*-anaphora problem. Markert *et al.* [2003] focused on the related phenomenon of *other*-anaphora.

7 Conclusion

In this paper, we have presented a machine learning approach to the identification and resolution of *one*-anaphora and achieved encouraging results; to our knowledge, this is the first learning-based system for resolving *one*-anaphora. There is scope for refinement to improve both the accuracy of identifying anaphoric uses of *one*, and of identifying the antecedent noun phrase that contains the semantic content required for interpreting the *one*-anaphor. Beyond this goal, there are more complex challenges awaiting in terms of *one*-anaphora interpretation.

Num of Ana in data set	**Step1: 7 features (f1-f7)**						**Step2: 5 features (f1-f5)**		**Overall**			**Baseline**	
	S1 feature	S1 sysout as Ana	S1 correct	S1 R	S1 P	S1 F1	S2 feature	S2 correct	R	P	F1	n-NP	n-Subj
240	ALL	281	192	80.0	68.3	73.7	ALL	119+0	49.6	42.3	**45.7**	28.8	23.8
							no f1	78+0	32.5	27.8	30.0		
							no f2	99+1	41.7	35.6	38.4		
							no f3	117+0	48.8	41.6	44.9		
							no f4	118+0	49.2	42.0	45.3		
							no f5	119+0	49.6	42.3	45.7		
	no f1	128	76	31.7	59.4	41.3	ALL	49+0	20.4	38.3	26.6	8.3	7.1
	no f2	126	91	37.9	72.2	49.7	ALL	60+0	25.0	47.6	32.8	12.5	13.8
	no f3	251	165	68.8	65.7	67.2	ALL	100+0	41.7	39.8	40.7	25.0	22.1
	no f4	282	185	77.1	65.6	70.9	ALL	114+0	47.5	40.4	43.7	27.1	23.3
	no f5	295	192	80.0	65.1	71.8	ALL	118+0	49.2	40.0	44.1	29.2	24.6
	no f6	276	187	77.9	67.8	72.5	ALL	115+0	47.9	41.7	44.6	27.5	23.3
	no f7	280	191	79.6	68.2	73.5	ALL	118+0	49.2	42.1	45.4	28.3	23.8

Table 4: Overall accuracy and baseline of identification and resolution of *one*-anaphora & contribution of the features. Step1 7 features are 1.positionInNP, 2.isOfPlural, 3.W_0 POS, 4.isAnimate, 5.W_{-1} POS, 6.isSubj, 7.isModalVerb. Step2 5 features are 1.hwPOSofAnte, 2.bothInPP, 3.AnteInRelClause, 4.AnteIsNearestNP, 5.AnteIsSubj

References

[Charniak, 2000] E. Charniak. A maximum-entropy inspired parser. In *Proceedings of NAACL*, pages 132–139, 2000.

[Dahl, 1985] D. A. Dahl. *The structure and function of one-anaphora in English*. PhD thesis, Univ. of Minnesota, 1985.

[Fellbaum, 1998] C. Fellbaum, editor. *WordNet: An electronic lexical database*. The MIT Press, 1998.

[Halliday and Hasan, 1976] M. A. K. Halliday and R. Hasan. *Cohesion in English*. Longman, 1976.

[Hankamer and Sag, 1976] J. Hankamer and I. Sag. Deep and surface anaphora. *Linguistic Inquiry*, 7:391–428, 1976.

[Kennedy and Boguraev, 1996] C. Kennedy and B. Boguraev. Anaphora for everyone: pronominal anaphora resolution without a parser. In *Proceedings of COLING*, pages 113–118, Copenhagen, Denmark, 1996.

[Lappin and Leass, 1994] S. Lappin and H. Leass. A syntactically based algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20:535–561, 1994.

[Luperfoy, 1991] S. Luperfoy. *Discourse Pegs: A Computational Analysis of Context-Dependent Referring Expressions*. PhD thesis, Univ. of Texas at Austin, 1991.

[Markert *et al.*, 2003] K. Markert, M. Nissim, and N. Modjeska. Using the web for nominal anaphora resolution. In *Proceedings of the EACL Workshop on Computational Treatment of Anaphora*, pages 39–46, 2003.

[MUC-7, 1997] MUC-7. Coreference task definition (v3.0, 13 Jul 97). In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1997.

[Ng and Cardie, 2002] V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of ACL*, 2002.

[Quinlan, 1993] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.

[Ramshaw and Marcus, 1995] L. Ramshaw and M. Marcus. Text chunking using transformation-based learning. In *Proceedings of the ACL Third Workshop on Very Large Corpora*, pages 82–94, 1995.

[Sidner, 1981] C. L. Sidner. Focusing for interpretation of pronouns. *American Journal of Computational Linguistics*, 7(4):217–231, 1981.

[Soon *et al.*, 2001] W. M. Soon, H. T. Ng, and D. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001.

[Vieira and Poesio, 2000] R. Vieira and M. Poesio. An empirically based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593, 2000.

[Webber, 1979] B. Webber. *A Formal Approach to Discourse Anaphora*. Garland Publishing Inc., 1979.

Robust Ontology Acquisition from Machine-Readable Dictionaries

Eric Nichols
Nara Inst. of Science and Technology
Nara, Japan
eric-n@is.naist.jp

Francis Bond
NTT Communication Science Labs
Nippon Telegraph and Telephone Co.
Keihanna, Japan
bond@cslab.kecl.ntt.co.jp

Daniel Flickinger
CSLI
Stanford University
California, U.S.A.
danf@csli.stanford.edu

Abstract

In this paper, we outline the development of a system that automatically constructs ontologies by extracting knowledge from dictionary definition sentences using Robust Minimal Recursion Semantics (RMRS), a semantic formalism that permits underspecification. We show that by combining deep and shallow parsing resources through the common formalism of RMRS, we can extract ontological relations in greater quality and quantity. Our approach also has the advantages of requiring a very small amount of rules and being easily adaptable to any language with RMRS resources.

1 Introduction

Ontologies are an important resource in natural language processing. They have been shown to be useful in tasks such machine translation, question answering, and word-sense disambiguation, among others where information about the relationship and similarity of words can be exploited. While there are large, hand-crafted ontologies available for several languages, such as WordNet for English [Fellbaum, 1998] and GoiTaikei for Japanese [Ikehara *et al.*, 1997], these resources are difficult to construct and maintain entirely by hand. They are, however, of proven utility in many NLP tasks, such as PP-attachment, where results using WordNet approach human accuracy (88.1% vs 88.2%), while the best methods using automatically constructed hierarchies still lag behind (at 84.6%) [Pantel and Lin, 2000]. Therefore, there is still a need to improve methods of both fully-automated and supervised construction of ontologies.

There is a great deal of work on the creation of ontologies from machine readable dictionaries (a good summary is [Wilkes *et al.*, 1996]), mainly for English. Recently, there has also been interest in Japanese [Tsurumaru *et al.*, 1991; Tokunaga *et al.*, 2001; Bond *et al.*, 2004]. Most approaches use either a specialized parser or a set of regular expressions tuned to a particular dictionary, often with hundreds of rules. In this paper, we take advantage of recent advances in both deep parsing and semantic representation to combine general purpose deep and shallow parsing technologies with a simple special relation extractor.

Our basic approach is to parse dictionary definition sentences with multiple shallow and deep processors, generating semantic representations of varying specificity. The semantic representation used is robust minimal recursion semantics (RMRS: Section 2.2). We then extract ontological relations using the most informative semantic representation for each definition sentence.

In this paper we discuss the construction of an ontology for Japanese using the the Japanese Semantic Database Lexeed [Kasahara *et al.*, 2004]. The deep parser uses the Japanese Grammar JACY [Siegel and Bender, 2002] and the shallow parser is based on the morphological analyzer ChaSen.

We carried out two evaluations. The first gives an automatically obtainable measure by comparing the extracted ontological relations by verifying the existence of the relations in exisiting WordNet [Fellbaum, 1998]and GoiTaikei [Ikehara *et al.*, 1997] ontologies. The second is a small scale human evaluation of the results.

2 Resources

2.1 The Lexeed Semantic Database of Japanese

The Lexeed Semantic Database of Japanese is a machine readable dictionary that covers the most common words in Japanese [Kasahara *et al.*, 2004]. It is built based on a series of psycholinguistic experiments where words from two existing machine-readable dictionaries were presented to multiple subjects who ranked them on a familiarity scale from one to seven, with seven being the most familiar [Amano and Kondo, 1999]. Lexeed consists of all open class words with a familiarity greater than or equal to five. An example entry for the word ドライバー *doraibā* "driver" is given in Figure 1, with English glosses added. The underlined material was not in Lexeed originally, we add it in this paper. *doraibā* "driver" has a familiarity of 6.55, and three senses. Lexeed has 28,000 words divided into 46,000 senses and defined with 75,000 definition sentences.

Useful hypernym relations can also be extracted from large corpora using relatively simple patterns (e.g., [Pantel *et al.*, 2004]). However, even a large newspaper corpus does not include all the familiar words of a language, let alone those words occurring in useful patterns [Amano and Kondo, 1999]. Therefore it makes sense to extract data from machine readable dictionaries (MRDs).



Figure 1: Entry for the Word *doraiba-* "driver" from Lexeed (with English glosses)

RMRS from JACY (deep)

```
hook(h1)
    proposition_m_rel(h1,h3:)
        qeq(h3:,h17)
    _jidousha_n(h4,x5:)
    udef_rel(h6,x5:)
        RSTR(h6,h7:)
        BODY(h6,h8:)
        qeq(h7:,h4)
    _unten_s_2(h9,e11:present:)
        ARG1(h9,x10:)
        ARG2(h9,x5:)
    _hito_n(h12,x10:)
        ING(h12:,h10001:)
    udef_rel(h13,x10:)
        RSTR(h13,h14:)
        BODY(h13,h15:)
        qeq(h14:,h12)
    proposition_m_rel(h10001,h16:)
        qeq(h16:,h9)
    unknown_rel(h17,e2:present:)
        ARG2(h17,x10:)
```

RMRS from ChaSen (shallow)

```
hook(h9)

    _jidousha_n(h1,x2)
    o_rel(h3,u4)


    _unten_s(h5,e6)
    suru_rel(h7,e8)

    _hito_n(h9,x10)
```

jidōsha wo unten suru hito ''a person who drives a car (lit: car-ACC drive do person)''

Real predicates are shown in **bold** font.

Figure 2: Deep and Shallow RMRS results for *doraibā*$_2$

2.2 Parsing Resources

We used the robust minimal recursion semantics (RMRS) designed in the Deep Thought project Callmeier *et al.* [2004], with tools from the Deep Linguistic Processing with HPSG Initiative (DELPH-IN: `http://www.delph-in.net/`).

Robust Minimal Recursion Semantics

Robust Minimal Recursion Semantics is a form of flat semantics which is designed to allow deep and shallow processing to use a compatible semantic representation, while being rich enough to support generalized quantifiers [Frank, 2004]. The full representation is basically the same as minimal recursion semantics [Copestake *et al.*, 2003]: a bag of labeled elementary predicates and their arguments, a list of scoping constraints, and a handle that provides a hook into the representation. The main difference is that handles must be unique, and there is an explicit distinction between grammatical and real predicates.

The representation can be underspecified in three ways: relationships can be omitted (such as message types, quantifiers and so on); predicate-argument relations can be omitted; and predicate names can be simplified. Predicate names are defined in such a way as to be as compatible as possible among different analysis engines (e.g. lemma-pos-sense, where sense is optional and the part of speech (pos) is drawn from a small set of general types (**n**oun, **v**erb, **s**ahen (verbal noun, ...)). The predicate **`unten_s`** is less specific than **`unten_s_2`** and thus subsumes it. In order to simplify the combination of different analyses, the results are indexed to the position in the original input sentence.

Examples of deep and shallow results for the same sentence 自動車を運転する人*jidōsha wo unten suru hito* "a person who drives a car (lit: car-ACC drive do person)" are given in Figure 2 (omitting the indexing). Real predicates are prefixed by an underbar (_). The deep parse gives information about the scope, message types and argument structure, while the shallow parse gives little more than a list of real and grammatical predicates with a hook.

Deep Parser (JACY and PET)

The Japanese grammar JACY [Siegel and Bender, 2002] was run with the PET System for the high-efficiency processing of typed feature structures [Callmeier, 2002].

Shallow Parser (based on ChaSen)

The part-of-speech tagger, ChaSen [Matsumoto *et al.*, 2000] was used for shallow processing of Japanese. Predicate names were produced by transliterating the pronunciation field and mapping the part-of-speech codes to the RMRS super types. The part-of-speech codes were also used to judge whether predicates were real or grammatical. Since Japanese is a head-final language, the hook value was set to be the handle of the rightmost real predicate.

3 Ontology Construction

As outlined in Section 1, our approach to ontology construction is to process a definition sentence with shallow and deep parsers and extract ontological relations from the most informative RMRS output. Here, we describe the algorithm used to extract ontological relations from an RMRS structure:

1. let P_i be the number of real predicates in the defining sentence
   - IF $P_i = 1$ (there is a unique real predicate) return: ⟨**`synonym`**: headword, predicate⟩
2. Initialize a stack of semantic relations to be processed with the semantic relation from the defining sentence's HOOK (the highest scoping handle)
3. Pop a semantic relation from the stack and check it against special predicates that require additional processing
   - When a relation indicating coordination or conjunction is found, locate all of its arguments and push them onto the stack for processing
   - IF a special predicate is found, extract its relations and add them to the stack
   - ELSE IF the current semantic relation is a real predicate, add it to list of extracted semantic heads

   Repeat until stack is empty
4. Return the ontological relations in the list of extracted semantic heads in the form: ⟨**`relation`**: headword, semantic_head⟩

Step 1. checks for a synonym relation, shown by a defining sentence containing a genus term with no differentia. Such a sentence will have a semantic representation with only a single real predicate.

In Step 2., for more complicated defining sentences, we try and find the genus term, looking first at the predicate with the widest scope. This is given by the RMRS's HOOK. The default ontological relation for the genus term is a hypernym.

Step 3. processes each semantic relation in the stack by searching for special predicates that require additional processing in order to retrieve the semantic head. Special predicates include explicit relation names (such as *ryaku* "abbreviation") and some grammatical predicates. This step identifies and processes special predicates, adding any results to the stack of unprocessed semantic relations. If a relation is not identified as being a special predicate, and it is a non-grammatical predicate, then it is accepted as a semantic head, and it is added to the list of extracted relations. Step 3 is repeated until the stack is empty.

Special predicates also give information about type of ontological relation that has been identified. They can confirm an implicit hypernym such as with *isshu* "one type" in Japanese or identify an entirely different relation, as in the case of the relation *part*, which identifies a meronym relationship in English or *meisho* "honorific name" identifying a name relation in Japanese. Specials predicates can also extract non-ontological relations such as domain.

Step 4. returns the list of all non-grammatical predicates once all semantic heads have been processed for special relations and no new results are produced.

This processing is following in the long tradition of parsing such special relationships (also called "empty heads", "function nouns" or "relators") [Guthrie *et al.*, 1990; Wilkes *et al.*, 1996]. Our main innovation is to extract them from the semantic representations produced by a deep and shallow

Special predicate	Ontological relation
Japanese	
isshu_n_1	hypernym
hitotsu_n_2	hypernym
soushou_n_1	hyponym
ryakushou_s_1	abbreviation
ryaku_s_1	abbreviation
keishou_n_1	name:honorific
zokushou_n_1	name:slang
meishou_n_1	name
bubun_n_1	meronym
ichibu_n_1	meronym

Table 1: Special predicates and their associated ontological relations

parsing, rather than using regular expressions or specially designed parsers.

4 Results and Evaluation

We summarize the relationships acquired in Table 2. The first two lines show thesaurus type relations: hypernyms and synonyms. The remaining four lines show other relations: abbreviations, names, meronyms and domains. Hypernyms and synonyms are by far the most common relations: fewer than 10% of entries are marked with an explicit relationship.

Results are shown for Lexeed using only the RMRS produced by ChaSen, using the results for JACY, and using the deepest possible result (JACY if it exists, backing off to ChaSen).

Results for ChaSen

Relation	Noun	Sahen	Verb	Other	Total
hypernym	42,235	8,176	9,237	3,346	62,994
synonym	7,278	776	2,005	933	10,992
Total	49,513	8,952	11,242	4,279	73,986

Results for JACY

Relation	Noun	Sahen	Verb	Other	Total
hypernym	31,374	6,748	6,619	2,029	46,770
synonym	7,831	801	2,220	1,048	11,900
abbreviation	154	7			161
domain	392	28			420
other	247				247
Total	39,998	7,584	8,839	3,077	59,498

Results for Deepest

Relation	Noun	Sahen	Verb	Other	Total
hypernym	45,014	9,647	10,305	3,299	68,265
synonym	81,51	827	2,257	1,254	12,489
abbreviation	154	7			161
domain	392	28			420
other	247				247
Total	53,958	10,509	12,562	4,553	81,582

Table 2: Results of Ontology Extraction (Lexeed)

As one would expect, the word based analysis using ChaSen finds more actual relationships, but does not provide enough information to find anything beyond implicit hypernyms and synonyms. The grammar based analyses have lower coverage, but allow us to extract some of the knowledge given explicitly in the lexicon.

Although the vast majority of relations extracted are hypernym and synonym relations, we extract several other kinds of knowledge, and are thus are closer to an ontology than a simple thesaurus.

We carried out two evaluations. The first was an automatic evaluation, comparing our extracted triples ⟨**relation**: word1, word2⟩ with existing resources. The second was a small scale hand evaluation of a sample of the relations.

4.1 Verification with Hand-crafted Ontologies

Because we are interested in comparing lexical semantics across languages, we compared the extracted ontology with resources in both the same and different languages.

We verified our results by comparing the hypernym links to the manually constructed Japanese ontology **GT**. It is a hierarchy of 2,710 semantic classes, defined for over 264,312 nouns [Ikehara *et al.*, 1997]. The semantic classes are mostly defined for nouns (and verbal nouns), although there is some information for verbs and adjectives. Senses are linked to **GT** semantic classes by the following heuristic: look up the semantic classes C for both the headword (w_i) and genus term(s) (w_g). If at least one of the index word's classes is subsumed by at least one of the genus' classes, then we consider the relationship confirmed (1).

$$\exists(c_h, c_g) : \{c_h \subset c_g; c_h \in C(w_h); c_g \in C(w_g)\} \quad (1)$$

In the event of an explicit hyponym relationship indicated between the headword and the genus, the test is reversed: we look for an instance of the genus' class being subsumed by the headword's class ($c_g \subset c_h$).

To test cross-linguistically, we looked up the headwords in a translation lexicon (**ALT-J/E** [Ikehara *et al.*, 1991] and EDICT [Breen, 2004]) and then did the confirmation on the set of translations $c_i \subset C(T(w_i))$. Although looking up the translation adds noise, the additional filter of the relationship triple effectively filters it out again.

For example, for ドライバー$_3$ *doraiba-*$_3$ "driver$_3$", **GT** does not find any relationship, as it does not have the golf club semantic class label for ドライバー*doraiba-*. However, looking up T(ドライバー) gives {*driver, screwdriver*} and the extracted hypernym is クラブ$_2$ *kurabu* "club". WordNet recognizes that *driver*$_5$ is a kind of *wood*$_8$ which is a kind of *club*$_5$ (using senses and relations from WordNet 2.0 [Fellbaum, 1998]). We thus simultaneously confirm the link is good; find an appropriate translation for this sense of ドライバー(and its hypernym); and link these to the appropriate WordNet synsems.

The results of the evaluation for Lexeed are shown in Table 3. Parts of speech are classified as "Noun", "Verb", "Sahen (Verbal Noun)" and "Other". Relations verified in either **GT** or WordNet are classed as verified. In these results, we only extract relations from the first definition sentence for each headword or when there is evidence of a **synonym** relation, as other definition sentences often provide clarification of the first sentence and may not contain useful ontological

information. However, extracting relations from all definition sentences results in a net loss in confirmation of less than five percent while extracting over 5,000 additional relations. This suggests that even secondary definition sentences contain information that can be exploited in building ontologies.

Our results using JACY achieve a confirmation rate of **63.31%** for nouns only and **55.74%** overall, besting Tokunaga *et al.* [2001], who reported 61.4% for nouns only. Our method also allows us to extract multiple relations from a single sentence by processing coordinate clauses. This allowed us to extract an extra 3,300 relations. Using the deepest RMRS parse available, we confirmed **57.68%** of the noun relations and **50.49%** overall. This is comparable to our results reported in [Bond *et al.*, 2004] with one important difference: we are now extracting over 10,000 more confirmed ontological relations.

GT and WordNet both lack complete cover - over half the relations were confirmed with only one resource. This shows that the machine readable dictionary is a useful source of these relations. Cross lingual checking was surprisingly effective, resources in one language can be used to bootstrap those in another, as seen in the Euro WordNet project.

4.2 Human Evaluation

One problem with using existing ontological resources to verify new relations is that only relations which are subsumed by the ontology being used for comparison can be verified. This poses a considerable problem for researchers who wish to extract new relations: be it from domains where such resources are unavailable, or in cases where existing resources are limited in scope, such as for verbs. In this case, it makes more sense to evaluate a selection of the results retrieved by hand than to rely completely on existing ontologies for verification.

In this spirit, we conducted a hand-verification of a selection of our automatically acquired relations. 1,471 relations were selected using a stratified method over the entirety of our results (every 35th relationship, ordered by link-type and then headword). In this evaluation we only consider synonyms and any relationships extracted from the first sentence: the second and subsequent definition sentences tend to contain other non-hypernomic information. The results were then evaluated by native speakers of Japanese were given the definition word, the semantic head we retrieved, and the posited relation type and asked to evaluate if the relation was accurate. They had access to the original lexicon.

The human judges found the relations presented to them to be accurate 88.99% of the time. In the 162 relations that were judged unacceptable, it was also determined that a relation did exist in 95 cases, but it was incorrect (i.e. a `synonym` in place of a `hypernym` and so on). These errors had three sources: the most common was a lack of identified explicit relationships; the next was lack of information from the shallow parse and the last was errors in the argument structure of the deep parse. Tokunaga *et al.* [2001] report slightly higher results for extracting noun relationships only (91.8%).

5 Discussion and Future Work

We were able to successfully combine deep and shallow processing to extract ontological information from lexical resources. We showed that, by using a well defined semantic representation, the extraction can be generalized so much that it can be used on very different dictionaries from different languages. This is an improvement on the common approach to using more and more detailed regular expressions (e.g. Tokunaga *et al.* [2001]). Although this provides a quick start, the results are not generally reusable. In comparison, the ChaSen-RMRS engine is immediately useful for a variety of tasks, such as question answering and information extraction [Callmeier *et al.*, 2004]

The other innovation of our approach is the cross-lingual evaluation. As a by-product of the evaluation we enhance the existing resources (such as **GT** or WordNet) by linking them, so that information can be shared between them. Further, we hope to use the cross-lingual links to fill in gaps in the monolingual resources. Finally, we can trivially extract links from the **GT** ontology to WordNet, thus combining two useful resources and allowing us to compare them in detail.

There are several areas to address in future work as we continue to pursue ontology acquisition. First, and foremost, in order to increase the quality of the ontological relations that are extracted, we need to improve the quality of the parsers to generate RMRS. With our HPSG parsers, this can be addressed by adjusting the parse ranking model to take into account the special features of dictionary definition sentences. In addition we are currently increasing the coverage by adding in treatments of more grammatical phenomena.

Another area of great interest is the acquisition of other ontological relations. For example, if we extend our special predicates to include the relation produced by various forms of negation, we may be able to extract antonym relations.

Finally, we would also like to use the links created during evaluation which link our ontologies to hand-crafted ontologies, to furtherlink together senses of words across languages. This kind of cross-lingual ontology would be of great use in applications like machine translation.

6 Conclusion

We have demonstrated how deep and shallow processing techniques can be used together to enrich the acquisition of ontological information by constructing an ontology for Japanese. Our approach requires few rules and is thus easy to maintain and expand, and it can be easily extended to cover any language that has RMRS resources. In future research, we plan to extend our processing to retrieve more ontological relations and to investigate means of improving the accuracy of output of both deep and shallow processors.

References

[Amano and Kondo, 1999] Shigeaki Amano and Tadahisa Kondo. *Nihongo-no Goi-Tokusei (Lexical properties of Japanese)*. Sanseido, 1999.

[Bond *et al.*, 2004] Francis Bond, Eric Nichols, Sanae Fujita, and Takaaki Tanaka. Acquiring an ontology for a fundamental vocabulary. In *20th International Conference on Computational Linguistics: COLING-2004*, pages 1319–1325, Geneva, 2004.

Results for ChaSen

Relation	Noun (%)	Sahen (%)	Verb (%)	Other (%)	Total (%)
hypernym	13124 / 27779 (47.24)	2489 / 5856 (42.50)	2599 / 6903 (37.65)	397 / 2218 (17.90)	18609 / 42756 (43.52)
synonym	5684 / 7278 (78.10)	606 / 776 (78.09)	1285 / 2005 (64.09)	323 / 933 (34.62)	7898 / 10992 (71.85)
total	18808 / 35057 (53.65)	3095 / 6632 (46.67)	3884 / 8908 (43.60)	720 / 3151 (22.85)	26507 / 53748 (49.32)

Results for JACY

Relation	Noun (%)	Sahen (%)	Verb (%)	Other (%)	Total (%)
hypernym	12757 / 21634 (58.97)	2033 / 5130 (39.63)	1884 / 5254 (35.86)	376 / 1527 (24.62)	17050 / 33545 (50.83)
synonym	6099 / 7831 (77.88)	626 / 801 (78.15)	1351 / 2220 (60.86)	360 / 1048 (34.35)	8436 / 11900 (70.89)
abbreviation	61 / 149 (40.94)	3 / 7 (42.86)	–/– (–)	–/– (–)	64 / 156 (41.03)
domain	68 / 344 (19.77)	7 / 28 (25.00)	–/– (–)	–/– (–)	75 / 372 (20.16)
other	125 / 225 (55.56)	–/– (–)	–/– (–)	–/– (–)	125 / 225 (55.56)
total	19110 / 30183 (63.31)	2669 / 5966 (44.74)	3235 / 7474 (43.28)	736 / 2575 (28.58)	25750 / 46198 (55.74)

Results for Deepest

Relation	Noun (%)	Sahen (%)	Verb (%)	Other (%)	Total (%)
hypernym	15703 / 29731 (52.82)	2723 / 7141 (38.13)	2762 / 7927 (34.84)	492 / 2350 (20.94)	21680 / 47149 (45.98)
synonym	6307 / 8151 (77.38)	643 / 827 (77.75)	1371 / 2257 (60.74)	409 / 1254 (32.62)	8730 / 12489 (69.90)
abbreviation	61 / 149 (40.94)	3 / 7 (42.86)	–/– (–)	–/– (–)	64 / 156 (41.03)
domain	68 / 344 (19.77)	7 / 28 (25.00)	–/– (–)	–/– (–)	75 / 372 (20.16)
other	125 / 225 (55.56)	–/– (–)	–/– (–)	–/– (–)	125 / 225 (55.56)
total	22264 / 38600 (57.68)	3376 / 8003 (42.18)	4133 / 10184 (40.58)	901 / 3604 (25.00)	30674 / 60391 (50.79)

Table 3: Results confirmed for Lexeed (for 46,000 senses)

[Breen, 2004] J. W. Breen. JMDict: a Japanese-mulitlingual dictionary. In *Coling 2004 Workshop on Multilingual Linguistic Resources*, pages 71–78, Geneva, 2004.

[Callmeier *et al.*, 2004] Ulrich Callmeier, Andreas Eisele, Ulrich Schäfer, and Melanie Siegel. The DeepThought core architecture framework. In *Proceedings of LREC-2004*, volume IV, Lisbon, 2004.

[Callmeier, 2002] Ulrich Callmeier. Preprocessing and encoding techniques in PET. In Stephan Oepen, Dan Flickinger, Jun-ichi Tsujii, and Hans Uszkoreit, editors, *Collabarative Language Engineering*, chapter 6, pages 127–143. CSLI Publications, Stanford, 2002.

[Copestake *et al.*, 2003] Ann Copestake, Daniel P. Flickinger, Carl Pollard, and Ivan A. Sag. Minimal Recursion Semantics. An introduction. 2003.

[Fellbaum, 1998] Christine Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[Frank, 2004] Anette Frank. Constraint-based RMRS construction from shallow grammars. In *20th International Conference on Computational Linguistics: COLING-2004*, pages 1269–1272, Geneva, 2004.

[Guthrie *et al.*, 1990] Louise Guthrie, Brian M. Slator, Yorick Wilkes, and Rebecca Bruce. Is there content in empty heads? In *13th International Conference on Computational Linguistics: COLING-90*, pages 138–143, Helsinki, 1990.

[Ikehara *et al.*, 1991] Satoru Ikehara, Satoshi Shirai, Akio Yokoo, and Hiromi Nakaiwa. Toward an MT system without pre-editing – effects of new methods in **ALT-J/E**–. In *Third Machine Translation Summit: MT Summit III*, pages 101–106, Washington DC, 1991. (`http://xxx.lanl.gov/abs/cmp-lg/9510008`).

[Ikehara *et al.*, 1997] Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentaro Ogura, Yoshifumi Ooyama, and Yoshihiko Hayashi. *Goi-Taikei — A Japanese Lexicon*. Iwanami Shoten, Tokyo, 1997. 5 volumes/CDROM.

[Kasahara *et al.*, 2004] Kaname Kasahara, Hiroshi Sato, Francis Bond, Takaaki Tanaka, Sanae Fujita, Tomoko Kanasugi, and Shigeaki Amano. Construction of a Japanese semantic lexicon: Lexeed. SIG NLC-159, IPSJ, Tokyo, 2004. (in Japanese).

[Matsumoto *et al.*, 2000] Yuji Matsumoto, Kitauchi, Yamashita, Hirano, Matsuda, and Asahara. *Nihongo Keitaiso Kaiseki System: Chasen*, version 2.2.1 manual edition, 2000. url=http://chasen.aist-nara.ac.jp.

[Pantel and Lin, 2000] Patrick Pantel and Dekang Lin. An unsupervised approach to prepositional phrase attachment using contextually similar words. In *38th Annual Meeting of the Association for Computational Linguistics: ACL-2000*, pages 101–108, Hong Kong, 2000.

[Pantel *et al.*, 2004] Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. Towards terascale knowledge acquisition. In *20th International Conference on Computational Linguistics: COLING-2004*, pages 771–777, Geneva, 2004.

[Siegel and Bender, 2002] Melanie Siegel and Emily M. Bender. Efficient deep processing of Japanese. In *Procedings of the 3rd Workshop on Asian Language Resources and International Standardization at the 19th International Conference on Computational Linguistics*, Taipei, 2002.

[Tokunaga *et al.*, 2001] Takenobu Tokunaga, Yasuhiro Syotu, Hozumi Tanaka, and Kiyoaki Shirai. Integration of heterogeneous language resources: A monolingual dictionary and a thesaurus. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium, NLPRS2001*, pages 135–142, Tokyo, 2001.

[Tsurumaru *et al.*, 1991] Hiroaki Tsurumaru, Katsunori Takesita, Itami Katsuki, Toshihide Yanagawa, and Sho Yoshida. An approach to thesaurus construction from Japanese language dictionary. In *IPSJ SIGNotes Natural Language*, volume 83-16, pages 121–128, 1991. (in Japanese).

[Wilkes *et al.*, 1996] Yorick A. Wilkes, Brian M. Slator, and Louise M. Guthrie. *Electric Words*. MIT Press, 1996.

The Necessity of Syntactic Parsing for Semantic Role Labeling

Vasin Punyakanok **Dan Roth** **Wen-tau Yih**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{punyakan,danr,yih}@uiuc.edu

Abstract

We provide an experimental study of the role of syntactic parsing in semantic role labeling. Our conclusions demonstrate that syntactic parse information is clearly most relevant in the very first stage – the pruning stage. In addition, the quality of the pruning stage cannot be determined solely based on its recall and precision. Instead it depends on the characteristics of the output candidates that make downstream problems easier or harder. Motivated by this observation, we suggest an effective and simple approach of combining different semantic role labeling systems through joint inference, which significantly improves the performance.

1 Introduction

Semantic parsing of sentences is believed to be an important task toward natural language understanding, and has immediate applications in tasks such information extraction and question answering. We study *semantic role labeling (SRL)* in which for each verb in a sentence, the goal is to identify all constituents that fill a semantic role, and to determine their roles, such as Agent, Patient or Instrument, and their adjuncts, such as Locative, Temporal or Manner.

The PropBank project [Kingsbury and Palmer, 2002], which provides a large human-annotated corpus of semantic verb-argument relations, has enabled researchers to apply machine learning techniques to improve SRL systems [Gildea and Palmer, 2002; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Pradhan *et al.*, 2003; Surdeanu *et al.*, 2003; Pradhan *et al.*, 2004; Xue and Palmer, 2004]. However, most systems rely heavily on the full syntactic parse trees. Therefore, the overall performance of the system is largely determined by the quality of the automatic syntactic parsers of which state of the art [Collins, 1999; Charniak, 2001] is still far from perfect.

Alternatively *shallow* syntactic parsers (i.e., chunkers and clausers), although not providing as much information as a full syntactic parser, have been shown to be more robust in their specific task [Li and Roth, 2001]. This raises the very natural and interesting question of quantifying the necessity of the full parse information to semantic parsing and whether it is possible to use only shallow syntactic information to build an outstanding SRL system.

Although PropBank is built by adding semantic annotation to the constituents on syntactic parse trees in Penn Treebank, it is not clear how important syntactic parsing is for building an SRL system. To the best of our knowledge, this problem was first addressed by Gildea and Palmer [2002]. In their attempt of using limited syntactic information, the parser was *very shallow* – clauses were not available and only chunks were used. Moreover, the pruning stage in [Gildea and Palmer, 2002] was too strict since only chunks are considered as argument candidates, meaning that over 60% of the arguments were not treated as candidates. As a result, the overall recall in their approach was very low. As we will demonstrate later, high recall of the pruning stage is in fact essential to a quality SRL system.

Using only the shallow parse information in an SRL system has largely been ignored until the recent CoNLL-04 shared task competition [Carreras and Màrquez, 2004]. In this competition, participants were restricted to only shallow parse information for their SRL systems. As a result, it became clear that the performance of the best shallow parse based system [Hacioglu *et al.*, 2004] is only 10% in F_1 below the best system that uses full parse information [Pradhan *et al.*, 2004]. In addition, there has not been a true quantitative comparison with shallow parsing. First, the CoNLL-04 shared task used only a subset of the data for training. Furthermore, its evaluation treats the continued and referential tags differently, which makes the performance metric stricter and the results worse. Second, an SRL system is usually complicated and consists of several stages. It is still unknown how much and where precisely the syntactic information helps the most.

The goal of this study is twofold. First, we make a fair comparison between SRL systems which use full parse trees and those exclusively using shallow syntactic information. This brings forward a better analysis on the necessity of full parsing in the SRL task. Second, to relieve the dependency of the SRL system on the quality of automatic parsers, we improve semantic role labeling significantly by combining several SRL systems based on different state-of-art full parsers.

To make our conclusions applicable to general SRL systems, we adhere to a widely used two step system architecture. In the first step, the system is trained to identify argument candidates for a given verb predicate. In the second step,

the system classifies the argument candidate into their types. In addition, it is also common to use a simple procedure to prune obvious non-candidates before the first step, and to use post-processing inference to fix inconsistent predictions after the second step. We also employ these two additional steps.

In our comparison between the systems using shallow and full syntactic information, we found the most interesting result is that while each step of the system using shallow information exhibits very good performance, the overall performance is significantly inferior to the system that uses full information. This necessity of full parse information is especially noticeable at the pruning stage. In addition, we produce a state-of-the-art SRL system by combining of different SRL systems based on two (potentially noisy) automatic full parsers [Collins, 1999; Charniak, 2001].

The rest of the paper is organized as follows. Section 2 gives a brief description of the semantic role labeling task and the PropBank corpus. Section 3 introduces the general architecture of an SRL system, including the features used in different stages. The detailed experimental comparison between using full parsing and shallow parsing is provided in Section 4, where we try to explain why and where the full parse information contributes to SRL. Inspired by the result, we suggests an approach that combines different SRL systems based on joint inference in Section 5. Finally, Section 6 concludes this paper.

2 Semantic Role Labeling (SRL) Task

The goal of the semantic-role labeling task is to discover the verb-argument structure for a given input sentence. For example, given a sentence " I *left* my pearls to my daughter-in-law in my will", the goal is to identify different arguments of the verb *left* which yields the output:

$[_{A0}$ I$]$ $[_{V}$ *left* $]$ $[_{A1}$ my pearls$]$ $[_{A2}$ to my daughter-in-law$]$ $[_{AM\text{-}LOC}$ in my will$]$.

Here A0 represents the *leaver*, A1 represents the *thing left*, A2 represents the *benefactor*, AM-LOC is an adjunct indicating the location of the action, and V determines the verb. In addition, each argument can be mapped to a constituent in its corresponding syntactic full parse tree.

Following the definition of the PropBank and CoNLL-2004 shared task, there are six different types of arguments labeled as A0-A5 and AA. These labels have different semantics for each verb as specified in the PropBank Frame files. In addition, there are also 13 types of adjuncts labeled as AM-*adj* where *adj* specifies the adjunct type. In some cases, an argument may span over different parts of a sentence, the label C-*arg* is used to specify the continuity of the arguments, as shown in the example below.

$[_{A1}$ The pearls$]$, $[_{A0}$ I$]$ $[_{V}$ *said*$]$, $[_{C\text{-}A1}$ were left to my daughter-in-law$]$.

Moreover in some cases, an argument might be a relative pronoun that in fact refers to the actual agent outside the clause. In this case, the actual agent is labeled as the appropriate argument type, *arg*, while the relative pronoun is instead labeled as R-*arg*. For example,

$[_{A1}$ The pearls$]$ $[_{R\text{-}A1}$ which$]$ $[_{A0}$ I$]$ $[_{V}$ *left*$]$, $[_{A2}$ to my daughter-in-law$]$ are fake.

The distribution of these argument labels is fairly unbalanced. In the official release of PropBank I, core arguments (A0–A5 and AA) occupy 71.26%, where the largest parts are A0 (25.39%) and A1 (35.19%). The rest portion is mostly the adjunct arguments (24.90%). The continued (C-*arg*) and referential (R-*arg*) arguments are relatively fewer, occupying 1.22% and 2.63% respectively. For more definitions of PropBank and the semantic role labeling task, readers can refer to [Kingsbury and Palmer, 2002] and [Carreras and Màrquez, 2004].

3 SRL System Architecture

Our SRL system consists of four stages: *pruning*, *argument identification*, *argument classification*, and *inference*. In particular, the goal of pruning and argument identification is to identify argument candidates for a given verb predicate. The system only classifies the argument candidate into their types in the stage of argument classification. Linguistic and structural constraints are incorporated in the inference stage to resolve inconsistent global predictions. This section describes how we build these four stages, including the features used in training the classifiers.

3.1 Pruning

When the full parse tree of a sentence is available, only the constituents in the parse tree are considered as argument candidates. In addition, our system exploits the heuristic rules introduced by Xue and Palmer [2004] to filter out simple constituents that are very unlikely to be arguments. The heuristic is a recursive process starting from the verb of which arguments to be identified. It first returns the siblings of the verb as candidates; then it moves to the parent of the verb, and collects the siblings again. The process goes on until it reaches the root. In addition, if a constituent is a PP (propositional phrase), its children are also collected.

3.2 Argument Identification

The argument identification stage utilizes binary classification to identify whether a candidate is an argument or not. When full parsing is available, we train and apply the binary classifiers on the constituents supplied by the pruning stage. When only shallow parsing is available, the system does not have the pruning stage, and also does not have constituents to begin with. Therefore, conceptually the system has to consider all possible subsequences (i.e., consecutive words) in a sentence as potential argument candidates. We avoid this by using a learning scheme by first training two classifiers, one to predict the beginnings of possible arguments, and the other the ends. The predictions are combined to form argument candidates that do not violate the following constraints.

1. Arguments cannot cover the predicate.
2. Arguments cannot overlap with the clauses (they can be embedded in one another).
3. If a predicate is outside a clause, its arguments cannot be embedded in that clause.

The features used in the full parsing and shallow parsing settings are described as follows.

Features when full parsing is available
Most of the features used in our system are standard features which include

- **Predicate and POS tag of predicate** features indicate the lemma of the predicate verb and its POS tag.
- **Voice** feature indicates passive/active voice of the predicate.
- **Phrase type** feature provides the phrase type of the constituent.
- **Head word and POS tag of the head word** feature provides the head word and its POS tag of the constituent. We use rules introduced by Collins [1999] to extract this feature.
- **Position** feature describes if the constituent is before or after the predicate relative to the position in the sentence.
- **Path** records the traversal path in the parse tree from the predicate to the constituent.
- **Subcategorization** feature describes the phrase structure around the predicate's parent. It records the immediate structure in the parse tree that expands to its parent.

We also use the following additional features.

- **Verb class** feature is the class of the active predicate described in PropBank Frames.
- **Lengths** of the target constituent, in the numbers of words and chunks separately.
- **Chunk** tells if the target argument is, embeds, overlaps, or is embedded in a chunk with its type.
- **Chunk pattern** encodes the sequence of chunks from the current words to the predicate.
- **Chunk pattern length** feature counts the number of chunks in the argument.
- **Clause relative position** feature is the position of the target word relative to the predicate in the pseudo-parse tree constructed only from clause constituent. There are four configurations—target constituent and predicate share the same parent, target constituent parent is an ancestor of predicate, predicate parent is an ancestor of target word, or otherwise.
- **Clause coverage** describes how much of the local clause (from the predicate) is covered by the target argument.
- **NEG** feature is active if the target verb chunk has `not` or `n't`.
- **MOD** feature is active when there is a modal verb in the verb chunk. The rules of the **NEG** and **MOD** features are used in a baseline SRL system developed by Erik Tjong Kim Sang [Carreras and Màrquez, 2004].

Features when only shallow parsing is available
Features used are similar to those used by the system with full parsing except those that need full parse trees to generate. For these types of features, we either try to mimic the features with some heuristics rules or discard them. The details of these features are as follows.

- **Phrase type** uses a simple heuristics to identify only VP, PP, and NP.
- **Head word and POS tag of the head word** are the rightmost word for NP, and leftmost word for VP and PP.
- **Shallow-Path** records the traversal path in the pseudo-parse tree constructed only from the clause structure and chunks.
- **Shallow-Subcategorization** feature describes the chunk and clause structure around the predicate's parent in the pseudo-parse tree.
- **Syntactic frame** features are discarded.

3.3 Argument Classification

This stage assigns the final argument labels to the argument candidates supplied from the previous stage. A multi-class classifier is trained to classify the types of the arguments supplied by the argument identification stage. In addition, to reduce the excessive candidates mistakenly output by the previous stage, the classifier can also classify the argument as *NULL* (meaning "not an argument") to discard the argument.

The features used here are the same as those used in the argument identification stage. However, when full parsing are available, an additional feature introduced by Xue and Palmer [2004] is used.

- **Syntactic frame** describes the sequential pattern of the noun phrases and the predicate in the sentence.

3.4 Inference

The purpose of this stage is to incorporate some prior linguistic and structural knowledge, such as "arguments do not overlap" or "each verb takes at most one argument of each type." This knowledge is used to resolve any inconsistencies of argument classification in order to generate final legitimate predictions. We use the inference process introduced by Punyakanok *et al.* [2004]. The process is formulated as an integer linear programming (ILP) problem that takes as inputs the confidences over each type of the arguments supplied by the argument classifier. The output is the optimal solution that maximizes the linear sum of the confidence scores (e.g., the conditional probabilities estimated by the argument classifier), subject to the constraints that encode the domain knowledge.

4 The Necessity of Syntactic Parsing

We study the necessity of syntactic parsing experimentally by observing the effects of using full parsing and shallow parsing at each stage of an SRL system. In Section 4.1, we first describe how we prepare the data, as well as the basic system including features and the learning algorithm. The comparison of full parsing and shallow parsing on the three stages (excluding the inference stage) is presented in the reversed order (Sections 4.2, 4.3, 4.4).

4.1 Experimental Setting

We use PropBank sections 02 through 21 as training data, and section 23 as testing. In order to apply the standard CoNLL-

04 evaluation script, our system conforms to both the input and output format defined in the shared task.

The CoNLL-04 evaluation metric is slightly more restricted since an argument prediction is only considered correct when all its *continued* arguments (C-*arg*) are correct and *referential* arguments (R-*arg*) are included – these requirements are often absent in previous SRL systems, given that they only occupy a very small percentage of the data. To provide a fair comparison, we also report the performance when discarding continued and referential arguments. Following the notation used in [Xue and Palmer, 2004], this evaluation metric is referred as "argM+", which considers all the core arguments and adjunct arguments. We note here that all the performance reported excludes V label which usually improves the overall performance if included.

The goal of the experiments in this section is to understand the effective contribution of full parsing versus shallow parsing using only the part-of-speech tags, chunks, and clauses. In addition, we also compare performance when using the correct (gold standard) versus using automatic parse data. The automatic full parse trees are derived using Charniak's parser [Charniak, 2001] (version 0.4). In automatic shallow parsing, the information is generated by a state-of-the-art POS tagger [Even-Zohar and Roth, 2001], chunker [Punyakanok and Roth, 2001], and clauser [Carreras and Màrquez, 2003].

The learning algorithm used is a variation of the Winnow update rule incorporated in SNoW [Roth, 1998; Roth and Yih, 2002], a multi-class classifier that is tailored for large scale learning tasks. SNoW learns a sparse network of linear functions, in which the targets (argument border predictions or argument type predictions, in this case) are represented as linear functions over a common feature space. It improves the basic Winnow multiplicative update rule in several ways. For example, a regularization term is added, which has the effect of trying to separate the data with a large margin separator [Grove and Roth, 2001; Hang *et al.*, 2002] and voted (averaged) weight vector is used [Freund and Schapire, 1999].

Experimental evidences have shown that SNoW activations correlate with the confidence of the prediction and can provide an estimate of probability to be used for both argument identification and inference. We use the softmax function [Bishop, 1995] to convert raw activation to conditional probabilities. Specifically, if there are n classes and the raw activation of class i is act_i, the posterior estimation for class i is

$$\text{score}(i) = p_i = \frac{e^{act_i}}{\sum_{1 \le j \le n} e^{act_j}}.$$

4.2 Argument Classification

To evaluate the performance gap between full parsing and shallow parsing in argument classification, we assume the argument boundaries are known, and only train classifiers to classify the labels of these arguments. In this stage, the only difference between *full parsing* and *shallow parsing* is the construction of three full parsing features: *path*, *subcategorization* and *syntactic frame*. As described in Section 3, *path* and *subcategorization* can be approximated by *shallow-path* and *shallow-subcategorization* using chunks and clauses. However, it is unclear how to mimic the syntactic frame feature since it relies on the internal structure of a full parse tree. Therefore, it does not have a corresponding feature in the shallow parsing case.

Table 1 reports the experimental results of argument classification when argument boundaries are known. Although full parsing features seem to help when using the gold standard data, the difference in F_1 is only 0.32% and 0.13% for the CoNLL-2004 and ArgM+ evaluation respectively. When the automatic (full and shallow) parsers are used, the gap is smaller.

	Full Parsing	Shallow Parsing
Gold	91.32	91.00
Auto	90.93	90.69
Gold (ArgM+)	90.67	91.54
Auto (ArgM+)	90.87	90.93

Table 1: The accuracy of argument classification when argument boundaries are known

Lesson When the argument boundaries are known, the performance of the full paring systems is about the same as the shallow parsing system.

4.3 Argument Identification

Argument identification is an important stage that effectively reduces the number of argument candidates after pruning. Given an argument candidate, an argument identifier is a binary classifier that decides whether or not the candidate should be considered as an argument. To evaluate the influence of full parsing in this stage, the candidate list used here is the pruning results on the gold standard parse trees.

Similar to the argument classification stage, the only difference between full-parse and shallow-parse is the use of *path* and *subcategorization* features. Again, we replace them with *shallow-path* and *shallow-subcategorization* when the binary classifier is trained using the shallow parsing information.

Table 2 reports the performance of the argument identifier on the test set using the direct predictions of the trained binary classifier. The recall and precision of the full parsing system are around 2 to 3 percents higher than the shallow parsing system on the gold standard data. As a result, the F_1 score is 2.5% higher. The performance on automatic parse data is unsurprisingly lower, but the difference between full parsing and shallow parsing is relatively the same. In terms of filtering efficiency, around 25% of the examples are predicted as positive. In other words, both argument identifiers filter out around 75% of the argument candidates after pruning.

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	96.53	93.57	95.03	93.66	91.72	92.68
Auto	94.68	90.60	92.59	92.31	88.36	90.29

Table 2: The performance of argument identification after pruning (based on the gold standard full parse trees)

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	92.13	95.62	93.84	88.54	94.81	91.57
Auto	89.48	94.14	91.75	86.14	93.21	89.54

Table 3: The performance of argument identification after pruning (based on the gold standard full parse trees) and with threshold=0.1

Since the recall in argument identification sets the upper bound of the recall in argument classification, in practice, the threshold that predicts examples as positive is usually lowered to allow more positive predictions. That is, a candidate is predicted as positive when its probability estimation is larger than the threshold. Table 3 shows the performance of the argument identifiers when the threshold is 0.1.

Since argument identification is just an intermediate step of a complete system, a more realistic evaluation method is to see how each final system performs. Table 4 and Table 5 report the final results in recall, precision, and F_1 in CoNLL and ArgM+ metrics. The F_1 difference is about 4.5% when using the gold standard data. However, when automatic parsers are used, shallow-parse is in fact slightly better. This may be due to the fact that shallow parsers are more accurate in chunk or clause predictions compared to a regular full parser [Li and Roth, 2001].

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	88.81	89.35	89.08	84.19	85.03	84.61
Auto	84.21	85.04	84.63	86.17	84.02	85.08

Table 4: The CoNLL-04 evaluation of the overall system performance when pruning (using the gold standard full parse trees) is available

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	89.02	89.57	89.29	84.46	85.31	84.88
Auto	84.38	85.38	84.87	86.37	84.36	85.35

Table 5: ArgM+ performance of the overall system when pruning (using the gold standard full parse trees) is available

Lesson Full parsing helps in argument identification. However, when the automatic shallow parser is more accurate than the full parser, using the full parsing information may not have advantages over shallow parsing.

4.4 Pruning

As shown in the previous two subsections, the performance difference of full parsing and shallow parsing is not large when the pruning information is given. We conclude that the main contribution of the full parse is in the pruning stage. Since the shallow parsing system does not have enough information for the pruning heuristics, we train two word based classifiers to replace the pruning stage. One classifier is trained to predict whether a given word is the start (S) of an argument; the other classifier is to predict the end (E) of an argument. If the product of probabilities of a pair of S and E predictions is larger than a predefined threshold, then this pair is considered as an argument candidate. The pruning comparison of using the classifiers and heuristics is shown in Table 6.

	Full Parsing			Classifier th=0.04		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	25.94	97.27	40.96	29.58	97.18	45.35
Auto	22.79	86.08	36.04	24.68	94.80	39.17

Table 6: The performance of pruning

Amazingly, the classifier pruning strategy seems better than the heuristics. With about the same recall, the classifiers achieve higher precision. However, to really compare systems using full parsing and shallow parsing, we still need to see the overall performance. We build two semantic role systems based on full parsing and shallow parsing. The full parsing system follows the pruning, argument identification, argument classification, and inference stages, as described earlier. For the shallow parsing system, pruning is replaced by the word-based pruning classifiers, and the rest stages are designed only to use shallow parsing information as described in previous sections. Table 7 and Table 8 show the overall performance in the two evaluation methods.

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	88.81	89.35	89.08	75.34	75.28	75.31
Auto	77.09	75.51	76.29	75.48	67.13	71.06

Table 7: The CoNLL-04 evaluation of the overall system performance

	Full Parsing			Shallow Parsing		
	Prec	Rec	F_1	Prec	Rec	F_1
Gold	89.02	89.57	89.29	75.35	75.20	75.27
Auto	77.09	75.57	76.32	75.54	67.14	71.09

Table 8: ArgM+ performance of the overall system

As indicated in the tables, the gap in F_1 between the full parsing and shallow parsing systems enlarges to more than 13% on the gold standard data. At first glance, this result seems to contradict our conclusion in Section 4.3. After all, if the pruning stage of the shallow parsing SRL system performs equally well or even better, the overall performance gap in F_1 should be small.

After we carefully examine the output of the word-based classifier pruning, we realize that it in fact filters out "easy" candidates, and leaves examples that are difficult to the later stages. To be specific, these argument candidates often overlap and differ only with one or two words. On the other hand, the pruning heuristics based on full parsing never outputs overlapping candidates. The following argument identification stage can be thought of as good in discriminating different types of candidates.

Lesson The most crucial contribution of full parsing is in pruning. The internal tree structure helps significantly in discriminating argument candidates, which makes the work easy to the following stages.

5 Combine Different SRL Systems

The empirical study in Section 4 indicates the performance of an SRL system primarily depends on the very first stage – pruning, which is derived directly from the full parse trees. This also means that in practice the quality of the syntactic parser is decisive to the quality of the SRL system. To improve semantic role labeling, one possible way is to combine different SRL systems through a joint inference stage, given that the systems are derived using different full parse trees.

To test this idea, we first build two SRL systems that use Collins' parser[1] and Charniak's parser respectively. In fact, these two parsers have noticeably different output. Applying punning heuristics on the output of Collins' parser produces a list of candidates with 81.05% recall. Although this number is significantly lower that 86.08% recall produced by Charniak's parser, the union of the two candidate lists still significantly improves recall to 91.37%. We construct the two systems by implementing the first three stages, namely pruning, argument identification, and argument classification. When a testing sentence is given, a joint inference stage is used to resolve the inconsistency of the output of argument classification in these two systems.

We first briefly describe the inference procedure introduced by Punyakanok *et al.* [2004]. Formally speaking, the argument classifier attempts to assign labels to a set of arguments, $S^{1:M}$, indexed from 1 to M. Each argument S^i can take any label from a set of argument labels, $\mathcal{P}$, and the indexed set of arguments can take a set of labels, $c^{1:M} \in \mathcal{P}^M$. If we assume that the argument classifier returns an estimated conditional probability distribution, $Prob(S^i = c^i)$, then, given a sentence, the inference procedure seeks an global assignment that maximizes the following objective function,

$$\hat{c}^{1:M} = \underset{c^{1:M} \in \mathcal{P}^M}{\operatorname{argmax}} \sum_{i=1}^{M} Prob(S^i = c^i), \qquad (1)$$

subject to linguistic and structural constraints. In other words, this objective function reflects the expected number of correct argument predictions, subject to the constraints.

When there are two or more argument classifiers from different SRL systems, a joint inference procedure can take the output estimated probabilities for these candidates as input, although some candidates may refer to the same phrases in the sentence. For example, Figure 1 shows the two candidate sets for a fragment of a sentence, "*..., traders say, unable to* ***cool*** *the selling panic in both stocks and futures.*" In this example, system A has two argument candidates, a_1 = "traders" and a_4 = "the selling panic in both stocks and futures"; system B has three argument candidates, b_1 = "traders", b_2 = "the selling panic", and b_3 = "in both stocks and futures".

[1] We use the Collins' parser implemented by Bikel [2004].

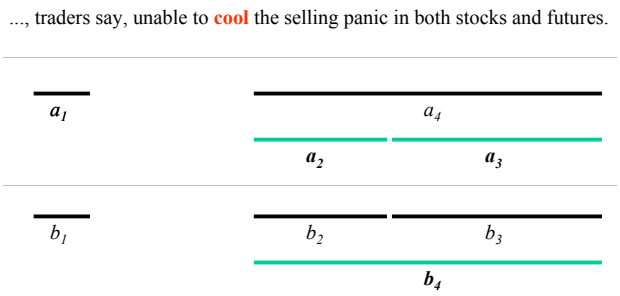


Figure 1: The output of two SRL systems: system A has two candidates, a_1 = "traders" and a_4 = "the selling panic in both stocks and futures"; system B has three argument candidates, b_1 = "traders", b_2 = "the selling panic", and b_3 = "in both stocks and futures". In addition, we create two phantom candidates a_2 and a_3 for system A that correspond to b_2 and b_3 respectively, and b_4 for system B that corresponds to a_4.

If we throw all these variables together into the inference procedure, then the final prediction will be more likely dominated by the system that has more candidates, which is system B in this example. The reason is because our objective function is the sum of the probabilities of all the candidate assignments.

This bias can be corrected by the following observation. Although system A only has two candidates, a_1 and a_4, it can be treated as it also has two additional *phantom* candidates, a_2 and a_3, where a_2 and b_2 refer to the same phrase, and so do a_3 and b_3. Similarly, system B has a phantom candidate b_4 that corresponds to a_4. Because system A does not really generate a_2 and a_3, we can assume that these two phantom candidates are predicted as *NULL* (i.e., not an argument). We assign the same prior distribution to each phantom candidate. In particular, the probability of the *null* class is set to be 0.55 based on empirical tests, and the probabilities of the rest classes are set based on their occurrence frequencies in the training data.

Tables 9 and 10 report the performance of individual systems, as well as the joint system. The joint system based on this straightforward strategy significantly improves the performance compared to the two original SRL systems in both recall and precision, and thus achieves a much higher F_1.

6 Conclusions

In this paper, we make a fair comparison between the SRL systems using full parse trees and using only shallow syntactic information. What we found confirms the necessity of full parsing for the SRL problem. In particular, this information is the most crucial in the pruning stage of the system, and relatively less important to the following stages. Inspired by this observation, we proposed an effective and simple approach that combines different SRL systems through a joint inference stage. The combined system significantly improves the performance compared to the original systems.

	Prec	Rec	F_1
Collins' Parse	75.92	71.45	73.62
Charniak's Parse	77.09	75.51	76.29
Combined Result	80.53	76.94	78.69

Table 9: The performance in CoNLL-04's evaluation of individual and combined SRL systems

	Prec	Rec	F_1
Collins' Parse	75.87	71.36	73.54
Charniak's Parse	77.09	75.57	76.32
Combined Result	80.56	76.99	78.73

Table 10: The performance in argM+'s evaluation of individual and combined SRL systems

Acknowledgments

We thank Dav Zimak, Kevin Small, and the anonymous referees for their useful comments. We are also grateful to Dash Optimization for the free academic use of Xpress-MP. This research is supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program, a DOI grant under the Reflex program, and an ONR MURI Award.

References

[Bikel, 2004] Dan Bikel. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4), December 2004.

[Bishop, 1995] C. Bishop. *Neural Networks for Pattern Recognition*, chapter 6.4: Modelling conditional distributions, page 215. Oxford University Press, 1995.

[Carreras and Màrquez, 2003] X. Carreras and L. Màrquez. Phrase recognition by filtering and ranking with perceptrons. In *Proc. of RANLP-2003*, 2003.

[Carreras and Màrquez, 2004] X. Carreras and L. Màrquez. Introduction to the conll-2004 shared tasks: Semantic role labeling. In *Proc. of CoNLL-2004*, 2004.

[Charniak, 2001] E. Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics*, Toulouse, France, 2001.

[Chen and Rambow, 2003] J. Chen and O. Rambow. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proc. of EMNLP-2003*, Sapporo, Japan, 2003.

[Collins, 1999] M. Collins. *Head-driven Statistical Models for Natural Language Parsing.* PhD thesis, Computer Science Department, University of Pennsylvenia, Philadelphia, 1999.

[Even-Zohar and Roth, 2001] Y. Even-Zohar and D. Roth. A sequential model for multi class classification. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 10–19, 2001.

[Freund and Schapire, 1999] Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[Gildea and Hockenmaier, 2003] D. Gildea and J. Hockenmaier. Identifying semantic roles using combinatory categorial grammar. In *Proc. of the EMNLP-2003*, Sapporo, Japan, 2003.

[Gildea and Palmer, 2002] D. Gildea and M. Palmer. The necessity of parsing for predicate argument recognition. In *Proc. of ACL 2002*, Philadelphia, PA, 2002.

[Grove and Roth, 2001] A. Grove and D. Roth. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141, 2001.

[Hacioglu *et al.*, 2004] K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. Semantic role labeling by tagging syntactic chunks. In *Proc. of CoNLL-04*, 2004.

[Hang *et al.*, 2002] T. Hang, F. Damerau, and D. Johnson. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637, 2002.

[Kingsbury and Palmer, 2002] P. Kingsbury and M. Palmer. From Treebank to PropBank. In *Proc. of LREC-2002*, Spain, 2002.

[Li and Roth, 2001] X. Li and D. Roth. Exploring evidence for shallow parsing. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 107–110, 2001.

[Pradhan *et al.*, 2003] S. Pradhan, K. Hacioglu, W. Ward, J. Martin, and D. Jurafsky. Semantic role parsing adding semantic structure to unstructured text. In *Proc. of ICDM-2003*, 2003.

[Pradhan *et al.*, 2004] S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Proc. of NAACL-HLT 2004*, 2004.

[Punyakanok and Roth, 2001] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS-13*, pages 995–1001, 2001.

[Punyakanok *et al.*, 2004] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proc. of COLING-2004*, 2004.

[Roth and Yih, 2002] D. Roth and W. Yih. Probabilistic reasoning for entity & relation recognition. In *Proc. of COLING-2002*, pages 835–841, 2002.

[Roth, 1998] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of AAAI*, pages 806–813, 1998.

[Surdeanu *et al.*, 2003] M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *Proc. of ACL 2003*, 2003.

[Xue and Palmer, 2004] N. Xue and M. Palmer. Calibrating features for semantic role labeling. In *Proc. of the EMNLP-2004*, pages 88–94, Barcelona, Spain, 2004.

Learning and Inference over Constrained Output

Vasin Punyakanok **Dan Roth** **Wen-tau Yih** **Dav Zimak**
Department of Computer Science
University of Illinois at Urbana-Champaign
{punyakan, danr, yih, davzimak}@uiuc.edu

Abstract

We study learning structured output in a discriminative framework where values of the output variables are estimated by local classifiers. In this framework, complex dependencies among the output variables are captured by constraints and dictate which global labels can be inferred. We compare two strategies, *learning independent classifiers* and *inference based training*, by observing their behaviors in different conditions. Experiments and theoretical justification lead to the conclusion that using inference based learning is superior when the local classifiers are difficult to learn but may require many examples before any discernible difference can be observed.

1 Introduction

Making decisions in real world problems involves assigning values to sets of variables where a complex and expressive structure can influence, or even dictate, what assignments are possible. For example, in the task of identifying named entities in a sentence, prediction is governed by constraints like "entities do not overlap." Another example exists in scene interpretation tasks where predictions must respect constraints that could arise from the nature of the data or task, such as "humans have two arms, two legs, and one head."

There exist at least three fundamentally different solutions to learning classifiers over structured output. In the first, structure is ignored; local classifiers are learned and used to predict each output component separately. In the second, learning is decoupled from the task of maintaining structured output. Estimators are used to produce global output consistent with the structural constraints only after they are learned for each output variable separately. Discriminative HMM, conditional models [Punyakanok and Roth, 2001; McCallum *et al.*, 2000] and many dynamic programming based schemes used in the context of sequential predictions fall into the this category. The third class of solutions incorporates dependencies among the variables into the learning process to directly induce estimators that optimize a global performance measure. Traditionally these solutions were generative; however recent developments have produced discriminative models of this type, including conditional random fields [Lafferty *et al.*, 2001], Perceptron-based learning of structured output [Collins, 2002; Carreras and Màrquez, 2003] and Max-Margin Markov networks which allow incorporating Markovian assumptions among output variables [Taskar *et al.*, 2004].

Incorporating constraints during training can lead to solutions that directly optimize the true objective function, and hence, should perform better. Nonetheless, most real world applications using this technique do not show significant advantages, if any. Therefore, it is important to discover the tradeoffs of using each of the above schemes.

In this paper, we compare three learning schemes. In the first, classifiers are learned independently (*learning only* (LO)), in the second, inference is used to maintain structural consistency only after learning (*learning plus inference* (L+I)), and finally inference is used while learning the parameters of the classifier (*inference based training* (IBT)). In semantic role labeling (SRL), it was observed [Punyakanok *et al.*, 2004; Carreras and Màrquez, 2003] that when the local classification problems are easy to learn, L+I outperforms IBT. However, when using a reduced feature space where the problem was no longer (locally) separable, IBT could overcome the poor local classifications to yield accurate global classifications.

Section 2 provides the formal definition of our problem. For example, in Section 3, we compare the three learning schemes using the online Perceptron algorithm applied in the three settings (see [Collins, 2002] for details). All three settings use the same linear representation, and L+I and IBT share the same decision function space. Our conjectures of the relative performance between different schemes are presented in Section 4. Despite the fact that IBT is a more powerful technique, in Section 5, we provide an experiment that shows how L+I can outperform IBT when there exist accurate local classifiers that do not depend on structure, or when there are too few examples to learn complex structural dependencies. This is also theoretically justified in Section 6.

2 Background

Structured output classification problems have many flavors. In this paper, we focus on problems where it is natural both to split the task into many smaller classification tasks and to solve directly as a single task. In Section 5.2, we consider the

semantic role-labeling problem, where the input $\mathcal{X}$ are natural language features and the output $\mathcal{Y}$ is the position and type of a semantic-role in the sentence. For this problem, one can either learn a set of local functions such as "is this phrase an argument of 'run'," or a global classifier to predict all semantic-roles at once. In addition, natural structural constraints dictate, for example, that no two semantic roles for a single verb can overlap. Other structural constraints, as well as linguistic constraints yield a restricted output space in which the classifiers operate.

In general, given an assignment $\mathbf{x} \in \mathcal{X}^{n_x}$ to a collection of input variables, $\mathbf{X} = (X_1, \ldots, X_{n_x})$, the structured classification problem involves identifying the "best" assignment $\mathbf{y} \in \mathcal{Y}^{n_y}$ to a collection of output variables $\mathbf{Y} = (Y_1, \ldots, Y_{n_y})$ that are consistent with a defined structure on $\mathbf{Y}$. This structure can be thought of as constraining the output space to a smaller space $\mathcal{C}(\mathcal{Y}^{n_y}) \subseteq \mathcal{Y}^{n_y}$, where $\mathcal{C} : 2^{\mathcal{Y}^*} \rightarrow 2^{\mathcal{Y}^*}$ constrains the output space to be structurally consistent.

In this paper, a structured output classifier is a function $h : \mathcal{X}^{n_x} \rightarrow \mathcal{Y}^{n_y}$, that uses a global scoring function, $f : \mathcal{X}^{n_x} \times \mathcal{Y}^{n_y} \rightarrow \mathbb{R}$ to assign scores to each possible example/label pair. Given input $\mathbf{x}$, it is hoped that the correct output $\mathbf{y}$ achieves the highest score among consistent outputs:

$$\hat{\mathbf{y}} = h(\mathbf{x}) = \underset{\mathbf{y}' \in \mathcal{C}(\mathcal{Y}^{n_y})}{\operatorname{argmax}} f(\mathbf{x}, \mathbf{y}'), \qquad (1)$$

where n_x and n_y depend on the example at hand. In addition, we view the global scoring function as a composition of a set of local scoring functions $\{f_y(\mathbf{x}, t)\}_{y \in \mathcal{Y}}$, where $f_y : \mathcal{X}^{n_x} \times \{1, \ldots, n_y\} \rightarrow \mathbb{R}$. Each function represents the *score* or confidence that output variable Y_t takes value y:

$$f(\mathbf{x}, (y_1, \ldots, y_{n_y})) = \sum_{t=1}^{n_y} f_{y_t}(\mathbf{x}, t)$$

Inference is the task of determining an optimal assignment $\mathbf{y}$ given an assignment $\mathbf{x}$. For sequential structure of constraints, polynomial-time algorithms such as Viterbi or CSCL [Punyakanok and Roth, 2001] are typically used for efficient inference. For general structure of constraints, a generic search method (e.g., beam search) may be applied. Recently, integer programming has also been shown to be an effective inference approach in several NLP applications [Roth and Yih, 2004; Punyakanok *et al.*, 2004].

In this paper, we consider classifiers with *linear representation*. Linear local classifiers are linear functions, $f_y(\mathbf{x}, t) = \boldsymbol{\alpha}^y \cdot \Phi^y(\mathbf{x}, t)$, where $\boldsymbol{\alpha}^y \in \mathbb{R}^{d_y}$ is a weight vector and $\Phi^y(\mathbf{x}, t) \in \mathbb{R}^{d_y}$ is a feature vector. Then, it is easy to show that the global scoring function can be written in the familiar form $f(\mathbf{x}, \mathbf{y}) = \boldsymbol{\alpha} \cdot \Phi(\mathbf{x}, \mathbf{y})$, where $\Phi^y(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n_y} \Phi^{y_t}(\mathbf{x}, t) I_{\{y_t = y\}}$ is an accumulation over all output variables of features occurring for class y, $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^{|\mathcal{Y}|})$ is concatenation of the $\boldsymbol{\alpha}^y$'s, and $\Phi(\mathbf{x}, \mathbf{y}) = (\Phi^1(\mathbf{x}, \mathbf{y}), \ldots, \Phi^{|\mathcal{Y}|}(\mathbf{x}, \mathbf{y}))$ is the concatenation of the $\Phi^y(\mathbf{x}, \mathbf{y})$'s. Then, the global classifier is

$$h(\mathbf{x}) = \hat{\mathbf{y}} = \underset{\mathbf{y}' \in \mathcal{C}(\mathcal{Y}^{n_y})}{\operatorname{argmax}} \boldsymbol{\alpha} \cdot \Phi(\mathbf{x}, \mathbf{y}').$$

3 Learning

We present several ways to learn the scoring function parameters differing in whether or not the structure-based inference process is leveraged during training. Learning consists of choosing a function $h : \mathcal{X}^* \rightarrow \mathcal{Y}^*$ from some hypothesis space, $\mathcal{H}$. Typically, the data is supplied as a set $\mathbf{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$ from a distribution $\mathrm{P}_{\mathcal{X},\mathcal{Y}}$ over $\mathcal{X}^* \times \mathcal{Y}^*$. While these concepts are very general, we focus on online learning of linear representations using a variant of the Perceptron algorithm (see [Collins, 2002]).

Learning Local Classifiers: Learning stand-alone local classifiers is perhaps the most straightforward setting. No knowledge of the inference procedure is used. Rather, for each example $(\mathbf{x}, \mathbf{y}) \in \mathbf{D}$, the learning algorithm must ensure that $f_{y_t}(\mathbf{x}, t) > f_{y'}(\mathbf{x}, t)$ for all $t = 1, \ldots, n_y$ and all $y' \neq y_t$. In Figure 3(a), an online Perceptron-style algorithm is presented where no global constraints are used. See [Har-Peled *et al.*, 2003] for details and Section 5 for experiments.

Learning Global Classifiers: We seek to train classifiers so they will produce the correct *global* classification. To this end, the key difference from learning locally is that feedback from the inference process determines which classifiers to modify so that together, the classifiers and the inference procedure yield the desired result. As in [Collins, 2002; Carreras and Màrquez, 2003], we train according to a global criterion. The algorithm presented here is an online procedure, where at each step a subset of the classifiers are updated according to inference feedback. See Figure 3(b) for details of a Perceptron-like algorithm for learning with inference feedback.

Note that in practice it is common for problems to be modeled in such a way that local classifiers are dependent on part of the output as part of their input. This sort of *interaction* can be incorporated directly to the algorithm for learning a global classifier as long as an appropriate inference process is used. In addition, to provide a fair comparison between LO, L+I, and IBP in this setting one must take care to ensure that the learning algorithms are appropriate for this task. In order to remain focused on the problem of training with and without inference feedback, the experiments and analysis presented concern only the local classifiers without interaction.

4 Conjectures

In this section, we investigate the relative performance of classifier systems learned with and without inference feedback. There are many competing factors. Initially, if the local classification problems are "easy", then it is likely that learning local classifiers only (LO) can yield the most accurate classifiers. However, an accurate model of the structural constraints could additionally increase performance (learning plus inference (L+I)). As the local problems become more difficult to learn, an accurate model of the structure becomes more important, and can, perhaps, overcome sub-optimal local classifiers. Despite the existence of a global solution, as the local classification problems become increasingly difficult, it is unlikely that structure based inference can fix poor classifiers learned locally. In this case, only training with inference feedback (IBT) can be expected to perform well.

Algorithm ONLINELOCALLEARNING
INPUT: $\mathbf{D}^{X,Y} \in \{\mathcal{X}^* \times \mathcal{Y}^*\}^m$
OUTPUT: $\{f_y\}_{y\in\mathcal{Y}} \in \mathcal{H}$

Initialize $\boldsymbol{\alpha}^y \in \mathbb{R}^{|\Phi^y|}$ for $y \in \mathcal{Y}$
Repeat until converge
for each $(\mathbf{x}, \mathbf{y}) \in \mathbf{D}^{X,Y}$ **do**
for $t = 1, \ldots, n_y$ **do**
$\hat{y}_t = \text{argmax}_y\, \boldsymbol{\alpha}^y \cdot \Phi^y(\mathbf{x}, t)$
if $\hat{y}_t \neq y_t$ **then**
$\boldsymbol{\alpha}^{y_t} = \boldsymbol{\alpha}^{y_t} + \Phi^{y_t}(\mathbf{x}, t)$
$\boldsymbol{\alpha}^{\hat{y}_t} = \boldsymbol{\alpha}^{\hat{y}_t} - \Phi^{\hat{y}_t}(\mathbf{x}, t)$

(a) Without inference feedback

Algorithm ONLINEGLOBALLEARNING
INPUT: $\mathbf{D}^{X,Y} \in \{\mathcal{X}^* \times \mathcal{Y}^*\}^m$
OUTPUT: $\{f_y\}_{y\in\mathcal{Y}} \in \mathcal{H}$

Initialize $\boldsymbol{\alpha} \in \mathbb{R}^{|\Phi|}$
Repeat until converge
for each $(\mathbf{x}, \mathbf{y}) \in \mathbf{D}^{X,Y}$ **do**
$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}\in\mathcal{C}(\mathcal{Y}^{n_y})}\, \boldsymbol{\alpha} \cdot \Phi(\mathbf{x}, \mathbf{y})$
if $\hat{\mathbf{y}} \neq \mathbf{y}$ **then**
$\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \hat{\mathbf{y}})$

(b) With inference feedback

Figure 1: Algorithms for learning without and with inference feedback. The key difference lies in the inference step (i.e. argmax). Inference while learning locally is trivial and the prediction is made simply by considering each label locally. Learning globally uses a global inference (i.e. $\text{argmax}_{\mathbf{y}\in\mathcal{C}(\mathcal{Y}^{n_y})}$) to predict global labels.

As a first attempt to formalize the difficulty of classification tasks, we define separability and learnability. A classifier, $f \in \mathcal{H}$, *globally separates* a data set $\mathbf{D}$ iff for all examples $(\mathbf{x}, \mathbf{y}) \in \mathbf{D}$, $f(\mathbf{x}, \mathbf{y}) > f(\mathbf{x}, \mathbf{y}')$ for all $\mathbf{y}' \in \mathcal{Y}^{n_y} \setminus \mathbf{y}$ and *locally separates* $\mathbf{D}$ iff for all examples $(\mathbf{x}, \mathbf{y}) \in \mathbf{D}$, $f_{y_t}(\mathbf{x}, t) > f_y(\mathbf{x}, t)$ for all $y \in \mathcal{Y} \setminus y_t$, and all $\mathbf{y}' \in \mathcal{Y}^{n_y} \setminus \mathbf{y}$. A learning algorithm $\mathcal{A}$ is a function from data sets to a $\mathcal{H}$. We say that $\mathbf{D}$ is *globally (locally) learnable* by $\mathcal{A}$ if there exists an $f \in \mathcal{H}$ such that f *globally (locally) separates* $\mathbf{D}$.

The following simple relationships exist between local and global learning: 1. local separability implies global separability, but the inverse is not true; 2. local separability implies local and global learnability; 3. global separability implies global learnability, but not local learnability. As a result, it is clear that if there exist learning algorithms to learn global separations, then given enough examples, IBT will outperform L+I. However, learning examples are often limited either because they are expensive to label or because some learning algorithms simply do not scale well to many examples. With a fixed number of examples, L+I can outperform IBT.

Claim 4.1 *With a fixed number of examples:*

1. *If the local classification tasks are separable, then L+I outperforms IBT.*
2. *If the task is globally separable, but not locally separable then IBT outperforms L+I only with sufficient examples. This number correlates with the degree of the separability of the local classifiers.*

5 Experiments

We present experiments to show how the relative performance of learning plus inference (L+I) compares to inference based training (IBT) when the quality of the local classifiers and amount of training data varies.

5.1 Synthetic Data

In our experiment, each example $\mathbf{x}$ is a set of c points in d-dimensional real space, where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_c) \in \mathbb{R}^d \times \ldots \times \mathbb{R}^d$ and its label is a sequence of binary variable, $\mathbf{y} = (y_1, \ldots, y_c) \in \{0, 1\}^c$, labeled according to:

$$\mathbf{y} = h(\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{C}(\mathcal{Y}^c)}{\text{argmax}} \sum_i y_i f_i(\mathbf{x}_i) - (1 - y_i) f_i(\mathbf{x}_i),$$

where $\mathcal{C}(\mathcal{Y}^c)$ is a subset of $\{0, 1\}^c$ imposing a random constraint[1] on $\mathbf{y}$, and $f_i(\mathbf{x}_i) = \mathbf{w}_i\mathbf{x}_i + \theta_i$. Each f_i corresponds to a local classifier $y_i = g_i(\mathbf{x}_i) = I_{f_i(\mathbf{x}_i)>0}$. Clearly, the dataset generated from this hypothesis is globally linearly separable. To vary the difficulty of local classification, we generate examples with various degree of linear separability of the local classifiers by controlling the fraction κ of the data where $h(\mathbf{x}) \neq g(\mathbf{x}) = (g_1(\mathbf{x}_1), \ldots, g_c(\mathbf{x}_c))$—examples whose labels, if generated by local classifiers independently, violate the constraints (i.e. $g(\mathbf{x}) \notin \mathcal{C}(\mathcal{Y}^c)$).

Figure 2 compares the performance of different learning strategies relative to the number of training examples used. In all experiments, $c = 5$, the true hypothesis is picked at random, and $\mathcal{C}(\mathcal{Y}^c)$ is a random subset with half of the size of $\mathcal{Y}^c$. Training is halted when a cycle complete with no errors, or 100 cycles is reached. The performance is averaged over 10 trials. Figure 2(a) shows the locally linearly separable case where L+I outperforms IBT. Figure 2(c) shows results for the case with the most difficult local classification tasks($\kappa = 1$) where IBT outperforms L+I. Figure 2(b) shows the case where data is not totally locally linearly separable($\kappa = 0.1$). In this case, L+I outperforms IBT when the number of training examples is small. In all cases, inference helps.

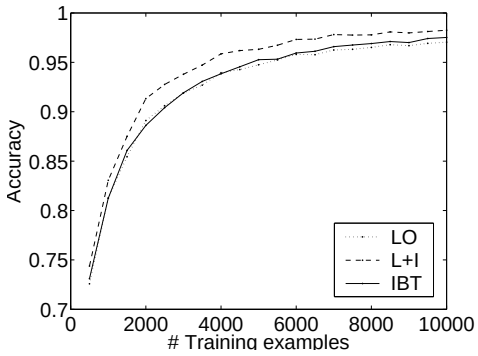
5.2 Real-World Data

In this section, we present experiments on two real-world problems from natural language processing – semantic role labeling and noun phrase identification.

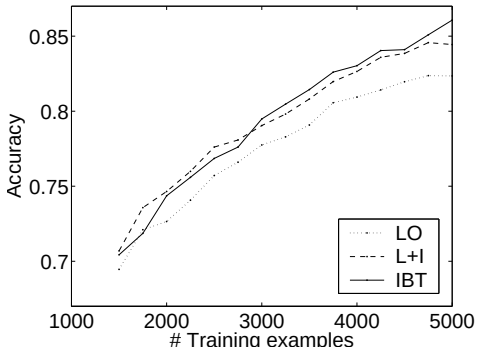
Semantic-Role Labeling

Semantic role labeling (SRL) is believed to be an important task toward natural language understanding, and has immediate applications in tasks such Information Extraction and

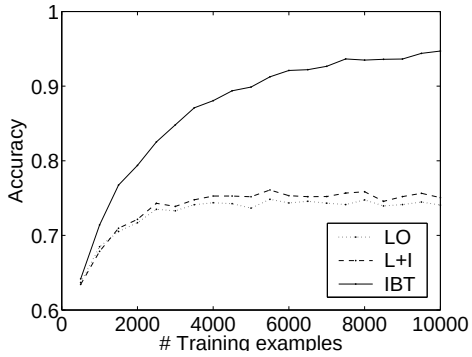
[1] Among the total 2^c possible output labels, $\mathcal{C}(\cdot)$ fixes a random fraction as legitimate global labels.



(a) $\kappa = 0, d = 100$



(b) $\kappa = 0.15, d = 300$



(c) $\kappa = 1, d = 100$

Figure 2: Comparison of different learning strategies in various degrees of difficulties of the local classifiers. $\kappa = 0$ implies locally linearly separability. Higher κ indicates harder local classification.

Question Answering. The goal is to identify, for each verb in the sentence, all the constituents which fill a semantic role, and determine their argument types, such as *Agent*, *Patient*, *Instrument*, as well as adjuncts such as *Locative*, *Temporal*, *Manner*, etc. For example, given a sentence " I *left* my pearls to my daughter-in-law in my will", the goal is to identify different arguments of the verb *left* which yields the output:

$[_{\text{A0}}$ I$]$ $[_{\text{V}}$ *left* $]$ $[_{\text{A1}}$ my pearls$]$ $[_{\text{A2}}$ to my daughter-in-law$]$ $[_{\text{AM-LOC}}$ in my will$]$.

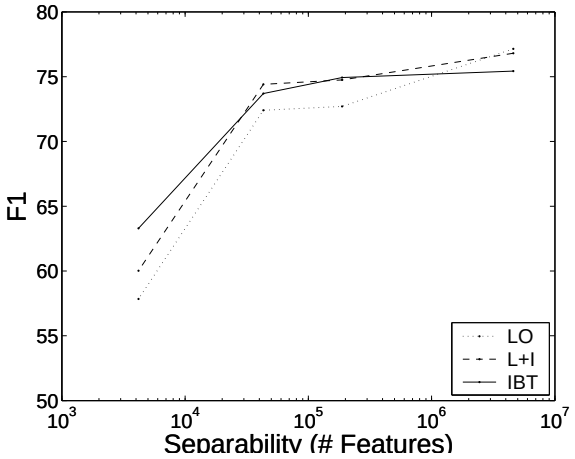


Figure 3: Results on the semantic-role labeling (SRL) problem. As the number of features increases, the difficulty of the local classification problem becomes easier, and the independently learned classifiers (LO) perform well, especially when inference is used after learning (L+I). Using inference during training (IBT) can aid performance when the learning problem is more difficult (few features).

Here A0 represents *leaver*, A1 represents *thing left*, A2 represents *benefactor*, AM-LOC is an adjunct indicating the location of the action, and V determines the verb.

We model the problem using classifiers that map constituent candidates to one of 45 different types, such as f_{AO} and f_{A1} [Kingsbury and Palmer, 2002; Carreras and Màrquez, 2003]. However, local multiclass decisions are insufficient. Structural constraints are necessary to ensure, for example, that no arguments can overlap or embed each other. In order to include both structural and linguistic constraints, we use a general inference procedure based on integer linear programming [Punyakanok *et al.*, 2004]. We use data provided in the CoNLL-2004 shared task [Carreras and Màrquez, 2003], but we restrict our focus to sentences that have greater than five arguments. In addition, to simplify the problem, we assume the boundaries of the constituents are given – the task is mainly to assign the argument types.

The experiments clearly show that IBT outperforms locally learned LO and L+I when the local classifiers are inseparable and difficult to learn. The difficulty of local learning was controlled by varying the number of input features. With more features, the linear classifier are more expressive and can learn effectively and L+I outperforms IBT. With less features the problem becomes more difficult and IBT outperforms L+I. See Figure 3.

Noun Phrase Labeling

Noun phrase identification involves the identification of phrases or of words that participate in a syntactic relationship. Specifically, we use the standard base Noun Phrases (NP) data set [Ramshaw and Marcus, 1995] taken from the Wall Street Journal corpus in the Penn Treebank [Marcus *et al.*, 1993].

The phrase identifier consists of two classifiers: one that

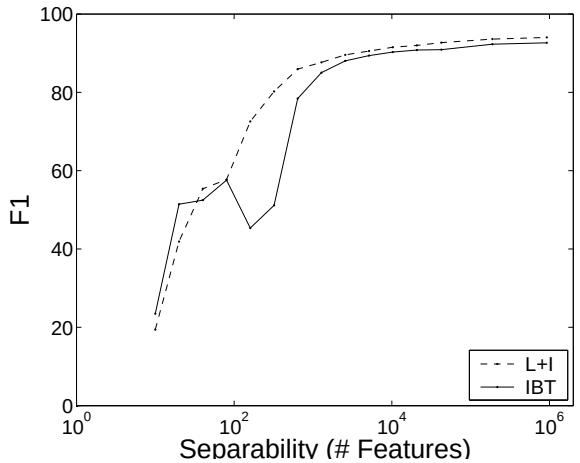


Figure 4: Results on the noun phrase (NP) identification problem.

detects the beginning, $f_[$, and a second that detects the end, $f_]$ of a phrase. The outcome of these classifiers are then combined in a way that satisfies structural constraints constraints (e.g. non-overlapping), using an efficient constraint satisfaction mechanism that makes use of the confidence in the classifiers' outcomes [Punyakanok and Roth, 2001].

In this case, L+I trains each classifier independently, and only during evaluation, the inference is used. On the other hand, IBT incorporates the inference into the training. For each sentence, each word position is processed by the classifiers, and their outcomes are used by the inference process to infer the final prediction. The classifiers are then updated based on the final prediction not on their own prediction before the inference.

As in the previous experiment, Figure 4 shows performance of two systems varied by the number of features. Unlike the previous experiment, the number of features in each experiment was determined by the frequency of occurrence. Less frequent features are pruned to make the task more difficult. The results are similar to the SRL task in that only when the problem becomes difficult IBT outperforms L+I.

6 Bound Prediction

In this section, we use standard VC-style generalization bounds from learning theory to gain intuition into when learning locally (LO and L+I) may outperform learning globally (IBT) by comparing the expressivity and complexity of each hypothesis space. When learning globally, it is possible to learn concepts that may be difficult to learn locally, since the global constraints are not available to the local algorithms. On the other hand, while the global hypothesis space is more expressive, it has a substantially larger representation. here we develop two bounds—both for linear classifiers on a restricted problem. The first upper bounds the generalization error for learning locally by assuming various degrees of separability. The second provides an improved generalization bound for globally learned classifiers by assuming separability in the more expressive global hypothesis space.

We begin by defining the *growth function* to measure the effective size of the hypothesis space.

Definition 6.1 (Growth Function) *For a given hypothesis class $\mathcal{H}$ consisting of functions $h : \mathcal{X} \to \mathcal{Y}$, the* growth function, *$\mathcal{N}_{\mathcal{H}}(m)$, counts the maximum number of ways to label any data set of size m:*

$$\mathcal{N}_{\mathcal{H}}(m) = \sup_{\mathbf{x}_1,\ldots,\mathbf{x}_m \in \mathcal{X}^m} |\{(h(\mathbf{x}_1),\ldots,h(\mathbf{x}_m))\,|h \in \mathcal{H}\}|$$

The well-known VC-style generalization bound expresses expected error, ϵ, of the *best* hypothesis, h_{opt} on unseen data. In the following theorem adapted from [Anthony and Bartlett, 1999][Theorem 4.2], we directly write the growth function into the bound,

Theorem 6.2 *Suppose that $\mathcal{H}$ is a set of functions from a set $\mathcal{X}$ to a set $\mathcal{Y}$ with growth function $\mathcal{N}_{\mathcal{H}}(m)$. Let $h_{\text{opt}} \in \mathcal{H}$ be the hypothesis that minimizes sample error on a sample of size m drawn from an unknown, but fixed probability distribution. Then, with probability $1-\delta$*

$$\epsilon \leq \epsilon_{\text{opt}} + \sqrt{\frac{32(\log(\mathcal{N}_{\mathcal{H}}(2m)) + \log(4/\delta))}{m}}. \qquad (2)$$

For simplicity, we first describe the setting in which a separate function is learned for each of a fixed number, c, of output variables (as in Section 5.1). Here, each example has c components in input $\mathbf{x} = (\mathbf{x}_1,\ldots,\mathbf{x}_c) \in \mathbb{R}^d \times \ldots \times \mathbb{R}^d$ and output $\mathbf{y} = (y_1,\ldots,y_c) \in \{0,1\}^c$.

Given a dataset $\mathbf{D}$, the aim is to learn a set of linear scoring functions $f_i(\mathbf{x}_i) = \mathbf{w}_i\mathbf{x}_i$, where $\mathbf{w}_i \in \mathbb{R}^d$ for each $i = 1,\ldots,c$. For LO and L+I, the setting is simple: find a set of weight vectors that, for each component, satisfy $y_i\mathbf{w}_i\mathbf{x}_i > 0$ for all examples $(\mathbf{x},\mathbf{y}) \in \mathbf{D}$. For IBT, we find a set of classifiers such that $\sum_i y_i\mathbf{w}_i\mathbf{x}_i > \sum_i y'_i\mathbf{w}_i\mathbf{x}_i$ for all $y' \neq y$ (and that satisfy the constraints, $y' \in \mathcal{C}(\mathcal{Y}^c)$).

As previously noted, when learning local classifiers independently (LO and L+I), one can only guarantee convergence when each local problem is separable – however, it is often the case that global constraints render these problems inseparable. Therefore, there is a lower bound, ϵ_{opt}, on the optimal error achievable. Since each component is a separate learning problem, the generalization error is thus

Corollary 6.3 *When $\mathcal{H}$ is the set of separating hyperplanes in $\mathbb{R}^d$,*

$$\epsilon \leq \epsilon_{\text{opt}} + \sqrt{\frac{32(d\log((em/d)) + \log(4/\delta))}{m}}. \qquad (3)$$

Proof sketch: We show that $\mathcal{N}_{\mathcal{H}}(m) \leq (em/d)^d$ when $\mathcal{H}$ is the class of threshold linear functions in d dimensions. $\mathcal{N}_{\mathcal{H}}(m)$ is precisely the maximum number of continuous regions an arrangement of m halfspaces in $\mathbb{R}^d$, which is $2\sum_{i=1}^{d}\binom{m-1}{i} \leq 2(e(m-1)/d)^d$. For $m > 1$, the result holds. See [Anthony and Bartlett, 1999][Theorem 3.1] for details. ■

On the other hand, when learning collectively with IBT, examples consist of the full vector $\mathbf{x} \in \mathbb{R}^{cd}$. In this setting, convergence is guaranteed (if, of course, such a function

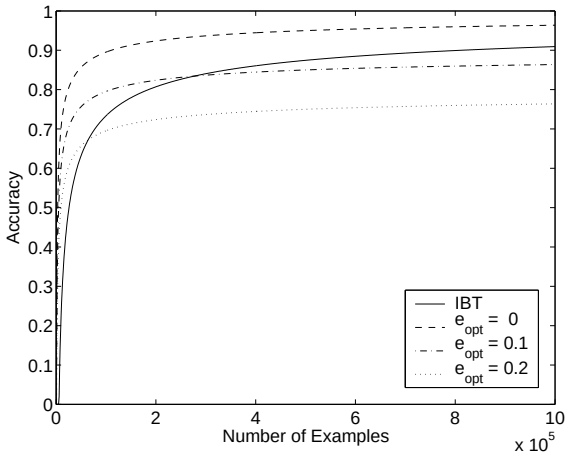


Figure 5: The VC-style generalization bounds predict that IBT will eventually outperform LO if the local classifiers are unable to find consistent classification ($\epsilon_{\text{opt}} > 0.0$, accuracy < 1). However, if the local classifiers are learnable ($\epsilon_{\text{opt}} = 0.0$, accuracy $= 1$), LO will perform well.

exists). Thus, the optimal error when training with IBT is $\epsilon_{\text{opt}} = 0$. However, the output of the global classification is now the entire output vector. Therefore, the growth function must account for exponentially many outputs.

Corollary 6.4 *When $\mathcal{H}$ is the set of decision functions over $\{0,1\}^c$, defined by $\text{argmax}_{\mathbf{y}' \in \mathcal{C}(\{0,1\}^c)} \sum_{i=1}^{c} y_i \mathbf{w}_i \mathbf{x}_i$, where $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_c) \in \mathbb{R}^{cd}$,*

$$\epsilon \leq \sqrt{\frac{32(cd\log(em/cd) + c^2 d + \log(4/\delta))}{m}}. \qquad (4)$$

Proof sketch: In this setting, we must count the effective hypothesis space – which is the effective number of different classifiers in weight space, $\mathbb{R}^{cd}$. As before, this is done by constructing an arrangement of halfspaces in the weight space. Specifically, each halfspace is defined by a single $((\mathbf{x}, \mathbf{y}), \mathbf{y}')$ pair that defines the region where $\sum_i y_i \mathbf{w}_i \mathbf{x}_i > \sum_i y'_i \mathbf{w}_i \mathbf{x}_i$. Because there are potentially $\mathcal{C}(2^c) \leq 2^c$ output labels and the weight space is cd-dimensional, the growth function is the size of the arrangement of $c2^c$ halfspaces in $\mathbb{R}^{cd}$. Therefore $\mathcal{N}_{\mathcal{H}}(m) \leq (em2^c/cd)^{cd}$. ■

Figure 5 shows a comparison between these two bounds, where the generalization bound curves on accuracy are shown for IBT (Corollary 6.4) and for LO and L+I (Corollary 6.3) with $\epsilon_{\text{opt}} \in \{0.0, 0.1, 0.2\}$. One can see that when separable, the accuracy=1 curve ($\epsilon_{\text{opt}} = 0.0$) in the figure outperforms IBT. However, when the problems are locally inseparable, IBT will eventually converge, whereas LO and L+I will not – these results match the synthetic experiment results in Figure 2. Notice the relationship between κ and ϵ_{opt}. When $\kappa = 0$, both the local and global problems are separable and $\epsilon_{\text{opt}} = 0$ As κ increases, the global problem remains separable and the local problems are inseparable ($\epsilon_{\text{opt}} > 0$).

7 Conclusion

We studied the tradeoffs between three common learning schemes for structured outputs, i.e. learning without the knowledge about structure (LO), using inference only after learning (L+I), and learning with inference feedback (IBT). We provided experiments on both real-world and synthetic data as well as a theoretical justification that support our main clams. – first, when the local classification is linearly separable, L+I outperforms IBT, and second, as the local problems become more difficult and are no longer linearly separable, IBT outperforms L+I, but only with sufficient number of training examples. In the future, we will seek a similar comparison for the more general setting where nontrivial interaction between local classifiers is allowed, and thus, local separability does not imply global separability.

8 Acknowledgments

We grateful Dash Optimization for the free academic use of Xpress-MP. This research is supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program, a DOI grant under the Reflex program, NSF grants ITR-IIS-0085836, ITR-IIS-0085980 and IIS-9984168, and an ONR MURI Award.

References

[Anthony and Bartlett, 1999] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[Carreras and Màrquez, 2003] X. Carreras and Lluís Màrquez. Online learning via global feedback for phrase recognition. In *Advances in Neural Information Processing Systems 15*, 2003.

[Collins, 2002] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, 2002.

[Har-Peled *et al.*, 2003] S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: A new approach to multiclass classification and ranking. In *Advances in Neural Information Processing Systems 15*, 2003.

[Kingsbury and Palmer, 2002] P. Kingsbury and M. Palmer. From Treebank to PropBank. In *Proceedings of LREC*, 2002.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML-01*, pages 282–289, 2001.

[Marcus *et al.*, 1993] M. P. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June 1993.

[McCallum *et al.*, 2000] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML-00*, Stanford, CA, 2000.

[Punyakanok and Roth, 2001] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *Advances in Neural Information Processing Systems 13*, 2001.

[Punyakanok *et al.*, 2004] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of COLING-04*, 2004.

[Ramshaw and Marcus, 1995] L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third Annual Workshop on Very Large Corpora*, 1995.

[Roth and Yih, 2004] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL-2004*, pages 1–8, 2004.

[Taskar *et al.*, 2004] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 16*, 2004.

Beyond TFIDF Weighting for Text Categorization in the Vector Space Model

Pascal Soucy
Coveo
Quebec, Canada
psoucy@coveo.com

Guy W. Mineau
Université Laval
Québec, Canada
guy.mineau@ift.ulaval.ca

Abstract

KNN and SVM are two machine learning approaches to Text Categorization (TC) based on the Vector Space Model. In this model, borrowed from Information Retrieval, documents are represented as a vector where each component is associated with a particular word from the vocabulary. Traditionally, each component value is assigned using the information retrieval TFIDF measure. While this weighting method seems very appropriate for IR, it is not clear that it is the best choice for TC problems. Actually, this weighting method does not leverage the information implicitly contained in the categorization task to represent documents. In this paper, we introduce a new weighting method based on statistical estimation of the importance of a word for a specific categorization problem. This method also has the benefit to make feature selection implicit, since useless features for the categorization problem considered get a very small weight. Extensive experiments reported in the paper shows that this new weighting method improves significantly the classification accuracy as measured on many categorization tasks.

1 Introduction

KNN and SVM are two machine learning approaches to Text Categorization (TC) based on the vector space model [Salton *et al*., 1975], a model borrowed from Information Retrieval (IR). Both approaches are known to be among the most accurate text categorizers [Joachims, 1998a; Yang and Liu, 1999]. In the vector space model, documents are represented as a vector where each component is associated with a particular word in the text collection vocabulary.

Generally, each vector component is assigned a value related to the estimated importance (some weight) of the word in the document. Traditionally, this weight was assigned using the TFIDF measure [Joachims, 1998a; Yang and Liu, 1999; Brank *et al*., 2002; Dumais *et al.*, 1998]. While this weighting method seems very appropriate for IR, it is not clear that it is the best choice for TC problems. Actually, this weighting method does not leverage the information implicitly contained in the categorization task to represent documents.

To illustrate this, let us suppose a text collection X and two categorization tasks *A* and *B*. Under the TFIDF weighting representation, each document in *X* is represented by the same vector for both *A* and *B*. Thus, the importance of a word in a document is seen as independent from the categorization task. However, we believe that this should not be the case in many situations. Suppose that *A* is the task that consists of classifying *X* in two categories: documents that pertain to *Computers* and documents that don't. Intuitively, words such as *computer*, *intel* and *keyboard* would be very relevant to this task, but not words such as *the* and *of*; for this reason, the former words should have a higher weight than the latter ones. Suppose, now that *B* consists of classifying *X* in two very different categories: documents written in English and documents written in other languages. It is arguable that in this particular task, words such has *the* (English stop word) and *les* (French stop word), are very relevant. However, under TFIDF, *the* would get a very small weight since its *IDF* (Inverse Document Frequency) would be low. In fact, it would get the same weight that was assigned for task *A*. While this example is somewhat an extreme case, we believe that a weighting approach could benefit from the knowledge about the categorization task at hand.

In this paper, we introduce a new weighting method based on statistical estimation of a word importance for a particular categorization problem. This weighting also has the benefit that it makes feature selection implicit since useless features for the categorization problem considered get a very small weight.

Section 2 presents both the TFIDF weighting function and the new weighting method introduced in this paper. Section 3 describes our evaluation test bed. In section 4, we report results that show significant improvements in terms of classification accuracy.

2 Weighting approaches in text categorization

2.1 TFIDF weighting

TFIDF is the most common weighting method used to describe documents in the Vector Space Model, particularly in IR problems. Regarding text categorization, this weighting function has been particularly related to two important machine learning methods: KNN and SVM. The TFIDF function weights each vector component (each of them relating to a word of the vocabulary) of each document on the following basis. First, it incorporates the word frequency in the document. Thus, the more a word appears in a document (e.g., its TF, term frequency is high) the more it is estimated to be significant in this document. In addition, IDF measures how infrequent a word is in the collection. This value is estimated using the whole training text collection at hand. Accordingly, if a word is very frequent in the text collection, it is not considered to be particularly representative of this document (since it occurs in most documents; for instance, stop words). In contrast, if the word is infrequent in the text collection, it is believed to be very relevant for the document. TFIDF is commonly used in IR to compare a query vector with a document vector using a similarity or distance function such as the cosine similarity function. There are many variants of TFIDF. The following common variant was used in our experiments, as found in [Yang and Liu, 1999].[1]

$$weight_{t,d} = \begin{cases} \log(tf_{t,d}+1)\log\dfrac{n}{x_t} & \text{if } tf_{t,d} \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $tf_{t,d}$ is the frequency of word t in document d, n is the number of documents in the text collection and x_t is the number of documents where word t occurs. Normalization to unit length is generally applied to the resulting vectors (unnecessary with KNN and the cosine similarity function).

2.2 Supervised weighting

[Debole and Sebastiani, 2003] have tested and compared some supervised weighting approaches that leverages on the training data. These approaches are variants of TFIDF weighting where the *idf* part is modified using common functions used to conduct feature selection. In this paper, their best finding is a variant of the Information Gain, the Gain Ratio. Respective to a category c_i, the Gain Ratio of the term t_k is:

$$GR(t_k, c_i) = \frac{IG(t_k, c_i)}{-\sum_{c \in \{c_i, \overline{c_i}\}} P(c)\log_2 P(c)} \quad (2)$$

[1] In [Joachims, 1998a], a slight variant is used where the *tf* is used without the logarithm function, but [Yang and Liu, 1999] reports no significant difference in classification accuracy whether the log is applied or not).

Another approach is presented in [Han 1999]. In this study, vector components are weighted using an iterative approach involving the classifier at each step. For each iteration, the weights are slightly modified and the categorization accuracy is measured using an evaluation set (a split from the training set). Convergence of weights should provide an optimal set of weights. While appealing (and probably a near optimal solution if the training data is the only information available to the classifier), this method is generally much too slow to be used, particularly for broad problems (involving a large vocabulary).

2.3 A Weighting Methods based on Confidence

The weighting method (named ConfWeight in the rest of the text) introduced in this paper is based on statistical confidence intervals. Let x_t be the number of documents containing the word t in a text collection and n, the size of this text collection. We estimate the proportion of documents containing this term to be:

$$\tilde{p} = \frac{x_t + 0.5z^2_{\alpha/2}}{n + z^2_{\alpha/2}} \quad (3)$$

Where $\tilde{p}$ is the Wilson proportion estimate [Wilson, 1927] and $z^2_{\alpha/2}$ is a value such that $\Phi(z_{\alpha/2}) = \alpha/2$, Φ is the t-distribution (Student's law) function when $n < 30$ and the normal distribution one when $n \geq 30$. So when $n \geq 30$[2], $\tilde{p}$ is:

$$\tilde{p} = \frac{x_t + 1.96}{n + 3.84} \quad (4)$$

Thus, its confidence interval at 95% is:

$$\tilde{p} \pm 1.96\sqrt{\frac{\tilde{p}(1-\tilde{p})}{n+3.84}} \quad (5)$$

Most categorization tasks can be formulated in a way to use only binary classifiers (e.g. a classifier that decides whether a document belongs to a specific category or not). Thus, for a task with n categories, there will be n binary classifiers.

For a given category, let us name $\tilde{p}_+$ the equation (4) applied to the positive documents (those who are labeled as being related to the category) in the training set and $\tilde{p}_-$ to those in the negative class. Now, we use the label *MinPos* for the lower range of the confidence interval of $\tilde{p}_+$, and the label *MaxNeg* for the higher range of that of $\tilde{p}_-$ according to (5) measured on their respective training set. Let now *MinPosRelFreq* be:

$$MinPosRelFreq = \frac{MinPos}{MinPos + MaxNeg} \quad (6)$$

[2] When $n < 30$ (which occurs for categories with few positive instances), the t-distribution was used instead of the normal law; thus, equations should be modified accordingly.

We now define the *strength* of term *t* for category +:

$$str_{t,+} = \begin{cases} \log_2(2 \cdot MinPosRelFreq) & \text{if } MinPos > MaxNeg \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Therefore, weight ≠ 0 iff the word appears proportionally more frequently in the + category than in the – category, even in the worst (measured by the confidence interval) estimation error scenario. There might be many categories where weight ≠ 0, since the categorization task is divided in *n* binary classifiers. We name the maximum strength of *t*:

$$\text{maxstr}(t) = \left(\max_{c \in Categories} (str_{t,c}) \right)^2 \quad (8)$$

Maxstr(t) is a global policy technique [Debole and Sebastiani, 2003], that is, the value is that of the best classifier and is thereafter used for all *n* binary classifiers. Using a global policy allows us to use the same document representation for all *n* binary classifiers. While local policies seem intuitively more appealing than global policies when the categorization task is divided in *n* binary problems, [Debole and Sebastiani, 2003] shown that global policies are at least as good as local policies. Note that a value of 0 for maxstr(*t*) is akin to a feature selection deciding to reject the feature.

Figure 1 presents an example to highlight the behavior of eq. (6) to (8). In this figure, *MinPos* is set to 0.5, which means that a hypothetic term occurs at least (recall that this value is the lower range of its relative document frequency confidence interval) in half the documents from the positive set. Then, the curves (labeled (6), (7) and (8) in the graph) consists of the resulting weights for different values of *MaxNeg*. Eq. (6) gives more weight to terms that occur more frequently (relative to the number of documents) in the positive category than in the negative one. Therefore, this weighting method favors features that are **proportionally** more frequent in the positive class. This weight decreases as *MaxNeg* increases. Eq. (7) scales the weight values linearly into the [0,1] range, so that the resulting weight is 0 when a term occurs at the same relative frequency in both classes or proportionally more frequently in the negative set. Finally, Eq. (8) makes the decrease faster, to reflect the rate at which features lose their "energy" as they are more evenly distributed among the positives and the negatives. As a consequence, very predictive features get a high weight, regardless of their absolute frequency (only proportion differences matter).

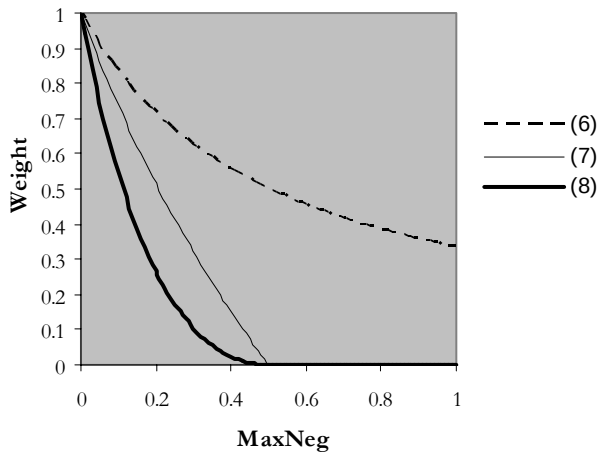


Figure 1: Weight when varying MaxNeg with a fixed MinPos = 0.5

As we are interested in weighting all training and testing documents components in the vector space model, we must use (8) with individual documents, taking the document term frequency into account. We define the `ConfWeight` of *t* in document *d* as:

$$ConfWeight_{t,d} = \log(tf_{t,d} + 1)\text{maxstr}(t) \quad (9)$$

Eq. (9) is quite similar to the `TFIDF` equation in (1): the first part weights the term according to its importance for the document while the second part weights the term globally. However, unlike `TFIDF`, `ConfWeight` uses the categorization problem to determine the weight of a particular term.

3 Methodology

3.1 Corpora

In this paper, three data sets previously studied in the literature have been selected. These datasets are: Reuters-21578, Ohsumed and the new Reuters Corpus Vol 1. Let us briefly describe these datasets.

Reuters-21578 [Lewis, 1997] is made of categories related to business news report. It is written using a limited vocabulary in a succinct manner. We used the ModApte [Lewis, 1997] split. There are 90 categories having at least one training and one testing document. These categories are highly unbalanced. Each document may be categorized in more than one category.

Ohsumed comes from a very large text collection (the MedLine Bibliographical Index) and is rarely used with all available categories and documents. We have chosen to split this text collection as done in [Lewis *et al*., 1996]. The result is a task comprising 49 closely related categories using a very technical vocabulary. Similarly to Reuters, a document may be classified in one or many categories.

Finally, Reuters Corpus Vol. 1 (RCV1) [Rose *et al*., 2001] is a newer text collection released by Reuters Corp. that consists of one full year of new stories. There are about 850,000 documents. 103 categories have documents assigned to them. This collection is very large, thus making

it a very challenging task for learning models such as SVM and KNN, which have polynomial complexity. Particularly, we were not able to use SVM with a large training set since SVM does not scale up very well to large text collections. Using our KNN implementation, we have limited the training set to the first 100,000 documents and the testing set to the next 100,000 documents[3]. An average of 3.15 categories is assigned to each testing document (over 315,000 total assignments).

3.2 Classifiers, feature selection and settings

The weighting method presented in this paper is intended to weight documents in the Vector Space Model. Thus, it can be used only with classifiers using this model. For this reason, we have evaluated our method using both KNN and SVM and compared the results obtained with `TFIDF` and `GainRatio` [Debole and Sebastiani, 2003] weighting.

We have used the SVM^{light} package [Joachims, 1998b] and the KNN classifier described in [Yang and Liu, 1999]. In our experiments with SVM, we divided each categorization task into *n* binary classification problems, as usual. In contrast, KNN is able to classify a document among the *n* categories using one multi-category classifier. To decide whether a document is classified or not in a particular category, thresholds were learned for each category [Yang and Liu, 1999]. `TFIDF` experiments were weighted using Eq. (1) and then normalized to unit length. `GainRatio` experiments were weighted as done by [Debole and Sebastiani, 2003].

To reach optimal classification accuracy, feature selection might be required. Thus, we have included feature selection in our tests. The Information Gain measure has been used to rank the features and many thresholds have been used to filter features out; with `ConfWeight`, in addition to the use of the Information Gain to select features, when *maxstr* (see Eq. 8) was 0, the feature was also rejected. Stop words were not removed and words were not stemmed.

4 Results and discussion

To assess classifier accuracy, a confusion matrix is created for each category:

	Classifier positive label	Classifier negative label
True positive label	A	B
True negative label	C	D

Table 1: Confusion matrix used to evaluate classifier accuracy

For instance, A (the true positives) is the number of documents labeled by the classifier to the category that are correct predictions. Similarly, B (the false negatives) is the number of documents that have not been labeled by the classifier to the category, but that should have.

For any category, the classifier *precision* is defined as A/(A+C) and the *recall* as A/(A+B). To combine these two measures in a single value, the F-measure is often used. The F-measure reflects the relative importance of recall versus precision. When as much importance is granted to precision as it is to recall we have the F1-measure:

$$F1 = \frac{(precision + recall)}{2 \cdot precision \cdot recall} \tag{10}$$

The F1-measure is an estimation of the breakeven point where precision and recall meets if classifier parameters are tuned to balance precision and recall. Since the F1 can be evaluated for each category, we get *n* different F1 values.

To compare two methods, it is needed to combine all the F1 values. In order to do that, two approaches are often used: the macro-F1 average and the micro-F1 average. The macro-F1 average is the simple average of all F1 values; thus each category gets the same weigh in the average. In counterpart, the micro-F1 average weighs large categories more than smaller ones. The micro-F1 is the F1 in (10) where A, B, C and D are global values instead of category-based ones. For instance, A in the micro-F1 is the total number of classifications made by the *n* classifiers that were good predictions. Micro-F1 has been widely used in Text Categorization [Lewis *et al*., 1996; Yang and Liu, 1999; Joachims, 1998a]. Table 2 includes micro-F1 results for SVM while Table 3 includes those of KNN. For each experiment, the best score (among `TFIDF`, `GainRatio` and `ConfWeight`) is bolded.

These results show that at low Information Gain thresholds, `ConfWeight` clearly outperforms both `TFIDF` and `GainRatio`. When more drastic term selection is conducted, overall scores tend to decrease for all three term weighting methods. Is it very interesting to note the very large difference between `ConfWeight` and `TFIDF` using KNN. This difference is particularly significant for a collection of the size of RCV1.

Figure 2, 3 and 4 show the curves resulting from the use of an increasing number of features (decreasing Information Gain thresholds) for each weighting method using KNN. Clearly, `ConfWeight` is the only weighting that doesn't suffer a decrease in accuracy as low-scored features are added. `TFIDF` results are less stable than `ConfWeight` and `GainRatio`, an observation that leads us to claim that `TFIDF` is very sensitive to the choice of feature selection settings.

While `GainRatio` is less sensitive to the presence of all terms (relevant or not) than `TFIDF`, `ConfWeight` seems not to need term selection at all, arguably due to its inherent term selection mechanism. We believe that `ConfWeight` can be used without feature selection and produce very good results.

[3] At the time these experiments were conducted, the LYRL2004 split was not yet released

IGain threshold	Weighting	Reuters 21578	Ohsumed
0	TFIDF	.848	.65
	GainRatio	.875	.702
	ConfWeight	**.877**	**.706**
0.001	TFIDF	.851	.679
	GainRatio	.875	.703
	ConfWeight	**.877**	**.707**
0.005	TFIDF	.875	.695
	GainRatio	.877	**.701**
	ConfWeight	**.882**	.697
0.01	TFIDF	.876	.692
	GainRatio	.877	.696
	ConfWeight	**.882**	**.697**
0.015	TFIDF	**.874**	.693
	GainRatio	.871	.694
	ConfWeight	**.874**	**.696**
0.03	TFIDF	.842	**.658**
	GainRatio	**.844**	**.658**
	ConfWeight	.836	**.658**

Table 2: SVM Micro-F1s by text collection and weighting method

IGain threshold	Weighting	Reuters 21578	Ohsumed	RCV1
0	TFIDF	.816	.588	.785
	GainRatio	.834	.659	.792
	ConfWeight	**.861**	**.683**	**.830**
.001	TFIDF	.819	.645	.812
	GainRatio	.833	.66	.812
	ConfWeight	**.862**	**.687**	**.833**
.005	TFIDF	.843	.621	.828
	GainRatio	.840	.661	.817
	ConfWeight	**.861**	**.683**	**.833**
.01	TFIDF	.856	.640	.823
	GainRatio	.834	.659	.822
	ConfWeight	**.864**	**.681**	**.824**
.015	TFIDF	.85	.655	.811
	GainRatio	.832	.654	**.812**
	ConfWeight	**.856**	**.675**	.811
.03	TFIDF	**.832**	.640	.757
	GainRatio	.799	.639	**.765**
	ConfWeight	.824	**.646**	.749

Table 3: KNN Micro-F1s by text collection and weighting method

Another interesting remark is that the best overall scores on each corpora, both using KNN and SVM, are obtained by `ConfWeight` (Reuters-21578: .882 with SVM and .864 with KNN; Ohsumed: .707 with SVM, .687 with KNN; RCV1 .833 with KNN).

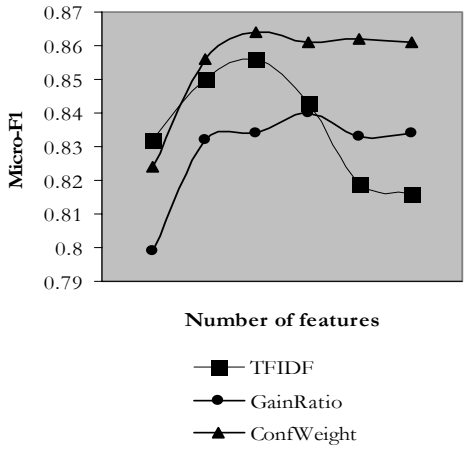


Figure 2: KNN Micro-F1s on Reuters-21578 as the number of feature increases

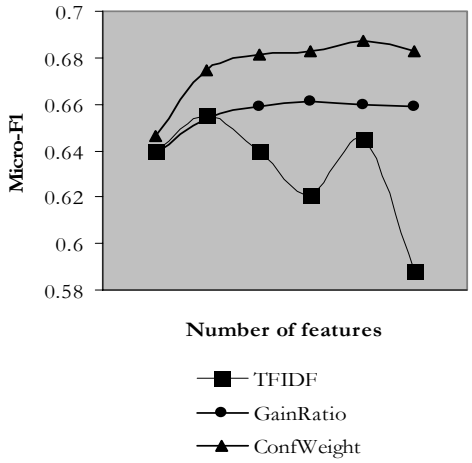


Figure 3: KNN Micro-F1s on Ohsumed as the number of feature increases

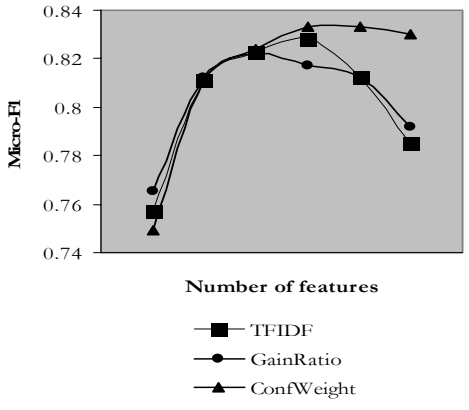


Figure 4: KNN Micro-F1s on RCV1 as the number of feature increases

Finally, we believe that ConfWeight is able to leverage the many features that get a low Information Gain score, which is not always the case with TFIDF and GainRatio. Let us take as an example the TFIDF behavior with SVM in table 2. At .005, there is much less features in the feature space than at .001. Adding features scored between .001 and .005 decreases the Micro-F1 for Reuters-21578 and Ohsumed. On the other hand, the accuracy with ConfWeight increases on Ohsumed if these same low-score features are added to the feature space, while results on Reuters-21578 stay about the same. Using only TFIDF, we might have concluded that features which have an Information Gain lower than 0.005 are harmful for most categorization tasks. Conversely, results so far using ConfWeight tend to show the relevancy and usefulness of low-score features in some settings.

5 Conclusions

In this paper, we have presented a new method (ConfWeight) to weight features in the vector-space model for text categorization by leveraging the categorization task. So far, the most commonly used method is TFIDF, which is unsupervised. To assess our new method, tests have been conducted using three well known text collections: Reuters-21578, Ohsumed and Reuters Corpus Vol. 1. As ConfWeight generally outperformed TFIDF and GainRatio on these text collections, our conclusion is that ConfWeight could be used as a replacement to TFIDF with significant accuracy improvements on the average, as shown in Tables 2 and 3. Moreover, ConfWeight has the ability to perform very well even if no feature selection is conducted, something depicted in the results presented in this paper. Actually, when a feature is irrelevant to the classification task, the weight it gets from ConfWeight is so low that this is merely equivalent to the feature rejection by a feature selection process. TFIDF, on the other hand, always yields a score higher than 0 (if the term occurs in the document for which TFIDF is computed) and this score is not related to the categorization problem, but only to the text collection as a whole. Since feature selection is not inherent to TFIDF, many additional parameters (for instance, the feature selection function to use and thresholds) need to be tuned to achieve optimal results.

[Debole and Sebastiani, 2003] argue for the use of supervised methods to weight features (GainRatio and ConfWeight are two such methods). Despite positive results in some settings, GainRatio failed to show that supervised weighting methods are generally higher than unsupervised ones. We believe that ConfWeight is a promising supervised weighting technique that behaves gracefully both with and without feature selection. Therefore, we advocate its use in further experiments.

References

[Brank *et al.*, 2002] J. Brank, M. Grobelnik, N. Frayling and D. Mladenic. Interaction of Feature Selection Methods and Linear Classification Models, In *Proc. of 19th Conf. on Machine Learning (ICML-02)*, Workshop on Text Learning.

[Debole and Sebastiani, 2003] F. Debole and F. Sebastiani. Supervised term weighting for automated text categorization. In *Proc. of SAC-03, 18th ACM Symposium on Applied Computing*, Melbourne, US, 2003, pp. 784-788.

[Dumais *et al*., 1998] S. Dumais, J. Platt, D. Heckerman and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. of the 1998 ACM 7th International Conference on Information and Knowledge Management*, 148-155, 1998.

[Han 1999] E.H. Han. Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification. PhD thesis, University of Minnesota, Oct.1999.

[Joachims, 1998a] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of the European Conference on Machine Learning,* Springer, 1998.

[Joachims, 1998b] T. Joachims, Making Large-Scale SVM Learning Practical. *LS8-Report*, 24, Universität Dortmund, 1998.

[Lewis *et al.*, 1996] D.D. Lewis, R. Schapire, J. Callan, and R. Papka. Training Algorithms for Linear Text Classifiers, In *Proc. of ACM SIGIR*, 298-306, 1996.

[Lewis, 1997] D.D. Lewis. *Reuters-21578 text categorization test collection*, Distrib. 1.0, Sept 26. 1997.

[Rose *et al.*, 2001] T.G. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1 - from yesterday's news to tomorrow's language resources. In *Proc. of the Third International Conference on Language Resources and Evaluation*, Spain, 29-31 May. 2001.

[Salton *et al.*, 1975] G. Salton, A. Wong, and C.S. Yang. *A vector space model for information retrieval*. Journal of the American Society for Information Science, 18(11):613-620, Nov. 1975.

[Yang and Liu, 1999] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR-99*, 1999.

[Wilson, 1927] E.B. Wilson. Probable Inference, the Law of Succession, and Statistical Inference. *Journal of the American Statistical Association*, 22, 209, 212. 1927.

Measuring Semantic Similarity by Latent Relational Analysis

Peter D. Turney
Institute for Information Technology
National Research Council Canada
M-50 Montreal Road, Ottawa, Ontario, Canada, K1A 0R6
peter.turney@nrc-cnrc.gc.ca

Abstract

This paper introduces Latent Relational Analysis (LRA), a method for measuring semantic similarity. LRA measures similarity in the semantic relations between two pairs of words. When two pairs have a high degree of relational similarity, they are analogous. For example, the pair cat:meow is analogous to the pair dog:bark. There is evidence from cognitive science that relational similarity is fundamental to many cognitive and linguistic tasks (e.g., analogical reasoning). In the Vector Space Model (VSM) approach to measuring relational similarity, the similarity between two pairs is calculated by the cosine of the angle between the vectors that represent the two pairs. The elements in the vectors are based on the frequencies of manually constructed patterns in a large corpus. LRA extends the VSM approach in three ways: (1) patterns are derived automatically from the corpus, (2) Singular Value Decomposition is used to smooth the frequency data, and (3) synonyms are used to reformulate word pairs. This paper describes the LRA algorithm and experimentally compares LRA to VSM on two tasks, answering college-level multiple-choice word analogy questions and classifying semantic relations in noun-modifier expressions. LRA achieves state-of-the-art results, reaching human-level performance on the analogy questions and significantly exceeding VSM performance on both tasks.

1 Introduction

This paper introduces Latent Relational Analysis (LRA), a method for measuring relational similarity. LRA has potential applications in many areas, including information extraction, word sense disambiguation, machine translation, and information retrieval.

Relational similarity is correspondence between relations, in contrast with *attributional similarity*, which is correspondence between attributes [Medin *et al.*, 1990]. When two words have a high degree of attributional similarity, we say they are *synonymous*. When two *pairs* of words have a high degree of relational similarity, we say they are *analogous*. For example, the word pair mason:stone is analogous to the pair carpenter:wood; the relation between mason and stone is highly similar to the relation between carpenter and wood.

Past work on semantic similarity measures has mainly been concerned with attributional similarity. For instance, Latent Semantic Analysis (LSA) can measure the degree of similarity between two words, but not between two relations [Landauer and Dumais, 1997].

Recently the Vector Space Model (VSM) of information retrieval has been adapted to the task of measuring relational similarity, achieving a score of 47% on a collection of 374 college-level multiple-choice word analogy questions [Turney and Littman, 2005]. The VSM approach represents the relation between a pair of words by a vector of frequencies of predefined patterns in a large corpus.

LRA extends the VSM approach in three ways: (1) the patterns are derived automatically from the corpus (they are not predefined), (2) the Singular Value Decomposition (SVD) is used to smooth the frequency data (it is also used this way in LSA), and (3) automatically generated synonyms are used to explore reformulations of the word pairs. LRA achieves 56% on the 374 analogy questions, statistically equivalent to the average human score of 57%. On the related problem of classifying noun-modifier relations, LRA achieves similar gains over the VSM. For both problems, LRA's performance is state-of-the-art.

To motivate this research, Section 2 briefly outlines some possible applications for a measure of relational similarity. Related work with the VSM approach to relational similarity is described in Section 3. The LRA algorithm is presented in Section 4. LRA and VSM are experimentally evaluated by their performance on word analogy questions in Section 5 and on classifying semantic relations in noun-modifier expressions in Section 6. We discuss the interpretation of the results, limitations of LRA, and future work in Section 7. The paper concludes in Section 8.

2 Applications of Relational Similarity

Many problems in text processing would be solved (or at least greatly simplified) if we had a black box that could take as input two chunks of text and produce as output a measure of the degree of similarity in the meanings of the two chunks. We could use it for information retrieval, ques-

tion answering, machine translation using parallel corpora, information extraction, word sense disambiguation, text summarization, measuring lexical cohesion, identifying sentiment and affect in text, and many other tasks in natural language processing. This is the vision that motivates research in paraphrasing [Barzilay and McKeown, 2001] and textual entailment [Dagan and Glickman, 2004], two topics that have lately attracted much interest.

In the absence of such a black box, current approaches to these problems typically use measures of attributional similarity. For example, the standard bag-of-words approach to information retrieval is based on attributional similarity [Salton and McGill, 1983]. Given a query, a search engine produces a ranked list of documents, where the rank of a document depends on the attributional similarity of the document to the query. The attributes are based on word frequencies; relations between words are ignored.

Although attributional similarity measures are very useful, we believe that they are limited and should be supplemented by relational similarity measures. Cognitive psychologists have also argued that human similarity judgements involve both attributional and relational similarity [Medin *et al.*, 1990].

Consider word sense disambiguation for example. In isolation, the word "plant" could refer to an industrial plant or a living organism. Suppose the word "plant" appears in some text near the word "food". A typical approach to disambiguating "plant" would compare the attributional similarity of "food" and "industrial plant" to the attributional similarity of "food" and "living organism" [Lesk, 1986; Banerjee and Pedersen, 2003]. In this case, the decision may not be clear, since industrial plants often produce food and living organisms often serve as food. It would be very helpful to know the relation between "food" and "plant" in this example. In the text "food *for* the plant", the relation between food and plant strongly suggests that the plant is a living organism, since industrial plants do not need food. In the text "food *at* the plant", the relation strongly suggests that the plant is an industrial plant, since living organisms are not usually considered as locations.

A measure of relational similarity could potentially improve the performance of any text processing application that currently uses a measure of attributional similarity. We believe relational similarity is the next step, after attributional similarity, towards the black box envisioned above.

3 Related Work

Let R_1 be the semantic relation between a pair of words, *A* and *B*, and let R_2 be the semantic relation between another pair, *C* and *D*. We wish to measure the relational similarity between R_1 and R_2. The relations R_1 and R_2 are not given to us; our task is to infer these hidden (latent) relations and then compare them.

In the VSM approach of Turney and Littman [2005], we create vectors, r_1 and r_2, that represent features of R_1 and R_2, and measure the similarity of R_1 and R_2 by the cosine of the angle θ between $r_1 = \langle r_{1,1}, \ldots, r_{1,n} \rangle$ and $r_2 = \langle r_{2,1}, \ldots r_{2,n} \rangle$:

$$\cos(\theta) = \frac{\sum_{i=1}^{n} r_{1,i} \cdot r_{2,i}}{\sqrt{\sum_{i=1}^{n} (r_{1,i})^2 \cdot \sum_{i=1}^{n} (r_{2,i})^2}} = \frac{r_1 \cdot r_2}{\|r_1\| \cdot \|r_2\|}.$$

We make a vector, *r*, to characterize the relationship between two words, *X* and *Y*, by counting the frequencies of various short phrases containing *X* and *Y*. Turney and Littman [2005] use a list of 64 joining terms, such as "of", "for", and "to", to form 128 phrases that contain *X* and *Y*, such as *"X* of *Y"*, *"Y* of *X"*, *"X* for *Y"*, *"Y* for *X"*, *"X* to *Y"*, and *"Y* to *X"*. These phrases are then used as queries for a search engine and the number of hits (matching documents) is recorded for each query. This process yields a vector of 128 numbers. If the number of hits for a query is *x*, then the corresponding element in the vector *r* is $\log(x+1)$.

Turney and Littman [2005] evaluated the VSM approach by its performance on 374 college-level multiple-choice SAT analogy questions, achieving a score of 47%. A SAT analogy question consists of a target word pair, called the *stem*, and five *choice* word pairs. To answer an analogy question, vectors are created for the stem pair and each choice pair, and then cosines are calculated for the angles between the stem vector and each choice vector. The best guess is the choice pair with the highest cosine. We use the same set of analogy questions to evaluate LRA in Section 5.

The best previous performance on the SAT questions was achieved by combining thirteen separate modules [Turney *et al.*, 2003]. The performance of LRA significantly surpasses this combined system, but there is no real contest between these approaches, because we can simply add LRA to the combination, as a fourteenth module. Since the VSM module had the best performance of the thirteen modules [Turney *et al.*, 2003], the following experiments focus on comparing VSM and LRA.

The VSM was also evaluated by its performance as a distance measure in a supervised nearest neighbour classifier for noun-modifier semantic relations [Turney and Littman, 2005]. The problem is to classify a noun-modifier pair, such as "laser printer", according to the semantic relation between the head noun (printer) and the modifier (laser). The evaluation used 600 noun-modifier pairs that have been manually labeled with 30 classes of semantic relations [Nastase and Szpakowicz, 2003]. For example, "laser printer" is classified as *instrument*; the printer uses the laser as an instrument for printing. A testing pair is classified by searching for its single nearest neighbour in the labeled training data. The best guess is the label for the training pair with the highest cosine; that is, the training pair that is *most analogous* to the testing pair, according to VSM. LRA is evaluated with the same set of noun-modifier pairs in Section 6.

4 Latent Relational Analysis

LRA takes as input a set of word pairs and produces as output a measure of the relational similarity between any two of the input pairs. LRA relies on three resources, (1) a search engine with a very large corpus of text, (2) a broad-coverage thesaurus of synonyms, and (3) an efficient im-

plementation of SVD. LRA does not use labeled data, structured data, or supervised learning.

LRA proceeds as follows:

1. Find alternates: For each word pair *A:B* in the input set, look in the thesaurus for the top num_sim words (in the following experiments, num_sim = 10) that are most similar to *A*. For each A' that is similar to *A*, make a new word pair $A':B$. Likewise, look for the top num_sim words that are most similar to *B*, and for each B', make a new word pair $A:B'$. *A:B* is called the *original* pair and each $A':B$ or $A:B'$ is an *alternate* pair. The intent is for alternates to have almost the same semantic relations as the original.

2. Filter alternates: For each original pair *A:B*, filter the 2 × num_sim alternates as follows. For each alternate pair, send a query to the search engine, to find the frequency of phrases that begin with one member of the pair and end with the other. The phrases cannot have more than max_phrase words (we use max_phrase = 5). Sort the alternate pairs by the frequency of their phrases. Select the top num_filter most frequent alternates and discard the remainder (we use num_filter = 3, so 17 alternates are dropped). This step tends to eliminate alternates that have no clear semantic relation.

3. Find phrases: For each pair (originals and alternates), make a list of phrases in the corpus that contain the pair. Query the search engine for all phrases that begin with one member of the pair and end with the other, with a minimum of min_inter intervening words and a maximum of max_inter intervening words (we use min_inter = 1, max_inter = 3 = max_phrase – 2). We ignore suffixes when searching for phrases that match a given pair. These phrases reflect the semantic relations between the words in each pair.

4. Find patterns: For each phrase found in the previous step, build patterns from the intervening words. A pattern is constructed by replacing any or all or none of the intervening words with wild cards (one wild card can only replace one word). For each pattern, count the number of pairs (originals and alternates) with phrases that match the pattern (a wild card must match exactly one word). Keep the top num_patterns most frequent patterns and discard the rest (we use num_patterns = 4,000). Typically there will be millions of patterns, so it is not feasible to keep them all.

5. Map pairs to rows: In preparation for building a matrix $\mathbf{X}$, suitable for SVD, create a mapping of word pairs to row numbers. For each pair *A:B*, create a row for *A:B* and another row for *B:A*. This will make the matrix more symmetrical, reflecting our knowledge that the relational similarity between *A:B* and *C:D* should be the same as the relational similarity between *B:A* and *D:C*. (Mason is to stone as carpenter is to wood. Stone is to mason as wood is to carpenter.) The intent is to assist SVD by enforcing this symmetry in the matrix.

6. Map patterns to columns: Create a mapping of the top num_patterns patterns to column numbers. For each pattern *P*, create a column for "$word_1$ *P* $word_2$" and another column for "$word_2$ *P* $word_1$". Thus there will be 2 × num_patterns columns in $\mathbf{X}$.

7. Generate a sparse matrix: Generate a matrix $\mathbf{X}$ in sparse matrix format. The value for the cell in row *i* and column *j* is the frequency of the *j*-th pattern (see step 6) in phrases that contain the *i*-th word pair (see step 5).

8. Calculate entropy: Apply log and entropy transformations to the sparse matrix [Landauer and Dumais, 1997]. Each cell is replaced with its logarithm, multiplied by a weight based on the negative entropy of the corresponding column vector in the matrix. This gives more weight to patterns that vary substantially in frequency for each pair.

9. Apply SVD: After log and entropy transformations, apply SVD to $\mathbf{X}$. SVD decomposes $\mathbf{X}$ into a product of three matrices $\mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are in column orthonormal form (i.e., the columns are orthogonal and have unit length) and Σ is a diagonal matrix of *singular values* (hence SVD) [Golub and Van Loan, 1996]. If $\mathbf{X}$ is of rank r, then Σ is also of rank r. Let Σ_k, where $k < r$, be the diagonal matrix formed from the top k singular values, and let $\mathbf{U}_k$ and $\mathbf{V}_k$ be the matrices produced by selecting the corresponding columns from $\mathbf{U}$ and $\mathbf{V}$. The matrix $\mathbf{U}_k\Sigma_k\mathbf{V}_k^T$ is the matrix of rank k that best approximates the original matrix $\mathbf{X}$, in the sense that it minimizes the approximation errors [Golub and Van Loan, 1996]. We may think of this matrix $\mathbf{U}_k\Sigma_k\mathbf{V}_k^T$ as a "smoothed" or "compressed" version of the original matrix. SVD is used to reduce noise and compensate for sparseness.

10. Projection: Calculate $\mathbf{U}_k\Sigma_k$ (we use k = 300, as recommended by Landauer and Dumais [1997]). This matrix has the same number of rows as $\mathbf{X}$, but only k columns (instead of 2 × num_patterns columns; in our experiments, that is 300 columns instead of 8,000). We do not use $\mathbf{V}$, because we want to calculate the cosines between row vectors, and it can be proven that the cosine between any two row vectors in $\mathbf{U}_k\Sigma_k$ is the same as the cosine between the corresponding two row vectors in $\mathbf{U}_k\Sigma_k\mathbf{V}_k^T$.

11. Evaluate alternates: Let *A:B* and *C:D* be any two word pairs in the input set. From step 2, we have (num_filter + 1) versions of *A:B*, the original and num_filter alternates. Likewise, we have (num_filter + 1) versions of *C:D*. Therefore we have $(\text{num\_filter} + 1)^2$ ways to compare a version of *A:B* with a version of *C:D*. Look for the row vectors in $\mathbf{U}_k\Sigma_k$ that correspond to the versions of *A:B* and the versions of *C:D* and calculate the $(\text{num\_filter} + 1)^2$ cosines (in our experiments, there are 16 cosines).

12. Calculate relational similarity: The relational similarity between *A:B* and *C:D* is the average of the cosines, among the $(\text{num\_filter} + 1)^2$ cosines from step 11, that are greater than or equal to the cosine of the original pairs, *A:B* and *C:D*. The requirement that the cosine must be greater than or equal to the original cosine is a way of filtering out poor analogies, which may be introduced in step 1 and may have slipped through the filtering in step 2. Averaging the cosines, as opposed to taking their maximum, is intended to provide some resistance to noise.

In our experiments, the input set contains from 600 to 2,244 word pairs. Steps 11 and 12 can be repeated for each two input pairs that are to be compared.

In the following experiments, we use a local copy of the Waterloo MultiText System (WMTS) as the search engine, with a corpus of about 5×10^{10} English words. The corpus

was gathered by a web crawler from US academic web sites [Clarke *et al.*, 1998]. The WMTS is a distributed (multiprocessor) search engine, designed primarily for passage retrieval (although document retrieval is possible, as a special case of passage retrieval). Our local copy runs on a 16-CPU Beowulf Cluster.

The WMTS is well suited to LRA, because it scales well to large corpora (one terabyte, in our case), it gives exact frequency counts (unlike most web search engines), it is designed for passage retrieval (rather than document retrieval), and it has a powerful query syntax.

As a source of synonyms, we use Lin's [1998] automatically generated thesaurus. Lin's thesaurus was generated by parsing a corpus of about 5×10^7 English words, consisting of text from the Wall Street Journal, San Jose Mercury, and AP Newswire [Lin, 1998]. The parser was used to extract pairs of words and their grammatical relations. Words were then clustered into synonym sets, based on the similarity of their grammatical relations. Two words were judged to be highly similar when they tended to have the same kinds of grammatical relations with the same sets of words.

Given a word and its part of speech, Lin's thesaurus provides a list of words, sorted in order of decreasing attributional similarity. This sorting is convenient for LRA, since it makes it possible to focus on words with higher attributional similarity and ignore the rest.

We use Rohde's SVDLIBC implementation of the Singular Value Decomposition, which is based on SVDPACKC [Berry, 1992].

5 Experiments with Word Analogy Questions

Table 1 shows one of the 374 SAT analogy questions, along with the relational similarities between the stem and each choice, as calculated by LRA. The choice with the highest relational similarity is also the correct answer for this question (quart is to volume as mile is to distance).

Table 1. Relation similarity measures for a sample SAT question.

Stem:		quart:volume	Relational similarity
Choices:	(a)	day:night	0.373725
	(b)	mile:distance	**0.677258**
	(c)	decade:century	0.388504
	(d)	friction:heat	0.427860
	(e)	part:whole	0.370172

LRA correctly answered 210 of the 374 analogy questions and incorrectly answered 160 questions. Four questions were skipped, because the stem pair and its alternates did not appear together in any phrases in the corpus, so all choices had a relational similarity of zero. Since there are five choices for each question, we would expect to answer 20% of the questions correctly by random guessing. Therefore we score the performance by giving one point for each correct answer and 0.2 points for each skipped question. LRA attained a score of 56.4% on the 374 SAT questions.

The average performance of college-bound senior high school students on verbal SAT questions corresponds to a score of about 57% [Turney and Littman, 2005]. The difference between the average human score and the score of LRA is not statistically significant.

With 374 questions and 6 word pairs per question (one stem and five choices), there are 2,244 pairs in the input set. In step 2, introducing alternate pairs multiplies the number of pairs by four, resulting in 8,976 pairs. In step 5, for each pair *A:B*, we add *B:A*, yielding 17,952 pairs. However, some pairs are dropped because they correspond to zero vectors (they do not appear together in a window of five words in the WMTS corpus). Also, a few words do not appear in Lin's thesaurus, and some word pairs appear twice in the SAT questions (e.g., lion:cat). The sparse matrix (step 7) has 17,232 rows (word pairs) and 8,000 columns (patterns), with a density of 5.8% (percentage of nonzero values).

Table 2 compares LRA to VSM with the 374 analogy questions. VSM-AV refers to the VSM using AltaVista's database as a corpus. The VSM-AV results are taken from Turney and Littman [2005]. We estimate the AltaVista search index contained about 5×10^{11} English words at the time the VSM-AV experiments took place. Turney and Littman [2005] gave an estimate of 1×10^{11} English words, but we believe this estimate was slightly conservative. VSM-WMTS refers to the VSM using the WMTS, which contains about 5×10^{10} English words. We generated the VSM-WMTS results by adapting the VSM to the WMTS.

Table 2. LRA versus VSM with 374 SAT analogy questions.

	VSM-AV	VSM-WMTS	LRA
Correct	176	144	210
Incorrect	193	196	160
Skipped	5	34	4
Total	374	374	374
Score	47.3%	40.3%	56.4%

All three pairwise differences in the three scores in Table 2 are statistically significant with 95% confidence, using the Fisher Exact Test. Using the same corpus as the VSM, LRA achieves a score of 56% whereas the VSM achieves a score of 40%, an absolute difference of 16% and a relative improvement of 40%. When VSM has a corpus ten times larger than LRA's corpus, LRA is still ahead, with an absolute difference of 9% and a relative improvement of 19%.

Comparing VSM-AV to VSM-WMTS, the smaller corpus has reduced the score of the VSM, but much of the drop is due to the larger number of questions that were skipped (34 for VSM-WMTS versus 5 for VSM-AV). With the smaller corpus, many more of the input word pairs simply do not appear together in short phrases in the corpus. LRA is able to answer as many questions as VSM-AV, although it uses the same corpus as VSM-WMTS, because Lin's [1998] thesaurus allows LRA to substitute synonyms for words that are not in the corpus.

VSM-AV required 17 days to process the 374 analogy questions [Turney and Littman, 2005], compared to 9 days for LRA. As a courtesy to AltaVista, Turney and Littman [2005] inserted a five second delay between each query. Since the WMTS is running locally, there is no need for delays. VSM-WMTS processed the questions in one day.

6 Experiments with Noun-Modifier Relations

This section describes experiments with 600 noun-modifier pairs, hand-labeled with 30 classes of semantic relations [Nastase and Szpakowicz, 2003]. We experiment with both a 30-class problem and a 5-class problem. The 30 classes of semantic relations include *cause* (e.g., in "flu virus", the head noun "virus" is the *cause* of the modifier "flu"), *location* (e.g., in "home town", the head noun "town" is the *location* of the modifier "home"), *part* (e.g., in "printer tray", the head noun "tray" is *part* of the modifier "printer"), and *topic* (e.g., in "weather report", the head noun "report" is about the *topic* "weather"). For a full list of classes, see Nastase and Szpakowicz [2003] or Turney and Littman [2005]. The 30 classes belong to 5 general groups of relations, *causal* relations, *temporal* relations, *spatial* relations, *participatory* relations (e.g., in "student protest", the "student" is the *agent* who performs the "protest"; *agent* is a *participatory* relation), and *qualitative* relations (e.g., in "oak tree", "oak" is a *type* of "tree"; *type* is a *qualitative* relation).

The following experiments use single nearest neighbour classification with leave-one-out cross-validation. For leave-one-out cross-validation, the testing set consists of a single noun-modifier pair and the training set consists of the 599 remaining noun-modifiers. The data set is split 600 times, so that each noun-modifier gets a turn as the testing word pair. The predicted class of the testing pair is the class of the single nearest neighbour in the training set. As the measure of nearness, we use LRA to calculate the relational similarity between the testing pair and the training pairs.

Following Turney and Littman [2005], we evaluate the performance by accuracy and also by the macroaveraged F measure [Lewis, 1991]. The F measure is the harmonic mean of precision and recall. Macroaveraging calculates the precision, recall, and F for each class separately, and then calculates the average across all classes.

There are 600 word pairs in the input set for LRA. In step 2, introducing alternate pairs multiplies the number of pairs by four, resulting in 2,400 pairs. In step 5, for each pair *A:B*, we add *B:A*, yielding 4,800 pairs. Some pairs are dropped because they correspond to zero vectors and a few words do not appear in Lin's thesaurus. The sparse matrix (step 7) has 4,748 rows and 8,000 columns, with a density of 8.4%.

Table 3 shows the performance of LRA and VSM on the 30-class problem. VSM-AV is VSM with the AltaVista corpus and VSM-WMTS is VSM with the WMTS corpus. The results for VSM-AV are taken from Turney and Littman [2005]. All three pairwise differences in the three F measures are statistically significant at the 95% level, according to the Paired T-Test. The accuracy of LRA is significantly higher than the accuracies of VSM-AV and VSM-WMTS, according to the Fisher Exact Test, but the difference between the two VSM accuracies is not significant. Using the same corpus as the VSM, LRA's accuracy is 15% higher in absolute terms and 61% higher in relative terms.

Table 4 compares the performance of LRA and VSM on the 5-class problem. The accuracy and F measure of LRA are significantly higher than the accuracies and F measures of VSM-AV and VSM-WMTS, but the differences between the two VSM accuracies and F measures are not significant. Using the same corpus as the VSM, LRA's accuracy is 14% higher in absolute terms and 32% higher in relative terms.

Table 3. Comparison of LRA and VSM on the 30-class problem.

	VSM-AV	VSM-WMTS	LRA
Correct	167	148	239
Incorrect	433	452	361
Total	600	600	600
Accuracy	27.8%	24.7%	39.8%
Precision	27.9%	24.0%	41.0%
Recall	26.8%	20.9%	35.9%
F	26.5%	20.3%	36.6%

Table 4. Comparison of LRA and VSM on the 5-class problem.

	VSM-AV	VSM-WMTS	LRA
Correct	274	264	348
Incorrect	326	336	252
Total	600	600	600
Accuracy	45.7%	44.0%	58.0%
Precision	43.4%	40.2%	55.9%
Recall	43.1%	41.4%	53.6%
F	43.2%	40.6%	54.6%

7 Discussion

The experimental results in Sections 5 and 6 demonstrate that LRA performs significantly better than the VSM, but it is also clear that there is room for improvement. The accuracy might not yet be adequate for practical applications, although past work has shown that it is possible to adjust the tradeoff of precision versus recall [Turney and Littman, 2005]. For some of the applications, such as information extraction, LRA might be suitable if it is adjusted for high precision, at the expense of low recall.

Another limitation is speed; it took almost nine days for LRA to answer 374 analogy questions. However, with progress in computer hardware, speed will gradually become less of a concern. Also, the software has not been optimized for speed; there are several places where the efficiency could be increased and many operations are parallelizable. It may also be possible to precompute much of the information for LRA, although this would require substantial changes to the algorithm.

The difference in performance between VSM-AV and VSM-WMTS shows that VSM is sensitive to the size of the corpus. Although LRA is able to surpass VSM-AV when the WMTS corpus is only about one tenth the size of the AV corpus, it seems likely that LRA would perform better with a larger corpus. The WMTS corpus requires one terabyte of hard disk space, but progress in hardware will likely make ten or even one hundred terabytes affordable in the relatively near future.

For noun-modifier classification, more labeled data should yield performance improvements. With 600 noun-modifier pairs and 30 classes, the average class has only 20 examples. We expect that the accuracy would improve sub-

stantially with five or ten times more examples, but it is time consuming and expensive to acquire hand-labeled data.

Another issue with noun-modifier classification is the choice of classification scheme for the semantic relations. The 30 classes of Nastase and Szpakowicz [2003] might not be the best scheme. Other researchers have proposed different schemes [Rosario and Hearst, 2001]. It seems likely that some schemes are easier for machine learning than others.

8 Conclusion

This paper has introduced a new method for calculating relational similarity, Latent Relational Analysis. The experiments demonstrate that LRA performs better than the VSM approach, when evaluated with SAT word analogy questions and with the task of classifying noun-modifier expressions. The VSM approach represents the relation between a pair of words with a vector, in which the elements are based on the frequencies of 64 hand-built patterns in a large corpus. LRA extends this approach in three ways: (1) the patterns are generated dynamically from the corpus, (2) SVD is used to smooth the data, and (3) a thesaurus is used to explore reformulations of the word pairs.

Just as attributional similarity measures have proven to have many practical uses, we expect that relational similarity measures will soon become widely used. Relational similarity plays a fundamental role in the mind and therefore relational similarity measures could be crucial for artificial intelligence [Medin *et al.*, 1990]. LRA may be a step towards the black box that we imagined in Section 2, with many potential applications in text processing.

In future work, we plan to investigate some potential applications for LRA. It is possible that the error rate of LRA is still too high for practical applications, but the fact that LRA matches average human performance on SAT analogy questions is encouraging.

Acknowledgments

For data and software used in this research, my thanks go to Michael Littman, Vivi Nastase, Stan Szpakowicz, Egidio Terra, Charlie Clarke, Dekang Lin, Doug Rohde, and Michael Berry.

References

[Banerjee and Pedersen, 2003] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 805-810, Acapulco, Mexico, 2003.

[Barzilay and McKeown, 2001] R. Barzilay and K. McKeown. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 50-57, Toulouse, France, 2001.

[Berry, 1992] M.W. Berry. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6(1): 13-49, 1992.

[Clarke *et al.*, 1998] C.L.A. Clarke, G.V. Cormack, and C.R. Palmer. An overview of MultiText. *ACM SIGIR Forum*, 32(2): 14-15, 1998.

[Dagan and Glickman, 2004] I. Dagan and O. Glickman, Probabilistic textual entailment: Generic applied modeling of language variability. In *Learning Methods for Text Understanding and Mining*, Grenoble, France, 2004.

[Golub and Van Loan, 1996] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Third edition. Johns Hopkins University Press, Baltimore, MD, 1996.

[Landauer and Dumais, 1997] T.K. Landauer and S.T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211-240, 1997.

[Lesk, 1986] M.E. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone. In *Proceedings of ACM SIGDOC '86*, pages 24-26, 1986.

[Lewis, 1991] D.D. Lewis. Evaluating text categorization. In *Proceedings of the Speech and Natural Language Workshop*, Asilomar, pages 312-318, 1991.

[Lin, 1998] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL '98)*, pages 768-774, Montreal, Canada, 1998.

[Medin *et al.*, 1990] D.L. Medin, R.L. Goldstone, and D. Gentner. Similarity involving attributes and relations: Judgments of similarity and difference are not inverses. *Psychological Science*, 1(1): 64-69, 1990.

[Nastase and Szpakowicz, 2003] V. Nastase and S. Szpakowicz. Exploring noun-modifier semantic relations. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, Tilburg, The Netherlands, pages 285-301, 2003.

[Rosario and Hearst, 2001] B. Rosario and M. Hearst. Classifying the semantic relations in noun-compounds via a domain-specific lexical hierarchy. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*, pages 82-90, 2001.

[Salton and McGill, 1983] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[Turney *et al.*, 2003] P.D. Turney, M.L. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, Borovets, Bulgaria, pages 482-489, 2003.

[Turney and Littman, 2005] P.D. Turney and M.L. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, in press, 2005.

Theory of Alignment Generators and Applications to Statistical Machine Translation

Raghavendra Udupa U. **Hemanta K. Maji**
IBM India Research Laboratory, New Delhi
{uraghave, hemantkm}@in.ibm.com

Abstract

Viterbi Alignment and *Decoding* are two fundamental search problems in Statistical Machine Translation. Both the problems are known to be **NP**-hard and therefore, it is unlikely that there exists an optimal polynomial time algorithm for either of these search problems. In this paper we characterize exponentially large subspaces in the solution space of *Viterbi Alignment* and *Decoding*. Each of these subspaces admits polynomial time optimal search algorithms. We propose a local search heuristic using a neighbourhood relation on these subspaces. Experimental results show that our algorithms produce better solutions taking substantially less time than the previously known algorithms for these problems.

1 Introduction

Statistical Machine Translation (SMT) is a data driven approach to Machine Translation [Brown *et al.*, 1993], [Al-Onaizan *et al.*, 1999], [Berger *et al.*, 1996]. Two of the fundamental problems in SMT are [Brown *et al.*, 1993]:

$$\mathbf{a}^* = \underset{\mathbf{a}}{argmax}\, Pr\left(\mathbf{f}, \mathbf{a}|\mathbf{e}\right) \quad (\mathbf{ViterbiAlignment})$$

$$(\mathbf{e}^*, \mathbf{a}^*) = \underset{\mathbf{e},\mathbf{a}}{argmax}\, Pr\left(\mathbf{f}, \mathbf{a}|\mathbf{e}\right) Pr\left(\mathbf{e}\right) \quad (\mathbf{Decoding})$$

Viterbi Alignment has a lot of applications in Natural Language Processing [Wang and Waibel, 1998], [Marcu and Wong, 2002]. While there exist simple polynomial time algorithms for computing the *Viterbi Alignment* for IBM Models 1-2, only heuristics are known for Models 3-5. Recently, [Udupa and Maji, 2005a] showed that the computation of *Viterbi Alignment* is **NP**-Hard for IBM Models 3-5.

Decoding algorithm is an essential component of all SMT systems [Wang and Waibel, 1997], [Tillman and Ney, 2000], [Och *et al.*, 2001], [Germann *et al.*, 2003], [Udupa *et al.*, 2004]. The problem is known to be **NP**-Hard for IBM Models 1-5 when the language model is a bigram model [Knight, 1999].

Unless $\mathbf{P} = \mathbf{NP}$, it is unlikely that there exist polynomial time algorithms for either *Viterbi Alignment* or *Decoding*. Therefore, there is a lot of interest in finding fast heuristics to find acceptable solutions for the problems.

Previous approaches to the *Viterbi Alignment* problem have focussed on defining a graph over all possible alignments where the connectivity is based on local modifications. The best known algorithm for *Decoding* (both in terms of speed and translation quality) employs a greedy search algorithm. However, these approaches can look at only polynomial number of alignments in polynomial time.

In this paper, we characterize exponentially large subspaces of the solution space of *Viterbi Alignment* and *Decoding*. We propose polynomial time optimal dynamic programming algorithms to solve *Viterbi Alignment* and *Decoding* when the search space is restricted to a particular subspace. There are exponentially many such subspaces. So, we perform a local search on these subspaces under a suitable neighbourhood relation. Though, there is no polynomial bound on the number of iterations of the local search, experiments show that our algorithms procduce better solutions in significantly less time than the current algorithms.

We introduce the notion of an *alignment generator* in Section 2. There are $m!$ distinct alignment generators (where m is the length of the source language sentence). Each alignment generator ($\mathbf{g_i}$, where $1 \leq i \leq m!$) is associated with an exponentially large subspace of alignments ($\mathcal{A}_i$). The solution space of *Viterbi Alignment* and *Decoding* can be written as the union of these subspaces. We next consider the graph induced on the set of $\mathcal{A}_i$s by a neighbourhood relation and perform local search on this graph. The explicit mathematical definition of $\mathcal{A}_i$ and the neighbourhood relation are presented in Section 3. We present polynomial time optimal algorithms for *Viterbi Alignment* and *Decoding* over a particular $\mathcal{A}_i$ (Section 4 and Section 5 respectively). Experiments show that our search algorithms produce better solutions taking subtantially less time than current algorithms (Section 6).

2 Preliminaries

IBM Models 1-5 assume that there is a hidden *alignment* between the source and target sentences.

Definition 1 (Alignment) *An alignment is a function* $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})} : \{1, \ldots, m\} \rightarrow \{0, 1, \ldots, l\}$.

$\{1, \ldots, m\}$ is the set of source positions and $\{0, 1, \ldots, l\}$ is the set of target positions. The target position 0 is known as the null position. We denote $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}(j)$ by $a_j^{(m,\,l)}$. The fertility of the target position i in alignment $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ is $\phi_i =$

$\sum_{j=1}^{m} \delta\left(a_j^{(m,\,l)}, i\right)$. If $\phi_i = 0$, then the target position i is an *infertile position*, otherwise it is a *fertile position*. Let $\{k, \ldots, k+\gamma-1\} \subseteq \{1, \ldots, l\}$ be the longest range of consecutive infertile positions, where $\gamma \geq 0$. We say that alignment $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ has an *infertility width* of γ.

Representation of Alignments: Since all source positions that are not connected to any of the target positions $1, \ldots, l$ are assumed to be connected to the null position (0), we can represent an alignment, without loss of generality, as a set of tuples $\{(j, a_j)\,|a_j \neq 0\}$. Let $m' = m - \phi_0$. Thus, a sequence representation of an alignment has m' tuples. By sorting the tuples in ascending order using the second field as the key, $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ can now be represented as a sequence $\left(j_1, a_{j_1}^{(m,\,l)}\right) \ldots \left(j_{m'}, a_{j_{m'}}^{(m,\,l)}\right)$ where the $a_{j_k}^{(m,\,l)}$s are non-decreasing with increasing k. Note that there are $(l+1)^m$ possible alignments for a given $m, l > 0$. Let $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}\big|_{(v,u)}$ be the alignment $\{1, \ldots, v\} \rightarrow \{0, \ldots, u\}$ where all the tuples $\left(j, a_j^{(m,\,l)}\right)$ such that $j > v$ or $a_j^{(m,\,l)} > u$ have been removed.

Definition 2 (Generator) *A generator is a bijection* $\mathbf{g}^{(\mathbf{m})} : \{1, \ldots, m\} \rightarrow \{1, \ldots, m\}$.

Representation of Generators: A generator $\mathbf{g}^{(\mathbf{m})}$ can be represented as a sequence of tuples $(j_1, 1) \ldots (j_m, m)$. Note that there are $m!$ different possible generators for a given $m > 0$. The identity function $\mathbf{g}^{(\mathbf{m})}(j) = j$ is denoted by $\mathbf{g_0^{(m)}}$.

Definition 3 ($\mathbf{g}^{(\mathbf{m})} \longrightarrow \mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$) *An alignment* $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ *(with tuple sequence* $\left(j_1, a_{j_1}^{(m,\,l)}\right) \ldots \left(j_{m'}, a_{j_{m'}}^{(m,\,l)}\right)$*) is said to be generated by a generator* $\mathbf{g}^{(\mathbf{m})}$ *(with tuple sequence* $(k_1, 1) \ldots (k_m, m)$*) if* $j_1 \ldots j_{m'}$ *is a subsequence of* $k_1 \ldots k_m$.

Note that the above definition does not take into account the *infertility* width of the alignment. Hence, we refine the above definition.

Definition 4 ($\mathbf{g}^{(\mathbf{m})} \xrightarrow{\gamma} \mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$) *An alignment* $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ *is said to be* γ*-generated by* $\mathbf{g}^{(\mathbf{m})}$*, if* $\mathbf{g}^{(\mathbf{m})} \longrightarrow \mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ *and the infertility width of* $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ *is at most* γ.

Definition 5 ($\mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}$)

$$\mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}} = \left\{\mathbf{a}^{(\mathbf{m},\,\mathbf{l})} \mid \mathbf{g}^{(\mathbf{m})} \xrightarrow{\gamma} \mathbf{a}^{(\mathbf{m},\,\mathbf{l})}\right\}$$

Thus, for every generator $\mathbf{g}^{(\mathbf{m})}$ and integer $\gamma \geq 0$ there is an associated family of alignments $\mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}$ for a given $l \in \mathbb{N}$. Further, $\mathcal{A}_{\gamma,\star}^{\mathbf{g}^{(\mathbf{m})}} = \cup_l \mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}$.

Swap Operation: Let $\mathbf{g}^{(\mathbf{m})}$ be a generator and $(j_1, 1) \ldots (j_k, k) \ldots (j_{k'}, k') \ldots (j_m, m)$ be its tuple sequence. The result of a SWAP operation on the kth and k'th tuples in the sequence is another tuple sequence $(j_1, 1) \ldots (j_{k'}, k) \ldots (j_k, k') \ldots (j_m, m)$. The new tuple sequence defines a generator $\mathbf{g}'^{(\mathbf{m})}$. The relationship between any two generators is given by:

Lemma 1 *If* $\mathbf{g}^{(\mathbf{m})}$ *and* $\mathbf{g}'^{(\mathbf{m})}$ *are two generators then* $\mathbf{g}'^{(\mathbf{m})}$ *can be obtained from* $\mathbf{g}^{(\mathbf{m})}$ *by a series of swap operations.*

Proof: Follows from the properties of permutations. ■

We can apply a sequence of $Swap$ operators to modify any given generator $\mathbf{g}^{(\mathbf{m})}$ to $\mathbf{g_0^{(m)}}$. The source position r_v $(1 \leq v \leq m)$ in $\mathbf{g}^{(\mathbf{m})}$ corresponds to the source position v in $\mathbf{g_0^{(m)}}$. For brevity, we denote a sequence $v_i\, v_{i+1} \ldots v_j$ by $\mathbf{v_i^j}$. The source sentence is represented by $\mathbf{f_1^m}$ and the target sentence is represented by $\mathbf{e_1^l}$.

3 Framework

We now describe the common framework for solving *Viterbi Alignment* and *Decoding*. The solution space for *Viterbi Alignment* can be written as $\cup_{\mathbf{g}^{(\mathbf{m})}}\ \mathcal{A}_{l,\,l}^{\mathbf{g}^{(\mathbf{m})}}$, and the solution space of *Decoding* can be approximated by $\cup_{\mathbf{g}^{(\mathbf{m})}}\ \mathcal{A}_{\gamma,\star}^{\mathbf{g}^{(\mathbf{m})}}$ for a large enough γ. The key idea is to find the optimal solution in an exponentially large subspace $\mathcal{A}$ corresponding to a generator ($\mathcal{A} = \mathcal{A}_{l,\,l}^{\mathbf{g}^{(\mathbf{m})}}$ and $\mathcal{A} = \mathcal{A}_{\gamma,\star}^{\mathbf{g}^{(\mathbf{m})}}$ for *Viterbi Alignment* and *Decoding* respectively). Given an optimal search algorithm for $\mathcal{A}$, we use Lemma 1 to augment the search space. Our framework is as follows

1. Choose any generator $\mathbf{g}^{(\mathbf{m})}$ as the initial generator.
2. Find the best solution for the problem in $\mathcal{A}$.
3. Pick a better generator $\mathbf{g}'^{(\mathbf{m})}$ via a swap operation on $\mathbf{g}^{(\mathbf{m})}$.
4. Repeat steps 2 and 3 until the solution cannot be improved further.

3.1 Algorithms

Viterbi Alignment

The algorithm for finding a good approximate solution to *Viterbi Alignment* employs a subroutine `Viterbi_For_Generator` to find the best solution in the exponentially large subspace defined by $\mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}$. The implementation of this subroutine is discussed in Section 4.

Algorithm 1 Viterbi Alignment

```
g^(m) ← g_0^(m).
while (true) do
    a^*(m,l) = Viterbi_For_Generator(g^(m), f_1^m, e_1^l).
    Find an alignment a'^(m,l) with higher score by using
    swap operations on a^*(m,l).
    if (a'^(m,l) = NIL) then
        break.
    end if
    g^(m) ← A generator of a'^(m,l).
end while
Output a^*(m,l)
```

Decoding
Our algorithm for *Decoding* employs a subroutine `Decode_For_Generator` to find the best solution in the exponentially large subspace defined by $\mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\star}$. The implementation of this subroutine is discussed in Section 5.

Algorithm 2 Decoding

1: $\mathbf{g}^{(\mathbf{m})} \leftarrow \mathbf{g}_0^{(\mathbf{m})}$.
2: **while** (`true`) **do**
3: $\left\langle \mathbf{e}_1^{*\mathbf{l}}, \mathbf{a}^{*(\mathbf{m},\,\mathbf{l})} \right\rangle =$ `Decode_For_Generator` $\left(\mathbf{g}^{(\mathbf{m})}, \mathbf{f}_1^{\mathbf{m}}\right)$.
4: Find an alignment $\mathbf{a}^{\prime(\mathbf{m},\,\mathbf{l})}$ with higher score by using swap operations on $\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})}$ and $\mathbf{e}_1^{*\mathbf{l}}$.
5: **if** $\left(\mathbf{a}^{\prime(\mathbf{m},\,\mathbf{l})} = \mathtt{NIL}\right)$ **then**
6: `break`.
7: **end if**
8: $\mathbf{g}^{(\mathbf{m})} \leftarrow$ A generator of $\mathbf{a}^{\prime(\mathbf{m},\,\mathbf{l})}$.
9: **end while**
10: Output $\mathbf{e}_1^{*\mathbf{l}}$.

3.2 Theory of Generators

In this section, we prove some important results on generators.

Structure Transformation Operations
We define a set of structure transformation operations on generators. The application of each of these operations on a generator modifies the structure of a generator and produces an alignment. Without loss of generality [1], we assume that the generator is the identity generator $\mathbf{g}_0^{(\mathbf{m})}$ and define the structural transformation operations on it. The tuple sequence for $\mathbf{g}_0^{(\mathbf{m})}$ is $(1,1)\ldots(m,m)$. Given an alignment $\mathbf{a}^{(\mathbf{j-1,i})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{j-1})}}_{\gamma,i}$ with $\phi_i > 0$, we explain the modifications introduced by each of these operators to the jth tuple.

1. SHRINK: Extend $\mathbf{a}^{(\mathbf{j-1,i})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{j-1})}}_{\gamma,i}$ to $\mathbf{a}^{(\mathbf{j,i})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{j})}}_{\gamma,i}$ such that $a_j^{(j,i)} = 0$.
2. MERGE: Extend $\mathbf{a}^{(\mathbf{j-1,i})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{j-1})}}_{\gamma,i}$ to $\mathbf{a}^{(\mathbf{j,i})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{j})}}_{\gamma,i}$ such that $a_j^{(j,i)} = i$.
3. (γ,k)-GROW $(0 \le k \le \gamma)$: Extend $\mathbf{a}^{(\mathbf{j-1,i})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{j-1})}}_{\gamma,i}$ to $\mathbf{a}^{(\mathbf{j,i+k+1})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{j})}}_{\gamma,i+k+1}$ such that $a_j^{(j,i+k+1)} = i+k+1$.

By removing the infertile positions at the end, we obtain the following result:

Lemma 2 *$\mathbf{a}^{(\mathbf{m},\,\mathbf{l})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\,l}$ and has at least one fertile position iff there exists s such that $0 \le s \le \gamma$, $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}\big|_{(m,\,l-s)} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\,l-s}$ and $\phi_{l-s} > 0$.*

[1] by permuting the given generator $\mathbf{g}^{(\mathbf{m})}$ to the identity generator $\mathbf{g}_0^{(\mathbf{m})}$

Theorem 1 *If $\mathbf{a}^{(\mathbf{m},\mathbf{l-s})} \in \mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\,l-s}$ such that $\phi_{l-s} > 0$ then it can be obtained from $\mathbf{g}_0^{(\mathbf{m})}$ by a series of* SHRINK, MERGE *and (γ,k)-*GROW *operations.*

Proof: WLOG we assume that the generator is $\mathbf{g}_0^{(\mathbf{m})}$. We provide a construction that has m phases. In the vth step, we construct $\mathbf{a}^{(\mathbf{m},\,\mathbf{l-s})}\big|_{(v,u)}$ from $\mathbf{g}_0^{(\mathbf{v})}$ in the following manner:

- We construct $\mathbf{a}^{(\mathbf{m},\,\mathbf{l-s})}\big|_{(v-1,u)}$ from $\mathbf{g}_0^{(\mathbf{v-1})}$ and employ a SHRINK operation on the tuple (v,v) if $a_v^{(m,\,l-s)} = 0$.
- We construct $\mathbf{a}^{(\mathbf{m},\,\mathbf{l-s})}\big|_{(v-1,u)}$ from $\mathbf{g}_0^{(\mathbf{v-1})}$ and employ a MERGE operation on the tuple (v,v) if $\phi_u > 1$.
- If $\phi_u = 1$: Let $\mathbf{a}^{(\mathbf{m},\,\mathbf{l-s})}\big|_{(v,u)}$ define $\phi_t = 0$ for $u-k \le t \le u-1$ and $\phi_{u-k-1} > 0$, where $0 \le k \le \gamma$. Construct $\mathbf{a}^{(\mathbf{m},\,\mathbf{l-s})}\big|_{v-1,u-k-1}$ from $\mathbf{g}_0^{(\mathbf{v-1})}$ and apply (γ,k)-GROW to the (v,v) tuple. (*Note*: If $(u-k-1) = 0$, then consider the alignment where $\{1,\ldots,v-1\}$ are all aligned to the null position)

As these are the only possibilities in the vth phase, at the end of the mth phase we get $\mathbf{a}^{(\mathbf{m},\mathbf{l-s})}$. ■

The dynamic programs for *Viterbi Alignment* and *Decoding* are based on ideas in Lemma 2 and Theorem 1.

Lemma 3 *$\left|\mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\,l}\right|$ is given by the coefficient of x^l in*

$$\left(\frac{\alpha_\gamma - 2}{x}\right)\left[(\alpha_\gamma - 2)\left(\frac{\alpha_\gamma^m - 1}{\alpha_\gamma - 1}\right) + 1\right] \tag{1}$$

, where $\alpha_\gamma = 2 + \sum_{k=0}^{\gamma} x^{k+1}$.

Proof: Refer [Udupa and Maji, 2005b] ■

Theorem 2 $\left|\mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\star}\right| = (\gamma+1)\left(\frac{(\gamma+1)(\gamma+3)^m+1}{\gamma+2}\right)$

Proof: Substitute $x = 1$ in Equation 1 ■

`Decode_For_Generator` finds the optimal solution over all possible alignments in $\mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\star}$ in polynomial time.

Theorem 3 *For a fixed l, $\left|\mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\,l}\right| = \Omega\left(2^m\right)$.*

Proof: Refer [Udupa and Maji, 2005b] ■

`Viterbi_For_Generator` finds the optimal solution over all possible alignments in $\mathcal{A}^{\mathbf{g}_0^{(\mathbf{m})}}_{\gamma,\,l}$ in polynomial time.

Lemma 4 *Let $\mathcal{S} = \left\{\mathbf{g}_1^{(\mathbf{m})}, \ldots, \mathbf{g}_N^{(\mathbf{m})}\right\}$ be a set of generators. Define span $(\mathcal{S}) = \cup_{i=1}^{N} \mathcal{A}^{\mathbf{g}_i^{(\mathbf{m})}}_{\gamma=l,\,l}$. Let $p(.)$ be a polynomial. If $l \ge 2$ and $N = \mathcal{O}(p(m))$, then there exists an alignment $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})}$ such that $\mathbf{a}^{(\mathbf{m},\,\mathbf{l})} \notin$ span $(\mathcal{S})$.*

Proof: Refer [Udupa and Maji, 2005b] ■

This Lemma shows that by considering any polynomially large set of generators we can not obtain the optimal solutions to either *Viterbi Alignment* or *Decoding*.

4 Viterbi Alignment

We now develop the algorithm for the subroutine `Viterbi_For_Generator` for IBM Model 3. Recall that this subroutine solves the following problem:

$$\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})} = \underset{\mathbf{a}^{(\mathbf{m},\,\mathbf{l})} \in \mathcal{A}_{\gamma,\,l}^{\mathbf{g}^{(\mathbf{m})}}}{argmax}\ Pr\left(\mathbf{f_1^m}, \mathbf{a}^{(\mathbf{m},\,\mathbf{l})} | \mathbf{e_1^l}\right).$$

Our algorithm has a time complexity polynomial in m and l. Without loss of generality, we assume that $\mathbf{g}^{(\mathbf{m})} = \mathbf{g_0^{(m)}}$.

Note that $\mathbf{a}^{*(\mathbf{m},\,\mathbf{l})}\big|_{(m,\,l-s)}$ (where $0 \leq s \leq \gamma$ and $\phi_{l-s} > 0$) can be obtained from $\mathbf{g_0}$ using the structure transformation operations described in Section 3.2. Therefore, we build the Viterbi alignment from left to right. We scan the tuple sequence from left to right in m phases and in any phase we determine the best structure transformation operation for that phase.

We consider all possible partial alignments between $\mathbf{f_1^v}$ and $\mathbf{e_1^u}$, such that $\phi_u = \varphi > 0$ and $\phi_0 = \varphi_0$. Let $B(u, v, \varphi_0, \varphi)$ be the best partial alignment between $\mathbf{f_1^v}$ and $\mathbf{e_1^u}$ and $A(u, v, \varphi_0, \varphi)$ be its score[2]. Here, φ_0 is the number of French words aligned to e_0 in the partial alignment.

The key idea here is to compute $A(u, v, \varphi_0, \varphi)$ and $B(u, v, \varphi_0, \varphi)$ recursively using Dynamic Programming. In the vth phase (corresponding to the tuple (v, v)), we consider each of the structure transformation operation.

1. SHRINK: If $\varphi_0 > 0$, the SHRINK operation extends $B(u, v-1, \varphi_0 - 1, \phi)$ and the score of the resulting partial alignment is

   $$s_1 = cA(u, v-1, \varphi_0 - 1, \varphi)$$

   where $c = \mathtt{t}(f_v|\mathtt{NULL})\,\frac{\mathtt{p}_1}{\mathtt{p}_0^2}\,\frac{(m-2\varphi_0+1)(m-2\varphi_0+2)}{(m-\varphi_0+1)\varphi_0}$.

2. MERGE: If $\varphi > 0$, the MERGE operation extends $B(u, v-1, \varphi_0, \varphi-1)$ and the score of the resulting partial alignment is

   $$s_2 = cA(u, v-1, \varphi_0, \varphi-1)$$

   where $c = \frac{\mathtt{n}(\varphi|e_u)\phi}{\mathtt{n}(\varphi-1|e_u)}\mathtt{t}(f_v|e_u)\,\mathtt{d}(r_v|u, l, m)$.

3. (γ, k)-GROW: Let

   $$\varphi' = \underset{k}{argmax}\, B(u-k-1, v-1, \varphi_0, k).$$

   (γ, k)-GROW extends $B(u-k-1, v-1, \varphi_0, \varphi')$ if $\varphi = 1$. If $u - k - 1 > 0$, the score of the resulting partial alignment is

   $$s_{3,k} = cA(u-k-1, v-1, \varphi_0, \varphi')$$

   where

   $$c = \mathtt{n}(1|e_u)\,\mathtt{t}(f_v|e_u)\,\mathtt{d}(r_v|u, l, m)\prod_{p=u-k}^{u-1}\mathtt{n}(0|e_p).$$

   If $u - k - 1 = 0$, then

   $$s_{3,k} = \binom{m-\varphi_0}{\varphi_0}\mathtt{p_0}^{m-2\varphi_0}\mathtt{p_1}^{\varphi_0}\prod_{p=1}^{v-1}\mathtt{t}(f_p|e_0)$$
   $$\times\,\mathtt{n}(1|e_u)\,\mathtt{t}(f_v|e_u)\,\mathtt{d}(r_v|u, l, m)\prod_{p=u-k}^{u-1}\mathtt{n}(0|e_p).$$

   We choose that operation which gives the best score. As a result, $B(u, v, \varphi_0, \varphi)$ now represents the best partial alignment so far and $A(u, v, \varphi_0, \varphi)$ represents its score. We have,

   $$A(u, v, \varphi_0, \varphi) = \underset{\substack{\mathbf{a}^{(\mathbf{v},\,\mathbf{u})} \in \mathcal{A}_{\gamma,\,u}^{\mathbf{g}^{(\mathbf{v})}} \\ \phi_u = \varphi,\ \phi_0 = \varphi_0}}{argmax}\ Pr\left(\mathbf{f_1^v}, \mathbf{a}^{(\mathbf{v},\,\mathbf{u})} | \mathbf{e_1^u}\right)$$

   We determine $A(u, v, \varphi_0, \varphi)$ and $B(u, v, \varphi_0, \varphi)$, for all $0 \leq u \leq l$, $0 \leq v \leq m$, $0 < \varphi \leq \varphi_{\max}$ and $0 \leq \varphi_0 \leq \frac{m}{2}$. To complete the scores of the alignments, we multiply each $A(u, m, \varphi_0, \varphi)$ by $\prod_{p=u+1}^{l}\mathtt{n}(0|e_p)$ for $l - \gamma \leq u \leq l$.

   Let

   $$(\hat{u}, \hat{\varphi}_0, \hat{\varphi}) = \underset{\substack{l-\gamma \,\leq\, u \,\leq\, l \\ 0 \,\leq\, \varphi_0 \,\leq\, \frac{m}{2} \\ 0 \,<\, \varphi \,\leq\, \varphi_{\max}}}{argmax}\ A(u, m, \varphi_0, \varphi).$$

   The Viterbi Alignment is, therefore, $B(\hat{u}, m, \hat{\varphi}_0, \hat{\varphi})$.

4.1 Time Complexity

The algorithm computes $B(u, v, \varphi_0, \varphi)$ and $A(u, v, \varphi_0, \varphi)$ and therefore, there are $\mathcal{O}\left(lm^2\varphi_{\max}\right)$ entries that need to be computed. Note that each of these entries is computed incrementally by a structure transformation operation.

- SHRINK operation requires $\mathcal{O}(1)$ time.
- MERGE operation requires $\mathcal{O}(1)$ time.
- (γ, k)-GROW ($0 \leq k \leq \gamma$) operation requires $\mathcal{O}(\varphi_{\max})$ time for finding the best alignment and another $\mathcal{O}(1)$ time to update the score[3].

Each iteration takes $\mathcal{O}(\gamma\varphi_{\max})$ time. Computation of the tables A and B takes $\mathcal{O}\left(lm^2\gamma\varphi_{\max}^2\right)$ time. The final step of finding the Viterbi alignment from the table entries takes $\mathcal{O}(m\gamma\varphi_{\max})$ time[4]. Thus, the algorithm takes $\mathcal{O}\left(lm^2\gamma\varphi_{\max}\right)$ time. In practice, γ and $\varphi_{\max}$ are constants. The time complexity of our algorithm is $\mathcal{O}\left(lm^2\right)$.

4.2 Space Complexity

There are $\mathcal{O}\left(lm^2\varphi_{\max}\right)$ entries in each of the two tables. Assuming γ and $\varphi_{\max}$ to be constants, the space complexity of `Viterbi_For_Generator` is $\mathcal{O}\left(lm^2\right)$.

IBM Models 4 and 5 Extending this procedure for IBM Models 4 and 5 is easy and we leave it to the reader to figure out the modifications.

[2] A naive implementation of the table $B(*, *, *, *)$ takes $\mathcal{O}(m)$ size for each element. It can instead be implemented as a table of decisions and a pointer to the alignment that was extended to obtain it. This makes the size of each entry $\mathcal{O}(1)$.

[3] Note: The product $\prod_{p=u-k}^{u-1}\mathtt{n}(0|e_p)$ can be calculated incrementally for each k, so that at each iteration only one multiplication needs to be done. Similarly, the score $\prod_{p=1}^{v-1}\mathtt{t}(f_p|\mathtt{NULL})$ can be calculated incrementally over the loop on v.

[4] The factor $\prod_{u+1}^{l}\mathtt{n}(0|e_p)$ can be calculated incrementally over the loop of u. So, it takes $\mathcal{O}(1)$ time in each iteration.

5 Decoding

The algorithm for the subroutine `Decode_For_Generator` is similar in spirit to the algorithm for `Viterbi_For_Generator`. We provide the details for IBM Model 4 as it is the most popular choice for *Decoding*. We assume that the language model is a trigram model. $\mathcal{V}_E$ is the target language vocabulary and $\mathcal{I}_E \subset \mathcal{V}_E$ is the set of infertile words. Our algorithm employs Dynamic Programming and computes the following:

$$\left(\mathbf{e_1^{*L}}, \mathbf{a}^{*(\mathbf{m},\mathbf{L})}\right) = \underset{\mathbf{a}^{(\mathbf{m},\mathbf{L})} \in \mathcal{A}^{\mathbf{g}^{(\mathbf{m})}}_{\gamma,\star},\ \mathbf{e_1^L}}{\operatorname{argmax}} Pr\left(\mathbf{f_1^m}, \mathbf{a}^{(\mathbf{m},\mathbf{L})} | \mathbf{e_1^L}\right)$$

Let $\mathbf{H_1^v}\left(e'', e', \varphi_0, \varphi, \rho\right)$ be the set of all partial hypotheses which are translations of $\mathbf{f_1^v}$ and have $e''e'$ as their last two words (ρ is the center of cept associated with e' and $\varphi > 0$ is the fertility of e'). Observe that the scores of all partial hypothesis in $\mathbf{H_1^v}\left(e'', e', \varphi_0, \varphi, \rho\right)$ are incremented by the same amount thereafter if the partial hypotheses are extended with the same sequence of operations. Therefore, for every $\mathbf{H_1^v}\left(v, e'', e', \varphi_0, \varphi, \rho\right)$ it suffices to work only on the best partial hypothesis in it.

The initial partial hypothesis is $h_0\left(.,.,0,0,0\right)$ with a score of $\mathtt{p_0}^m$. Initially, the hypothesis set $\mathcal{H}$ has only one hypothesis, $h_0\left(.,.,0,0,0\right)$. We scan the tuple sequence of $\mathbf{g_0^{(m)}}$ left to right in m phases and build the translation left to right. In the vth phase, we extend each partial hypothesis in $\mathcal{H}$ by employing the structure transformation operations. Let $h_1^{v-1}\left(e'', e', \varphi_0, \varphi, \rho\right)$ be the partial hypothesis which is being extended.

- SHRINK: We create a new partial hypothesis $h_1^v\left(e'', e', \varphi_0 + 1, \varphi, \rho\right)$ whose score is the score of $h_1^{v-1}\left(e'', e', \varphi_0, \varphi, \rho\right)$ times

  $$\mathtt{t}\left(f_v|\mathtt{NULL}\right)\frac{\mathtt{p_1}}{\mathtt{p_0}^2}\frac{\left(m-2\varphi_0-1\right)\left(m-2\varphi_0\right)}{\left(m-\varphi_0\right)\left(\varphi_0+1\right)}.$$

  We put the new hypothesis in to $\mathbf{H_1^v}\left(e'', e', \varphi_0 + 1, \varphi, \rho\right)$.

- MERGE: We create a new partial hypothesis $h_1^v\left(e'', e', \varphi_0, \varphi + 1, \rho\right)$ whose score is the score of $h_1^{v-1}\left(e'', e', \varphi_0, \varphi, \rho\right)$ times

  $$\frac{\mathtt{n}\left(\varphi+1|e'\right)}{\mathtt{n}\left(\varphi|e'\right)}\mathtt{t}\left(f_v|e'\right)\frac{\mathtt{d_{new}}}{\mathtt{d_{old}}}.$$

  Here, $\mathtt{d_{old}}$ is the distortion probability score of the last tableau before expansion, $\mathtt{d_{new}}$ is the distortion probability of the expanded hypothesis after r_v is inserted into the tableau and ρ' is the new center of the cept associated with e'. We put the new hypothesis into $\mathbf{H_1^v}\left(e'', e', \varphi_0, \varphi + 1, \rho'\right)$.

- (γ, k)-GROW: We choose k words $\left(e^{(1)}, \ldots, e^{(k)}\right)$ from $\mathcal{I}_E$ and one word $\left(e^{(k+1)}\right)$ from $\mathcal{V}_E$. We create a new partial hypothesis $h_1^v\left(e', e^{(1)}, \varphi_0, 1, r_v\right)$ if $k = 0$, $h_1^v\left(e^{(k)}, e^{(k+1)}, \varphi_0, 1, r_v\right)$ otherwise. The score of this hypothesis is the score of $h_1^{v-1}\left(e'', e', \varphi_0, \varphi, \rho\right)$ times

  $$\mathtt{n}\left(1|e_{k+1}\right)\mathtt{t}\left(f_v|e_{k+1}\right)\mathtt{d}\left(r_v - \rho|\mathcal{A}\left(e'\right), \mathcal{B}\left(f_v\right)\right)\prod_{p=1}^{k}\mathtt{n}\left(0|e_p\right)$$

  We put the new hypothesis in to $\mathbf{H_1^v}\left(e^{k+1}, e^k, \varphi_0, 1, r_v\right)$.

At the end of the phase, we retain only the best hypothesis for each $\mathbf{H_1^v}\left(e'', e', \varphi_0, \varphi, \rho\right)$. After m phases, we append at most γ infertile words to all hypotheses in $\mathcal{H}$. The hypothesis with the best score in $\mathcal{H}$ is the output of the algorithm.

5.1 Time Complexity

At the beginning of vth phase, there are at most $\mathcal{O}\left(\left|\mathcal{V}_E\right|^2 v^2 \varphi_{\max}\right)$ distinct partial hypotheses. It takes $\mathcal{O}\left(1\right) + \mathcal{O}\left(\varphi_{\max}\right) + \mathcal{O}\left(\left|\mathcal{I}_E\right|^\gamma \left|\mathcal{V}_E\right| \varphi_{\max}\right)$ time to extend a hypothesis. Therefore, the algorithm takes totally $\mathcal{O}\left(\left|\mathcal{V}_E\right|^3 \left|\mathcal{I}_E\right|^\gamma m^3 \varphi_{\max}^2\right)$ time. Now, since $|\mathcal{V}_E|$, $|\mathcal{I}_E|$, $\varphi_{\max}$ and γ are assumed to be constant, the time complexity of the algorithm is $\mathcal{O}\left(m^3\right)$.

5.2 Space Complexity

In each phase, we need $\mathcal{O}\left(\left|\mathcal{V}_E\right|^2 m^2 \varphi_{\max}\right)$ space. Assuming $|\mathcal{V}_E|$ and $\varphi_{\max}$ to be constant, the space complexity is $\mathcal{O}\left(m^2\right)$.

Extending this procedure for IBM Models 3 and 5 is easy.

6 Experiments and Results

6.1 Decoding

We trained the models using the GIZA++ tool [Och, 2001] for the translation direction French to English. For the current set of experiments, we set $\gamma = 1$ and $\varphi_{\max} = 10$. To compare our results with those of a state of the art decoder, we used the Greedy decoder [Marcu and Och, 2004]. Our test data consisted of a corpus of French sentences with sentence length in the range $06-10,\ 11-15, \ldots, 56-60$. Each sentence class had 100 French sentences. In Table 1, we compare the mean logscores (negative logarithm of the probability) of each of the length classes. Lower logscore indicates better probability score. We observe that the scores for our algorithm are (20%) better than the scores for Greedy algorithm. In the same table, we report the average time required for decoding for each sentence class. Our algorithm is much faster than the Greedy algorithm for most length classes. This demonstrates the power of our algorithm to search an exponentially large subspace in polynomial time.

We compare the NIST and BLEU scores of our solutions with the Greedy solutions. Our NIST scores are 14% better while our BLEU scores are 16% better than those of the Greedy solutions.

6.2 Viterbi Alignment

We compare our results with those of local search algorithm of GIZA++ [Och, 2001]. We set $\gamma = 4$ and $\varphi_{\max} = 10$. The initial generator for our algorithm is obtained from the solution of the local search algorithm.

In Table 3, we report the mean logscores of the alignments (for each length class) found by our algorithm and the local search algorithm. Our scores are about 5% better than those of the local search algorithm.

	Time (sec.)		Score	
Class	Our	Greedy	Our	Greedy
06-10	000.18	000.08	025.31	028.09
11-15	000.89	000.40	036.35	041.36
16-20	001.86	001.47	048.49	054.52
21-25	003.74	003.82	061.90	072.29
26-30	005.41	008.21	073.62	086.04
31-35	007.70	016.62	087.40	101.98
36-40	011.23	029.09	098.84	116.62
41-45	013.29	048.19	112.06	133.10
46-50	016.83	074.03	122.88	146.28
51-55	020.22	112.82	135.31	160.75
56-60	024.21	168.64	146.21	175.53

Table 1: Mean Time and Mean logscores

	BLEU		NIST	
Class	Our	Greedy	Our	Greedy
06-10	0.3285	0.2696	4.6922	4.2705
11-15	0.2942	0.2701	5.0521	4.7236
16-20	0.3335	0.2867	5.6887	5.1301
21-25	0.2816	0.2328	5.4845	4.8847
26-30	0.3261	0.3014	5.6388	5.0480
31-35	0.2666	0.2330	5.2857	4.6997
36-40	0.3347	0.2988	5.8144	5.1785
41-45	0.3303	0.2812	6.0273	5.1311
46-50	0.3620	0.3092	6.2733	5.2489
51-55	0.3686	0.3032	6.5108	5.4288
56-60	0.3627	0.3030	6.4538	5.4089

Table 2: BLEU and NIST Scores

7 Conclusions

It would be interesting to see if the theory of alignment generators can be applied successfully for other problems in NLP.

References

[Al-Onaizan *et al.*, 1999] Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. Och, D. Purdy, N. Smith, and D. Yarowsky. Statistical machine translation: Final reprot. *JHU Workshop*, 1999.

[Berger *et al.*, 1996] A. Berger, P. Brown, S. Della Pietra, V. Della Pietra, A. Kehler, and R. Mercer. Language translation apparatus and method using context-based translation models. *United States Patent 5,510,981*, 1996.

[Brown *et al.*, 1993] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[Germann *et al.*, 2003] U. Germann, M. Jahr, D. Marcu, and K. Yamada. Fast decoding and optimal decoding for machine translation. *Artificial Intelligence*, 2003.

[Knight, 1999] Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4), 1999.

Class	Our	Greedy
06-10	017.51	017.76
11-15	026.77	027.49
16-20	037.63	038.98
21-25	046.96	048.61
26-30	057.31	059.74
31-35	068.46	071.46
36-40	079.83	083.59
41-45	090.04	094.24
46-50	102.24	107.74
51-55	112.65	119.40
56-60	122.15	129.46

Table 3: Viterbi Alignment Logscores

[Marcu and Och, 2004] D. Marcu and F. Och. Greedy decoder for statistical machine translation. `http://www.isi.edu/licensed-sw/rewrite-decoder`, 2004.

[Marcu and Wong, 2002] D. Marcu and W. Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–139. Philadelphia, 2002.

[Och *et al.*, 2001] F. Och, N. Ueffing, and H. Ney. An efficient a* search algorithm for statistical machine translation. In *Proceedings of the ACL 2001 Workshop on Data-Driven Methods in Machine Translation*, pages 55–62. Toulouse, France, 2001.

[Och, 2001] F. Och. Giza++. `http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html`, 2001.

[Tillman and Ney, 2000] C. Tillman and H. Ney. Word reordering and dp-based search in statistical machine translation. In *Proceedings of the 18th COLING*, pages 850–856. Saarbrucken, Germany, 2000.

[Udupa and Maji, 2005a] R. Udupa and H. Maji. On the computational complexity of statistical machine translation. 2005.

[Udupa and Maji, 2005b] R. Udupa and H. Maji. Theory of alignment generators (extended version). `http://www.geocities.com/hemanta_maji/algo.pdf`, 2005.

[Udupa *et al.*, 2004] R. Udupa, T. Faruquie, and H. Maji. An algorithmic framework for the decoding problem in statistical machine translation. In *Proceedings of COLING*. Geneva, Switzerland, 2004.

[Wang and Waibel, 1997] Y. Wang and A. Waibel. Decoding algorithm in statistical machine translation. In *Proceedings of the 35th ACL*, pages 366–372. Madrid, Spain, 1997.

[Wang and Waibel, 1998] Y. Wang and A. Waibel. Modeling with structures in statistical machine translation. In *Proceedings of the 36th ACL*. Montreal, Canada, 1998.

Analogy Generation with HowNet*

Tony Veale
University College Dublin,
Creative Language Systems Group, Dept. of Computer Science,
Dublin 4, Ireland.
Tony.Veale@UCD.ie

Abstract

Analogy is a powerful boundary-transcending process that exploits a conceptual system's ability to perform controlled generalization in one domain and re-specialization into another. The result of this semantic leap is the transference of meaning from one concept to another from which metaphor derives its name (literally: to carry over). Such generalization and re-specialization can be achieved using a variety of representations and techniques, most notably abstraction via a taxonomic backbone, or selective projection via structure-mapping on propositional content. In this paper we explore the extent to which a bilingual lexical ontology for English and Chinese, called HowNet, can support both approaches to analogy.

1 Introduction

Theories of analogy and metaphor are typically based either on structure-mapping (e.g., [Falkenhainer *et al.*1989; Veale and Keane, 1997]) or on abstraction e.g., [Hutton, 1982; Fass, 1988; Way, 1991; Veale, 2003). While the former is most associated with analogy, the latter has been a near-constant in the computational treatment of metaphor. Structure-mapping assumes that the causal behaviour of a concept is expressed in an explicit, graph-theoretic form so that unifying sub-graph isomorphisms can be found between different propositional representations. In contrast, abstraction theories assume that analogous concepts, even if far removed in ontological terms, will nonetheless share a common hypernym that will capture their causal similarity. Thus, we should expect an analogous pairing like *cancer* and *assassin* to have very different immediate hypernyms but to ultimately share a behavioural abstraction like *kill-agent* (e.g., see [Veale, 2003]).

With a well known lexical ontology like WordNet (see [Miller, 1995]), both structure-mapping and taxonomic approaches are problematic. The idea that a one-size-fits-all representation like WordNet will actually provide a hypernym like *kill-agent* seems convenient almost to the point of incredulity. As much as we want our ontologies to anticipate future analogies with these pro-active categorizations, most off-the-shelf ontologies simply do not possess such convenient terms (see [Wong, 2004]. Similarly, WordNet lacks the propositional content that is the necessary grist for a structure-mapping approach. The semantic content that would ideally fill this role is not explicit, but implicitly resides in the unstructured textual glosses that annotate each lexical concept.

In this paper we explore whether another lexical ontology, the Chinese/English HowNet system (see [Dong, 1988; Carpuat *et al.* 2002; Wong, 2004]), is better suited to the demands of analogy generation. HowNet combines a taxonomic backbone with an explicit, if somewhat sparse and under-specified, propositional semantics. This combination allows us to evaluate the extent to which both structure-mapping and abstraction theories of analogy can be supported by the same lexical ontology.

2 Past Work

That analogy and metaphor operate across multiple levels of conceptual abstraction has been well known since classical times. Aristotle first provided a compelling taxonomic account of both in his Poetics (see [Hutton, 1982] for a translation), and computationalists have been fascinated by this perspective ever since. While the core idea has survived relatively unchanged, one must discriminate theories that apparently presume a static type-hierarchy to be sufficient for all abstraction purposes (e.g., [Fass, 1998]), from theories that posit the need for a dynamic type hierarchy (e.g., [Way, 1991; Veale, 2003]). One must also differentiate theories that have actually been implemented (e.g., [Fass, 1988; Veale, 2003,2004]) from those that are either notional or that seem to court computational intractability (e.g., [Hutton, 1982; Way, 1991]). Perhaps most meaningfully, one must differentiate theories and implementations that assume hand-crafted, purpose-built ontologies (e.g., [Fass, 1988]) from those that exploit an existing large-scale resource like WordNet (e.g., [Veale, 2003,2004]). The latter approach side-steps any possible charge of hand-crafting by working only with third-party resources, but at the cost of living with their perceived flaws and inadequacies.

Structure-Mapping theory is founded on the premise that the most satisfying analogies are those that operate at the

* The generous support of Enterprise Ireland is acknowledged.

causal level of representation, since causality allows an analogy to offer a deep explanation for a poorly understood phenomenon (e.g., see [Falkenhainer *et al.* 1989]) Thus, *the atom as miniature solar-system* is a satisfying analogy because both source and target are causally structured around the notion of rotation. Furthermore, when comparing agents or artefacts (e.g., see [Veale and Keane, 1997]), this causality can be captured by considering the functional or behavioural commonality between target and source: a footballer can be meaningfully described as a gladiator or a warrior since each exhibits competitive behaviour, and a scalpel can be compared to a sabre, a sword or a cleaver since each has a cutting behaviour.

By employing a single lexical resource, HowNet, to implement both the taxonomic abstraction *and* the structure-mapping theories of analogy, we have as a secondary goal a demonstration that both perspectives are not fundamentally opposed. Structure-mapping can be seen as a form of structural-abstraction, where one abstracts out the causal backbone of a concept, while taxonomic abstraction, if derived from the relational structure of a concept, can also be seen as a highly selective form of structure-mapping.

3 Comparing WordNet and HowNet

HowNet and WordNet each reflect a different view of semantic organization. WordNet is *differential* in nature: rather than attempting to express the meaning of a word explicitly, WordNet instead differentiates words with different meanings by placing them in different synonym sets, and further differentiates these synsets from one another by assigning them to different positions in its taxonomy. In contrast, HowNet is *constructive* in nature. It does not provide a human-oriented textual gloss for each lexical concept, but instead combines sememes from a less discriminating taxonomy to compose a semantic representation of meaning for each word sense.

For example, the lexical concept *surgeon*|医生 is given the following semantic definition in HowNet:

surgeon|*医生*≡*{human*|*人:HostOf={Occupation*|*职位},*
domain={medical|*医}},*
{doctor|*医治:agent={~}}}*

which can be glossed thus: "a surgeon is a human with an occupation in the medical domain who acts as the agent of a doctoring activity." The *{~}* construct serves as a self-reference, to mark the location of the concept being defined in the given semantic structure. The oblique reference offered by the tilde serves to make the definition more generic, so that many different concepts can conceivably employ the same definition. Thus, HowNet uses the above definition not only for surgeon, but for medical workers in general, from orderlies to nurses, internists and neurologists.

Perhaps because HowNet relies less on hierarchical differentiation, it has a considerably less developed middle ontology than WordNet. For instance, most kinds of person in HowNet, from mathematicians to hobos, are placed directly under the hypernym *human*|人, eschewing the intermediate concepts like *{professional}, {specialist}* and *{worker}* that give substance to WordNet's middle ontology. We note that HowNet does indeed define these concepts – but unlike WordNet, it does so at the leaf level where they add nothing to the internal structure of the taxonomy.

4 Analogy Via Relational Signatures

The skeletal nature of HowNet semantic definitions, combined with the wide-spread use of *{~}* as a generic reference, suggests how HowNet might support an efficient approach to analogical recall. By indexing each concept on a reduced form of its semantic definition – a *relational signature* – analogies will correspond to collisions between concepts with different definitions but with identical signatures. Such an approach can be efficiently implemented using simple string hashing of signatures, to detect analogical collisions between kitchens and factories, generals and admirals, ballet dancers and acrobats, or cruise missiles and arrows. The devil here is in the *lack* of detail: because HowNet's definitions are frequently imprecise and fail to fully specify a concept, they allows others – potential analogues – to occupy the same reduced semantic space. The further we exacerbate this deficiency, indexing each definition on an increasingly diluted version of itself, the more distant and creative will be the analogies that are generated. For example, excluding the hypernym of a definition, or its domain markings, facilitates analogies between people and non-people, such as pests and persecutors, or hackers and viruses.

To implement both the abstraction and structure-mapping theories of analogy, we will explore the effectiveness of two kinds of relational signatures in the current work: atomic signatures based on taxonomic abstraction, and more template-like signatures based on generalized propositional content in which place-holder variables have been added.

4.1 Taxonomic Abstraction with HowNet

Given the general impoverishment of HowNet's middle ontology (at least compared with that of WordNet), abstraction-based signatures should not be based directly on taxonomic organization. Rather, by instead deriving taxonomic signatures from the relational structure of a concept's semantic definition, we can better capture the functional and behavioral nature of the concepts concerned. We can do this by focusing on that part of each definition that contains an explicit self-reference in the form *{~}*. For instance, consider the following semantic definition of *repairman*|*修理工*:

{human|*人:HostOf={Occupation*|*职位},*
{repair|*修理:agent={~}}}*

Noting the relational position of *{~}*, we can infer that a repairman is the agent of a repairing activity. Expressing this as a taxonomic abstraction, we can reify the combination of activity and role to create a new taxonomic term *repair-agent*, of which repairman will be an instance. From an analogical perspective, *repair-agent* thus serves as a good relational signature for *repairman*|*修理*.

Further noting that the HowNet taxonomy defines the predicate *repair*|*修理* as a specialization of the reinstatement predicate *resume*|恢复, we can further establish *repair-agent* as a specialization of *resume-agent*. This double layer of abstraction effectively establishes a new, parallel taxonomy that organizes lexical-concepts according to their analogical potential, rather than their formal taxonomic properties. As shown in Figure 1 then, *resume-agent* will encompass not only *repair-agent*, but *doctor-agent*, since HowNet defines the predicate *doctor*|*医治* as a specialization of *resume*|恢复.

resume-agent
- *repair-agent*
  - repairman|修理工
  - watchmaker|钟表匠
- *doctor-agent*
  - surgeon|医生
  - herbalist|药农
- *amend-agent*
  - reviser|修订者

Figure 1: Portion of a new three-level abstraction hierarchy derived from HowNet's relational structures.

In general, taxonomic signatures are generated as follows: given a semantic fragment *F:role={~}* in a HowNet definition of a concept C, we create the signatures *F-role* and *F'-role*, where *F'* is the immediate HowNet hypernym of *F*, which in turn is the immediate hypernym of *C*. The role in question might be *agent, patient, instrument,* or any other role supported by HowNet, such as *target, content,* etc.

AlterForm-instrument
- *cut-instrument*
  - knife|刀
  - razor|剃刀
- *stab-instrument*
  - sword|宝剑
  - lance|长矛
- *split-instrument*
  - grater|擦菜板
  - glasscutter|玻璃刀
- *break-instrument*
  - scissors|剪
  - chainsaw|油锯
- *dig-instrument*
  - pickaxe|镐
  - chisel|凿

Figure 2: a hierarchy of taxonomic signatures that facilitates analogy between instruments that "alter the form" of others.

Each concept is thus assigned two different taxonomic signatures: a direct signature (*F-role*) based on the specific relational structure of the concept, and another more general signature (*F'-role*) that is abstracted from this direct signature. These signatures effectively form an alternate taxonomy by which the lexical concepts in HowNet can be organized for analogical purposes. Figure 2 above illustrates a partial hierarchy derived from the HowNet semantics of various form-altering tools.

This additional layer of abstraction is necessary to facilitate creative analogy between semantically distant concepts. Nonetheless, we note that since HowNet's designers have already exercised a certain degree of metaphoric license, even concepts with the same direct signature can exhibit a surprising degree of semantic variety.

MakeBad-agent
- *kill-agent*
  - assassin|刺客
  - Death|死神
- *attack-agent*
  - intruder|侵略者

Figure 3: diversity among concepts with the same signatures.

This diversity, as illustrated by Figure 3, means that the analogy "Death is an assassin" can be generated without recourse to a more abstract signature.

4.2 Structure-Mapping with HowNet

The structure-mapping approach also strives for abstraction, not through the selective creation of new taxonyms but through a form of structural *rarefaction.* Structure-mapping theory places particular emphasis on the causal backbone of a concept's propositional content, which is usually projected unchanged from one domain to another ([Falkenhainer *et al.* 1989)]. Based on this isomorphic alignment of relational structures, the entities contained in each structure are typically placed into a 1-to-1 correspondence with one another. The attributive modifiers of these entities play a more peripheral role in structure-mapping, but in approaches like *Sapper* [Veale and Keane, 1997] they often serve as a literal grounding for an analogy.

The semantic definitions provided by HowNet are already so skeletal and under-specified that we can operate on the assumption that they represent the relational backbone of a concept's meaning. So in generating a set of structure-mapping signatures for a given concept, we will assume that each signature preserves the general form of a single proposition. Consider the HowNet definition of *blind person*|*盲人*:

{human|人:
{disable|知道:
OfPart={part|部件:PartPosition={eye|眼},
whole= {human|人}}
experiencer={~},
scope={look|看}}}

In other words, a blind person has "a disability of the eye that affects one's ability to look". One finds precisely the same propositional structure in the HowNet definition of *lame*

person|*拐子*, except that *eye*|*眼* is replaced with l*eg*|*腿* and *look*| *看* is replaced with *walk*| *走*. The goal of a structure-mapping approach is to capture this semantic isomorphism while identifying *eye:leg* and *look:walk* as cross-domain counterparts. One way to do this is to generalize from each definition a signature that, by virtue of identicality, signals a structural equivalence between individual definitions. For instance, the common structural signature for *blind person*|*盲人* and *lame person*|*拐子* might look like this:

{?:{ill|*病态:OfPart={?},experiencer={~},scope={?}}}*

Generalized structural signatures like this can be generated using the following 7-step process:

1. Split each definition into multiple propositions, and generate a separate signature for each.
2. If a proposition describes a noun concept, replace its taxonomic head with a *?* marker. In contrast, if a proposition describes a verb concept, replace its taxonomic head with its most specific hypernym.
3. Replace the conceptual arguments bound to each case-role of a predicate with the variable marker *{?}*. These markers will indicate positions in the signature where 1-to-1 correspondences between source and target structures can be made.
4. When a propositional sub-structure corresponds to the definition of another HowNet concept, replace the entire sub-structure with a *{?}* variable marker.
5. Replace predicates by their immediate hypernyms in the HowNet taxonomy. Thus, both *repair*|*修理* in the definition of *repairman*|*修理*, and *doctor*|*医治* in the definition of surgeon|医生, should be replaced by the hypernym *resume*|恢复 when generating their respective signatures.
6. Remove any explicit domain tag in a proposition from the corresponding signature (e.g., the assignment *domain={medical*|*医}* in the definition of *surgeon*|*医生*). This is necessary since analogy is meant to transcend domain boundaries.
7. Generalize any attributive value to its immediate hypernym. As such, step 3 above should not variablize the arguments of the attributive relations *modifier, manner, restrictive, host* or *content*.

Following these 7 steps, the following structural signatures will be assigned to each of the concepts *surgeon*|*医生*, *repairman*| *修理*, *reviser*| *修订者*, *watchmaker*| *钟表匠* and *herbalist*|*药农*:

{?:HostOf={?}}
{?: resume|恢复*:agent={~}}}*

[Note: because the *HostOf* relation always occurs with the binding *Occupation*|*职位* in HowNet, only the latter signature is retained as an analogical index]

More structural richness is exhibited by the lexical concepts *apostle*|*使徒* and *insider*|*局内人*, whose HowNet definitions are shown below.

apostle|*使徒*
≡ *{human*|*人:*
{believe|*修理:*
agent={~},
content={humanized|*拟人},*
domain={religion|*宗教}}}*

person who knows inside story|*个中人*
≡ *{human*|*人:*
{know|*知道:*
agent={~},
content={fact|*事情:*
modifier=covert|*隐秘}}}*

These are also assigned the same structural signature:

signature ≡ {?: {HaveKnowledge|*有知:*
agent={~},
content={?}}

The sub-structure *{fact*| *事情:modifier=covert*| *隐秘}* has been completely variablized within the signature of *person who knows inside story*| *个中人* since this corresponds to the HowNet definition of *secret*|*秘事* (see step 4). Analogically then, an apostle is a religious insider who knows the inside scoop on a deity (denoted *humanized*|*拟人* in HowNet).

5 Comparative Evaluation

Consider first the composition of the HowNet version used in this research. It contains 95,407 unique lexical concepts (excluding synonyms) and 23,507 unique semantic definitions. Clearly then, these definitions are under-specified to the extent that many are shared by non-identical concepts (such as *cart*|*板车* and *bicycle*|*单车*, which HowNet simply defines as manual vehicles). Furthermore, 90% of these definitions comprise a single proposition, while 8% comprise two propositions and only 2% comprise three or more.

We evaluate the taxonomic and structure-mapping approaches using four criteria: *coverage* – the percentage of unique HowNet definitions from which a valid signature can be derived; *recall* – the percentage of definitions for which at least one analogical counterpart can be found; *parsimony*/*precision* – the percentage of effective signatures that can actually be used to generate analogies (since a parsimonious approach will precisely generate just those signatures that are analogically useful); and *richness* – the complexity of the mappings involved, as measured by the average number of entity correspondences per analogy.

5.1 Evaluating Taxonomic Abstraction

5.1.1 Taxonomic Coverage

Since taxonomic signatures exploit occurrences of *{~}* for their generation, both the coverage and recall of the taxonomic abstraction approach depend crucially on the wide-spread usage of this reflexive construct.

However, of the 23,507 unique definitions in HowNet, just 6430 employ this form of self-reference. The coverage offered by taxonomic signatures is therefore just 27% of the available definitions.

5.1.2 Taxonomic Recall

For these 6430 self-referential definitions, 1579 unique direct signatures are generated. In turn, another 838 abstract signatures are derived from these via predicate generalization. In total, 2219 unique taxonomic signatures are generated, revealing that in 8% of cases, the abstract signature of one definition corresponds to the direct signature of another.

A majority of these signatures (59%) serve to generate analogies for 6184 semantic definitions. The overall recall rate then is 26%. The most productive taxonomic signature is *control_agent*, which serves to analogically co-index 210 unique definitions.

5.1.3 Taxonomic Parsimony/Precision

Overall, 1,315 of all 2219 taxonomic signatures prove useful in co-indexing two or more definitions, while 904 taxonomic signatures are associated with just a single definition. The parsimony of the taxonomic approach is thus 59%.

5.1.4 Taxonomic Richness

Only one mapping, at the gross level of source and target concepts, can be generated by the taxonomic approach. For instance, the approach can recognize that *blind person|盲人* and *lame person|拐子* are analogous by virtue of sharing the taxonomic signature *disable-experiencer*. However, it cannot recursively determine the entity mappings *eye:leg* and *look:walk*. The taxonomic approach thus has a uniform mapping richness of 1.

5.2 Evaluating Structure-Mapping

5.2.1 Structure-Mapping Coverage

A structure-mapping signature can be generated for every semantic definition in HowNet. In principle then, the coverage of this approach is 100%. In practice, however, 10% of HowNet's semantic definitions contain no real structure beyond the specification of a hypernym or a domain tag. The maximum coverage of structure-mapping then, as limited to definitions with relational structure, is 90%.

5.2.2 Structure-Mapping Recall

HowNet's 21,761 unique structured definitions comprise 21,929 unique propositions. From these, 21,159 unique structural signatures are derived (many are generalizations of other signatures), serving to find analogues for 14,370 definitions. The recall for structure-mapping is thus 61%.

The most productive structural signature is:

{component.部分:whole={?}}

which serves to analogically co-index 397 unique semantic definitions.

5.2.3 Structure-Mapping Parsimony/Precision

With 79% of all structural signatures serving to index just a single definition, the parsimony of the structure-mapping approach must be judged as a low 21%.

5.2.4 Structure-Mapping Richness

Most analogies (64%) generated using the structure-mapping approach imply two entity mappings, 25% imply three entity mappings, and 11% imply four or more. The average mapping richness of a structure-mapped analogy is thus 2.48.

5.3 Analysis of Results

The results of this comparison, as summarized in Table 1 below, force us to draw some important conclusions about the utility of HowNet for analogy.

	Taxonomic	**Structure-Map**	**Combo**
Coverage	.27	.90	.90
Recall	.26	.61	.72
Parsimony	.59	.21	.24
Richness	1	2.48	2.24

Table 1: Comparison of both approaches to analogy in HowNet

First, though the taxonomic approach is capped by the limited use of self-reference among HowNet definitions, it demonstrates a recall rate that closely approaches this ceiling, managing to find analogies of non-trivial complexity for 1 in 4 HowNet definitions. Because of its broader coverage, structure-mapping does considerably better, generating analogies for 3 in 5 definitions. A combination of both approaches ("combo" in Table 1) generates analogies for almost 3 in 4 definitions, which is most encouraging given the creative demands of analogy generation. This is especially so as we have considered here analogies between unique definitions, not unique words. A given definition-level analogy can be lexically realized in many, sometimes hundreds, of different ways.

6 Conclusions and Future Work

We conclude then that HowNet contains sufficient structure to realistically support both a taxonomic abstraction view *and* a structure-mapping view of analogy generation. Nonetheless, we need to investigate additional sources of semantic content to further increase the recall of the abstraction approach. For this, we must turn to the implicit content hidden in HowNet's lexico-semantic structure.

Perhaps surprisingly, one source of implicit semantic content is orthography. Most Chinese entries in HowNet are multi-character – and thus multi-morpheme – terms whose composite orthography affords a kind of semantic transparency that other writing systems (e.g., that of English) do not possess. Thus, 手术刀, meaning "scalpel", is a composite not just of characters but of ideas, for 手术 means "surgery" and 刀 means "knife". Likewise, 哲学家, which translates as

"philosopher", is a composition of 哲学 ("philosophy") and 家 ("specialist" or "scientist"). In turn, *philosophy|哲学* is ontologized by HowNet as a specialization of *knowledge|知识*, as is *logic|辩学*, *mathematics|数学*, *lexicography|词典学* and even *midwifery|产科学*. By decomposing compound terms in this way, and by generalizing the extracted modifiers, a three-level taxonomy can be constructed to complement that which is formed by taxonomic signatures. From these examples alone, the partial taxonomy of Figure 4 can be derived:

knowledge-human
  mathematics-human
    mathematician|数学家
  philosophy-human
    philosopher|哲学家
  midwifery-human
    midwife|产科
  Buddhism-human
    Buddhist|佛教徒

Figure 4: Portion of an alternate three-level hierarchy derived from Chinese compound terms (prefix term + hypernym).

The analogical potential of such an alternative signature scheme becomes clear when one notices that it immediately supports Plato's classical analogy of philosopher as midwife.

References

[Carpuat *et al.*, 2002] Carpuat, M., Ngai, G. Fung, P., and Church, K.W. Creating a Bilingual Ontology: A Corpus-Based Approach for Aligning WordNet and HowNet. *The Proceedings of GWC 2002, the 1st Global WordNet conference*, Mysore, India.

[Dong, 1988] Dong, Z. Knowledge Description: What, How and Who? *The Proceedings of the International Symposium on Electronic Dictionaries*, Tokyo, Japan.

[Hutton, 1982] Hutton, J. *Aristotle's Poetics*. Norton.

[Way, 1991] Way, E. C. Knowledge Representation and Metaphor. *Studies in Cognitive systems*. Kluwer Academic Publishers: Holland.

[Miller, 1995] Miller, G. WordNet: A Lexical Database for English. *Comms. of the ACM*, 38(11).

[Fass, 1988] Fass, D. An Account of Coherence, Semantic Relations, Metonymy, and Lexical Ambiguity Resolution. *In: S. Small, G. W. Cottrell and M. K. Tanenhaus (eds.): Lexical Ambiguity Resolution: Perspectives from Psycholinguistics, Neuropsychology and A.I.* Morgan Kauffman: CA.

[Falkenhainer *et al.*, 1989] Falkenhainer, B.; Forbus, K.; and Gentner, D**.** Structure-Mapping Engine: Algorithm and Examples. *Artificial Intelligence*, 41, pages 1-63.

[Wong, 2004] Wong, S.H.S. Fighting Arbitrariness in WordNet-like Lexical Databases – A Natural Language Motivated Remedy. *The proceedings of GWC 2004, the 2nd Global WordNet conference*, Sojka, Pala, Smrz, Fellbaum, Vossen (eds.).

[Veale and Keane, 1997] Veale, T. and Keane, M.T. The Competence of Sub-Optimal Structure Mapping on Hard Analogies. *The proceedings of IJCAI'97, the International Joint Conference on Artificial Intelligence*, Nagoya, Japan.

[Veale, 2003] Veale. T. Dynamic Type Creation in Metaphor Interpretation and Analogical Reasoning: A Case-Study with WordNet. *The proceedings of ICCS2003, the 2003 International Conference on Conceptual Structures*, Dresden, Germany

[Veale, 2004] Veale, T. WordNet sits the S.A.T.: A Knowledge-Based Approach to Lexical Analogy. *The proceedings of ECAI'2004, the 16th European Conf. on Artificial Intelligence*. John Wiley: London.

Meaning development versus predefined meanings in language evolution models

Paul Vogt[1,2]
[1]Language Evolution and Computation, School of Philosophy, Psychology and Language Sciences
University of Edinburgh, 40 George Square, Edinburgh EH8 9LL, U.K.
[2]Induction of Linguistic Knowledge / Computational Linguistics
Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands
paulv@ling.ed.ac.uk

Abstract

This paper investigates the effect of predefining semantics in modelling the evolution of compositional languages versus allowing agents to develop these semantics in parallel with the development of language. The study is done using a multi-agent model of language evolution that is based on the Talking Heads experiment. The experiments show that when allowing a co-evolution of semantics with language, compositional languages develop faster than when the semantics are predefined, but compositionality appears more stable in the latter case. The paper concludes that conclusions drawn from simulations with predefined meanings, which most studies use, may need revision.

1 Introduction

The field of evolutionary linguistics is a rapidly growing field in contemporary cognitive science. Many studies are based on computational modelling, where the researchers typically study aspects of language evolution using models that integrate AI techniques such as multi-agent systems, evolutionary computation, machine learning, natural language processing and robotics. See [Cangelosi and Parisi, 2002] for an extensive overview.

One of the most prominent aspects of human language that is researched concerns the origins and evolution of grammatical structures, such as *compositionality*. Compositionality refers to representations (typically utterances in languages) in which the meaning of the whole is a function of the meaning of its parts. Studies into the origins and evolution of compositionality have yielded models that can successfully explain how compositionality may emerge. Most models have semantic structures built in, so the agents only have to acquire a mapping from signals to these meanings, together with their syntactic structures [Brighton, 2002; Kirby, 2002; Smith *et al.*, 2003]. Only few models have considered how compositional structures can arise through a co-evolution between syntax and semantics, where the semantics are grounded through interactions with the world and develop from scratch [Steels, 2004; Vogt, 2005].

Naturally, the two approaches yield different results. Whereas in [Brighton, 2002; Kirby, 2002; Smith *et al.*, 2003] it takes a number of generations until compositionality arises, in studies where the syntax co-develops with the semantics compositionality arises from the first generation [Steels, 2004; Vogt, 2005]. Why is this difference? In this paper, the answer is sought by starting off with an implementation of Vogt's model, which was based on Kirby's model, but without predefined semantics, and then comparing this with a model in which the semantics is predefined, as in Kirby's model.

The next section presents some background relating to Kirby's and Vogt's model. Section 3 presents the model which is our starting point. The results are presented in Section 4 and discussed in Section 5.

2 Background

In the context of this paper, compositionality is defined as a representation of which the meaning of the whole can be described as a function of the meaning of its parts. For instance, the meaning of the expression "red apple" is a function of the meaning of "red" and the meaning of "apple". As a consequence, it is possible to substitute one part with another to form a new meaning as in the expression "green apple". In contrast, there are holistic representations in which the meaning of the whole cannot be described as a function of the meaning of its parts. For instance, the expression "kick the bucket" in the meaning of `dying` is a holistic phrase.

It has been repeatedly shown that compositional structures can arise from initially holistic structures (i.e. structures with no compositionality) using the *iterated learning model* (ILM) [Brighton, 2002; Kirby, 2002; Smith *et al.*, 2003]. In the ILM the population at any time consists of adults and children.[1] The children learn from the linguistic behaviour of adults. After a learning episode (or *iteration*), the adults are removed from the population, the children becomes new adults and new children enter the population and the process repeats. Kirby and others have shown that, given an induction mechanism that can induce compositional structures, an initially holistic language can change into a compositional one after a number of iterations, provided the language is transmitted through a *bottleneck*.

The transmission bottleneck entails that children only observe a part of the expressible meanings of the language. Assuming the children are equipped with a learning mechanism

[1]In most ILMs there is only one adult and one child.

to discover and store compositional structures whenever possible, these structures tend to remain in the language because they allow an agent to communicate about previously unseen meanings. Suppose, for instance, you only have observed the expressions "ab", "ad" and "cb" meaning `p(m)`, `q(m)` and `p(n)` resp.. If you have no way to discover a compositional structure, you would not be able to express the meaning `q(n)`. However, if you have the ability to acquire compositional structures such as `S -> A/x B/y`, where `A/m -> a` or `A/n -> c`, and `B/p(x) -> b` or `B/q(x) -> d`, you would be able to form the sentence "cd" to express the meaning `q(n)`. If an agent has acquired the language through a bottleneck, it may have to produce expressions about previously unseen meanings when it has become an adult. If there is no bottleneck, the children are expected to have learnt the entire language, so no compositionality is required and typically does not evolve.

In Kirby's model, all agents (adults and children alike) are equipped with the same predefined predicate argument semantics. Naturally this is unrealistic, since it is widely acknowledged that human children are born without innate semantics. The question is therefore: to what extent does predefining the agents' semantics influence the results of such simulations?

To investigate this question, Vogt [2005] implemented a simulation based on Kirby's model, but without predefining the semantics. In Vogt's model (described below), the semantics of individual agents develop in parallel with the language learning. This way, the semantics of adult agents differ from the children's.

Vogt [2005] showed that, even without a bottleneck, relatively high levels of compositionality developed very early in the simulation. It was hypothesised that this rapid emergence of compositionality has to do with the statistical nature of the input given to the agents (both from the environment and signals). Furthermore, it was shown that under certain conditions, e.g., when a bottleneck on transmission was absent, this compositionality was unstable. In those cases, the languages gradually (though sometimes suddenly) transformed into holistic languages. One possible explanation for such a transition was based on the ontogenetical development of meaning.

The study in this paper investigates the effect of predefining the semantics as opposed to semantic development with respect to the rapid development of compositionality and to the stability of the compositional structures. More concrete, a number of simulations will be presented in which the model is gradually changed from the model reported by Vogt [2005] to a model with predefined semantics as reported in Kirby [2002]. The aim is to verify the two possible explanations mentioned.

3 The model

The model is implemented in Vogt's Talking Heads simulator THSim [Vogt, 2003].[2] In this simulation, a population of agents can develop a language that allows them to communicate about a set of geometrical coloured objects that form the agents' world. Language development is controlled by agents playing a series of *guessing games*, cf. [Steels, 1997]. The description of the model provided below lacks many details and motivations. Unfortunately this is unavoidable due to the lack of space in this paper. For further details, consult [Vogt, 2005].

	action	result
1	sensing the environment	context
2	select topic	topic
3	discrimination game	meaning
4	decoding	expression
5	encoding	topic
6	evaluate success	feedback
7	induction	grammar

Table 1: A brief outline of the guessing game. Step 2 and 4 are performed only by the speaker, steps 5 and 7 only by the hearer, and all other steps by both agents.

The guessing game (GG) is briefly outlined in Table 1. The game is played by two agents: a speaker and a hearer. Both agents sense the situation (or *context*) of the game. The context C consists of a given number of geometrical coloured objects. For each object o_j, the agents extract a feature vector $\mathbf{f}_j = \{f_r, f_g, f_b, f_s\}$, where f_i are features in a specified *quality dimension* [Gärdenfors, 2000]. These dimensions relate to a sensed quality, which in the current implementation is one of the components of the rgb colour space (f_r, f_g, f_b) or a shape feature f_s indicating the shape of the object.

The speaker selects one object from the context as the *topic* o_t of the game (step 2, Table 1) and plays a *discrimination game* [Steels, 1997] to find a category that distinguishes the topic o_t from the other objects in the context $o_j \in C \backslash \{o_t\}$ (step 3). To this aim, each individual agent constructs an ontology containing *categorical features* (CF), which are points c_{ik} in some quality dimension i. Each feature f_i is categorised with that categorical feature c_{ik} such that the distance $|f_i - c_{ik}|$ is smallest.

Combining the CFs of each quality dimension constructs a category $\mathbf{c}_j$ for object o_j. The category $\mathbf{c}_j$ is a point in a n-dimensional space. If the category for the topic o_t is different from the categories of all other objects in the context, the DG succeeds. If no distinctive category is found, then each feature f_i of the topic's feature vector $\mathbf{f}_t$ is used as an exemplar to form new CFs c_{ik}, which are added to the ontology. This way, the ontology is built up from scratch.

The distinctive category resulting from the DG serves as the 'whole meaning' of o_t, where the whole meaning can be considered as a holistic semantic category. Semantic categories are represented by *conceptual spaces* [Gärdenfors, 2000]. A conceptual space is spanned by n quality dimensions and contain n-dimensional categories $\mathbf{c}_j$ formed from n CFs c_i. The holistic semantic category is spanned by all 4 quality dimensions. Other semantic categories include any possible configuration of quality dimensions, such as a colour conceptual space (spanned by the `rgb` qualities), a 1-dimensional shape space and a 'redgreen' space spanned by the `r` and `g` components of the `rgb` space. The whole

[2] Available at http://www.ling.ed.ac.uk/~paulv/thsim.html.

R_1	S -> greycircle/$[.5_r, .5_g, .5_b, .6_s]$
R_2	S -> A/rgb B/s
R_3	A -> red/$[1_r, 0_g, 0_b]$
R_4	B -> square/$[1_s]$
R_5	B -> triangle/$[0_s, .11_s]$

Table 2: A constructed example grammar. In the example S, A and B are non-terminals. R_1 is a holistic rule that rewrites to an expression greycircle with meaning $[.5_r, .5_g, .5_b, .6_s]$. R_2 is a compositional rule that rewrites to non-terminals A and B that form linguistic categories associated with the semantic categories indicated by the quality dimensions rgb (for the colour space) and s (for the shape space). Each non-terminal rewrites to another expression and meaning. (Note that an expression can be associated with more than one meaning as in R_5.)

meaning can thus be formed by a holistic category, of by a combination of more lower dimensional categories. There are two constraints in the current model: (1) All quality dimensions should be used exactly once. (2) Only combinations of 2 semantic categories are allowed. During their development, agents learn (or *discover*) which semantic categories are useful to form a compositional language. (In one experiment, though, these are predefined.)

A composition $\oplus R_i$ may be of only one holistic rule (e.g., $\oplus R_i = R_1$ in Table 2), or of several rules, such as $\oplus R_i = R_2 \circ R_3 \circ R_4$, which combines a colour with shape (note that the operator $\circ$ is used to connect terminal rules (R_3 and R_4) with compositional rules such as R_2). Each rule R_i has a rule weight ρ_i and – for the terminal rules – each possible meaning of a rule has the weight weight μ_j, both indicating the effectiveness of a rule and meaning.[3] These weights are used to compute a score s_i indicating the effectiveness of the rule based on past experiences. If the rule R_i is a terminal rule, $s_i = \rho_i \mu_j$; if R_i is a compositional rule (such as R_2), $s_i = \rho_i$. Given the composition $\oplus R_i$, one can calculate the composition score $\sigma(\oplus R_i) = \prod_{\oplus R_i} s_i$. When a rule/meaning pair is used successfully in the guessing game, the weights ρ_i and μ_j are increased, while the weights of competing rules/meanings are inhibited. The score $\sigma(\oplus R_i)$ is used to select among competing compositions.

When the speaker obtained a distinctive category from the DG, it tries to *decode* an expression (step 4, Table 1). To this aim, the speaker searches its grammar G for compositions of rules that match the distinctive category (or meaning). So, the meaning $[.5_r, .5_g, .5_b, .6_s]$ can be decoded into the expression greycircle using R_1 and meaning $[1_r, 0_g, 0_b, 1_s]$ into redsquare using $R_2 \circ R_3 \circ R_4$. (Note that elements such as 1_r and 1_s are CFs with value 1 and dimension r and s respectively.) If there is more than one composition that can decode a meaning, the composition with the highest score $\sigma(\oplus R_i)$ is selected. This implements a selection mechanism favouring effective compositions.

If there is no composition that decodes the meaning, a new rule is constructed that either decodes a part of the meaning, or that decodes the meaning holistically. The former case is true if the speaker wanted to decode, for instance, meaning $[1_r, 0_g, 0_b, .6_s]$ using the grammar of Table 2. The part $[1_r, 0_g, 0_b]$ can be decoded using composition $R_2 \circ R_3 \circ ?x$, where $?x$ is the undecoded part. A new rule of the form B -> circle/$[.6_s]$ is then constructed to fill in $?x$. In the model, expressions are constructed at random from a limited alphabet Σ of size $|\Sigma| = 15$ and with length l, where $2 \leq l \leq 8$, with a frequency distribution $p(l) \propto \frac{1}{l}$ to implement a bias toward short strings.

The decoded expression is uttered to the hearer, which tries to *encode* this expression (step 5). The hearer does not know which object $o_i \in C$ is the topic, and its aim is to guess this using the verbal hint received. To do this, the hearer first plays a DG for each individual object $o_i \in C$, thus resulting in a set of possible meanings for the expression. Then the hearer searches its grammar for compositions that can parse the expression and of which the meaning matches one of the possible meanings. If there are more than one, the hearer selects the one with highest score $\sigma(\oplus R_i)$ and guesses that the object of the corresponding meaning is the topic.

This information is passed back to the speaker, who verifies if the guess is correct and provides the hearer with feedback regarding to the outcome (step 6). If the game succeeds, the weights of used rules are increased, while competing ones are inhibited using the equation $w = \eta w + (1 - \eta)\chi$, where w is a weight (ρ_i or μ_j), and $\chi = 1$ if the weight is increased and $\chi = 0$ if it is inhibited. (Note that a rule is competing if it can decode an expression for the topic or encode the expression received. Meanings compete when they belong to the same used rule, but are not the ones used, see, e.g., R_5.) If the game fails, the speaker provides the hearer with the correct topic and the hearer will then try to induce new knowledge (step 7).

Induction proceeds in up to three of the following steps (if one step succeeds, induction stops):

1. **Exploitation.** If there is a composition that partially decodes the expression, the remaining gaps are filled. For instance, if the hearer hears "redcircle" and the topic (indicated by the speaker) has distinctive category $[1_r, 0_g, 0_b, .6_s]$, then it can decode the expression partially using composition $R_3 \circ R_4 \circ ?x$, covering meaning $[1_r, 0_g, 0_b]$. In that case, the hearer will add the rule B -> circle/$[.6_s]$ to its grammar.

2. **Chunking.** If exploitation fails, the hearer will investigate whether it can chunk the expression-meaning pair based on stored (holistic) expression-meaning pairs received in the past. (These pairs are stored in an instance base.) This is done by looking for alignments in the received expression with stored expressions and alignments in the distinctive category with stored meanings. If more chunks are possible, the chunk that has occurred most frequently or that has the largest common string is actually pursued.

   For instance, if the instance base contains an entry with expression-meaning pair "pinkcircle"-$[1_r, .7_g, .7_b, .6_s]$ and the hearer tries to induce knowledge from receiving the expression-meaning pair "greencircle"-

[3] The actual implementation of μ_j is more complex, but for the purpose of this paper, the current explanation suffices.

$[0_r, 1_g, 0_b, .6_s]$, the following alignment is present: "green<u>circle</u>"-$[0_r, 1_g, 0_b, \underline{.6_s}]$.[4] Based on this alignment the following new rules are added: `S -> C/rgb D/s`, `C -> pink/`$[1_r, .7_g, .7_b]$, `C -> green/`$[0_r, 1_g, 0_b]$ and `D -> circle/`$[.6_s]$.

3. **Incorporation.** If chunking fails too, the hearer will incorporate the expression-meaning pair holistically. For instance, if the expression-meaning pair is: "wateva"-$[0_r, .2_g, .2_b, .7_s]$, the rule `S -> wateva/`$[0_r, .2_g, .2_b, .7_s]$ is added to the grammar.

At the end of the induction step, a post-operation called *generalise and merge* is performed to generalise even further and to eliminate redundancies. Redundancies can occur when new rules are created that are basically similar to already existing ones. The above example of chunking, for instance, yielded the construction of the rule `S -> C/rgb D/s` with additional filler rules. This rule, however, is in essence the same as rule R_3 from Table 2. Therefore, this new rule is removed (*merged*) and the non-terminals of the additional filler rules are replaced with the letters `A` and `B`. In addition, given a newly added rule `B -> circle/`$[.6_s]$ and the rules of Table 2, rule R_1 can be chunked (*generalised*) as well, leading to the new rule `A -> grey/`$[.5_r, .5_g, .5_b]$. (Note that the old rules such as R_1 remain in the grammar.)

The language game model is integrated with the iterated learning model. In this model, the population contains N adults and N children. Whenever a GG is played, the speaker is selected from the adult population and the hearer from the child population. After a given number of guessing games, all adults are replaced by the children, and new children enter the population. Such an iteration then repeats.

4 Results

A number of experiments were done with the above model, three of which are reported here. Each condition was carried out both without and with a transmission bottleneck. If a transmission bottleneck was imposed, a set of 60 objects were selected at random from the world of 120 objects at the start of each iteration. These 60 objects were then the objects about which the agents could communicate during that iteration.

Each experiment contained a population of 3 adults and 3 children. The experiments were run for 250 iterations of 6000 guessing games each. At the end of an iteration, the population was tested before the adults were replaced by the children and new children entered the population. In each test phase, 200 contexts were constructed. In every context one object was chosen as topic, and all agents then tried to encode an expression, which in turn all other agents tried to decode. From this phase, various measures were calculated of which compositionality is the measure reported in this paper. Compositionality is calculated as the ratio between the number of compositional rules used (both encoded and decoded) and the total number of utterances produced and interpreted. The testing phase always used the entire world of 120 objects.

[4] Alignments are only allowed if they appear at the start or the end of an expression, and if they are connected.

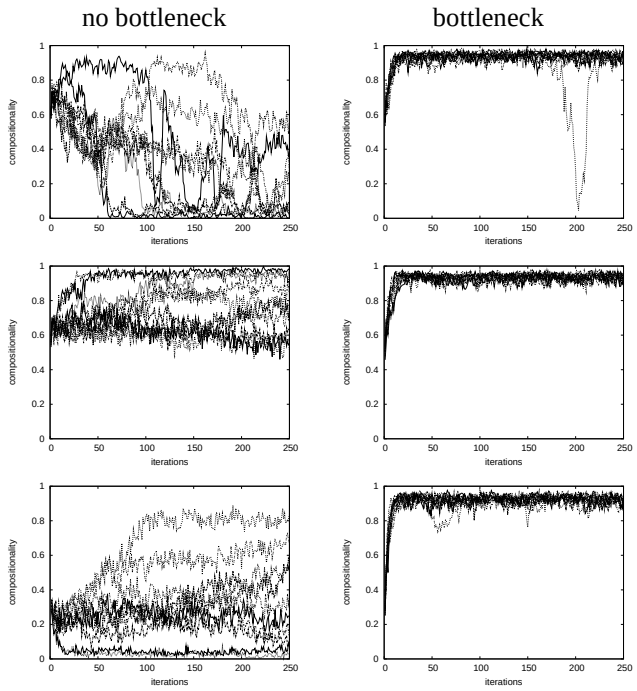


Figure 1: Compositionality as a function of iterations for experiments 1 (top), 2 (middle) and 3 (bottom). All experiments were done both without (left) and with (right) a transmission bottleneck. Each line represents one simulation run.

	no bottleneck		bottleneck	
	2	250	2	250
1	0.74±0.01	0.16±0.11	0.69±0.04	0.93±0.01
2	0.68±0.01	0.76±0.14	0.64±0.04	0.93±0.02
3	0.30±0.03	0.35±0.26	0.48±0.03	0.91±0.02

Table 3: The results of the first three experiments, both without and with bottleneck. For each condition compositionality is given for the end of the 2nd iteration and the end of the 250st iteration.

Experiment 1 The first experiment was done with exactly the same model as described in the previous section. Here all meanings developed from scratch in parallel to the development of the grammar.

Experiment 2 In experiment two, all agents were given a repertoire of categorical features. With these CFs, the agents could form categories right from the start, but they did not have any idea how to combine the different dimensions to form conceptual spaces as the basis of the semantic structures.

Experiment 3 In the third experiment, the semantic structure was predefined. All agents were given CFs as in experiment 2 and an additional constraint that semantic categories (i.e. conceptual spaces) were either colours, shapes or the whole (holistic) meaning.

Figure 1 and Table 3 show the results of these three experi-

ments both without (left) and with (right) a transmission bottleneck. The development of compositionality in experiment 1 is shown in the top graphs. As can be seen, in both cases compositionality emerged to a rather high level in the first few iterations (0.74 ± 0.01 without and 0.69 ± 0.04 with a bottleneck, see Table 3, row 1). After that, compositionality tended to be instable and eventually died away in favour of holistic languages if no bottleneck was present (this was confirmed in simulations run for 500 iterations). With a transmission bottleneck, compositionality rapidly kept rising to a stable value of around 0.93. Only one simulation showed a sudden dramatic decrease in compositionality around iteration 200, but soon recovered to its previous level.

When the agents have predefined CFs, but had to learn the semantic structures (experiment 2, middle graphs), the initial levels of compositionality were slightly lower than in experiment 1. In the case where there was no bottleneck, compositionality remained in most cases at the same level, and in some cases even increased to levels near 1. When there was a bottleneck, compositionality rose in the same way to similar levels as in experiment 1.

In the case where the semantics, including their structures, were predefined (experiment 3, bottom graphs), the results were quite different from the other two experiments. In both cases, the initial levels were substantially lower, although – in contrast to the other experiments – compositionality was lower without a bottleneck (0.30 ± 0.03) than with a bottleneck (0.48 ± 0.03). When no bottleneck was imposed, compositionality increased to 0.8 in one simulation run, it rose in a few other cases, but often it remained at the same level or decreased. When a bottleneck was imposed, the compositionality revealed a similar increase as in the two other experiments with a bottleneck.

5 Discussion

The experiments in this paper investigate the hypothesis that the statistical nature of the semantic structure can explain the rapid development of compositionality and the hypothesis that the instability of compositionality in some circumstances had to do with the ontogenetical development of meaning. The experiments further investigate the effect of imposing a bottleneck on the transmission of language on the development of compositionality.

All experiments reveal that when a bottleneck is imposed, compositionality develops rapidly to a high and stable degree, thus confirming the results achieved earlier by Kirby and others [Brighton, 2002; Kirby, 2002; Smith *et al.*, 2003]. When no bottleneck is imposed, the behaviour of the different experiments is more different from each other, which will be discussed in the remainder of this section.

There are two types of observations: First, when no semantic *structure* is imposed (experiments 1 and 2), high levels of compositionality are achieved within two iterations. Second, when no categorical features are predefined (experiment 1), compositionality is unstable, whereas in other cases, compositionality is either stable (experiment 2) or may emerge at a later stage (experiment 3).

Let me start discussing the first observation. The problems the agents learn is to discover regularities in the semantic space (which reflects the feature space of the objects) in combination with regularities in the signal space. It was hypothesised that the rapid development of compositionality in experiment 1 (and 2) was caused by the statistical properties of the semantic space on one hand and the signal space on the other [Vogt, 2005]. Because signals tend to be short and are constructed from a limited alphabet, the likelihood of finding alignments in this space is high. Since this aspect is not varied, all experiments had the same likelihood.

X	Y	$P_1 \cdot P_2$
r	gbs	0.297
g	rbs	0.200
b	rgs	0.256
rg	bs	0.117
rb	gs	0.144
gb	rs	0.117
rgb	s	0.075

Table 4: Probability of finding co-occurring structures in dimension X in two different games, assuming the values in Y differ. Assuming that x_i can have any possible value in the space of X (indicated by $x_i \in X$) and y_i any value in the space of Y, then $P_1 = P(x_1 \in X \wedge y_1 \in Y)$ and $P_2 = P(x_2 = x_1 \wedge y_2 \in Y \backslash \{y_2\}))$. The values are based on the distribution of feature values, which are not presented in this text.

The chances for finding regularities in the semantic space, however, is much higher when no structure is predefined. In experiment 3, the structure for combining the `rgb` space with the shape `s` space was imposed. Given there are 12 colours and 10 shapes, there are $12 \times 10 = 120$ different ways to form semantic structures of these 120 objects. However, when no structure is imposed, the combinations `r-gbs`, `g-rbs`, `b-rgs`, `rg-bs`, `rb-gs` and `bg-rs` are possible ways to combine the 4 quality dimensions as well. Since, for instance, in the `r` dimension there are 40 different objects (4 colours and 10 shapes) that can have feature value $f_r = 0$. In a rough estimation, the probability P_1 of finding a one of these objects in game is proportional to $\frac{40}{120}$. The probability P_2 of finding *another* object with $f_r = 0$ in another game is $P_2 \propto \frac{39}{120}$. So, the probability of finding a co-occurring structure in the `r` dimension with $f_r = 0$ in two consecutive games is: $P_1 \cdot P_2 \propto 0.108$. Now, if we look at objects of the same colour, e.g., blue (i.e. with $f_r = 0, f_b = 1$ and $f_g = 0$), then the chance of finding a blue object in the first game is $P_1 \propto \frac{10}{120}$ and a different blue object in the second game is $P_2 \propto \frac{9}{120}$, so finding this co-occurrence is $P_1 \cdot P_2 \propto 0.006$.

Table 4 shows the co-occurrence probabilities for all objects involving similar values in one or more dimensions of the colour space. The values shown are the probabilities summed over all possible values in a conceptual space. As the table shows, when considering the entire colour space `rgb`, the co-occurrence probability is much lower than when searching for a co-occurring structure in the `r` space. So clearly, when an agent is able to construct any possible combination of semantic categories as in experiment 1, the

chances of finding co-occurring structures is much higher than when it is restricted only to using `rgb-s` as in experiment 3. Hence, compositionality in experiment 1 emerges faster than in experiment 3.

Interestingly, when there is a bottleneck on the transmission of language and stable compositional systems emerge, the most dominant rules (i.e. the type of rules that were used most frequently) combine the colour space `rgb` and the shape space `s`. When there is no bottleneck, the most frequently used compositional rule combined `r` with `gbs`. This can be understood by realising that when half of the objects are discarded, the co-occurrence probability in the `rgb-s` structure is more or less maintained, while the other probabilities are affected differently in each iteration. This is because the distribution of colours and shapes is less skewed than the distributions in the other dimensions. (Recall that each iteration a different set of objects is selected to learn from.) As a result, the `rgb-s` structure can be learnt most reliably.

The second observation that compositionality is unstable in the absence of bottlenecks when categorical features are not predefined can be explained by realising the consequences of the gradual development of CFs. The foremost difference is that at the start of an iteration adults have a well developed set of CFs, while the children have none. During development, the children gradually acquire a similar set of CFs, but initially these CFs are overgeneralisations of the final categories. Nevertheless, the children may – in certain situations – categorise some objects distinctively, thus successfully finishing a discrimination game.

As a consequence, these children can successfully learn the words associated with these categories. However, since the CFs are overgeneralisations of the adults' CFs, the children's CFs may be associated with more than one word. This is even more the case when realising that different adults can have different words to name a CF. When the children acquire more fine grained CFs, the words may become associated with different CFs than was the case for the adults. This way, the words can drift through the conceptual space. When the language is compositional, the movement in one semantic category affects a larger part of the language than it would when the language is holistic. Hence, holistic languages are more stable and thus easier to learn. When the CFs are predefined, there is less or no reason for overgeneralisations. This, in turn, allows children to acquire any compositional structure of their adults more reliably, which – in part – explains why compositionality is more stable (experiment 2) or even emerges (experiment 3). The results of experiment 3 are very similar to those reported in [Smith *et al.*, 2003], while the results of experiment 2 is highly dissimilar. It is yet unclear whether the predefined semantics of Smith et al. are more similar to the ones used in experiment 2 or 3, though the results suggest that the semantics are more similar to the one used in experiment 3.

Concluding, both hypotheses tested in this paper are confirmed. In addition, the major findings of [Brighton, 2002; Kirby, 2002; Smith *et al.*, 2003] that compositionality emerges under the pressure of a transmission bottleneck are confirmed in all tested conditions. Hence, the assumption they made on predefining semantics to simplify their models appears valid. However, predefining the meanings – either with or without structure – can affect the results achieved in simulations of language evolution to a great extent, so one should be very careful in interpreting results achieved with such ungrounded models. This does not mean that the current grounded model on language evolution is sufficiently realistic to make hard claims regarding the evolution of language. However, the model is more realistic than ungrounded models in that in this model children do not have the same full-fletched semantic structures as adults.

Acknowledgements

This work is sponsored by a EC Marie Curie Fellowship and a VENI grant provided by the Netherlands Organisation for Scientific Research (NWO). The author thanks Andrew and Kenny Smith and three anonymous reviewers for comments on earlier versions of this manuscript.

References

[Brighton, 2002] H. Brighton. Compositional syntax from cultural transmission. *Artificial Life*, 8(1):25–54, 2002.

[Cangelosi and Parisi, 2002] A. Cangelosi and D. Parisi, editors. *Simulating the Evolution of Language*. Springer, London, 2002.

[Gärdenfors, 2000] P. Gärdenfors. *Conceptual Spaces*. Bradford Books, MIT Press, 2000.

[Kirby, 2002] S. Kirby. Learning, bottlenecks and the evolution of recursive syntax. In Ted Briscoe, editor, *Linguistic Evolution through Language Acquisition: Formal and Computational Models*. Cambridge University Press, 2002.

[Smith *et al.*, 2003] K. Smith, H. Brighton, and S. Kirby. Complex systems in language evolution: the cultural emergence of compositional structure. *Advances in Complex Systems*, 6(4):537–558, 2003.

[Steels, 1997] L. Steels. The synthetic modeling of language origins. *Evolution of Communication*, 1(1):1–34, 1997.

[Steels, 2004] L. Steels. Constructivist development of grounded construction grammars. In W. Daelemans, editor, *Proceedings Annual Meeting of Association for Computational Linguistics*, 2004.

[Vogt, 2003] P. Vogt. THSim v3.2: The Talking Heads simulation tool. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, editors, *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*. Springer Verlag Berlin, Heidelberg, 2003.

[Vogt, 2005] P. Vogt. The emergence of compositional structures in perceptually grounded language games. Submitted for publication, 2005.

Automatic Semantic Role Labeling for Chinese Verbs

Nianwen Xue, Martha Palmer
CIS Department
University of Pennsylvania
Philadelphia, PA 19104, USA
{xueniwen,mpalmer}@linc.cis.upenn.edu

Abstract

Recent years have seen a revived interst in semantic parsing by applying statistical and machine-learning methods to semantically annotated corpora such as the FrameNet and the Proposition Bank. So far much of the research has been focused on English due to the lack of semantically annotated resources in other languages. In this paper, we report first results on semantic role labeling using a pre-release version of the Chinese Proposition Bank. Since the Chinese Proposition Bank is superimposed on top of the Chinese Treebank, i.e., the semantic role labels are assigned to constituents in a treebank parse tree, we start by reporting results on experiments using the hand-crafted parses in the treebank. This will give us a measure of the extent to which the semantic role labels can be bootstrapped from the syntactic annotation in the treebank. We will then report experiments using a fully automatic Chinese parser that integrates word segmentation, POS-tagging and parsing. This will gauge how successful semantic role labeling can be done for Chinese in realistic situations. We show that our results using hand-crafted parses are slightly higher than the results reported for the state-of-the-art semantic role labeling systems for English using the Penn English Proposition Bank data, even though the Chinese Proposition Bank is smaller in size. When an automatic parser is used, however, the accuracy of our system is much lower than the English state-of-the-art. This reveals an interesting cross-linguistic difference between the two languages, which we attempt to explain. We also describe a method to induce verb classes from the Proposition Bank "frame files" that can be used to improve semantic role labeling.

1 Introduction

Recent efforts on semantic annotation have made it possible to train domain-independent semantic systems [Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Hacioglu *et al.*, 2003; Sun and Jurafsky, 2004; Pradhan *et al.*, 2004; Xue and Palmer, 2004]. Most of the semantic annotation projects focus on the predicate-argument structure, which represents a predicate and a number of arguments that are expected of this predicate. Generally each expected argument is assigned a label that marks the role this argument plays in relation to its predicate. It is in the level of generalization these role labels represent that the various annotation efforts differ. The most general are a limited set of roles such as agent and theme that are globally meaningful [Chen *et al.*, 2004]. The role labels used in FrameNet [Baker *et al.*, 1998] are less general in that they are meaningful only with respect to a specific situation, more formally known as a semantic *frame*. For example, the label *Byr* is only meaningful in the "Commercial_transaction" frame. One reflection of this reduced generality is that it is realized with a small class of predicates that indicate transaction, e.g. *purchase*, *rent*. The least general are the labels used in the Propbank annotation. The Propbanks [Palmer *et al.*, 2005; Xue and Palmer, 2003] use predicate-specific labels *ARG0*, *ARG1*, ... *ARGn* for arguments and *ARGM* combined with a secondary tag to mark adjunct-like elements. The secondary tags indicate types of adjuncts and represent generalizations across all verbs.

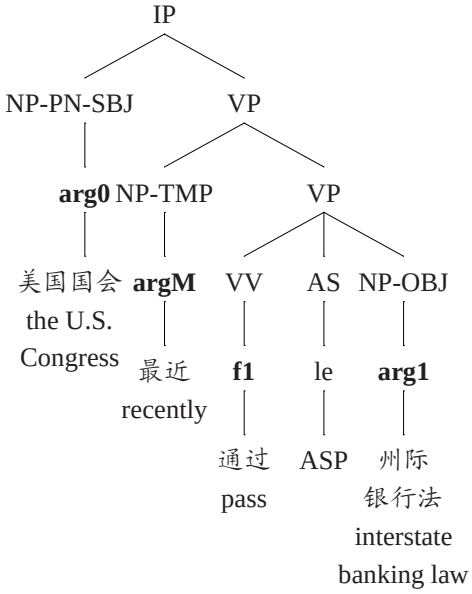
The predicate-specific approach of the Propbank annotation builds a solid foundation for making high-level generalizations in a bottom-up manner, if broader generalizations are needed. There is generally a straightforward mapping from the numbered role labels to more general roles such as agent and theme. It is much harder to derive these semantic concepts from syntactic representation because an argument may not always be realized, and when it is, it may not always be realized in the same syntactic position as a result of syntactic alternations [Levin, 1993], etc.. In addition, different senses of a verb take different sets of arguments that demonstrate different syntactic patterns. Thus, predicate-argument structure recognition at this level represents a crucial leap towards proper representation of semantic structure from the syntactic structure.

So far, most of the work on automatic semantic role labeling has been devoted to English due to the lack of resources in other languages. In this paper we present first results on the Chinese Proposition Bank [Xue and Palmer, 2003]. While Sun and Jurafsky [2004] did preliminary work on Chinese semantic role labeling using 10 selected verbs, our experi-

ments will be on the all the verbs in the Chinese Proposition Bank. We discuss the linguistic annotation of the Chinese Proposition Bank in Section 2. We describe our experiments using hand-crafted parses in Section 3. In particular, we describe a method of automatically inducing verb classes and use this information as features in semantic role labeling. In Section 4, we describe our experiments using a fully automatic Character-based parser. Section 5 discusses our experimental results and the facilitating and exacerbating factors in Chinese semantic role labeling in comparison with English. Section 6 concludes this paper.

2 The Chinese Proposition Bank

In this section we briefly examine the annotation scheme of the Penn Chinese Propbank [Xue and Palmer, 2003]. The Chinese Propbank is based on the Chinese Treebank [Xue *et al.*, To apear], which is a 500K-word corpus annotated with syntactic structures. The semantic annotation in the Propbank is added to the appropriate constituents in a syntactic tree. This is illustrated in Table 1. The newly added semantic role labels are in bold. The predicate 通过"pass" in Table 1 has two numbered arguments, *ARG0* 美国国会"U.S. Congress" and *ARG1* 州际银行法"interstate banking law". It also has a temporal semantic adjunct, *ARGM-TMP* 最近"recently". In addition to the arguments and adjuncts, the predicate 通过"pass" is also annotated with a *frameset* identifier, *f1*, which uniquely identifies the frameset of this verb instance.



The U.S. Congress recently passed the interstate banking law.

Table 1: A treebank tree annotated with semantic role labels and frameset ID

The frameset represents a major sense defined by the set of arguments a predicate takes. For example, (1) and (2) represents two different framesets of the predicate 通过"pass" because its arguments in those two cases are different. The two arguments for frameset *f1* can be described as "a legislative body" and "the law, regulations, etc. that are passed", while the two arguments for frameset *f2* are "an entity in motion" and "a location the entity passes through". Note that the two instances of 通过"pass" within (1) or (2) do not belong to different framesets since they share the same set of arguments, even though the arguments may be realized in different places, if realized at all. In (1b), the "legislative body" argument is dropped and the "law" argument is realized in the subject position instead of the object position (1a).

(1) a. [**arg0** 美国 国会] 最近 [**f1** 通过] 了 [**arg1**
the U.S. Congress recently pass ASP
州际 银行法] 。
interstate banking law .
"The U.S. Congress passed the interstate banking law recently."

b. [**arg1** 州际 银行法] 最近 [**f1** 通过] 了 。
interstate banking law recently pass ASP .
"The interstate banking law passed recently."

(2) a. [**arg0** 火车] 正在[**f2** 通过] [**arg1** 隧道] 。
train now pass tunnel .
"The train is passing through the tunnel."

b. [**arg1** 火车] 正在[**f2** 通过] 。
train 正在 pass .
"The train is passing through."

The task of semantic role labeling is to use the role labels as categories and classify each argument as belonging to one of these categories.

3 Semantic role tagging with hand-crafted parses

In this section we describe a system that does semantic role labeling using Gold Standard parses in the Chinese Treebank as input. To be used in real-world natural language applications, a semantic role tagger has to use automatically produced constituent boundaries either from a parser or some other means, but experiments with Gold Standard input will help us evaluate how much of a challenge it is to map a syntactic representation to a semantic representation, which may very well vary from language to language.

3.1 Classifier

For our purposes here, we use a Maximum Entropy classifier with a tunable Gaussian prior in the Mallet Toolkit[1]. The Maximum Entropy classifier does multi-category classification and thus can be straightforwardly applied to the problem here. The classifier can be tuned to minimize overfitting by adjusting the Gaussian prior.

3.2 Architecture

The Propbank annotation is predicate-centered in that sense that only constituents that are semantic arguments and adjuncts (in a loose sense) to a predicate are annotated. Since

[1] http://mallet.cs.umass.edu

the treebank sentences are very long and generally contain several verbs, the majority of the constituents are not related to the predicate in question. One obvious strategy is to assign a *NULL label* to the unannotated constituents, but it is a known fact that when negative samples (*NULL* constituents in this case) overwhelm positive samples (constituents that are actually annotated), the classifier will be heavily biased towards *NULL* constituents. Most systems find a way to filter out some of the negative samples to make the classification task more balanced. For example, [Hacioglu *et al.*, 2003] uses a two-stage architecture where a binary classifier is first used to label all the constituents as either *NULL* or *NON-NULL*, and then a multi-category classifier is run on the *NON-NULL* constituents to assign the semantic role labels. Xue and Palmer [2004] uses a three-stage architecture in which some negative samples are first filtered out with heuristics that exploit the syntactic structures represented in a parse tree. A binary classifier is then applied to further separate the positive samples from the negative samples and finally a multiply-category classified is applied to assign the semantic role labels to the positive samples. In our experiments we adopt the same three-stage strategy described in [Xue and Palmer, 2004]. First the heuristic algorithm is adapted to take advantage of the syntactic structures in the Penn Chinese Treebank, which uses a different set of syntactic categories. The algorithm starts from the predicate that anchors the annotation, and first collects all the syntactic complements of this predicate, which are represented as sisters to the predicate. It then iteratively moves one level up to the parent of the current node till it reaches the appropriate top-level node. This top-level node can be an IP, CP, or NP depending on what their parent nodes are. For example, if the parent node of a CP is an NP, then this CP is very likely a relative clause and the head of this NP is a possible argument to the predicate, via a trace. In this case the algorithm will collect all sisters of this CP. At each level, the system has a procedure to determine whether that level is a coordination structure or a modification structure. The system only considers a constituent to be a potential candidate if it is an adjunct to the current node. Punctuation marks at all levels are skipped. It is worth pointing out that the functional tags and traces are not used to determine the candidates to allow for fair comparison with results from using automatic parses, where they are not available. After this initial procedure, a binary classifier is applied to distinguish the positive samples from the negative samples. A lower threshold is used for positive samples then negative samples to maximize the recall so that we can pass along as many as positive samples as possible to the final stage, which is the multi-category classification.

3.3 Data

In all our experiments we use a pre-release version of the Chinese Proposition Bank. This version of the Chinese Proposition Bank [Xue and Palmer, 2003] consists of stand-off annotation on the first 760 articles (`chtb_001.fid` to `chtb_931.fid`) of the Penn Chinese Treebank[2]. This chunk of the data has 250K words and 10,364 sentences. The total number of verb types in this chunk of the data is 4,854.[3]. Following the convention of the English semantic role labeling experiments, we divide the training and test data by the number of articles, not by the verb instances. This pretty much guarantees that there will be unseen verbs in the test data. For all our experiments on semantic role labeling, 661 files (`chtb_100.fid` to `chtb_931.fid`) are used as training data and the other 99 files (`chtb_001.fid` to `chtb_099.fid`) are held out as test data. However, our parser is trained on all the data in most current version of the Penn Chinese Treebank except for the test data that has been set aside. That is, in addition to the training data for the semantic role labeling experiments, it also uses the rest of the treebank which has not been propbanked.

3.4 Features

One characteristic of feature-based semantic role modeling is that the feature space is generally large. This is in contrast with the low-level NLP tasks such as POS tagging, which generally have a small feature space. A wide range of features have been shown to be useful in previous work on semantic role labeling [Gildea and Jurafsky, 2002; Pradhan *et al.*, 2004; Xue and Palmer, 2004] and we suspect that many more will be tested before they will settle down with a core set of features. In their preliminary work on Chinese semantic role labeling, Sun and Jurafsky [2004] has successfully adapted a number of the features to Chinese. In our experiments more features that have been described in recent work on English semantic role labeling are adapted to Chinese. We briefly discuss these features and explain at an intuitive level why these are useful features for semantic role labeling where necessary. Our focus, however, will be on how verb classes can be induced from "frame files" from the Penn Proposition Bank and be used as features, which we will discuss in greater detail in the next subsection. The features that we use are listed below:

- *Position*: The position is defined in relation to the predicate verb and the values are *before* and *after*.
- *path*: The path between the constituent in focus and the predicate.
- *Head word and its part of speech*: The head word and its part-of-speech is often a good indicator of the semantic role label of a constituent.
- *Predicate*: The verb itself.
- *subcat frame*: The rule that expands the parent of the verb.
- *Phrase type*: The syntactic category of the constituent in focus.
- *First and last word of the constituent in focus*
- *Phrase type of the sibling to the left*
- *syntactic frame*: The syntactic frame consists of the NPs that surround the predicate verb. This feature is defined

[2]The most current version (CTB5.0) of the Penn Chinese Treebank has 507K words, 825K Chinese characters, 18,716 sentences and 890 articles.

[3]These include the so-called stative verbs, which roughly correspond to adjectives in English.

by the position of the constituent in focus in relation to this syntactic frame [Xue and Palmer, 2004].

- *Combination features*: predicate-head word combination, predicate-phrase type combination.

The reason why the position feature is useful is obvious since constituents receiving a particular semantic role label may occur in some typical positions. For example, the majority of the adjuncts, *ARGM*s occur before the verb in Chinese. The path feature, defined as the route from the constituent in focus to the predicate, represents a more "fine-grained" position. While the values for the simple position feature are just "BEFORE" or "AFTER", the values for the path feature can represent syntactic notions like "subject" or "object". For example, a subject may be represented as "NP↑IP↓VP↓VV" and an object may be represented as "VV↑VP↓NP". Intuitively the path feature is more informative than simple position features but they are also sparse because they are more specific.

The head word and its part-of-speech are clearly informative for semantic role labeling. For example, a noun phrase headed by "jintian/today" is very likely to be a temporal element. So is a prepositional phrase with the head word "zai/at". However, for propositional phrases, the preposition is not always the most informative element. Sometimes the head word of the NP complement is more predictive of the semantic category. For example, in the prepositional phrase "zai/at beijing/Beijing", the NP head "beijing/Beijing" is more telling of the fact that it indicates a location. So for prepositional phrases we use both the preposition and the head noun as features in our system. As has been discussed in Sun and Jurafsky, the head word feature also tends to be sparse, especially given the smaller size of the Penn Chinese Treebank. The chance of seeing a word in the test data that also occurs in the training data is small. The POS tag serves as one form of backoff: constituents headed by words that have the same part-of-speech are likely to receive the same semantic role labels as well.

The first-word-in-the-constituent and the phrase label of the left sibling features are from [Pradhan *et al.*, 2004] and the interested reader is referred to their work for an explanation of why these are useful features. We also implemented features that are described in Xue and Palmer [2004]. These include the syntactic template features, predicate head word combination features and predicate phrase type combination features. For details of these features the reader is referred to [Xue and Palmer, 2004].

3.5 Using verb classes to improve semantic role labeling

With the current experiment setup, as is also the case in most of the work on semantic role labeling, training data and testing data are not divided by verb instances but by the number of articles. As a result, it is expected that the verb instances are not evenly divided. It is even possible that some verbs can only be found in the training data and other verbs can only be found in the test data. By our count, there are 4,666 verb types in the training data and 1,067 verb types in the test data. 188 verb types that occur in the test data are absent from the training data and conversely, 3,787 verb types are in the training data only. Since the semantic role labels are defined with regard to the individual verbs, this can be a real problem since the model learned in the training process does not optimally fit with the test data if different verbs are involved. Fortunately, many verbs have similar argument structures and therefore are annotated with similar semantic role labels in the Chinese Proposition Bank. For examples, verbs like "jiada/enlarge", "jiaju/make more drastic", "jiakuai/accelerate", "jiaqiang/strengthen", "jiashen/deepen", "jiasu/accelerate", "jiazhong/give more weight" "jiagao/make higher" all take two arguments, a theme that undergoes a change of state and an external force or agent that brings about the change of state. These verbs are uniformly annotated and they all have two numbered arguments *arg0* and *arg1* with *arg0* denoting the cause and *arg1* denoting the theme. It would make sense to group these verbs together into a class and use this information as features in the semantic role labeling task. Having a membership in a particular class says something about the predicate-argument structure of a verb and when a verb is absent in the training data, which is the familiar sparse data problem, the class information may tell the system how to label the semantic roles of the verbs belonging to a particular class.

Although to our knowledge no such classification exists for Chinese verbs based on the predicate-argument structures, a rough classification can be automatically derived from the frames files, which are created to guide the Propbank annotation. We classified the verbs along three dimensions, the number of arguments, the number of framesets and selected syntactic alternations.

Number of arguments: Verbs in the Chinese Proposition Bank can have one to five arguments, with the majority of them having one, two or three arguments. Verbs with zero argument are auxiliary verbs [4] like "bi/will", "deyi/be able to", "gai/should", "gan/dare", "ke/may", "ken/be willing to", "neng/can", "nenggou/can", "xu/must", "yingdang/should" and some other light verbs. Verbs that have five arguments are change of state verbs like "yanchang/lengthen", "suoduan/shorten" "jiangdi/lower", "tigao/increase", "kuoda/enlarge", "suoxiao/make smaller". These verbs generally take as arguments a theme that undergoes the change of state, the original state, the new state, the range of the change and the cause or agent that brings about the change.

Number of framesets: A frameset roughly corresponds to a major sense. This information is used because it is common that the different framesets of a verb can have different number of arguments. For example, verbs like "pingheng/balance" can be used either as a non-stative verb, in which case it means "balance", or a stative verb, in which case it means "balanced". When it is used an a non-stative verb, it takes two arguments, the thing or situation that is balanced and the balancer, the entity that

[4] One could say that the argument of the auxiliary verbs is the entire proposition, but in this phase of the Chinese Proposition Bank, auxiliary verbs are not annotated.

maintains the balance. When it is used as a static verb, obviously it only takes a single argument.

Syntactic alternations: We also represent certain type of syntactic alternations. One salient type of syntactic alternation is the well-known "subject of intransitive / object of the transitive" alternation described in detail in Levin [Levin, 1993]. Chinese verbs that demonstrate this alternation pattern include "chuban/publish". For example, "zhe/this ben/CL shu/book" plays the same semantic role even if it is the subject in "zhe/this ben/CL shu/book chuban/publish le/AS" and the object in "zhe/this jia/CL chubanshe/publishing house chuban/publish le/ASP zhe/this ben/CL shu/book".

Thus each verb will belong to a class with a symbol representing each of the three dimensions. For example, a given verb may belong to the class "C1C2a", which means that this verb has two framesets, with the first frameset having one argument and the second having two arguments. The "a" in the second frameset represents a type of syntactic alternation. 40 classes are derived in this manner.

Such a classification scheme will undoubtedly prove to be linguistically unsophisticated. Verbs that have the same number of arguments may have different types of arguments, and the current classifications system do not pick up these distinctions. However, our experiments show that such a simple classification, when used as features, significantly improves the semantic role labeling task.

4 Using automatic parses

Previous work [Sun and Jurafsky, 2004] on Chinese semantic role labeling uses a parser that assume correct (hand-crafted) segmentation. As word segmentation is a very challenging problem that has attracted a large body of research by itself, it is still unclear how well semantic role tagging in Chinese can be performed in realistic situations. In our experiments, we use a fully automatic parser that integrates segmentation, POS tagging and parsing. Our parser is similar to [Luo, 2003]. The parser is trained on CTB5.0, using the training data described in the previous section. Tested on the held-out test data, the labeled precision and recall are 81.83% and 82.91% respectively for all sentences. The results are comparable with those reported in Luo [Luo, 2003], but they cannot be directly compared with most of the results reported in the literature, where correct segmentation is assumed. In addition, in order to account for the differences in segmentation, each character has to be treated as a leaf of the parse tree. This is in contrast with word-based parsers where words are terminals. Since semantic role tagging is performed on the output of the parser, only constituents in the parse tree are candidates. If there is no constituent in the parse tree that shares the same text span with an argument in the manual annotation, the system cannot possibly get a correct annotation. In other words, the best the system can do is to correctly label all arguments that have a constituent with the same text span in the parse tree.

5 Results and Discussion

The results of the semantic role labeling are presented in Table 2. The verb class information, when used as features, improves the semantic role labeling accuracy by about 1 percent when the Gold Standard parses are used. When the parser is used, however, the improvement is insignificant. The 93.9 percent that our system achieved using hand-crafted parses is fairly high considering the fact that the state-of-the-art semantic role labeling systems trained on the English Proposition Bank [Palmer *et al.*, 2005] is around 93 percent [Pradhan *et al.*, 2004; Xue and Palmer, 2004] and the English Proposition Bank is a much larger corpus, with 1 million words. There are several facilitating factors for Chinese semantic role labeling when hand-craft parses are provided as input. First of all, Chinese verbs appear to be less polysemous, at least the ones that occur in the Penn Chinese Treebank. Of the 4854 verbs in this version of the Chinese Proposition bank, only 62 verbs have 3 or more framesets. In contrast, 294 verbs out of the 3635 verbs in the Penn English Proposition Bank has 3 or more framesets. When a verb is less polysemous, the arguments of the verb tend to be realized in a more uniform manner in syntax. As a result, the argument labels are easier to predict from their structure in a correct parse tree. Chinese seems to compensate this fact by using a larger number of verbs. This becomes obvious when we consider the fact that 4854 verbs are from just 250K words and the 3635 verbs in the English Proposition Bank is from one million words! A related fact is that adjectives in Chinese are traditionally counted as verbs and they generally have only one argument with a much simpler syntactic realization.

experiments	class	p (%)	r(%)	f(%)
known constituents (Gold)	no	n/a	n/a	92.7
known constituents (Gold)	yes	n/a	n/a	93.9
unknown constituents (Gold)	no	90.4	90.3	90.3
unknown constituents (Gold)	yes	91.4	91.1	91.3
unknown constituents (parser)	no	66.1	57.0	61.2
unknown constituents (parser)	yes	67.0	56.4	61.3

Table 2: Results

We also believe that a more subtle explanation for the higher semantic role labeling accuracy given the annotation of the Chinese Treebank is the fact that the Chinese Treebank has richer structure. By using less flat structures and more hierarchical structures, the Chinese Treebank resolves some of the attachment ambiguities that are useful for semantic role labeling. For example, the complement and adjunct in a VP in the Chinese Treebank are attached in different syntactic configurations with regard to the verb. Since complements are generally numbered arguments and adjuncts are generally *ARGM*s, the semantic role labeler can take advantage of this fact it tries to determine when a constituent is a numbered argument or an adjunct.

This apparent advantage in Chinese semantic role labeling is diminished when an automatic parser is used. First of all, the hierarchical structures in the hand-craft parses that aid semantic role labeling are hard to recover with an automatic parser. Resolving the many attachment ambiguities caused by the hierarchical structures in language is the one of the most difficult problems in parsing literature. Parsing Chinese in a realistic scenario is especially difficult given that word seg-

mentation must be performed as a prerequisite. Chinese word segmentation is a non-trivial problem that has been the topic of a lot of research in the Chinese language processing community. It is still a very much unsolved problem, with the current state-of-the-art accuracy being around 96 percent [Sproat and Emerson, 2003]. This is in contrast with the much less difficult problem of English tokenization. The segmentation errors are very difficult to recover in later parsing stages. Another property of Chinese that exacerbates parsing is the lack of formal clues. For example, determining whether a word is a verb or noun in Chinese can be difficult because Chinese does not have a rich inflection system to aid POS tagging.

6 Conclusions and Future Work

We presented first experimental results on Chinese semantic role labeling using a pre-release version of the Chinese Proposition Bank. We show that verb classes, induced from the predicate-argument information in the "frame files", helps semantic role labeling. We also show that given Gold Standard parses, Chinese semantic role labeling can be performed with considerable accuracy. In fact, even though the Chinese Proposition Bank is a smaller corpus the English Proposition Bank, we achieved results that are slightly higher than the state-of-the-art English semantic role labeling systems. On the other hand, even though our experiments using a fully automatic parser yields promising results, the accuracy is much lower than the English state-of-the-art. We suggest that the lower accuracy when an automatic parser is used is due to the fact that determining the argument boundaries in Chinese using a parser is a very challenging task because word segmentation is a prerequisite and word segmentation errors are hard to recover in the parsing stage. The lack of formal clues such as inflection morphology also makes POS tagging a very difficult task. Since parsing assumes accurate word segmentation and POS tagging, word segmentation and POS tagging errors naturally translate into parsing errors. Once these bumps are overcome, however, semantic analysis actually gets easier, compared with English, due to the lower degree of polysemy.

Naturally there is a lot to be done in Chinese semantic role labeling, with the first order of business being to improve Chinese parsing. We also need to conduct more experiments with the features to figure out which features are most useful for Chinese.

Acknowledgements

We would like to thank Zixin Jiang, Meiyu Chang and Gang Chen for their annotation efforts. This work is funded by the NSF ITR via grant 130-1303-4-541984-XXXX-2000-1070.

References

[Baker *et al.*, 1998] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of COLING/ACL*, pages 86–90, Montreal, Canada, 1998.

[Chen *et al.*, 2004] Keh-Jiann Chen, Chu-Ren Huang, Feng-Yi Chen, Chi-Ching Luo, Ming-Chung Chang, and Chao-Jan Chen. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Anne Abeillé, editor, *Building and Using Parsed Corpora*. Kluwer, 2004.

[Gildea and Jurafsky, 2002] D. Gildea and D. Jurafsky. Automatic labeling for semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[Gildea and Palmer, 2002] Dan Gildea and Martha Palmer. The Necessity of Parsing for Predicate Argument Recognition. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, Philadelphia, PA, 2002.

[Hacioglu *et al.*, 2003] Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Daniel Jurafsky. Shallow Semantic Parsing Using Support Vector Machines. Technical Report CSLR-2003-1, Center for Spoken Language Research at the University of Colorado, 2003.

[Levin, 1993] Beth Levin. *English Verbs and Alternations: A Preliminary Investigation*. Chicago: The Unversity of Chicago Press, 1993.

[Luo, 2003] Xiaoqiang Luo. A Maximum Entropy Chinese Character-Based Parser. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, Sapporo, Japan, 2003.

[Palmer *et al.*, 2005] Martha Palmer, Dan Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1), 2005.

[Pradhan *et al.*, 2004] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. Shallow Semantic Parsing Using Support Vector Machines. In *Proceedings of NAACL-HLT 2004*, Boston, Mass., 2004.

[Sproat and Emerson, 2003] Richard Sproat and Tom Emerson. The First International Chinese Word Segmentation Bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, Sapporo, Japan, 2003.

[Sun and Jurafsky, 2004] Honglin Sun and Daniel Jurafsky. Shallow semantic parsing of chinese. In *Proceedings of NAACL 2004*, Boston, USA, 2004.

[Xue and Palmer, 2003] Nianwen Xue and Martha Palmer. Annotating the Propositions in the Penn Chinese Treebank. In *The Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing*, Sapporo, Japan, 2003.

[Xue and Palmer, 2004] Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, 2004.

[Xue *et al.*, To apear] Nianwen Xue, Fei Xia, Fu dong Chiou, and Martha Palmer. The Penn Chinese Treebank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, To apear.

Extraction of Hierarchies Based on Inclusion of Co-occurring Words with Frequency Information

Eiko Yamamoto Kyoko Kanzaki Hitoshi Isahara
Computational Linguistics Group
National Institute of Information and Communications Technology,
3-5 Hikari-dai, Seika-cho, Souraku-gun, Kyoto 619-0289, Japan
eiko@nict.go.jp kanzaki@nict.go.jp isahara@nict.go.jp

Abstract

In this paper, we propose a method of automatically extracting word hierarchies based on the inclusion relations of word appearance patterns in corpora. We applied the complementary similarity measure (CSM) to determine a hierarchical structure of word meanings. The CSM is a similarity measure developed for recognizing degraded machine-printed text. There are CSMs for both binary and gray-scale images. The CSM for binary images has been applied to estimate one-to-many relations, such as superordinate-subordinate relations, and to extract word hierarchies. However, the CSM for gray-scale images has not been applied to natural language processing. Here, we apply the latter to extract word hierarchies from corpora. To do this, we used frequency information for co-occurring words, which is not considered when using the CSM for binary images. We compared our hierarchies with those obtained using the CSM for binary images, and evaluated them by measuring their degree of agreement with the EDR electronic dictionary.

1 Introduction

The hierarchical relations of words are useful language resources. Progress is being made in lexical database research, notably with hierarchical semantic lexical databases such as WordNet [Miller *et al.*, 1990] and the EDR electronic dictionary [1995], which are used for natural language processing (NLP) research worldwide. These databases are essential for enabling computers, and even humans, to fully understand the meanings of words because the lexicon is the origin of language understanding and generation. In current thesauri with hierarchical relations, words are categorized manually and classified in a top-down manner based on human intuition. This may be a practical way of developing a lexical database for NLP. However, these word hierarchies tend to vary greatly depending on the lexicographer. In fact, each thesaurus includes original hierarchical relations that differ from those in other thesauri. There is often disagreement as to the make-up of a hierarchy. In addition, hierarchical relations based on different data may be needed depending on the user. A statistical method of creating hierarchies from corpora would thus be useful. We therefore attempted to automatically extract hierarchies most suited to the information that a user handles. To do this, we extract hypernym-hyponym relations between two words from corpora and then build hierarchies by connecting these relations. As the initial task, we attempted to extract hierarchies of abstract nouns that co-occur with adjectives in Japanese.

In finding word hierarchies in corpora, it is usual to use patterns, such as "a part of," "is a," "such as," or "and," obtained from the corpora [Hearst, 1992; Berland and Charniak, 1999; Caraballo, 1999]. Methods for extracting hypernyms of entry words from definition sentences in dictionaries [Tsurumaru *et al.*, 1986; Shoutsu *et al.*, 2003] and methods using collocations retrieved from corpora [Nakayama and Matsumoto, 1997] have been described previously. A hybrid method that uses both dictionaries and the dependency relations of words taken from a corpus has also been reported [Matsumoto *et al.*, 1996]. Recently, a similarity measure developed for recognizing degraded machine-printed text [Hagita and Sawaki, 1995] was used to estimate one-to-many relations, such as that of superordinate-subordinate, from a corpus [Yamamoto and Umemura, 2002]. This measure is called the complementary similarity measure (CSM) for binary images and indicates the degree of the inclusion relation between two binary vectors. In that study, each binary vector corresponds to certain appearance patterns for each term in a corpus. The CSM for binary images has also been applied to extract word hierarchies from corpora [Yamamoto *et al.*, 2004] and to trace the distribution of abstract nouns on a self-organizing semantic map [Kanzaki *et al.*, 2004].

In the experiments described in this paper, we attempted to use the CSM for gray-scale images [Sawaki *et al.*, 1997] to extract hypernym-hyponym relations between two words. Specifically, we used not only binary vectors with elements of 0 or 1, but also vectors consisting of weights based on the frequencies of co-occurring words in corpora. We compared the hierarchies extracted using the CSM for gray-scale images with those extracted using the CSM for binary images. Finally, to verify the effectiveness of our approach, we

evaluated our hierarchies by measuring the degree to which they agreed with the EDR electronic dictionary.

2 Experimental Data

A good deal of linguistic research has focused on the syntactic and semantic functions of abstract nouns [Nemoto, 1969; Takahashi, 1975; Kanzaki *et al.*, 2003]. In the example, "*Yagi* (goat) *wa seishitsu* (nature) *ga otonashii* (gentle) (Goats have a gentle nature)," Takahashi [1975] recognized that the abstract noun "*seishitsu* (nature)" is a hypernym of the attribute expressed by the predicative adjective "*otonashi* (gentle)." To classify adjectives on the basis on these functions, Kanzaki *et al.* [2003] defined such abstract nouns that co-occur with adjectives as hypernyms of these adjectives. They produced linguistic data for their research by automatically extracting the co-occurrence relations between abstract nouns and adjectives from corpora.

In our experiment, we used the same corpora and abstract nouns as Kanzaki *et al.*. The corpora consist of 100 novels, 100 essays, and 42 years' worth of newspaper articles, including 11 years of the Mainichi Shinbun, 10 years of the Nihon Keizai Shinbun, 7 years of the Sangyoukinyuuryuutsuu Shinbun, and 14 years of the Yomiuri Shinbun. The abstract nouns were selected from 2 years' worth of the Mainichi Shinbun newspaper articles by Kanzaki *et al.*.

However, we produced our experimental data using only the sentences in the corpora that could be parsed using KNP, which is a Japanese parser developed at Kyoto University. The parsed data consisted of 354 abstract nouns and 6407 adjectives and included the following examples (the number after each adjective is the frequency which the abstract noun co-occurs with the adjective):

OMOI (feeling): *ureshii* (glad) 25, *kanashii* (sad) 396, *shiawasena* (happy) 6, …

KIMOCHI (thought): *ureshii* (glad) 204, *tanoshii* (pleasant) 87, *hokorashii* (proud) 40, …

KANTEN (viewpoint): *igakutekina* (medical) 9, *rekishitekina* (historical) 17, ...

3 Complementary Similarity Measure

As mentioned above, we used the complementary similarity measure (CSM) to estimate the hierarchical relations between word pairs. The CSM was developed for recognizing degraded machine-printed text [Hagita and Sawaki, 1995; Sawaki *et al.*, 1997]. There are two kinds of CSMs, one for binary images and one for gray-scale images.

3.1 Complementary similarity measure for binary images

The CSM for binary images was developed as a character recognition measure for binary images and is designed to be robust against heavy noise or graphical design [Hagita and Sawaki, 1995]. It was applied to estimate one-to-many relationships between words by Yamamoto and Umemura [2002]. They estimated these relations from the inclusion relations between the appearance patterns of two words. An appearance pattern is expressed as an n-dimensional binary feature vector. Let $F = (f_1, f_2, \ldots, f_i, \ldots, f_n)$, where $f_i = 0$ or 1, and let $T = (t_1, t_2, \ldots, t_i, \ldots, t_n)$, where $t_i = 0$ or 1, be the feature vectors of the appearance patterns for two words. The *CSM* of F to T is then defined as follows:

$$CSM(F,T) = \frac{ad - bc}{\sqrt{(a+c)(b+d)}}$$

$$a = \sum_{i=1}^{n} f_i \cdot t_i, \qquad b = \sum_{i=1}^{n} f_i \cdot (1 - t_i),$$

$$c = \sum_{i=1}^{n} (1 - f_i) \cdot t_i, \quad d = \sum_{i=1}^{n} (1 - f_i) \cdot (1 - t_i),$$

$$n = a + b + c + d$$

In our experiment, each "word" is an abstract noun, and n is the number of adjective types in the corpus (6407). Therefore, a indicates the number of adjective types co-occurring with both abstract nouns and b indicates the number of adjective types co-occurring only with the abstract noun corresponding to F. In contrast, c indicates the number of adjective types co-occurring only with the abstract noun corresponding to T and d indicates the number of adjective types that do not co-occur with either abstract noun.

3.2 Complementary similarity measure for gray-scale images

The CSM for gray-scale images is an extension of the CSM for binary images. Although the CSM for binary images is robust against graphical design, it is strongly affected by binarization or scanning conditions [Sawaki *et al.*, 1997].

The CSM for binary images is a special case of the four-fold point correlation coefficient. Therefore, Sawaki *et al.* [1997] defined the CSM for gray-scale images as a general form of the four-fold point correlation coefficient. Because it handles gray-scale images directly, this CSM is less affected by binarization or scanning conditions.

Let $F_g = (f_{g1}, f_{g2}, \ldots, f_{gi}, \ldots, f_{gn})$, where $f_{gi} = 0$ through 1, and let $T_g = (t_{g1}, t_{g2}, \ldots, t_{gi}, \ldots, t_{gn})$, where $t_{gi} = 0$ through 1, be the feature vectors of two gray-scale patterns. Then, the CSM_g of F_g to T_g is defined as follows:

$$CSM_g(F_g, T_g) = \frac{a_g d_g - b_g c_g}{\sqrt{nT_{g2} - T_g^2}}$$

$$a_g = \sum_{i=1}^{n} f_{gi} \cdot t_{gi}, \qquad b_g = \sum_{i=1}^{n} f_{gi} \cdot (1 - t_{gi}),$$

$$c_g = \sum_{i=1}^{n} (1 - f_{gi}) \cdot t_{gi}, \quad d_g = \sum_{i=1}^{n} (1 - f_{gi}) \cdot (1 - t_{gi}),$$

$$T_g = \sum_{i=1}^{n} t_{gi}, \quad T_{g2} = \sum_{i=1}^{n} t_{gi}^2$$

In our experiment, f_{gi} and t_{gi} are the weights based on the frequency at which an abstract noun co-occurred with an i-th type of adjective. In this paper, we used the following weighting function, where *Freq(noun, adj)* is the frequency at which the abstract noun co-occurs with the adjective:

$$Weight(noun, adj) = \frac{Freq(noun, adj)}{Freq(noun, adj) + 1} .$$

We paid particular attention to situations in which the noun co-occurred with the adjective. If the noun does not co-occur with the adjective, that is, *Freq*(*noun, adj*) is 0, the weight is 0.0. If *Freq*(*noun, adj*) is 1, it is 0.5. If *Freq*(*noun, adj*) was more than 1, the weight increased gradually until it approaches 1.0. This is because information on whether or not the noun co-occurs with the adjective is more important than information on how many times the noun co-occurs with the adjective.

4 Hierarchy Extraction Process

Word hierarchies were extracted as follows, where "TH" is a threshold value for each word pair under consideration;

1. Compute the similarity between appearance patterns for each pair of words. The hierarchical relation between the two words in a pair is determined by the similarity value. The pair is expressed as (X, Y), where X is a hypernym of Y and Y is a hyponym of X.
2. Sort the pairs by their normalized values and eliminate pairs with values below TH.
3. For each abstract noun C:
   - i Choose the hypernym-hyponym pair (C, D) with the highest value. This pair (C, D) is placed in the initial hierarchy.
   - ii Choose a pair (D, E) such that the hyponym E is not contained in the current hierarchy and (D, E) has the highest value among the pairs where word D, at the bottom of the current hierarchy, is a hypernym.

     Connect the hyponym E to D at the bottom of the current hierarchy.
   - iii Choose another pair (E, F) according to the previous step, and repeat the process until no more such pairs can be chosen.
   - iv Choose a pair (B, C) such that the hypernym B is not contained in the current hierarchy and (B, C) has the highest value among the pairs where the top word C of the current hierarchy is a hyponym.

     Connect the hypernym B in front of C at the top of the current hierarchy.
   - v Choose another pair (A, B) according to the previous step, and repeat the process until no more such pairs can be chosen.
4. If a short hierarchy is included in a longer hierarchy and the order of the words stays the same, the short one is dropped from the list of hierarchies.

5 Parameters

The conditions of our experiments were set as follows:

- CSM for binary images: TH = 0.2;
- CSM for gray-scale images: TH = 0.12.

If we set TH to a low value, it is possible to obtain long hierarchies. When the TH is too low, however, the number of word pairs that have to be considered becomes overwhelming and the reliability of the measurement decreases. We experimentally set TH as shown above so as to obtain "*koto* (matter)" as the top word of all hierarchies. Because "*koto*" co-occurred with the most number of adjectives, we predicted that "*koto*" would be at the top of the hierarchies.

6 Comparison and Evaluation

6.1 Overlap between extracted hierarchies

First, we compared the hierarchies obtained using the CSM for binary images (CSMb) and the CSM for gray-scale images (CSMg). Table 1 lists the number of extracted hierarchies. The CSMb extracted 189 hierarchies, while the CSMg extracted 178. There were only 28 common hierarchies with most of them having depths ranging from 3 to 6. One of the common hierarchies is shown below.

koto (matter) ---
jyoutai (state) ---
kankei (relation) ---
tsunagari (relationship) ---
en (ties/bonds).

The number of hierarchies obtained by the CSMg that included one or more hierarchies obtained by the CSMb was higher than the number obtained by the CSMb that included one or more obtained by the CSMg, i.e., (D) < (E). This suggests that the CSMg might be able to extract longer hierarchies than the CSMb can.

	Type of hierarchy	*Number*
(A)	Hierarchies obtained by CSMb	189
(B)	Hierarchies obtained by CSMg	178
(C)	Common hierarchies	28
(D)	Inclusion of CSMg hierarchies in CSMb hierarchies	5
(E)	Inclusion of CSMb hierarchies in CSMg hierarchies	38

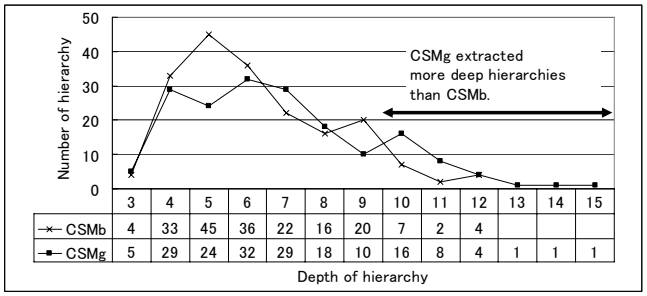
Table 1: Number of extracted hierarchies

The CSMg hierarchies shown below include one or more CSMb hierarchies. The underlined nouns are those that appear in one of the CSMb hierarchies.

koto (matter) ---
tokoro (point) ---
imeeji (image) ---
inshou (impression) ---
gaiken (appearance) ---
monogoshi (manner) ---
kihin (elegance) ---
hinkaku (grace) ---
kettou (pedigree) ---
kakei (family line)

koto (matter) ---
tokoro (point) ---
shigusa (behavior) ---
omokage (visage) ---
kawaisa (loveliness)

In fact, the depths of the CSMb hierarchies ranged from 3 to 12, while the depths of the CSMg hierarchies ranged from 3 to 15. We also found that the CSMg extracted more deep hierarchies than the CSMb but fewer shallow hierarchies. Overall, these results show that the CSMg extracted deeper hierarchies than did the CSMb, though the number of extracted hierarchies was smaller (see Figure 1).



	3	4	5	6	7	8	9	10	11	12	13	14	15
CSMb	4	33	45	36	22	16	20	7	2	4			
CSMg	5	29	24	32	29	18	10	16	8	4	1	1	1

Figure 1: Distribution of hierarchies by depth

6.2 Agreement with the concept hierarchies in the EDR electronic dictionary

Next, we compared each of the hierarchies obtained by the CSMb and the CSMg with the concept hierarchies for adjectives in the EDR electronic dictionary.

In this paper, we extracted the hierarchies from corpora consisting mostly of newspaper articles. Because the newspaper articles cover a wide range of topics, the information we obtain from our corpora is general knowledge. Therefore, it is reasonable to compare our hierarchy of abstract nouns with existing general purpose hierarchies such as the EDR hierarchy.

The number of hierarchies for adjectives obtained from the EDR dictionary is 932, and the maximum depth is 14, whereas our hierarchies had maximum depths of 12 and 15, as noted above (Figure 1). Because both the EDR and our two types of hierarchies had similar maximum depths, it was appropriate to evaluate our hierarchies by comparing them with those of the EDR dictionary.

Hierarchy in the EDR electronic dictionary

The EDR electronic dictionary [1995], which was developed for advanced processing of natural language by computers, is composed of 11 sub-dictionaries, including a concept dictionary, word dictionaries, bilingual dictionaries, etc.

Although we could verify and analyse our extracted hierarchies by comparing them with the EDR dictionary, there were two problems with this approach. First, many concepts in the EDR dictionary are defined by sentences or phrases, whereas the concepts in our extracted hierarchies are defined by abstract nouns only. Therefore, we replaced the sentences and phrases in the EDR concept definitions with sequences of words for a more accurate comparison. Secondly, there was a difference of words between those used to define concepts in the EDR and the abstract nouns used for our extracted hierarchies. To solve this problem, we extracted synonyms from the EDR dictionary and added synonyms to both the words in the EDR concept definitions and the abstract nouns in our hierarchies. We thus transformed the conceptual hierarchies of adjectives in the EDR dictionary into hierarchies consisting of sequences of word sets to enable a comparison with our hierarchies consisting of adjective hypernyms.

Measurement of agreement level

For the comparison, we measured the degree of agreement with an EDR hierarchy for each extracted hierarchy, i.e., we counted the number of nodes that agreed with nodes in the corresponding EDR hierarchy, while maintaining the order of each hierarchy. A node in one of our hierarchies is a set of abstract nouns and their synonyms. We represented the node by *Node*(*abstract noun, synonym*$_1$*, synonym*$_2$*, ...*), where *Node* was a name identifying the node, while a node in an EDR hierarchy is a set of sequences of words and synonyms and was represented by *Node*(*content word1, synonym1*$_1$*, synonym1*$_2$*, ..., content word2, synonym2*$_1$*, synonym2*$_2$*, ...*). Therefore, if a word in a node of one of our hierarchies was included in a node of an EDR hierarchy, we considered that the nodes agreed. For example, if our hierarchy is

A(***a, a'***, *a"*) ---
B(***b***, *b'*, ***b"***) ---
C(*c, c', c"*) ---
D(***d***, *d', d"*)

and a corresponding EDR hierarchy is

P(***a, a'***, *x, x'*) ---
Q (***b, b"***) ---
R(*r, r', r"*) ---
S(*s, s'*, ***d***, *f, f', f"*) ---
T(*t, t', g, g'*),

we count three agreement nodes, because *A, B, D* match to *P, Q, S*, respectively. The bold words indicate words which match between our hierarchy and an EDR hierarchy. And we define the level of agreement as three.

In comparing hierarchies, we found cases in which a hypernym and its hyponym in our hierarchy were treated as synonyms in the EDR electronic dictionary. For example, consider the following hierarchy obtained using our approach:

koto (matter) ---
  tokoro (point) ---
    imeeji (image) ---
      funiki (atmosphere) ---
        kuuki (atmosphere in a place) ---
          kanjyou (feeling/emotion) ---
            shinjyou (one's feelings/one's sentiment) ---
              shinkyou (mental state/one's heart) ---
                kangai (deep emotion) ---
                  omoide (memories).

In the EDR electronic dictionary, each word is linked to its concept, and a synonymous relation is defined as words linked to the same concept. That is, we can gather synonyms via EDR dictionary. In fact, in the above hierarchy, we obtained "*shinjyou* (one's feelings/one's sentiment)" and "*shinkyou* (mental state/one's heart)" as synonyms of "*kanjyou* (feeling/emotion)" from the EDR dictionary. Also, we know that "*kuuki* (atmosphere in a place)" is a synonym of "*funiki* (atmosphere)".

If we count the agreement of the above hierarchy with the EDR dictionary strictly, the level of agreement is 6. The agreement nodes are "*koto* (matter) -- *tokoro* (point) -- *imeeji* (image) -- *funiki* (atmosphere) or *kuuki* (atmosphere in a place) -- *kanjyou* (feeling/emotion), *shinjyou* (one's feelings/one's sentiment), or *shinkyou* (mental state/one's heart) -- *omoide* (memories)". However, if we accept hypernym-hyponym relations among synonyms, the agreement level is 9. In this case, the agreement nodes are "*koto* (matter) -- *tokoro* (point) -- *imeeji* (image) -- *funiki* (atmosphere) -- *kuuki* (atmosphere in a place) -- *kanjyou* (feeling/emotion) -- *shinjyou* (one's feelings/one's sentiment) -- *shinkyou* (mental state/one's heart) -- *omoide* (memories)".

Results

Table 2 shows the distribution of CSMb hierarchies for various agreement levels. Table 3 shows the same results for the CSMg. The numbers in *italics* in the tables indicate that the number of hierarchies at that depth which are completely included in an EDR hierarchy. In the last column, we show the average agreement level at each depth. The value in the bottom right-hand corner is the average agreement level for all hierarchies.

Figure 2 is a graph of the average agreement level at each depth shown in Tables 2 and 3. In Figure 2, except at the depths of 8 and 9, the average agreement levels for the CSMg hierarchies are higher than those of the CSMb hierarchies. As shown in Tables 2 and 3, the deeper hierarchies tended to have higher agreement levels. Therefore, we consider that overall the CSMg hierarchies were closer to the EDR hierarchies than were the CSMb hierarchies. That is, the CSMg hierarchies were more in accordance with human intuition than were the CSMb hierarchies.

We also verified the ability of the CSM to estimate hypernym-hyponym relations between two nouns. Some of noun pairs whose relations were estimated by two CSMs were opposite each other. Table 4 shows some such pairs. For each of them, the CSMg estimated that the noun on the left was a hypernym and the one on the right was a hyponym. The CSMb estimated the reverse.

Depth of hierarchy	*Agreement level* 1	2	3	4	5	6	7	8	9	*Ave. level*
3	1	2	***1***							2.00
4		6	18	***9***						3.09
5		7	23	12	***3***					3.24
6		4	12	9	7	***4***				3.86
7		2	2	10	4	3	***1***			4.32
8			1	6	6	3				4.69
9			1	4	5	4	5	1		5.55
10			2		2	2			1	5.29
11			1			1				4.50
12			1	1					2	6.25
Overall ave.										4.28

Table 2: Distribution of CSMb hierarchies for various agreement levels

Depth of hierarchy	*Agreement level* 1	2	3	4	5	6	7	8	9	*Ave. level*
3	1	3	***1***							2.50
4		6	13	***10***						3.14
5		3	9	9	***3***					3.50
6		1	11	12	6	***2***				3.91
7		1	5	10	8	5				4.38
8			4	5	7	2				4.39
9				6	1	3				4.70
10					2	6	4	3	1	6.69
11			1	2	1	3	1			5.13
12								1	3	8.75
13							1			7.00
14								1		8.00
15									1	9.00
Overall ave.										5.47

Table 3: Distribution of CSMg hierarchies for various agreement levels

Noun pair	
tokoro (point)	***imeeji*** (image)
tokoro (point)	***men*** (side)
tokoro (point)	***inshou*** (impression)
tokoro (point)	***seikaku*** (character)
tokoro (point)	***seishitsu*** (property)
tokoro (point)	***kanshoku*** (touch)
kimochi (thought/feeling/intention)	***omoi*** (feeling/mind/expectation)
kagayaki (brightness)	***koutaku*** (gloss)
kuukan (space)	***men*** (side)
kotoba (speech)	***iken*** (opinion)
kokoro (mind)	***shinjyou*** (one's feelings/one's sentiment)
hiyori (fine weather)	***ondo*** (temperature)

Table 4: Noun pairs estimated oppositely

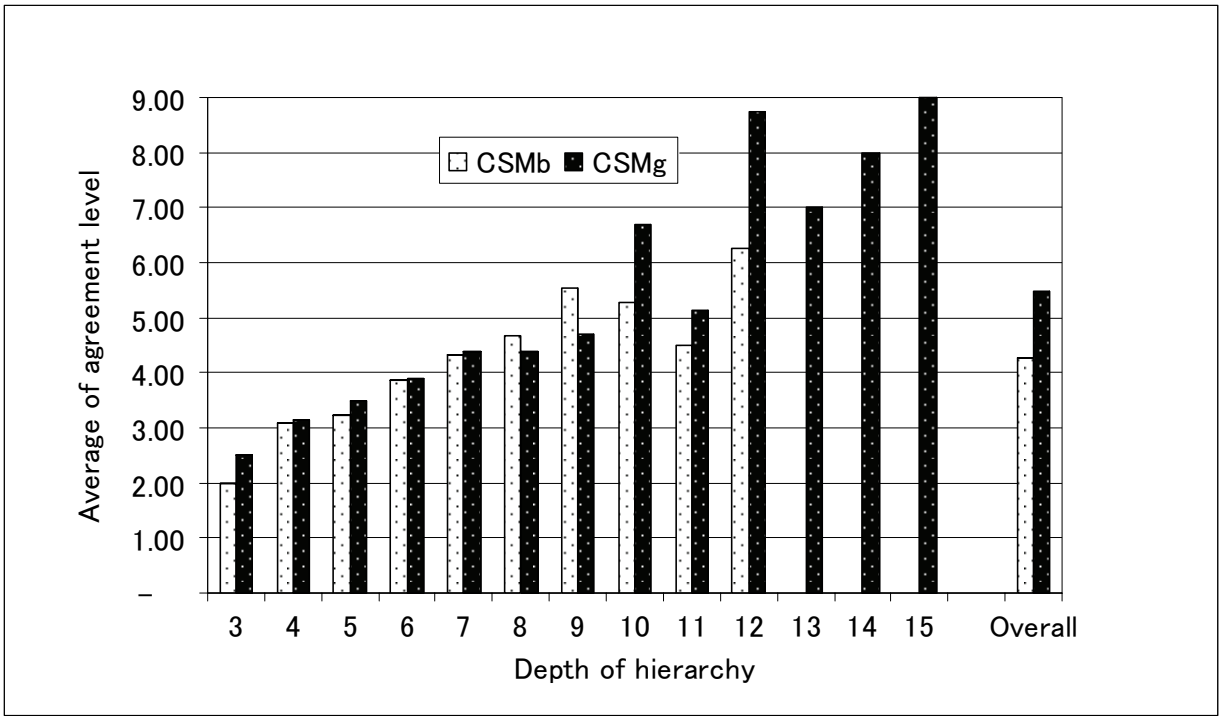


Figure 2: Comparison of CSMb and CSMg hierarchies by average agreement level

In our experiment, there were 836 such pairs. As the total number of pairs considered was 17201, these pairs amounted to less than 5%. We also found that in most cases these pairs appeared in the middle of a hierarchy.

Let us consider the hypernym-hyponym relation between "*kimochi* (thought/feeling/intention)" and "*omoi* (feeling/mind/expectation)". A CSMg hierarchy including "*kimochi*" and "*omoi*" was as follows:

koto (matter) ---
tokoro (point) ---
imeeji (image) ---
inshou (impression) ---
kanji (feeling/sense) ---
kibun (feeling/mood) ---
kimochi (thought/feeling/intention) ---
omoi (feeling/mind/expectation) ---
negai (wish) ---
nen (desire).

Here, "*kimochi*" was estimated as a hypernym of "*omoi*." However, CSMb estimated the opposite, i.e., "*omoi*" was a hypernym of "*kimochi*." We examined the values given by the CSMb and CSMg to "*kimochi*" and "*omoi*" (see Table 5).

(*F, T*)	("*omoi*", "*kimochi*")	("*kimochi*", "*omoi*")	*Diff.*
CSMb	0.8094	0.8064	0.0030
CSMg	0.7632	0.7700	0.0068

Table 5: Differences in CSM values for "*omoi*" and "*kimochi*"

As shown in Table 5, the values of CSMg in both directions were similar, as was the case for CSMb. As the CSM is a measure of inclusion, if the values of the CSM for two words in both directions are similar, it might mean that the two words are synonymous. Both the CSMg and CSMb estimated that "*kimochi*" and "*omoi*" were similar. However, due to the very small differences in values, they extracted opposite results. In fact, in the EDR electronic dictionary, "*kimochi*" and "*omoi*" are synonymous because both have the meaning of "feeling". The pairs estimated oppositely by the two CSMs may have a synonymous relation. In future work, we will introduce the CSM-based definition to estimate two words as synonyms.

A small number of word pairs were estimated oppositely by the two CSMs, with large differences in the CSM value of each direction. For example, CSMg estimated that "*tokoro* (point)" was a hypernym of "*imeeji* (image)" and CSMb estimated that "*imeeji*" was a hypernym of "*tokoro*" (see Table 6).

(*F, T*)	("*tokoro*", "*imeeji*")	("*imeeji*", "*tokoro*")	*Diff.*
CSMb	0.6767	0.7156	+0.0389
CSMg	0.6631	0.6468	-0.0163

Table 6: Differences in CSM values for "*tokoro*" and "*imeeji*"

The introduction of frequency information for CSMg may be the reason for this difference. In future work, we will analyze this in more detail.

7 Conclusion

We proposed a method of automatically extracting hierarchies based on the inclusion relations of the appearance patterns of words from corpora. In this paper, we described our attempts to extract objective hierarchies of abstract nouns co-occurring with adjectives in Japanese. We applied the complementary similarity measure for gray-scale images (CSMg) to search for better hierarchical word structures. In our experiment, we found that the CSMg could extract word hierarchies from corpora, even though it was developed for recognizing degraded machine-printed text. We also compared the CSMg with the CSM for binary images (CSMb), and found that the CSMg hierarchies were more in accordance with human intuition than were the CSMb hierarchies, as measured by their degree of agreement with the EDR electronic dictionary. As a next step, it would be interesting to compare this method to existing statistical methods such as agglomerative clustering. It would also be necessary to estimate confidence.

In this paper, we thus verified the suitability of the proposed method for extracting hierarchies from corpora. We consider that hierarchies tuned to specific corpora could be used for query expansion in information retrieval for specific domains. Our future work will include extracting hierarchies of key words in corpora and trying to utilize them as a special thesaurus for domain-oriented information retrieval.

References

[Berland and Charniak, 1999] Berland, M. and Charniak, E. Finding parts in very large corpora, In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 57-64, 1999.

[Caraballo, 1999] Caraballo, S. A. Automatic construction of a hypernym-labeled noun hierarchy from text, In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 120-126, 1999.

[EDR, 1995] *EDR Electronic Dictionary*. 1995.

http://www2.nict.go.jp/kk/e416/EDR/index.html

[Hagita and Sawaki, 1995] Hagita, N. and Sawaki, M. Robust recognition of degraded machine-printed characters using complementary similarity measure and error-correction learning, In *Proceedings of the SPIE – The International Society for Optical Engineering*, 2442: pp. 236-244, 1995.

[Hearst, 1992] Hearst, M. A. Automatic acquisition of hyponyms from large text corpora, In *Proceedings of the 14th International Conference on Computational Linguistics*, pp. 539-545, 1992.

[Kanzaki *et al.*, 2003] Kanzaki, K., Ma, Q., Yamamoto, E., Murata, M. and Isahara, H. Adjectives and their abstract concepts --- toward an objective thesaurus from semantic map. In *Proceedings of the Second International Workshop on Generative Approaches to the Lexicon*, pp. 177-184, 2003.

[Kanzaki *et al*., 2004] Kanzaki, K., Yamamoto, E., Ma, Q. and Isahara, H. Construction of an objective hierarchy of abstract concepts via directional similarity. In *Proceedings of 20th International Conference on Computational Linguistics,* Vol. II , pp. 1147-1153, 2004.

[Matsumoto *et al.*, 1996] Matsumoto, Y., Sudo, S., Nakayama, T. and Hirao, T. Thesaurus construction from multiple language resources, In *IPSJ SIG Notes NL-93*, pp. 23-28, 1996.

[Miller *et al.*, 1990] Miller, A., Beckwith, R., Fellbaum, C., Gros, D., Millier, K. and Tengi, R. *Five Papers on WordNet, Technical Report CSL Report 43*, Cognitive Science Laboratory, Princeton University, 1990.

[Nakayama and Matsumoto, 1997] Nakayama, T. and Matsumoto, Y. Positioning nouns in a classification-based thesaurus, In *IPSJ SIG Notes NL-120*, pp. 103-108, 1997.

[Nemoto, 1969] Nemoto, K. Combination of noun with "ga-case" and adjective, *Language Research for the Computer 2*, National Language Research Institute, pp. 63-73, 1969 (In Japanese).

[Sawaki *et al.*, 1997] Sawaki, M., Hagita, N. and Ishii, K. Robust character recognition of gray-scaled images with graphical designs and noise, In *Proceedings of the International Conference on Document Analysis and Recognition,* pp. 491-494, 1997.

[Shoutsu *et al.*, 2003] Shoutsu, Y., Tokunaga, T. and Tanaka, H. The integration of Japanese dictionary and thesaurus, In *IPSJ SIG Notes NL-153*, pp. 141-146, 2003.

[Takahashi, 1975] Takahashi, T. Various phase related to part-whole relation investigated in sentence, *Studies in the Japanese Language 103*, The Society of Japanese Linguistics, pp. 1-16, 1975 (In Japanese).

[Tsurumaru *et al.*, 1986] Tsurumaru, H., Hitaka, T. and Yoshida, S. Automatic extraction of hierarchical relation between words, In *IPSJ SIG Notes NL-83*, pp. 121-128, 1986.

[Yamamoto and Umemura, 2002] Yamamoto, E. and Umemura, K. A similarity measure for estimation of one–to-many relationship in corpus, In *Journal of Natural Language Processing*, pp. 45-75, 2002.

[Yamamoto *et al.*, 2004] Yamamoto, E., Kanzaki, K. and Isahara, H. Hierarchy extraction based on inclusion of appearance, In *ACL04 Companion Volume to the Proceedings of the Conference*, pp. 149-152, 2004.

PHILOSOPHICAL FOUNDATIONS

Attribution of Knowledge to Artificial Agents and their Principals*

Samir Chopra
Department of Computer and Information Science
Brooklyn College
Brooklyn, NY 11210
schopra@sci.brooklyn.cuny.edu

Laurence White
Rue Franklin 157
Brussels B-1000
Belgium
laurencefwhite@gmail.com

Abstract

We consider the problem of attribution of knowledge to artificial agents and their legal principals. When can we say that an artificial agent *X* knows *p* and that its principal can be attributed the knowledge of *p*? We offer a pragmatic analysis of knowledge attribution and apply it to the legal theory of artificial agents and their principals.

1 Introduction

An agent's principal is its employer, or any other legal person engaging the agent to carry on transactions on its behalf. A problem commonly faced by courts is deciding when to attribute the knowledge in possession of an agent to its principal. If the agent in question is an *artificial* one, how should the courts decide that a) the agent *knows* the proposition in question and b) that *this knowledge can be attributed to the agents' principal*? We need a philosophical account of knowledge attribution that does justice to the first question, and thereby aids in the resolution of the second – legal – problem. Conversely, legal resolutions of these issues will aid us in a solution of the philosophical problem – just as legal findings on personhood can clarify philosophical debates over the nature of personal identity. As the delegation of tasks to artificial agents increases, so will cases that encounter the need for decisions hinging on these debates, thus rendering more urgent the need for a solution.

The plan of this paper is as follows. In Section 2, we present our analysis of knowledge attribution. In Section 3, (within limitations of space) we illustrate its plausibility by examples and arguments. In Section 4 we describe the legal problem of attributing knowledge held by agents to their principals (while critically examining current legal doctrine in this area) and show how our analysis is of help. While we concentrate on British Commonwealth and US law, this paper does not purport to be a complete survey of the relevant law in the jurisdictions dealt with. When citing non-academic sources such as case reports, we use the legal mode of citation.

* We thank Rohit Parikh, John Sutton and the referees for helpful comments.

2 Agents' knowledge: a pragmatic analysis

Consider the following knowledge claim: *X* knows *p*. Philosophers have long considered the conditions under which such a claim could be made going back to Plato's *Theatetus*, which analyzed knowledge as *justified true belief*: i.e., *X* knows *p* iff:

1. *X* believes *p*
2. *p* is true
3. *X* is justified in believing *p*

[Gettier, 1963] has shown by a series of counterexamples that this analysis is flawed. Despite considerable effort, no satisfactory analysis of knowledge has emerged that does justice to these or newer counterexamples (largely due to difficulties in defining a satisfactory notion of justification). Knowledge attribution has long been recognized as ripe for a treatment grounded in a more *pragmatic* understanding.

In our analysis of knowledge attribution, *X* knows *p* iff:

1. *X* has *ready access* to *p*
2. *p* is true
3. *X* can *make use of the informational content* of *p* (equivalently, *X* can exercise certain capacities dependent on its knowing *p)*

We retain the truth condition of the original analysis and introduce two new conditions. Condition 1 uses the notion of access to, or easy availability of, the proposition *p*. Condition 3 – which replaces the notion of justification – implies counterfactuals such as "If *X* did not know *p*, then *Y*" – where *Y* is a statement like "*X* is not able to exercise capacity *C*". Since *X* is able to exercise capacity *C*, it knows *p*. (See dispositional [Levi and Morgenbesser, 1964] or functionalist accounts [Armstrong, 1980] for similar notions.)

3 The Case for the Pragmatic Analysis

The following examples illustrate the intuitions underlying our analysis:

1. As I walk down the street, I am asked by a passerby, "Excuse me, do you know the time?" I answer "Yes" as I reach for my cellphone to check and inform him what time it is. This example, due to Andy Clark [2003], shows that we take ourselves to know those items of information that are easily accessible[1] and can be easily used. Clark's example is part of an extended argument for distributed cognition through external tools and memory stores not confined to the inside of our craniums. For our analysis we note that information at hand can be described as information that we 'know' (I could not claim to know my friends' telephone numbers if, on being asked, I were to reply, "I can't remember").

2. A friend wants to buy me a book as a gift. He asks me for my shipping address so that he can send me the book. I direct him to my wish-list at Amazon.com saying, "Amazon *knows* my shipping address". Indeed, the shopping cart software on that site does. When my friend has decided which books he wants to buy, he pays and picks the shipping option. Amazon generates a shipping invoice complete with shipping address. The shopping cart software is able to discharge its functions using that piece of information. I had stored the shipping address on Amazon precisely for such future use. I could store an alternate shipping address and *forget its details*, since Amazon 'knows' it and will ship to it if anyone decides to send me a book at that address. Note that parts of my address could be obtained from Amazon's database (by Amazon.com programmers) by writing specific queries (e.g., "What street does customer *X* live on?"). In a weaker sense, then, Amazon then also knows which of its customers lives in postal code 11205.

3. I have to attend a meeting at the university campus branch located in the city center. With directions for the meeting written down on a piece of paper that I keep in my pocket, I head out the office door. As I do so, my office-mate asks me "Do you know where the meeting will be held?" I answer, "Yes" as I hurry towards the next train. Here, Gricean semantics [Grice, 1975] is at play: *if I said I did not know the meeting's location, I would be misleading my questioner*. This example is crucial in showing that knowledge claims are connected to the pragmatics of speech. To deny a valid knowledge attribution in this case would be to say something misleading or something whose semantic value is considerably less than the knowledge claim.

In the Amazon example the agent is able to use my address to fulfill its functions. An alternative locution would be to say, "Amazon *has* my address", but what purpose would be served other than an avoidance of intentional vocabulary? If it did not ship to the correct address, Amazon could not use as a defense the claim that it did not know (or 'have access to') my address. It was stored in their database and had been used successfully in the past. Amazon is capable of informing a potential customer that it is unable to ship goods since it does not 'know' the recipient's address (or credit card number). If Amazon has access to my address but it has changed in the meantime, then it is natural to say that Amazon *does not know my address since it would not be able to perform the function of shipping books to me*.

When we say, "Amazon.com knows my shipping address", our analysis implies that:

1. Amazon has ready access to my shipping address in its databases.
2. The shipping address is correct.
3. Amazon is able to perform capacities dependent upon its knowing my address (it is able to make use of the informational content of the address).

Furthermore, the relevant counterfactuals are true: if it did not know my address, Amazon's core functionality with respect to its interactions with me would not be achievable; if Amazon did not know my shipping address, it would not be able to send books to me; if Amazon did not know my address, it would not be able to send me a bill; but it is able to do so; hence it knows my address. This kind of analysis is readily extended to other kinds of agents that take actions based on information at their disposal. An artificial agent's actions could be described in much the same way as a human agent's – "The pricebot sent me a quote because it knew my preferences".

Our analysis may be fruitfully contrasted with conventional formal analysis, in which an agent's belief corpus is taken to be the set of propositions that the agent is committed to (the agent answers "Yes" when asked "Do you believe *p*?"). Formally, *p* is derivable using the inference machinery built into that agent's architecture. Thus I could say of an artificial agent that it knows or believes *p* if *p* is derivable from its knowledge base. But to limit knowledge attri-

[1] [Parikh, 1994] has described knowledge as statements that the agent is capable of deriving (from a given set of premises) within some reasonably tractable bounds as opposed to those implied by epistemic closure (if *X* knows *p*, and *X* knows that *p* implies *q* then *X* knows *q*). Parikh's account limits knowledge to those *q*'s that meet the tractability condition (thus providing an analysis of knowledge in terms of the abilities of the agent as we do).

bution to those agents that are capable of deriving the proposition *p* using a formally specifiable inference mechanism would be to put the proverbial cart before the horse. We feel comfortable making the claim that the cat knows a mouse is behind the door though we do not have the foggiest idea of what kind of inferential mechanism is at hand. The cat reveals its knowledge through its actions. Whatever kind of retrieval or inference mechanism is at work, it enables the cat to go about its tasks. Similarly for an artificial agent – it reveals its knowledge of *p* through the ready availability of the proposition in facilitating the agent's functionality. We do not discount a hybrid architecture that employs a deductive database that can infer further information applying rules to a set of stored facts. In that case we would say that the agent in question knows all the facts derivable from its database as well – subject to tractability conditions as in [Parikh, 1994].

Where does an agent's epistemology come into the picture? If an agent elicits information from humans then the responsibility of ensuring the accuracy of the information is the user's. If the user inputs incorrectly, the agent is in possession of false information and we do not make the knowledge attribution. Thus, the software artifact inherits its epistemology from the humans that supply it information and carry out data entry. This should not lead us to think that artificial agents do not have an independent epistemology. Pricebots that read price information on remote web pages acquire knowledge autonomously, by using their file-reading mechanisms, presumably equipped with error checking and validation routines that guarantee it will not read in garbage (the software equivalent of a reliable sensor). The accuracy with which these agents acquire information is a function of their design and the code that runs on them – very similar to human agents, the accuracy of whose beliefs is a function of how well their senses work in conjunction with background knowledge and their reasoning powers.

We would not want to say that an artificial agent knows a proposition if the proposition is simply stored in the agent's knowledge base but is not accessible for use by the agent. In that case, we would say that the information in question is stored in the agent but the agent does not know it, since it is unable to access or use it. Note that when files are deleted from a computer the information ordinarily does not vanish, it simply becomes a target for over-writing. The information is not accessible any more without elaborate recovery methods, and hence the computer's operating system is reasonably enough said not to have access to it any more.

In the case of Amazon, it is possible that not a single human being employed by Amazon knows my address. It is conceivable that when the shipping invoice is printed out by the software, a human clerk will pick it up and attach it to the box of books in question without bothering to check any further whether the address is correct or not. The software has been treated as a reliable source of information with regards to the address – and thus humans might accurately claim that they learned a proposition from a software agent. When a book is purchased, my address has been used by Amazon.com without any human knowing it. What sense would it make to say that Amazon did not know the address? Alternative locutions for describing this functionality of Amazon's would be artificial. What could we say – that Amazon has access to this true information, and can use it? The parallels with knowledge attributions to human agents should be clear. For human agents, on our analysis, are said to know a proposition *p* when we can make such a claim. If I know that 619 times 3 is 1857 but cannot open a safe with this combination, then I do not know the combination to the safe since I cannot open it but I do know the product of 619 and 3.

In making knowledge attributions, there is a parallel between humans and artificial agents. The ease with which we slip into the intentional attribution when it comes to Amazon.com is an indication of this similarity. The intentional stance is used when it is possible to give the best explanations of behavior using it. What kind of behavior would we able to predict? We could predict Amazon's responses to certain queries. For instance, we could say that Amazon knows the ISBN number for *How Green Was My Valley* since it would be able to produce that number on request. But as we have argued, we could also predict Amazon's success in certain tasks – Amazon could demonstrate its knowledge of the ISBN number of *How Green Was My Valley* by shipping me that book and none other.

Condition 2 of our analysis is crucial (as in most analyses of knowledge) since if the shipping address in question were incorrect we would not say that Amazon knows the shipping address. The locution we would employ if Amazon were to use the incorrect shipping address would be "Amazon shipped my books to what it thought (or believed) was my correct address". We would not make the claim that Amazon knows my address if in fact, the address is false (since it is possible to have a false belief).

One way to deny that artificial agents can know propositions would be to ask, "*Who* does the knowing in the case of the artificial agent?" Our response would be that the same could be asked of humans, and in the absence of any philosophically satisfactory analysis of personal identity there is no reason to believe that a stronger condition should be placed on artificial agents. Below, we suggest that the correct legal treatment of artificial agents is to assimilate them to human agents, but without the personhood possessed by human agents. If the same view is adopted on the philosophical perspective, it becomes otiose to ask who does the knowing in the case of an artificial agent – other than the agent itself, of course.

4 The legal doctrine of attributed knowledge

This inclusion of the ready-to-hand in the knowledge of an agent has close and instructive parallels in the legal doctrine of attributed knowledge. Under this doctrine, the law may impute to a principal knowledge – relating to the subject matter of the agency – which the agent acquires while acting on behalf of its principal within the scope of its authority. The scope of the agent's authority refers to those transactions that the principal has authorized the agent to conduct. In some circumstances, knowledge gained by the agent outside the scope of the agency can also be attributed to the agent's principal.

Once knowledge is attributed to the principal, it is deemed to be known by the principal and it is no defense for the principal to claim that he did not know the information in question, for example, because the agent failed in its duty to convey the information to the principal.

The doctrine of attributed knowledge has many applications, and is used generally in civil law contexts in cases where the knowledge of the principal is relevant. For instance, legal consequences attach to principals knowingly receiving trust funds, or having notice of claims of third parties to property received, or knowingly making false statements.

The doctrine has close parallels with our analysis above, which extends the concept of knowledge to include the information that we retain in storage devices – including written documents – that are ready-to-hand. From this perspective, a human agent is akin to a knowledge storage device under the control of a principal. Below, we suggest that artificial agents can be thought of similarly. But first, we explore the basis of the doctrine and its application to the modern company.

4.1 A duty to communicate?

While the doctrine of attributed knowledge is pervasive in the legal systems under discussion, its precise doctrinal basis is still a matter of some dispute [DeMott, 2003].

One explanation of the doctrine relies on the supposed identity of principal and agent, whereby the law sees them as one person for some purposes. However, this theory lacks explanatory power, and does not explain the public policy justification for the rule.

Another explanation put forward for attributed knowledge is that the law presumes that agents will carry out their duties to communicate information to their principals. For example, in the standard practitioner's text *Halsbury's Laws of England*, the scope of an agent's duty to communicate determines the existence and the timing of any attributed knowledge of the agent.[2] Under this approach, the doctrine operates on a pre-existing duty to convey information to deem that the duty has been discharged.

In the US, the common law of agency does not require as a precondition an *existing duty to communicate* the information to the principal [DeMott, 2003]. As Langevoort [2003] points out, the courts' description of attribution as *the presumption that the agent has fulfilled its duty of candor in conveying information* is not correct, since attribution applies *even* where interaction between principal and agent creates enough scope of discretion that no transmission of information is expected (or occurs).

In England the "duty to communicate" has been abandoned as the explanatory basis of attribution of knowledge.[3] Similarly, Australian courts have inferred attribution of knowledge in the absence of a duty to communicate information in cases where the task assigned to the agent included making appropriate disclosures. [4]

We believe the rejection of the duty to communicate is correct on policy grounds To require such a duty in order to attribute knowledge held by agents to their principals would encourage principals to ask agents to shield them from inconvenient information, and would put principals acting through agents in a better position than principals acting directly. Such an approach is also incompatible with modern information management practices within companies, and we discuss why below.

However, the fact that agents are *capable* of communication is important to the attribution of knowledge. In terms of our analysis of knowledge, a lack of capacity to communicate information would render the first and/or third conditions unfulfilled – i.e., that the principal has ready access to the knowledge held by the agent, or that the principal can make use of its informational content. *The capacity to communicate* therefore plays an explanatory role when thinking about how artificial agents fit within this legal schema.

4.2 Attribution of knowledge to companies

A company is a special kind of organization that, in modern legal systems, is recognized as a legal person in its own right. How, then, does a company gain knowledge in the eyes of the law? Apart, possibly, from knowledge gained "directly" by the Board or general meeting of a company, only through the attribution to it of knowledge gained by its agents (i.e., its directors, employees or contractors). By the

[2] Vol. 2(1) (Fourth Edition Reissue) Agency, para 164.

[3] *El Ajou v Dollar Land Holdings plc & Anor* [1994] 2 All ER 685, at 703–4 per Hoffmann L.J. See also [Reynolds, 2001], Article 97(1) at paragraph 8-207.

[4] *Permanent Trustee Australia Limited v FAI General Insurance Company Ltd (in Liq)* [2003] HCA 25 at paragraph 87.

doctrine of attribution, the company is deemed to gain the knowledge that is gained by the natural persons (i.e., humans) engaged by it. [5]

The large modern company illustrates why the "duty to communicate" cannot found the attribution of knowledge. Given the company is an abstract entity, the only way to make sense of such a duty would be in terms of communication to other agents (such as immediate superiors), who are required to communicate it "directly" to the company as embodied by the Board of directors (or general meeting). Since in modern corporations authority to enter and administer contracts is usually delegated to relatively junior staff members, it would be absurd if all the knowledge that had legal consequences for a company had to be communicated upwards in this way. Instead, most information within the modern corporation remains with lower-level officers, and is only passed upwards in summary terms – or when there is some exceptional reason to do so, such as a dispute with outside parties. Abandoning the "duty to communicate" allows the legal system to acknowledge information managed in accordance with modern decentralized practices.

Today the most common way for information to be stored and controlled by low-level officers is by inputting it into the company's information systems. Some of these systems can be queried by senior managers – but it has never, to our knowledge, been suggested that this is essential to the attribution. To what extent could information systems – artificial agents – themselves be treated by the legal system as agents for the purposes of attribution of knowledge?

4.3 Artificial agents as agents for knowledge imputation purposes

In [Chopra and White, 2004], following [Kerr, 1999], it was argued that the legal system should and could extend the legal treatment of human agents to artificial agents, with appropriate modifications. Artificial agents, on this approach, would have a legal status akin to slaves in Roman law – that is, with capacity to enter contracts on behalf of their principals, but without contracting capacity or legal personhood in their own right.

We believe a similar move can and should be made with respect to the imputation of knowledge. On this approach, knowledge gained by artificial agents employed by corporations could be attributed to the corporations themselves, where that knowledge would be attributed to the corporation in the case of a human agent.

Our analysis could be utilized in a legal context for the task of determining what is known by the artificial agent in question.[6]

The scope of the agency would be those transactions that the artificial agent has been deployed to conduct. Not all the agent's knowledge would necessarily be attributed to the principal. For example, an agent could conceivably act for two principals and in accordance with the law of agency, knowledge gained in the course of one agency is not always attributed to the other principal.[7] A natural person could deploy an artificial agent, and in that instance the agent's knowledge would be attributed to the principal in the same circumstances.

Surprisingly, there is a paucity of judicial pronouncements on the possibility of attribution of knowledge held by artificial agents. Some tangential judicial support for such a treatment of artificial agents was given recently, but it was made clear that it did not (yet) represent the law. In the Australian case *Commercial Union v Beard & Ors*[8], the issue arose whether a fact contained in a news clipping, filed in a company paper file, was "known" to an insurer for the purposes of the relevant statute. If it was known to the insurer, the party taking out insurance was relieved of the obligation of making disclosure of the fact to the insurer.

The majority found that a matter could be "known" by the insurer company if it were contained in the "current formal records" of the company. This term appeared to include the minutes of the company's Board meetings. However, the majority held that the extract of the news clipping in question was "not a record of [the insurer] and it was not contained in any file to which officers of [the insurer] were expected to have recourse for the purposes of the subject insurance."[9] As a result, they found that the contents were not "known" to the company for the purposes of the statute.

The minority judge, however, seemed to discount that anything could be "known" to the company merely by being contained in a record, while acknowledging that such a view had its attractions [emphasis added]:

> We were not referred to any authority for the proposition that, in the absence of actual knowledge on the part of relevant officers of a company, the company may, nevertheless, "know" a matter, where the relevant information is contained in a company file. *I*

[5] See *Halsbury's Laws of England*, Companies, 7(1) (2004 Reissue), paragraph 441: How a company may act.

[6] The 'actual knowledge' of human agents is treated by the Courts as self-evident and not needing further analysis: see the five-fold categorization of knowledge in *Baden v. Société Générale*, [1993] 1 W.L.R. 509, 575-76.

[7] On these cases, English and US law take divergent and sometimes confusing approaches: see [DeMott, 2003]; [Reynolds, 2001], at paragraph 8-210; *Permanent Trustee Australia Co Ltd v FAI General Insurance Co Ltd* (2001) 50 NSWLR 679 at 697 per Handley JA.

[8] [1999] NSWCA 422

[9] per Davies AJA at paragraph 63

> *find the proposition an attractive one*. In circumstances, which are undoubtedly common today, where important information relating to the conduct of a company's business is stored in the company's computer system, *from which it may be readily obtained*, the suggestion that such material is part of the company's knowledge is certainly appealing. However…the present state of authority does not permit a finding that the information so stored becomes "known" to the company until it is transferred into the mind of an officer, who is relevantly engaged in the transaction in question[10].

The emphasis on being readily obtainable echoes condition 1 of our analysis of knowledge. Indeed, we suggest that, had the contents of the news clipping been stored in an information system, rather than a paper file, so as to be *readily available* to the human officers conducting the insurance transaction, the result should have been different in the *Commercial Union* case. The prohibitive cost of insisting on cross-checks being made of all paper files before proceeding with any transaction without fear of legal consequences is obviously significantly reduced if those files are held electronically and therefore ready-to-hand to employees of the company generally.

The example suggests that information systems that are mere accumulations of records may not qualify as agents for attribution purposes. We suggest that the knowledge held by artificial agents will only be attributed to a corporation to the extent that the agent permits ready access by other (human or artificial) agents to its contents. In this way, while the duty to communicate is not necessary for the imputation of knowledge, the *ability* to communicate so as to make information readily accessible to others – and not just to passively store information – might well be.

Considering the company as a knowing agent in its own right for a moment, paper files, to which officers are not expected to have recourse in conducting particular transactions, could be equated with the 'dead' information contained within, but not accessible to, an artificial agent – such as information written on a hard disk, but not readily accessible to the user through the operating system without deploying specialized software.

5 Conclusion

We have presented a philosophical and legal analysis of knowledge attribution and suggested that the courts could make use of the analysis when deciding whether to attribute knowledge to artificial agents and principals, such as corporations, employing those artificial agents to conduct transactions on their behalf. In our discussion of Amazon, we did not bother to distinguish between the corporation (a legal entity) and the software agents operated by the corporation. We think that for the reasons mentioned in our analysis above, it can make sense to attribute knowledge to both the artificial agents operated by a corporation and the corporation itself. We also pointed out close and instructive parallels between the philosophical analysis and the legal doctrine. We look forward to the first cases where legally salient information known only to artificial agents is nevertheless attributed to the corporations operating those agents on the basis of the above-outlined principles.

[10] per Foster AJA at paragraph 73

References

[Armstrong, 1980] David Armstrong. *The Nature of Mind*. St. Lucia, Queensland, 1980. University of Queensland Press.

[Chopra and White, 2004] Samir Chopra and Laurence White. Artificial Agents - Personhood in Law and Philosophy. In *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI 2004*, pages 635-639. Amsterdam, The Netherlands, 2004. IOS Press.

[Clark, 2003] Andy Clark. *Natural-Born Cyborgs: Minds, Technologies, and the Future of Human Intelligence*. Oxford, United Kingdom, 2003. Oxford University Press.

[DeMott, 2003] Deborah A. DeMott. When is a Principal Charged with an Agent's Knowledge? *Duke Journal of Comparative and International Law,* 13:291-320, 2003.

[Gettier, 1963] Edmund Gettier. Is Justified True Belief Knowledge? *Analysis*, 23:121-23, 1963.

[Grice, 1975] H.P Grice. Logic and Conversation. In Peter Cole and Jerry L. Morgan, eds. 1975. *Syntax and Semantics, 3: Speech Acts*, pages 41-58. New York, United States, 1975. Academic Press.

[Kerr, 1999] Ian R. Kerr. Providing for Autonomous Electronic Devices in the Uniform Electronic Commerce Act. Web-only paper at www.ulcc.ca/en/cls/index.cfm?sec=4, 1999. Uniform Law Conference of Canada.

[Langevoort, 2003] Donald C. Langevoort. Agency Law inside the Corporation: Problems of Candor and Knowledge. *University of Cincinnati Law Review*, 71:1187-1231, Summer, 2003.

[Levi and Morgenbesser, 1964] Isaac Levi and Sidney Morgenbesser. Belief and Disposition. *American Philosophical Quarterly*, 1(3):221-232, 1964.

[Lewis, 1996] David Lewis. Elusive Knowledge. *Australasian Journal of Philosophy*. 74:549-567, 1996.

[Parikh, 1994] Rohit Parikh. Logical Omniscience. In *Logic and Computational Complexity, LNCS 960*, pages 22-29, Heidelberg, Germany, 1995. Springer-Verlag.

[Reynolds, 2001] F.M.B. Reynolds. *Bowstead and Reynolds on Agency*. 17th ed. London, United Kingdom, 2001. Sweet & Maxwell.

PsychSim: Modeling Theory of Mind with Decision-Theoretic Agents

David V. Pynadath and Stacy C. Marsella
Information Sciences Institute, University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292 USA
{marsella,pynadath}@isi.edu

Abstract

Agent-based modeling of human social behavior is an increasingly important research area. A key factor in human social interaction is our beliefs about others, a theory of mind. Whether we believe a message depends not only on its content but also on our model of the communicator. How we act depends not only on the immediate effect but also on how we believe others will react. In this paper, we discuss PsychSim, an implemented multiagent-based simulation tool for modeling interactions and influence. While typical approaches to such modeling have used first-order logic, PsychSim agents have their own decision-theoretic model of the world, including beliefs about its environment and recursive models of other agents. Using these quantitative models of uncertainty and preferences, we have translated existing psychological theories into a decision-theoretic semantics that allow the agents to reason about degrees of believability in a novel way. We discuss PsychSim's underlying architecture and describe its application to a school violence scenario for illustration.

1 Introduction

People interact within a rich social framework. To better understand people's social interactions, researchers have increasingly relied on computational models [Liebrand *et al.*, 1998; Prietula *et al.*, 1998]. Models of social interaction have also been used to create social training environments where the learner explores high-stress social interactions in the safety of a virtual world [Marsella *et al.*, 2000]. A key factor in human social interaction is our beliefs about others, a *theory of mind* [Whiten, 1991]. Our decisions to act are influenced by how we believe others will react. Whether we believe a message depends not only on its content but also on our model of the communicator. Giving its importance in human social interaction, modeling theory of mind can play a key role in enriching social simulations.

Typical approaches to modeling theory of mind in a computational framework have relied on first-order logic to represent beliefs and goals. However, such representations are often insensitive to the distinctions among conflicting goals that people must balance in a social interaction. For example, psychological research has identified a range of goals that motivate classroom bullies (e.g., peer approval, sadism, tangible rewards). All bullies share the same goals, but it is the relative priorities that they place on them that leads to the variations in their behavior. Resolving the ambiguity among equally possible, but unequally plausible or preferred, options requires a quantitative model of uncertainty and preference. Unfortunately, more quantitative frameworks, like decision theory and game theory, face their own difficulties in modeling human psychology. Game theoretic frameworks typically rely on concepts of equilibria that people rarely achieve in an unstructured social setting like a classroom. Decision theoretical frameworks typically rely on assumptions of rationality that people constantly violate.

We have developed a social simulation tool, *PsychSim* [Marsella *et al.*, 2004], that operationalizes existing psychological theories as boundedly rational computations to generate more plausibly human behavior. PsychSim allows a user to quickly construct a social scenario where a diverse set of entities, groups or individuals, interact and communicate. Each entity has its own preferences, relationships (e.g., friendship, hostility, authority) with other entities, private beliefs, and mental models about other entities. The simulation tool generates the behavior for these entities and provides explanations of the result in terms of each entity's preferences and beliefs. The richness of the entity models allows one to explore the potential consequences of minor variations on the scenario. A user can play different roles by specifying actions or messages for any entity to perform.

A central aspect of the PsychSim design is that agents have fully specified decision-theoretic models of others. Such quantitative recursive models give PsychSim a powerful mechanism to model a range of factors in a principled way. For instance, we exploit this recursive modeling to allow agents to form complex attributions about others, enrich the messages between agents to include the beliefs and preferences of other agents, model the impact such recursive models have on an agent's own behavior, model the influence observations of another's behavior have on the agent's model of that other, and enrich the explanations provided to the user. The decision-theoretic models in particular give our agents the ability to judge degree of credibility of messages in a subjective fashion that can consider the range of influences that

sway such judgments in humans.

The rest of this paper describes PsychSim's underlying architecture in more detail, using a school bully scenario for illustration. The agents represent different people and groups in the school setting. The user can analyze the simulated behavior of the students to explore the causes and cures for school violence. One agent represents a bully, and another represents the student who is the target of the bully's violence. A third agent represents the group of onlookers, who encourage the bully's exploits by, for example, laughing at the victim as he is beaten up. A final agent represents the class's teacher trying to maintain control of the classroom, for example by doling out punishment in response to the violence.

2 The Agent Models

We embed PsychSim's agents within a decision-theoretic framework for quantitative modeling of multiple agents. Each agent maintains its independent beliefs about the world, has its own goals and it owns policies for achieving those goals. The PsychSim framework is an extension to the Com-MTDP model [Pynadath and Tambe, 2002] of agent teamwork. To extend the Com-MTDP framework to social scenarios (where the agents are pursuing their own goals, rather than those of a team), we designed novel agent models for handling belief update and policy application, as described in Section 2.3.

2.1 Model of the World

Each agent model starts with a representation of its current state and the Markovian process by which that state evolves over time in response to the actions performed.

State

Each agent model includes several features representing its "true" state. This state consists of objective facts about the world, some of which may be hidden from the agent itself. For our example bully domain, we included such state features as `power(agent)`, to represent the strength of an agent. `trust(truster,trustee)` represents the degree of trust that the agent `truster` has in another agent `trustee`'s messages. `support(supporter,supportee)` is the strength of support that an agent `supporter` has for another agent `supportee`. We represent the state as a vector, $\vec{s}^t$, where each component corresponds to one of these state features and has a value in the range $[-1, 1]$.

Actions

Agents have a set of actions that they can choose to change the world. An action consists of an action type (e.g., `punish`), an agent performing the action (i.e., the actor), and possibly another agent who is the object of the action. For example, the action `laugh(onlooker, victim)` represents the laughter of the onlooker directed at the victim.

World Dynamics

The state of the world changes in response to the actions performed by the agents. We model these dynamics using a transition probability function, $T(\vec{s}, \vec{a}, \vec{s}')$, to capture the possibly uncertain effects of these actions on the subsequent state:

$$\Pr(\vec{s}^{t+1} = \vec{s}' | \vec{s}^t = \vec{s}, \vec{a}^t = \vec{a}) = T(\vec{s}, \vec{a}, \vec{s}') \qquad (1)$$

For example, the bully's attack on the victim impacts the power of the bully, the power of the victim, etc. The distribution over the bully's and victim's changes in power is a function of the relative powers of the two—e.g., the larger the power gap that the bully enjoys over the victim, the more likely the victim is to suffer a big loss in power.

2.2 Preferences

PsychSim's decision-theoretic framework represents an agent's incentives for behavior as a reward function that maps the state of the world into a real-valued evaluation of benefit for the agent. We separate components of this reward function into two types of subgoals. A goal of **Minimize/maximize** `feature(agent)` corresponds to a negative/positive reward proportional to the value of the given state feature. For example, an agent can have the goal of maximizing its own power. A goal of **Minimize/maximize** `action(actor, object)` corresponds to a negative/positive reward proportional to the number of matching actions performed. For example, the teacher may have the goal of minimizing the number of times any student teases any other.

We can represent the overall preferences of an agent, as well as the relative priority among them, as a vector of weights, $\vec{g}$, so that the product, $\vec{g} \cdot \vec{s}^t$, quantifies the degree of satisfaction that the agent receives from the world, as represented by the state vector, $\vec{s}^t$. For example, in the school violence simulation, the bully's reward function consists of goals of maximizing `power(bully)`, minimizing `power(victim)`, and maximizing `laugh(onlookers, victim)`. By modifying the weights on the different goals, we can alter the motivation of the agent and, thus, its behavior in the simulation.

2.3 Beliefs about Others

As described by Sections 2.1 and 2.2, the overall decision problem facing a single agent maps easily into a partially observable Markov decision problem (POMDP) [Smallwood and Sondik, 1973]. Software agents can solve such a decision problem using existing algorithms to form their beliefs and then determine the action that maximizes their reward given those beliefs. However, we do not expect people to conform to such optimality in their behavior. Thus, we have taken the POMDP algorithms as our starting point and modified them in a psychologically motivated manner to capture more human-like behavior. This "bounded rationality" better captures the reasoning of people in the real-world, as well as providing the additional benefit of avoiding the computational complexity incurred by an assumption of perfect rationality.

Nested Beliefs

The simulation agents have only a *subjective* view of the world, where they form beliefs, $\vec{b}^t$, about what they *think* is the state of the world, $\vec{s}^t$. Agent A's beliefs about agent B have the same structure as the real agent B. Thus, our agent belief models follow a recursive structure, similar to previous work on game-theoretic agents [Gmytrasiewicz and Durfee, 1995]. Of course, the nesting of these agent models is potentially unbounded. However, although infinite nesting is required for modeling optimal behavior, people rarely use such

deep models [Taylor *et al.*, 1996]. In our school violence scenario, we found that 2-level nesting was sufficiently rich to generate the desired behavior. Thus, the agents model each other as 1-level agents, who, in turn, model each other as 0-level agents, who do *not* have any beliefs. Thus, there is an inherent loss of precision (but with a gain in computational efficiency) as we move deeper into the belief structure.

For example, an agent's beliefs may include its subjective view on states of the world: "The bully believes that the teacher is weak", "The onlookers believe that the teacher supports the victim", or "The bully believes that he/she is powerful." These beliefs may also include its subjective view on beliefs of other agents: "The teacher believes that the bully believes the teacher to be weak." An agent may also have a subjective view of the *preferences* of other agents: "The teacher believes that the bully has a goal to increase his power." It is important to note that we also separate an agent's subjective view of itself from the real agent. We can thus represent errors that the agent has in its view of itself (e.g., the bully believes himself to be stronger than he actually is).

Actions affect the beliefs of agents in several ways. For example, the bully's attack may alter the beliefs that agents have about the state of the world—such as beliefs about the bully's power. Each agent updates its beliefs according to its subjective beliefs about the world dynamics. It may also alter the beliefs about the bully's preferences and policy. We discuss the procedure of belief update in Section 2.4.

Policies of Behavior

Each agent's policy is a function, $\pi(\vec{b})$, that represents the process by which it selects an action or message based on its beliefs. An agent's policy allows us to model critical psychological distinctions such as reactive vs. deliberative behavior. We model each agent's real policy as a bounded lookahead procedure that seeks to maximize expected reward simulating the behavior of the other agents and the dynamics of the world in response to the selected action/message. Each agent i computes a quantitative value, $V_a(\vec{b}_i^t)$, of each possible action, a, given its beliefs, $\vec{b}_i^t$.

$$V_a(\vec{b}_i^t) = \vec{g}_i \cdot \vec{b}_i^t + \sum_{\vec{b}^{t+1}} V(\vec{b}^{t+1}) \Pr(\vec{b}^{t+1} | \vec{b}_i^{t+1}, a, \vec{\pi}_{\neg i}(b_i^{t+1})) \tag{2}$$

$$V(\vec{b}^t) = \vec{g}_i \cdot \vec{b}_i^t + \sum_{\vec{b}^{t+1}} V(\vec{b}^{t+1}) \Pr(\vec{b}^{t+1} | \vec{b}_i^{t+1}, \vec{\pi}(b_i^{t+1})) \tag{3}$$

Thus, an agent first uses the transition function, T, to project the immediate effect of the action, a, and then projects another N steps into the future, weighing each state against its goals, $\vec{g}$. At the first step, agent i uses its model of the policies of all of the other agents, $\pi_{\neg i}$, and, in subsequent steps, it uses its model of the policies of all agents, including itself, π. Thus, the agent is seeking to maximize the expected reward of its behavior as in a POMDP. However, PsychSim's agents are only boundedly rational, given that they are constrained, both by the finite horizon, N, of their lookahead and the possible error in their belief state, $\vec{b}$. By varying N for different agents, we can model entities who display different degrees of reactive vs. deliberative behavior in their thinking.

Stereotypical Mental Models

If we applied this full lookahead policy within the nested models of the other agents, the computational complexity of the top-level lookahead would quickly become infeasible as the number of agents grew. To simplify the agents' reasoning, these mental models are realized as simplified stereotypes of the richer lookahead behavior models of the agents themselves. For our simulation model of a bullying scenario, we have implemented mental models corresponding to *attention-seeking*, *sadistic*, *dominance-seeking*, etc. For example, a model of an attention-seeking bully specifies a high priority on increasing the approval (i.e., `support`) that the other agents have for it, a dominance-seeking bully specifies a high priority on increasing its power as paramount, and a bully agent specifies a high priority on hurting others.

These simplified mental models also include potentially erroneous beliefs about the policies of other agents. Although the real agents use lookahead exclusively when choosing their own actions (as described in Section 2.3), the agents *believe* that the other agents follow much more reactive policies as part of their mental models of each other. PsychSim models reactive policies as a table of "Condition⇒Action" rules. The left-hand side conditions may trigger on an *observation* of some action or a *belief* of some agent (e.g., such as the bully believing himself as powerful). The conditions may also be more complicated combinations of these basic triggers (e.g., a *conjunction* of conditions that matches when each and every individual condition matches).

The use of these more reactive policies in the mental models that agents have of each other achieves two desirable results. First, from a human modeling perspective, the agents perform a shallower reasoning that provides a more accurate model of the real-world entities they represent. Second, from a computational perspective, the direct action rules are cheap to execute, so the agents gain significant efficiency in their reasoning.

2.4 Modeling Influence and Belief Change

Messages

Messages are attempts by one agent to influence the beliefs of another. Messages have four components: source, recipients, subject, and content. For example, the teacher (source) could tell the bully (recipient) that the principal (subject of the message) will punish violence by the bully (content). Messages can refer to beliefs, preferences, policies, or any other aspect of other agents. Thus, a message may make a claim about a state feature of the subject ("the principal is powerful"), the beliefs of the subject ("the principal believes that he is powerful"), the preferences of the subject ("the bully wants to increase his power"), the policy of the subject ("if the bully thinks the victim is weak, he will pick on him"), or the stereotypical model of the subject ("the bully is selfish").

Influence Factors

A challenge in creating a social simulation is addressing how groups or individuals influence each other, how they update their beliefs and alter behavior based on any partial observation of, as well as messages from, others. Although many psychological results and theories must inform the modeling

of such influence (e.g., [Cialdini, 2001; Abelson *et al.*, 1968; Petty and Cacioppo, 1986]) they often suffer from two shortcomings from a computational perspective. First, they identify factors that affect influence but do not operationalize those factors. Second, they are rarely comprehensive and do not address the details of how various factors relate to each other or can be composed. To provide a sufficient basis for our computational models, our approach has been to distill key psychological factors and map those factors into our simulation framework. Here, our decision-theoretic models are helpful in quantifying the impact of factors in such a way that they can be composed. Specifically, a survey of the social psychology literature identified the following key factors:

Consistency: People expect, prefer, and are driven to maintain consistency, and avoid cognitive dissonance, between beliefs and behaviors.

Self-interest: The inferences we draw are biased by self-interest (e.g., motivated inference) and how deeply we analyze information in general is biased by self-interest.

Speaker's Self-interest: If the sender of a message benefits greatly if the recipient believes it, there is often a tendency to be more critical and for influence to fail.

Trust, Likability, Affinity: The relation to the source of the message, whether we trust, like or have some group affinity for him, all impact whether we are influenced by the message.

Computational Model of Influence

To model such factors in the simulation, one could specify them exogenously and make them explicit, user-specified factors for a message. This tactic is often employed in social simulations where massive numbers of simpler, often identical, agents are used to explore emergent social properties. However, providing each agent with quantitative models of itself and, more importantly, of other agents gives us a powerful mechanism to model this range of factors in a principled way. We model these factors by a few simple mechanisms in the simulation: *consistency*, *self-interest*, and *bias*. We can render each as a quantitative function of beliefs that allows an agent to compare alternate candidate belief states (e.g., an agent's original $\vec{b}$ vs. the $\vec{b}'$ implied by a message).

Consistency is an evaluation of the degree to which a potential belief agreed with prior observations. In effect, the agent asks itself, "If this belief holds, would it better explain the past better than my current beliefs?". We use a Bayesian definition of consistency based on the relative likelihood of past observations given the two candidate sets of beliefs (e.g., my current beliefs with and without believing the message). An agent assesses the quality of the competing explanations by a re-simulation of the past history. In other words, it starts at time 0 with the two worlds implied by the two candidate sets of beliefs, projects each world forward up to the current point of time, and computes the probability of the observation it received. In particular, the consistency of a sequence of observations, $\omega^0, \omega^1, \ldots$, with a given belief state, $\vec{b}$, corresponds to:

$$\begin{aligned} &\text{consistency}(\vec{b}^t, [\omega^0, \omega^1, \ldots, \omega^{t-1}]) \\ &= \Pr\left([\omega^0, \omega^1, \ldots, \omega^{t-1}] \,\middle|\, \vec{b}^t\right) \\ &\propto \sum_{\tau=0}^{t-1} \sum_{a \in A} V_a(\vec{b}^\tau) \Pr(\omega^\tau | a, \vec{b}^\tau) \end{aligned} \quad (4)$$

The value function, V, computed is with respect to the agent performing the action at time τ. We are summing the value of the observed action to the acting agent, given the set of beliefs under consideration. The higher the value, the more likely that agent is to have chosen the observed action, and, thus, the higher the degree of consistency.

Self-interest is similar to consistency, in that the agent compares two sets of beliefs, one which accepts the message and one which rejects it. However, while consistency evaluates the past, we compute self-interest by evaluating the future using Equation 3. An agent can perform an analogous computation using its beliefs about the sender's preferences to compute the sender's self-interest in sending the message.

Bias factors represent subjective views of the message sender that influence the receiver's acceptance/rejection of the message. We treat support (or affinity) and trust as such a bias on message acceptance. Agents compute their support and trust levels as a running history of their past interactions. In particular, one agent increases (decreases) its trust in another, when the second sends a message that the first decides to accept (reject). Similarly, an agent increases (decreases) its support for another, when the second selects an action that has a high (low) reward, with respect to the preferences of the first. In other words, if an agent selects an action a, then the other agents modify their support level for that agent by a value proportional to $\vec{g} \cdot \vec{b}$, where $\vec{g}$ corresponds to the goals and $\vec{b}$ the new beliefs of the agent modifying its support.

Upon receiving any information (whether message or observation), an agent must consider all of these various factors in deciding whether to accept it and how to alter its beliefs (including its mental models of the other agents). For a message, the agent determines acceptance using a weighted sum of the five components: consistency, self-interest, speaker self-interest, trust and support. Whenever an agent observes an action by another, it checks whether the observation is consistent with its current beliefs (including mental models). If so, no belief change is necessary. If not, the agent evaluates alternate mental models as possible new beliefs to adopt in light of this inconsistent behavior. Agents evaluate these possible belief changes using the same weighted sum as for messages.

Each agent's decision-making procedure is sensitive to these changes that its actions may trigger in the beliefs of others. Each agent accounts for the others' belief update when doing its lookahead, as Equations 2 and 3 project the future beliefs of the other agents in response to an agent's selected action. Similar to work by [de Rosis *et al.*, 2003] this mechanism provides PsychSim agents with a potential incentive to deceive, if doing so leads the other agents to perform actions that lead to a better state for the deceiving agent.

We see the computation of these factors as a toolkit for the user to explore the system's behavior under existing theories, which we can encode in PsychSim. For example, the elaboration likelihood model (ELM) [Petty and Cacioppo, 1986] argues that the way messages are processed differs according to the relevance of the message to the receiver. High relevance or importance would lead to a deeper assessment of the message, which is consistent with the self-interest calculations our model performs. PsychSim's linear combination of factors is roughly in keeping with ELM because self-interest values of high magnitude would tend to dominate.

3 PsychSim in Operation

The research literature on childhood aggression provides interesting insight into the role that theory of mind plays in human behavior. Investigations of bullying and victimization [Schwartz, 2000] have identified four types of children; we focus here on *nonvictimized aggressors*, those who display proactive aggression due to positive outcome expectancies for aggression. Children develop expectations on the likely outcomes of aggression based on past experiences (e.g., did past acts of aggression lead to rewards or punishment). This section describes the results of our exploration of the space of different nonvictimized aggressors and the effectiveness of possible intervention strategies in dealing with them.

3.1 Scenario Setup

The user sets up a simulation in PsychSim by selecting generic agent models that will play the roles of the various groups or individuals to be simulated and specializing those models as needed. In our bullying scenario, we constructed generic bully models that compute outcome expectancies as the expected value of actions (V_a from Equation 2). Thus, when considering possible aggression, the agents consider the immediate effect of an act of violence, as well as the possible consequences, including the change in the beliefs of the other agents. In our example scenario, a bully has three subgoals that provide incentives to perform an act of aggression: (1) to change the power dynamic in the class by making himself stronger, (2) to change the power dynamic by weakening his victim, and (3) to earn the approval of his peers (as demonstrated by their response of laughter at the victim). Our bully agent models the first incentive as a goal of maximizing `power(bully)` and the second as minimizing `power(victim)`, both coupled with a belief that an act of aggression will increase the former and decrease the latter. The third incentive seeks to maximize the `laugh` actions directed at the victim, so it must consider the actions that the other agents may take in response.

For example, a bully motivated by the approval of his classmates would use his mental model of them to predict whether they would laugh along with him. We implemented two possible mental models of the bully's classmates: *encouraging*, where the students will laugh at the victim, and *scared*, where the students will laugh only if the teacher did not punish them for laughing last time. Similarly, the bully would use his mental model of the teacher to predict whether he will be punished or not. We provide the bully with three possible mental models of the teacher: *normal*, where the teacher will punish the bully in response to an act of violence; *severe*, where the teacher will more harshly punish the bully than in the *normal* model; and *weak*, where the teacher never punishes the bully.

The relative priorities of these subgoals within the bully's overall reward function provide a large space of possible behavior. When creating a model of a specific bully, PsychSim uses a fitting algorithm to automatically determine the appropriate weights for these goals to match observed behavior. For example, if the user wants the bully to initially attack a victim and the teacher to threaten the bully with punishment, then the user specifies those behaviors and the model parameters are fitted accordingly [Pynadath and Marsella, 2004]. This degree of automation significantly simplifies simulation setup. In this experiment, we selected three specific bully models from the overall space: (1) *dominance-seeking*, (2) *sadistic*, and (3) *attention-seeking*, each corresponding to a goal weighting that favors the corresponding subgoal.

3.2 Experimental Results

PsychSim allows one to explore multiple tactics for dealing with a social issue and see the potential consequences. Here, we examine a decision point for the teacher after the bully has attacked the victim, followed by laughter by the rest of the class. At this point, the teacher can punish the bully, punish the whole class (including the victim), or do nothing. We explore the impact of different types of proactive aggression by varying the type of the bully, the teacher's decision to punish the bully, the whole class, or no one, and the mental models that the bully has of the other students and the teacher.

A successful outcome is when the bully does not choose to act out violently toward the victim the next time around. By examining the outcomes under these combinations, we can see the effects of intervention over the space of possible classroom settings. Table 1 shows all of the outcomes, where we use the "*" wildcard symbol to collapse rows where the outcome was the same. Similarly, a row with "¬severe" in the **Teacher** row spans the cases where the bully's mental model of the teacher is either *normal* or *weak*.

We first see that the PsychSim bully models meets our intuitive expectations. For example, we see from Table 1 that if the bully thinks that the teacher is too weak to ever punish, then no immediate action by the teacher will change the bully from picking on the victim. Thus, it is critical for the teacher to avoid behavior that leads the bully to form such mental models. Similarly, if the bully is of the *attention-seeking* variety, then punishment directed at solely himself will not dissuade him, as he will still expect to gain peer approval. In such cases, the teacher is better off punishing the whole class.

We can see more interesting cases as we delve deeper. For example, if we look at the case of a *sadistic* bully when the teacher punishes the whole class, we see that bully can be dissuaded only if he thinks that the other students will *approve* of his act of violence. This outcome may seem counterintuitive at first, but the *sadistic* bully is primarily concerned with causing suffering for the victim, and thus does not mind being punished if the victim is punished as well. However, if the bully thinks that the rest of the class is *encouraging*, then the teacher's punishment of the whole class costs him peer

Bully Type	Punish Whom?	Model of Students	Model of Teacher	Success?
Sadistic	bully	*	¬severe	N
			severe	Y
	class	scared	*	N
		encouraging	¬severe	N
			severe	Y
	no one	*	¬severe	N
			severe	Y
Attention Seeking	bully	*	*	N
	class	scared	weak	N
			normal	Y
			severe	Y
		encouraging	*	N
	no one	*	*	N
Dominance Seeking	*	*	weak	N
			normal	Y
			severe	Y

Table 1: Outcomes of intervention strategies

approval. On the other hand, if the bully thinks that the rest of the class is already *scared*, so that they will not approve of his violence, then he has no peer approval to lose.

Such exploration can offer the user an understanding of the potential pitfalls in implementing an intervention strategy. Rather than providing a simple prediction of whether a strategy will succeed or not, PsychSim maps out the key conditions, in terms of the bully's preferences and beliefs, on which a strategy's success depends. PsychSim provides a rich space of possible models that we can systematically explore to understand the social behavior that arises out of different configurations of student psychologies. We are continuing to investigate more class configurations and the effects of possible interventions as we expand our models to cover all of the factors in school aggression identified in the literature.

4 Conclusion

We have discussed PsychSim, an environment for multi-agent simulation of human social interaction that employs a formal decision-theoretic approach using recursive models. This approach allows us to model phenomena rarely if at all addressed in simulated worlds. Within PsychSim, we have developed a range of technology to simplify the task of setting up the models, exploring the simulation, and analyzing results. This includes new algorithms for fitting multi-agent simulations. There is also an ontology for modeling communications about theory of mind. Finally, there is an analysis/perturbation capability that supports the iterative refinement of the simulation by reporting sensitivities in the results, as well as potentially interesting perturbations to the scenario. We have exploited the recursive models to provide a psychologically motivated computational model of how agents influence each other's beliefs. We believe PsychSim has a range of innovative applications, including computational social science and the model of social training environments. Our current goals are to expand the exploration already begun in the school violence scenario and begin evaluating the application of PsychSim there and in these other areas.

References

[Abelson *et al.*, 1968] R.P. Abelson, E. Aronson, W.J. McGuire, T.M. Newcomb, M.J. Rosenberg, and P.H. Tannenbaum, eds. *Theories of Cognitive Consistency: A Sourcebook*. Rand McNally, Chicago, IL, 1968.

[Cialdini, 2001] R. Cialdini. *Influence: Science and Practice*. Allyn and Bacon, Boston, MA, 2001.

[de Rosis *et al.*, 2003] F. de Rosis, C. Castelfranchi, V. Carofiglio, and G. Grassano. Can computers deliberately deceive? A simulation tool and its application to Turing's imitation game. *Computational Intelligence*, 19(3):253–263, 2003.

[Gmytrasiewicz and Durfee, 1995] P.J. Gmytrasiewicz and E.H. Durfee. A rigorous, operational formalization of recursive modeling. In *ICMAS*, pp. 125–132, 1995.

[Liebrand *et al.*, 1998] W. Liebrand, A. Nowak, and R. Hegselmann, editors. *Computer Modeling of Social Processes*. Sage, London, UK, 1998.

[Marsella *et al.*, 2000] S.C. Marsella, W.L. Johnson, and C. LaBore. Interactive pedagogical drama. In *Agents*, pp. 301–308, 2000.

[Marsella *et al.*, 2004] S.C. Marsella, D.V. Pynadath, and S.J. Read. PsychSim: Agent-based modeling of social interactions and influence. In *ICCM*, pp. 243–248, 2004.

[Petty and Cacioppo, 1986] R. Petty and J. Cacioppo. *Communication and persuasion: Central and peripheral routes to attitude change*. Springer, New York, NY, 1986.

[Prietula *et al.*, 1998] M. Prietula, K. Carley, and L. Gasser, eds. *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press, Menlo Park, CA, 1998.

[Pynadath and Marsella, 2004] D.V. Pynadath and S.C. Marsella. Fitting and compilation of multiagent models through piecewise linear functions. In *AAMAS*, pp. 1197–1204, 2004.

[Pynadath and Tambe, 2002] D.V. Pynadath and M. Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *AAMAS*, pp. 873–880, 2002.

[Schwartz, 2000] D. Schwartz. Subtypes of victims and aggressors in children's peer groups. *Journal of Abnormal Child Psychology*, 28:181–192, 2000.

[Smallwood and Sondik, 1973] R.D. Smallwood and E.J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[Taylor *et al.*, 1996] J. Taylor, J. Carletta, and C. Mellish. Requirements for belief models in cooperative dialogue. *User Modeling and User-Adapted Interaction*, 6:23–68, 1996.

[Whiten, 1991] A. Whiten, editor. *Natural Theories of Mind*. Basil Blackwell, Oxford, UK, 1991.

The Altricial-Precocial Spectrum for Robots *

Aaron Sloman
http://www.cs.bham.ac.uk/~axs/
School of Computer Science
University of Birmingham. UK

Jackie Chappell
http://www.biosciences.bham.ac.uk/staff/staff.htm?ID=90
School of Biosciences
University of Birmingham, UK

Abstract

Several high level methodological debates among AI researchers, linguists, psychologists and philosophers, appear to be endless, e.g. about the need for and nature of representations, about the role of symbolic processes, about embodiment, about situatedness, about whether symbol-grounding is needed, and about whether a robot needs any knowledge at birth or can start simply with a powerful learning mechanism. Consideration of the variety of capabilities and development patterns on the precocial-altricial spectrum in biological organisms will help us to see these debates in a new light.

1 Introduction

Attempts to design intelligent systems often encounter two objections: (a) for non-trivial systems the task is impossibly difficult, and (b) a system that has been designed is not intelligent since everything it does is a result of the designer's decisions, not its own. These and similar arguments lead some people to the conclusion that intelligent systems must start off without any knowledge, and acquire what knowledge and skills they need by interacting with the environment, possibly guided by a teacher, using only general learning capabilities.

This view assumes that something like human adult intelligence could be a product of neonates born with very little innate knowledge, but possessing a powerful general-purpose learning mechanism, e.g. reinforcement learning using positive and negative reinforcement based on drives, needs, and aversions. Then all knowledge would gradually be built up by continual shaping of internal and external responses to various combinations of internal and external stimuli. There have been several AI robotics projects and synthetic agent projects based on this line of thought (e.g. [Grand, 2004; McCauley, 2002; Weng, 2004], among others.)

This view ignores some powerful genetic influences in biological organisms. Most species show a mosaic of skills and abilities some of which develop mainly under genetic influence, while others grow (usually gradually) as a result of learning and experience. Even learning mechanisms can have constraints tailored by natural selection to the probability of certain events co-occurring in their environment. For example, Domjan and Wilson [1972] showed that rats readily learned to associate the taste of water with sickness, and ringing bells with electric shock, but could not learn to associate noise with sickness or taste with shock.

Our ability to design artificial systems could benefit from a better understanding of the very varied spectrum of cases found in nature. At one extreme are biological species, often labelled 'precocial', whose young are born or hatched relatively well developed and able to cope with tasks that are far beyond what our current robots can do: newly hatched chickens look for food which they peck, and new-born deer can, within hours, find a nipple to feed from, and run with the herd. At another extreme (often labelled 'altricial'), animals are born or hatched helpless and underdeveloped, yet sometimes grow into adults with far more impressive cognitive skills than their precocial cousins, e.g. humans, hunting mammals and nest-building birds. How?

Precocial skills that appear in a relatively constant form in different environments can be fine-tuned by the individual experience of the animal. In contrast, altricial capabilities can vary widely within a species, as languages do between cultures, and as many kinds of competence do from one generation to another within a culture. Many children now have knowledge and skills beyond the wildest dreams of their parents at the same age. It is implausible that a combination of environmental differences and some simple general learning mechanism, or any known learning algorithms, could produce such variation. Evolution seems to have found something more powerful, which supports and enhances cultural evolution.

We propose that in addition to knowledge-free reinforcement learning mechanisms and the unlearned kinds of knowledge acquisition (like maturation, and peri-natally generated precocial skills), there are capabilities produced by powerful mechanisms characteristic of the altricial end of the spectrum, that explain how adult members of the same species can have hugely varying knowledge and skills. These mechanisms (a) aquire many discrete chunks of knowledge through play and exploratory behaviour which is not directly reinforced, and (b) combine such chunks in novel ways both in solving problems and in further play and exploration. This process

*Many thanks to colleagues on the EC CoSy project.

depends on a collection of very abstract 'internal' skills and knowledge with a strong innate component – though the collection can develop over time, especially in humans.

So evolution provides much information that animals do not have to learn: some used in overt behaviour in precocial skills and some in more abstract internal behaviour (e.g. skill-bootstrapping behaviour) towards the altricial end. We present a number of conjectures about how this happens and discuss implications for robot designers.

1.1 Precocial vs. altricial developmental mode

Conjecture 1: *There is no sharp altricial/precocial distinction between species, nor a continuum of cases, but a spectrum of capabilities, with many discontinuities. Each species exhibits a unique collection of skills from different parts of the spectrum. A similar variety of designs may be useful in robots.*

A number of factors determine the extent to which a behaviour or skill is modified by the animal's own experience. The outcomes of these factors map very approximately on to the precocial and altricial developmental modes, though altricial species may have some precocial skills and *vice versa*. Analysis of these issues will prove valuable in deciding which sorts of robots will need 'precocial' capabilities, which 'altricial' capabilities, and what sorts of mechanisms are needed.

Factors promoting precocial skills and knowledge

The following will tend to favour evolution of precocial skills over altricial ones (if both are possible).

- **Opportunity**: If it would be very difficult for the animal to learn a skill given the constraints of its environment the skill has to be precocial. e.g. young blackcaps migrating for the first time on their own [Helbig, 1996], and the species-specific 'password' vocalisation in a brood parasite, the brown-headed cowbird [Hauber *et al.*, 2001], which is raised by other species.
- **Risk**: Learning depends on exposure to certain stimuli at the right time. If failing to perform a vital skill correctly would result in animals dying before reproduction, the skill is more likely to be precocial. e.g. predator recognition in Atlantic salmon, [Hawkins *et al.*, 2004], flight in cliff-nesting altricial birds.
- **Time constraints**: If learning would take too long to meet a need, a skill is more likely to be precocial. e.g. running in wildebeest calves, who must keep up with the herd soon after birth.

The young of species with many precocial skills start highly competent, requiring little or no parental care other than a degree of protection from predators.[1] They appear to have a great deal of information about structures in the world, how to perceive them, and how to behave in relation to them, even if they can be changed later in subtle ways as a result of adaptive processes, reinforcement learning or the like.

Since evolution can, and often does, provide huge amounts of information in the form of knowledge and skills. should not a large amount of precocial information be the norm, if all those millions of years of evolution are not to be wasted?

[1]Note, however, that species with precocial young may have to invest more resources in their young pre-natally.

Factors promoting altricial skills and knowledge

In conditions opposite to those outlined above (abundant opportunity for learning, low risk outcome, no time constraints), altricial skills will tend to be favoured. In addition, the following factors are also important:

- **Unpredictability**: If the environment is very variable or unpredictable, there will be no consistent pattern for natural selection to act on. The fact that fleeing prey are relatively unpredictable, while grass tends to be highly predictable might explain why predatory mammals and birds tend to have altricial developmental patterns, while grazers have many precocial skills.
- **New niches**: This is related to unpredictability. If an animal encounters conditions which have never been encountered before in the evolutionary history of its species, individual learning will be the only possible response. Some niche variation can be accommodated by individual adaptation and calibration. But not when quite new concepts and new behavioural structures are required, e.g. if opponents often invent new weapons, or new defences against old weapons and strategies.

If altricial skills provide so much flexibility and diversity of responses, why don't more animals show altricial skills? A partial answer is cost: neural tissue is expensive to build and maintain (in humans, the brain consumes about 20% of the basal metabolic rate). Learning can be time-consuming and risky. Young animals need to be protected from the harsh realities of life while acquiring altricial skills, so the intensity and duration of parental care can also be significant. The extended commitment of resources by parents to guarding, feeding, and in some cases helping their offspring to gain hunting skills, can both endanger the adults (make them more vulnerable to predation, deprive them of food given to the offspring), and reduce their breeding frequency.

1.2 The altricial-precocial spectrum for robots

Analysis of the altricial/precocial trade-offs already explored by evolution can provide a basis for a new understanding of the varied requirements and tradeoffs in robot design. Applications where tasks and environments are fairly static and machines need to be functional quickly, and where mistakes during learning could be disastrous (e.g. flight control) require precocial skills (with some adaptation and self-calibration), whereas others require altricial capabilities, e.g. where tasks vary widely and change in complex ways over time, and where machines need to learn to cope without being sent for re-programming (e.g. robots caring for the elderly). Architectures, mechanisms, forms of representation, and types of learning may differ sharply between the two extremes. And the end results of altricial learning by the same initial architecture may differ widely, as happens in humans.

1.3 Combining precocial and altricial skills

Precocial skills can provide sophisticated abilities at birth. Altricial capabilities add the potential to adapt to changing needs and opportunities. So it is not surprising that many species have both. However evolution seems to have discovered something deeper than just a mixture of some innate modules and some learning capabilities. Humans, and presumably some other animals, seem to have a *hybrid* architecture where the mechanisms for learning are

themselves a mixture of precocial and altricial mechanisms.

Conjecture 2: *Evolution 'discovered' and deployed the power of an architecture towards the more sophisticated, altricial, end of the spectrum, based not on specific knowledge about the environment, but on mechanisms using abstract knowledge about ways in which physically possible environments can differ. So powerful altricial learning arises from sophisticated genetically determined (i.e. precocial) learning mechanisms, which may be expanded during development.*

Instead of starting off able to recognise and react to specific stimuli, an altricial learner might have mechanisms that use knowledge about how to generate varied exploratory actions and how to seek recognisable patterns.[2] Such 'playful' actions reveal which among possible patterns are instantiated in the environment, and which ways of combining various kinds of pre-programmed or learnt action capabilities have effects that are 'interesting' not merely in terms of useful or harmful effects, but also according to criteria such as symmetry, elegance and power (e.g. producing big effects with little effort, or making something work that previously failed), thereby bootstrapping a wide variety of competences through interactions with the environment.[3]

2 What are the mechanisms?

This may sound like standard reinforcement learning, except for some innate biases to learn about certain classes of things only a subset of which are actually learnt under the influence of any particular environment [Cummins and Cummins, 2005]. But such mechanisms produce only gradual changes in behaviour, and it seems that at least some species, including humans, apes, and some corvids are — in addition — capable of kinds of discontinuous learning where hard problems are solved creatively by recombining previously acquired concepts, knowledge chunks and action chunks.

2.1 Gradual change *vs* learning chunks.

Conjecture 3: *Altricial learning is not always driven by biological needs and desires (e.g. for food, shelter, escape, mating, etc.) using rewards and punishments to drive gradual change through positive and negative feedback; instead the key feature is constant (often unmotivated) experimentation with external and internal actions during which re-usable* chunks *of information are learnt about what can be done and what can occur, and the preconditions and effects. Chunks are discovered both in passively perceived inputs (e.g. using self-organising classifiers to chunk sensory inputs) and also in behaviour patterns and their consequences.*

Re-use, in new combinations, of previously stored chunks, might explain the creative problem-solving of New Caledonian crows [Chappell and Kacelnik, 2004; Weir *et al.*, 2002]) and chimpanzees [Kohler, 1927], the ability of social animals to absorb a culture; and, in humans, the ability to learn and use a rich and highly structured language. A key feature of such processes is learning new ontologies (discussed further below.)

2.2 Combinatorics: the origins of syntax

For these kinds of learning, mechanisms that gradually change weights or probabilities do not suffice: instead some chunks once learnt are explicitly referenced and available to be combined with other chunks, so as *immediately* to produce external and internal states and processes that are qualitatively (e.g. structurally) *novel* for the individual.[4]

Conjecture 4: *Some altricial species can not only store, label and categorise input and output 'chunks' that can be re-used later, but can also combine them to form larger chunks that are explored, and if found 'interesting' also stored, labelled, categorised, etc. so that they become available as new, larger units for future actions. A simple example would be picking something up and transferring something to one's mouth, being combined to form a new unit to pick up an object and bring it to one's mouth. Another example is combining moving right and moving left in a repeated sequence which can produce changes of visual input that provide useful information about 3-D structure [Philipona* et al.*, 2003].*

A simple demonstration of how this can enormously reduce search spaces, originally suggested by Oliver Selfridge over 20 years ago can be found in a tutorial program file provided with the Poplog system.[5] In that example the 'agent' is given tasks which it attempts to solve using its primitive actions, previously learnt useful actions, and combinations of primitive and previously learnt actions formed by simple syntactic operations, including use of a 'repeat construct'. If it is given a complex task initially it searches for a solution and gives up after a certain length of search. But as it gradually builds up useful re-usable chunks the space of actions in which it searches in a given number of steps gets more and more complex. If suitably trained by giving it the right problems to 'play' with it can suddenly become able to solve a class of quite difficult 'counting' problems.

This is a very simple example, but it illustrates a general point: that an altricial learning agent may need to be able to learn many things that are not intrinsically rewarding, but which are stored as a result of innately driven exploratory drives and then provide a basis for newer more complex explorations and learning – which could be specially well suited to the environment in which the simpler chunks (including perceptual and action chunks) have been learnt.

We have here a kind of syntactic competence which generates structures with compositional semantics.

Conjecture 5: *Not only external, but also internal actions – e.g. actions of goal formation, problem-solving, concept formation – may be composed, forming layers of complexity.*

If the methods of syntactic composition are themselves subject to the same process then the result may be production of ontological layers in many parts of the system including perceptual layering, action layering and various kinds of internal layering of control and description. Different kinds of competence may produce different kinds of layering.

[2] Manuela Viezzer's PhD research on discovery of affordances illustrates this. See http://www.cs.bham.ac.uk/~mxv

[3] For more on interestingness see [Colton *et al.*, 2000]

[4] [Boden, 1990] distinguishes historical and personal creativity.

[5] http://www.cs.bham.ac.uk/research/poplog/teach/finger

2.3 Possible benefits of the hybrid design

There may be several different biological advantages gained by the hybrid architecture over the more obvious 'precocial' and 'altricial' architectures, which can be framed in terms of ways of overcoming resource limitations.

Advantage over precocial strategies

- The capabilities required in adults of some species may be too rich to be encoded in the genome if there is a 'space' limitation). (Genetic information capacity as a scarce resource)
- The evolutionary history of some species may not have provided any contexts in which certain currently useful capabilities could have been selected for, or may constrain the direction in which evolution can proceed (i.e. phylogenetic constraints) (Evolutionary history is a biological resource, and may have limitations like any other biological resource.)

Advantage over altricial strategies

- Although rapid, one-trial reinforcement learning is possible, reinforcement generally requires multiple exposures to a pairing of stimulus and reinforcer, and such slower forms of learning (involving gradual, adaptive, shaping, processes) may not be able to cope with the very varied environments and challenges facing adults of some species. (Time to learn as a limited resource)
- For some new problems an individual may have to produce a discontinuous change from previous behaviours, i.e. novel, creative behaviour, as opposed to interpolating or extrapolating in a space that is already well explored. That is because prior learning does not always take the individual 'close' to problems that can occur. (Individual learning opportunities as a limited resource)
- The benefits of precocial strategies may be outweighed by the risks of over-commitment in a particular direction. If there are constraints on types of learning that make certain sorts of developmentally fixed competence incompatible with kinds of learning e.g. because the prior commitment provides the wrong building-blocks for the required new competence and because it is difficult to re-wire or otherwise re-implement the required building blocks. In that case, the advantages of being precocial may be best abandoned in favour of the advantages of flexibility: provided that parents are available to care for helpless young. (Flexibility as a limited resource.)

All this points to the need for learning, decision-making, and acting capabilities that support discontinuous development either in the evolution of the species, or in learning by the individual, without adversely affecting fitness.

2.4 Effect of body and physical environment

Although innate altricial bootstrapping mechanisms encode a great deal of abstract know-how enabling specific kinds of development, the space of possible learnt concepts and action schemas permitted by such mechanisms (the 'epigenetic landscape') may be very large, leaving the environment scope to determine which subset of that space is actually learnt in both early and late development. Even within one environment, different kinds of bodies and different kinds of initial brain mechanisms will produce variation: (a) in the kinds of chunks that can be learnt, (b) in the number of different chunks that can be learnt (c) in the variety of forms of combination of previously learnt chunks.

Having two hands with five fingers each with several (more or less independently controllable) joints can produce huge combinatorial possibilities. Having grippers that can move independently of the eyes (i.e. hands vs beaks) can make a huge difference to perceivable patterns. Different designs for mouths (or beaks) and tongues or vocal mechanisms may also produce different combinatorial spaces to be explored.

A fixed learning mechanism may be able to acquire different sorts of reusable chunks and perhaps even different ways of combining chunks, when it is combined with different physical bodies, or different physical environments. Compare the different behavioural and conceptual results of exploratory play in children surrounded by rocks, sand, sticks, stones, mud and water, on the one hand and children surrounded by stackable blocks and cups, dolls, toy animals, meccano sets, lego sets, computers with mice and keyboards, on the other.[6]

Such learners start with both a control system that performs systematically varied internal and external operations driving these mechanisms, as well as mechanisms for learning, storing, labelling and re-using various kinds of chunks. (Compare macro-formation mechanisms in planners, e.g. HACKER [Sussman, 1975] and SOAR's chunking.)

So the variety of sensory and motor chunks learnt will depend on (a) the physical design of the organism or robot, e.g. having independently movable limbs, digits, jaws, tongue, neck, head, eyeballs, ears, with motion changed in various ways, such as accelerating, decelerating, changing direction etc. (b) the richness of the organism's environment, particularly during development (e.g. learning in rats improves with environmental enrichment [Iuvone *et al.*, 1996]). (c) the genetically determined collection of internal operations on data-structures, especially (d) the 'syntactic' mechanisms available for combining old chunks into new structures[7] in perception (e.g. parsing), or in action (e.g. synthesising plans), and (e) whether those syntactic mechanisms can be applied not only to representations of external actions, but also to representations of internal actions such as operators used in problem formation, plan construction and debugging (as in Sussman's HACKER).

So different altricial species will differ in many ways related to these differences. In particular some may have much richer combinatorial competence than others.

Some of the processes are clearly amenable to cultural influences: e.g. what sorts of toys and games are found in the environment during early development will make a difference to which ontologies can be acquired. The processes could lead to different kinds of understanding and representation of space, time, motion and causality in different species, and perhaps in different cultures within a species.

After mechanisms for chunking and composing evolved in forms that we share with some other mammals, further

[6] An example of such exploration can be seen in this video of a child aged 11 months exploring affordances in a world with yogurt: http://www.cs.bham.ac.uk/~axs/fig/josh23_0040.mpg

[7] Arp [2005] has recently made a similar point, but citing only the special case of visualisation. [Cummins and Cummins, 2005] propose only innate learning biases not these innate meta-level modules.

typical evolutionary processes of duplication plus differentiation could have produced variants that were particularly suited to human language, which, if used internally, would have enormously amplified all the effects discussed above. But our discussion implies that a powerful form of *internal* language developed earlier, to support processes of altricial bootstrapping that we appear to share with some non-verbal altricial species [Sloman, 1979].

3 Symbol 'attachment', not 'grounding'

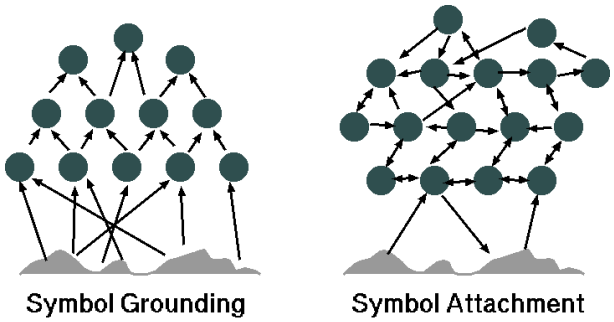
The existence of precocial species demonstrates that sophisticated visual and other apparatus need not be learned, implying that the semantic content of the information structures is somehow determined by pre-existing structures and how they are used. This eliminates, at least for such precocial species, one of the supposed explanations of where meaning comes from ("There is really only one viable route from sense to symbols: from the ground up" [Harnad, 1990]).

How then can innate abstract structures determine meaning, as we claim happens for precocial species? Our answer, presented in [Sloman, 1985; 1987], is inspired by ideas about models in meta-mathematics and proposals in philosophy of science about scientific terms that could not possibly get their meaning by abstraction from perceptual processes, (e.g. 'electron', 'quark', 'neutrino', 'gene', 'economic inflation', 'grammar').

A (consistent) formal system determines a non-empty class of possible (Tarskian) models, which may be either abstract mathematical objects or collections of objects, properties, relations, events, processes etc. in the world. For a given formal system not every such collection can be a model, though it may have many different models. As (non-redundant) axioms are added the set of possible models decreases. So the meaning becomes increasingly specific. A formal system can determine precise meanings for its theoretical terms, by ruling out the vast majority of possible things in the universe as referents.

There is typically residual ambiguity, but, as Carnap [1947] and other philosophers have shown, this residual indeterminacy of meaning may be eliminated, or substantially reduced, by the use of bridging rules or 'meaning postulates'. As Strawson [1959] noted, further ambiguity in reference to individuals can in some cases be eliminated or reduced by causal relations between the event of use of symbols and the things referred to.

We can call this process 'symbol attachment', because it is a process in which a complex symbolic structure with rich structurally-determined meaning can be pinned down to a particular part of the world. This is very different from the theory of symbol grounding which requires all meaning to be derived bottom up by abstraction from sensory experience of instances, which is actually a very old philosophical theory (concept empiricism) put forward hundreds of years ago by empiricist philosophers and refuted by Kant. Our impression is that many of the AI researchers who now use the phrase 'symbol grounding' have ignored its original meaning (arising out of concept empiricism) and actually use it, unwittingly, to refer to something like what we have called



There are many intermediate cases between pure symbol grounding and extreme symbol-attachment. Arrows on the left indicate abstraction of concepts from instances or sub-classes. On the right arrows indicate a mixture of constraints that help to determine possible interpretations (models) of the whole system and/or make predictions that enable the system to be used or tested.

'symbol attachment'. We suggest that in altricial species:
(a) symbol attachment, in which structures produced by a combination of evolution, and exploratory 'play' and problem solving, acquire most of their meaning from their formal structure, with perceptual/action bridging rules to reduce their ambiguity, is far more important than what is normally called 'symbol grounding', and
(b) learning of chunks and combinatory procedures tailored to the current environment but constrained by powerful innate boot-strapping mechanisms that evolved to cope with general features of the environment (e.g. concerning space, time, motion, causation, etc.) is far more powerful than most forms of learning now being studied.

4 Conclusion

The rapid, automatic, non-need-driven collection/creation of a store of labelled, reusable, perceptual and action chunks, along with 'syntactic' mechanisms for combinatorial extensions of those 'basic' chunks, provides a rich and extendable store of rapidly deployable cognitive resources, using mechanisms that will already sound familiar to many AI researchers who have worked on planning, problem solving, reasoning, 3-D vision, and language understanding, though we have said very little about how they might work.

It may be that if some of this happens while the brain is still growing the result can be 'compiled' into hardware structures that support further development in powerful ways, explaining why the process needs to start while the infant is still physically under-developed. (This is very vague.)

We are not saying that precocial species cannot learn and adapt but that the amount of variation of which they are capable is more restricted and the processes are much slower (e.g. adjusting weights in a neural net trained by reinforcement learning, as contrasted with constructing new re-usable components for combinatorially diverse perceptual and action mechanisms).

So we suggest that in addition to the experiments where we pre-design working systems with combinations of different sorts of competence in order to understand ways in which such competences might work and cooperate, we should also

begin the longer term exploration of architectures and mechanisms for a robot towards the altricial end of the spectrum described above, growing its architecture to fit opportunities, constraints and demands provided by the environment. Some further details regarding requirements for such architectures can be found in [Sloman *et al.*, 2005].

Readers with some knowledge of the history of philosophy will recognise all this as an elaboration of ideas put forward by I. Kant in his *Critique of Pure Reason* around 1780. He would have loved AI. Future tasks include relating this work to the ideas Piaget was trying to express, in his theories about a child's construction of reality. using inadequate conceptual tools (group theory, truth tables), and to the taxonomy of types of cognitive evolution in [Heyes, 2003].

Application domains where tasks and environments are fairly static and machines need to be functional quickly, require precocial skills (possibly including some adaptation and self-calibration), whereas others require altricial capabilities, e.g. where tasks and environments vary widely and change in complex ways over time, and where machines need to learn how to cope without being sent for re-programming. Architectures, mechanisms, forms of representation, and types of learning may differ sharply between the two extremes. And the end results of altricial learning by the same initial architecture may differ widely.

If all this is correct, it seems that after evolution discovered how to make physical bodies that grow themselves, it discovered how to make virtual machines that grow themselves. Researchers attempting to design human-like, chimp-like or crow-like intelligent robots will need to understand how. Whether computers as we know them can provide the infrastructure for such systems is a separate question.

References

[Arp, 2005] Robert Arp. Scenario visualization: One explanation of creative problem solving. *Journal of Consciousness Studies*, 12(3):31–60, 2005.

[Boden, 1990] M. A. Boden. *The Creative Mind: Myths and Mechanisms*. Weidenfeld & Nicolson, London, 1990.

[Carnap, 1947] R. Carnap. *Meaning and necessity: a study in semantics and modal logic*. Chicago University Press, Chicago, 1947.

[Chappell and Kacelnik, 2004] J Chappell and A Kacelnik. New Caledonian crows manufacture tools with a suitable diameter for a novel task. *Animal Cognition*, 7:121–127, 2004.

[Colton *et al.*, 2000] Simon Colton, Alan Bundy, and Toby Walsh. On the notion of interestingness in automated mathematical discovery. *Int. Journ. of Human-Computer Studies*, 53(3):351–375, 2000.

[Cummins and Cummins, 2005] D Cummins and R Cummins. Innate modules vs innate learning biases. *Cognitive Processing: International Quarterly Journal of Cognitive Science*, 3(3-4):19–30, 2005.

[Domjan and Wilson, 1972] M Domjan and N E Wilson. Specificity of cue to consequence in aversion learning in the rat. *Psychonomic Science*, 26:143–145, 1972.

[Grand, 2004] Steve Grand. *Growing Up with Lucy: How to Build an Android in Twenty Easy Steps*. Weidenfeld & Nicolson, London, 2004.

[Harnad, 1990] S. Harnad. The Symbol Grounding Problem. *Physica D*, 42:335–346, 1990.

[Hauber *et al.*, 2001] M E Hauber, S A Russo, and P W Sherman. A password for species recognition in a brood-parasitic bird. *Proceedings of the Royal Society of London B*, 268:1041–1048, 2001.

[Hawkins *et al.*, 2004] L A Hawkins, A E Magurran, and J D Armstrong. Innate predator recognition in newly-hatched atlantic salmon. *Behaviour*, 141:1249–1262, 2004.

[Helbig, 1996] A J Helbig. Genetic basis, mode of inheritance and evolutionary changes of migratory directions in palearctic warblers (aves: Sylviidae). *Journal of Experimental Biology*, 199:49–55, 1996.

[Heyes, 2003] C.M. Heyes. Four routes of cognitive evolution. *Psychological Review*, 110(713-727), 2003.

[Iuvone *et al.*, 1996] L Iuvone, M C Geloso, and E Dell'Anna. Changes in open field behavior, spatial memory, and hippocampal parvalbumin immunoreactivity following enrichment in rats exposed to neonatal anoxia. *Experimental Neurology*, 139:25–33, 1996.

[Kohler, 1927] W. Kohler. *The Mentality Of Apes*. Routledge & Kegan Paul, London, 1927. 2nd edition.

[McCauley, 2002] L. McCauley. *Neural Schemas: Toward a Comprehensive Mechanism of Mind*. PhD thesis, University of Memphis, 2002. http://csrg.cs.memphis.edu/mccauley/McCauley-Dis.pdf.

[Philipona *et al.*, 2003] D. Philipona, J.K.O'Regan, and J.-P. Nadal. Is there something out there? Inferring space from sensorimotor dependencies. *Neural Computation*, 15(9), 2003. http://nivea.psycho.univ-paris5.fr/Philipona/space.pdf.

[Sloman *et al.*, 2005] A. Sloman, R.L. Chrisley, and M. Scheutz. The architectural basis of affective states and processes. In M. Arbib and J-M. Fellous, editors, *Who Needs Emotions?: The Brain Meets the Machine*. Oxford University Press, Oxford, New York, 2005. Online at http://www.cs.bham.ac.uk/research/cogaff/0-INDEX03.html#200305.

[Sloman, 1979] A. Sloman. The primacy of non-communicative language. In M. MacCafferty and K. Gray, editors, *The analysis of Meaning: Informatics 5 Proceedings ASLIB/BCS Conference, Oxford, March 1979*, pages 1–15, London, 1979. Aslib. Online at http://www.cs.bham.ac.uk/research/cogaff/.

[Sloman, 1985] A. Sloman. What enables a machine to understand? In *Proc 9th IJCAI*, pages 995–1001, Los Angeles, 1985.

[Sloman, 1987] A. Sloman. Reference without causal links. In J.B.H. du Boulay, D.Hogg, and L.Steels, editors, *Advances in Artificial Intelligence - II*, pages 369–381. North Holland, Dordrecht, 1987.

[Strawson, 1959] P. F. Strawson. *Individuals: An essay in descriptive metaphysics*. Methuen, London, 1959.

[Sussman, 1975] G.J. Sussman. *A computational model of skill acquisition*. American Elsevier, 1975.

[Weir *et al.*, 2002] A A S Weir, J Chappell, and A Kacelnik. Shaping of hooks in New Caledonian crows. *Science*, 297(9 August 2002):981, 2002.

[Weng, 2004] J. Weng. SAIL and Dav Developmental Robot Projects, 2004. http://www.cse.msu.edu/~weng/research/LM.html.

PLANNING

Model Compilation for Real-Time Planning and Diagnosis with Feedback

Anthony Barrett
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, M/S 126-347, Pasadena, CA 91109
anthony.barrett@jpl.nasa.gov

Abstract

This paper describes MEXEC, an implemented micro executive that compiles a device model that can have feedback into a structure for subsequent evaluation. This system computes both the most likely current device mode from *n* sets of sensor measurements and the *n-1* step reconfiguration plan that is most likely to result in reaching a target mode – if such a plan exists. A user tunes the system by increasing *n* to improve system capability at the cost of real-time performance.

1 Introduction

Over the past decade spacecraft complexity has exploded with increasingly ambitions mission requirements. Relatively simple flyby probes have been replaced with more capable remote orbiters, and these orbiters are slowly becoming communications relay satellites for even more ambitious mobile landers like the current Mars Exploration Rover, the planned Mars Science Lab, and the suggested aerobot at Titan. With this increased complexity there is also an increased probability that components will break and in unexpected ways with subtle interactions. While traditional approaches hand-craft rule-based diagnosis and recovery systems, the difficulty in creating these rule bases quickly gets out of hand as component interactions become more subtle. Model-based approaches address this issue, but their acceptance has been retarded by the complexity of the underlying evaluation systems when compared with a simple rule evaluator whose performance is guaranteed to be linear in the number of rules [Darwiche, 2000].

This paper combines ideas from Livingston [Williams and Nayak, 1996] with results in knowledge compilation for diagnosis [Darwiche, 1998] and planning [Barrett, 2004] to create MEXEC, a micro executive that is both model-based and has an onboard evaluation system whose simplicity is comparable to that of a rule evaluator. This involves taking a device model and compiling it into a structure that facilitates determining both a system's current mode and how to reconfigure to a desired target mode in linear time. Thus the system's architecture consists of an offline compiler and an online evaluator (see figure 1).

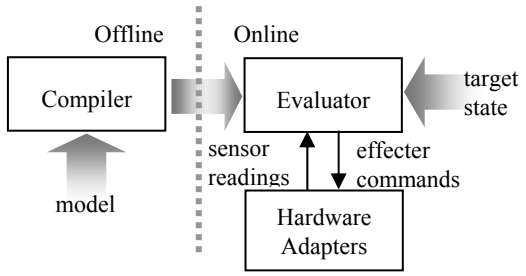


Figure 1: Online/Offline architecture for MEXEC

In addition an online performance guarantee, the global system reasoning during compilation handles feedback loops without any explicit mechanism. Also, the same compiled structure facilitates both mode identification and planning. Evaluating it one way computes the most likely current mode given *n* sets of measurements, and evaluating it another way computes the best *n-1* step reconfiguration plan given current and target modes – if such a plan exists.

This paper starts by defining our device representation language and compares it with Livingston's. It next presents a simple example device and shows how to compile it into an internal representation for linear time planning and diagnosis. The subsequent section shows how the structure is evaluated for both planning and diagnosis. To provide some realism, the implementation is described with a number of experiments. Finally the paper ends with a discussion of future work and conclusions.

2 Representing Devices

MEXEC's device modeling language is a simplified yet equally expressive variant of Livingstone's MPL. We model a device as a connected set of components, where each component operates in one of a number of modes. Essentially, each mode defines the relationships between a component's inputs and its outputs. More precisely, we use five constructs to define: types of connections, abstract relations, components with modes and relations between inputs and outputs, modules to define multiple component

subsystems, and the top-level system being diagnosed. The following conventions facilitate defining our syntax.

- A word in italic denotes a parameter, like *value*.
- Ellipsis denotes repetition, like *value*…
- Square brackets denote optional contents, like [*value*].
- A vertical bar denotes choice between options, like false | true.

With these conventions the entire language's syntax is defined in figure 2, which has constructs to respectively define connection types, well-formed formulas with arguments, user defined relations, components, modules, and a system. Just like Livingstone, system *name*'s structure is a connected set of components and modules with input/output ports, but unlike Livingston these inputs and outputs can form feedback loops and are statically defined in :connections sensed connections (:sensors) and the commanded connections (:affectors). While the *wff*s in components can only refer to variables in local component ports, the *wff*s in :constraint entries can refer variables defined in :structure elements as well as locally defined variables.

Another divergence from Livingstone involves the modeling of components. While the syntax is similar, the semantics revolves around the concept of cost. Essentially a mode's cost denotes now unlikely it is irrespective of any information, and a transition's cost denotes how unlikely it is when its preconditions hold. While getting costs from Livingston's probabilities is a simple matter of taking a probability's negative log, our formalism makes users directly specify costs to reflect that the number specified is manually guessed, just like a probability.

(**defvalues** *ctype* (*value*…))

wff → **:false** | **:true** | (**:not** *wff*) | (**:and** *wff*…) | (**:or** *wff*…) | (**=** *cname value*) | (**==** *cname cname*) | (*rname arg*…)

arg → *wff* | *cname* | *value*

(**defrelation** *rname* (*parameter*…) *wff*)

(**defcomponent** *stype*
  :ports ((*ctype cname*)…)
  :modes ((*mname* [**:cost** *int*] [**:model** *wff*])…)
  :transitions (({*mname* | *} -> *mname wff* [**:cost** *int*])…))

(**defmodule** *stype*
  :ports ((*ctype cname*)…)
  :connections ((*ctype cname*)…)
  :structure ((*stype sname* (*cname*…))…)
  [**:constraint** *wff*])

(**defsystem** *name*
  :sensors ((*ctype cname*)…)
  [**:affectors** ((*ctype cname*)…)]
  [**:connections** ((*ctype cname*)…)]
  :structure ((*stype sname* (*cname*…))…)
  [**:constraint** *wff*])

Figure 2: Syntax of device modeling language

3 Model Compilation

To provide an example of a modeled device, consider the following system, which has a single siderostat for tracking a star within an interferometer. This system is kept as simple as possible in order to facilitate its use as a running example in the rest of the paper. It starts by defining the values and then defines a component using the values and finally defines a system in terms of the component. One semantic restriction not mentioned in the syntax is that a definition cannot be used until after it has appeared. This keeps modelers from crafting infinite recursive definitions.

```
(defvalues boolean (false true))
(defvalues command (idle track none))
(defcomponent siderostat
   :ports  ((command in)(boolean valid))
   :modes ((Tracking :model (= valid true)
             :cost 20)
           (Idling :model (= valid false)
             :cost 5)
           (unknown :cost 1000))
   :transitions
   ((Tracking -> Idling (= in idle))
    (Idling -> Tracking (= in track))
    (* -> unknown :true :cost 1000))))
(defsystem tst
   :sensors ((boolean o))
   :affectors ((command c))
   :structure ((siderostat sw (c o))))
```

3.1 Model to CNF

Compiling a device model starts by taking a system definition and recursively expanding its modules using the defmodules until only components remain with an optional conjunctive constraint. Since the example lacked any defmodules, this step results in a single component called "sw" which is a siderostat in the following list, where c is a command effecter and o is a Boolean observation sensor

```
((siderostat sw (c o)))
```

As the example implies, name substitution occurs during the expansion. Inputs and outputs are replaced by actual parameter names – in and valid respectively become c and o. While not visible in the example, components get unique names by prefixing each structure element name with the current module name. For instance, if tst were a module, sw would become tst*sw, and the connection names would be similarly prefixed. This naming convention facilitates allowing module constraints that refer to connections and modes in substructure.

After determining components, their mode definitions are converted into a Boolean expression. This involves building an equation with the following form, where *sname* is the component's name, and each disjunctive entry is for a different mode *mname* with model *wff*. Within this form, notice the subscripts that vary from 0 to *n-1*.

```
(:and (:or (:not (= sname*mode mname))
           wff_i)… )
```

For example, if *n* were one in our example the resulting equation would be the following.

```
(:and (:or (:not (= sw*mode0 Tracking))
           (= o true))
      (:or (:not (= sw*mode0 Idling))
           (= o false)))
```

For higher *n*, the disjuncts are replicated for each step and transition disjuncts are added. Transition disjuncts take on the following form, where *sname* is the component name, X denotes the X^th component transition, frm_x/to_x respectively denote the transition's source/destination, and $wff_{x,i}$ denotes its precondition at step i. When frm_x is "*", the frm_x constraint is dropped to denote that the transition is always enabled.

```
(:or (:not (= sname*trans_i X))
     (:and (= sname*mode_i frm_x)
           (= sname*mode_i+1 to_x) wff_x,i))
```

Finally, these user-defined disjuncts are supplemented with system-defined disjuncts for not transitioning at all. They look respectively as follows, where the noop equation's size depends on the number of transitions in order to avoid choosing no transition when some transition is enabled.

```
(:or (:not (= sname*trans_i noop))
     (:and (== sname*mode_i sname*mode_i+1)
           (:or (:not (= sname*mode_i frm_x))
                (:not wff_x,i))… )))
```

Finally, with these constructs the compiler turns the a set of components into a single Boolean equation, conjoins that equation with the conjunction of :constraint *wff*s, and subsequently flatten the equation into a CNF form.

3.2 CNF to DNNF

Unfortunately finding a minimal satisfying assignment to a CNF equation is an NP-complete problem, and more compilation is needed to achieve linear time evaluation. Fortunately results from knowledge compilation research [Darwiche and Marquis, 2002] show how to convert the CNF representation into Decomposable Negation Normal Form (DNNF). It turns out that this form of logical expression can be evaluated in linear time to compute either the most likely diagnosis or an optimal *n* level plan.

While DNNF has been defined previously in terms of a Boolean expression, we make a slight extension for variable logic equations, where the negation of a variable assignment can be replaced by a disjunct of all other possible assignments to that same variable. This extends the DNNF formalism to constraint satisfaction problems.

Definition 1: A variable logic equation is in Decomposable Negation Normal Form if (1) it contains no negations and (2) the subexpressions under each conjunct refer to disjoint sets of variables.

Just as in the Boolean case, there are multiple possible variable logic DNNF expressions equivalent to the CNF and the objective is to find one that is as small as possible. Since Disjunctive Normal Form is also DNNF, the largest DNNF equivalent is exponentially larger than the CNF. Fortunately much smaller DNNF equivalents can often be found. The approach here mirrors the Boolean approach to finding a d-DNNF [Darwiche, 2002] by first recursively partitioning the CNF disjuncts and then traversing the partition tree to generate the DNNF.

The whole purpose for partitioning the disjuncts is to group those that refer to the same variables together and those that refer to different variables in different partitions. Since each disjunct refers to multiple variables, it is often the case that the disjuncts in two sibling partitions will refer to the same variable, but minimizing the cross partition variables dramatically reduces the size of the DNNF equation. This partitioning essentially converts a flat conjunct of disjuncts into an equation tree with internal AND nodes and disjuncts of literals at the leaves, where the number of propositions appearing in multiple branches below an AND node is minimized.

Mirroring the Boolean compiler, partitioning is done by mapping the CNF equation to a hyper-graph, where nodes and hyper-arcs respectively correspond to disjuncts and variables. The nodes that each hyper-arc connects are determined by the disjuncts where the hyper-arc's corresponding variable appears. Given this hyper-graph, a recursive partitioning using a probabilistic min-cut algorithm [Wagner and Klimmek, 1996] computes a relatively good partition tree for the disjuncts, and generalizing this algorithm by weighting the hyperarcs with associated variable cardinalities does even better. See Figure 3 for an extremely simple example with two disjuncts and three variables whose cardinalities are 2. From the equation tree perspective, there is an AND node on top above disjuncts at the leaves. The branches of the AND node share the variable b, which is recorded in the top node's *Sep* set.

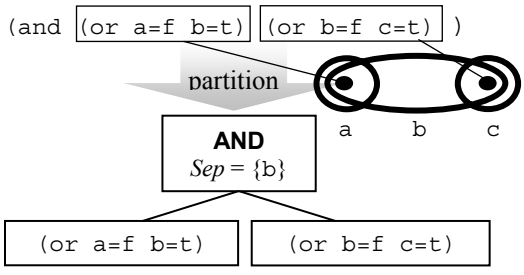


Figure 3: Example of partitioning a CNF equation

Once the equation tree is computed, computing the DNNF involves extracting each AND node's associated shared variables using the equality

$$eqn = \bigvee_{c \in domain(v)} \left(v = c \wedge eqn \setminus \{v = c\} \right),$$

where *eqn*\{v=c} is an equation generated by replacing disjuncts containing v=c with *True* and removing assignments to v from other disjuncts. If a disjunct ever ends up with no assignments, it becomes *False*.

More formally, the DNNF equation is recursively defined on the equation tree using the two formulas below, where the first and second apply to internal and leaf nodes respectively. The first formula's *instances*(*N.Sep*,α) refers to the set of possible assignments to the vector of variables in *N.Sep* that are consistent with α. For instance, running these formulas over Figure 3's tree starts by calling *dnnf*(*root*,*True*), and the instances are `b=t` and `b=f` since only `b` is in *root.Sep*, and both assignments agree with *True*. In general the number of consistent instances grows exponentially with *N.Sep*, leading to the use of min-cut to reduce the size of *N.Sep* for each partition.

$$dnnf(N,\alpha) \equiv \bigvee_{\beta \in instances(N.Sep,\alpha)} \left(\beta \wedge \bigwedge_{c \in N.kids} dnnf(c, \alpha \wedge \beta)\right)$$

$$dnnf(disj,\alpha) \equiv \begin{cases} True & \text{if } \alpha \Rightarrow disj \\ \bigvee_{\beta \in disj \,\&\, \alpha \not\Rightarrow \neg\beta} \beta & \text{if } \exists \beta \supset \alpha \not\Rightarrow \neg\beta \\ False & \text{Otherwise} \end{cases}$$

While walking the equation tree does provide a DNNF equation that can be evaluated in linear time, two very important optimizations involve merging common sub-expressions to decrease the size of the computed structure and caching computations made when visiting a node for improving compiler performance [Darwiche, 2002]. In Figure 4 there were no common sub-expressions to merge, and the resulting DNNF expression appears below.

```
(or (and b=t c=t) (and b=f a=f))
```

4 Onboard Evaluation

To illustrate a less trivial DNNF expression, consider the Figure 4 for the siderostat DNNF. Actually this is a slight simplification of the generated DNNF in that it was generated from a siderostat model that lacked the unknown mode – where the omission was motivated by space limitations. This expression's top rightmost AND node has three children, and each child refers to a unique set of variables. From top to bottom these disjoint sets respectively are

{sw*mode$_1$}, {o$_1$}, and {sw*mode$_0$, sw*trans$_0$, o$_0$, c$_0$}.

Given that DNNF AND nodes have a disjoint branches property, finding optimal satisfying variable assignments becomes a simple three-step process:

1. associate costs with variable assignments in leaves;
2. propagate node costs up through the tree by either assigning the min or sum of the descendents' costs to an OR or AND node respectively; and
3. if the root's cost is 0, infinity, or some other value then respectively return default assignments, failure, or descend from the root to determine and return the variable assignments that contribute to its cost.

Actually, the algorithm is a little more general in that step 2 computes the number of min-cost assignments, and step 3 can extract any one of them in linear time.

4.1 Mode Estimation

Evaluating a DNNF structure to determine component modes starts by assigning costs to the *name*`*mode`$_0$ variables, where these costs come from the :cost entry associated with each mode in the original model, and missing cost entries are assumed to be zero. For instance, none of the transitions have associated costs in the model, resulting in assigning zero to the *name*`*trans`$_0$ leave costs. Finally, sensed values are assigned either zero or infinity depending on the value sensed. In this case the sensed values for o$_0$ and o$_1$ were both true.

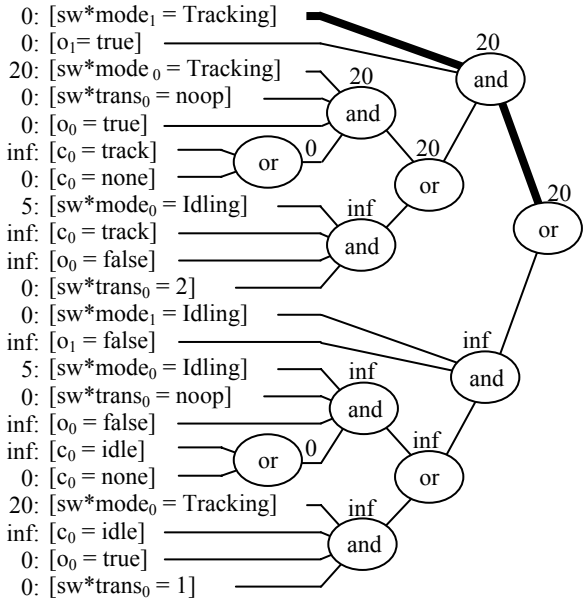


Figure 4: Evaluating a 2 level DNNF structure to determine the siderostat mode from two sets of observations.

Following the simple propagation step, the associated node costs appear above the nodes in Figure 4. Note that the cost is of the top level node is 20. This value is used to prune the search when descending down the tree to determine the assignment to `sw*mode`$_1$, which is the most likely siderostat mode that matches the observations.

While this approach assumes forgetting of old state information, it can be enhanced to either remember the most likely last state or a set of likely last states using a particle filter approach. Since the only difference between such approaches revolves around leaf cost assignments, the requisite changes are very manageable.

4.2 Reconfiguration Planning

When evaluating a DNNF structure for a reconfiguration plan, a cost is assigned to each variable using a number of planning dependent preferences. First, not performing an action has zero cost. This results in associating zero with all leaves that set transitions to noops. Second, leaves denoting other transitions are assigned costs that come from :cost entries associated with transitions. In the example all of these costs are assumed to be zero. Finally

*name**mode$_0$ and *name**mode$_n$ entries are assigned costs that depend respectively on the current and target mode. The leaf assignments that are consistent with these modes will cost zero and inconsistent leaves get an infinite cost. For instance, Figure 5 documents the evaluation to take a currently tracking siderostat and make it idle. In this case the cost is propagated up and then it is used to guide the descent to find the desired cost of c_0, the effecter variable.

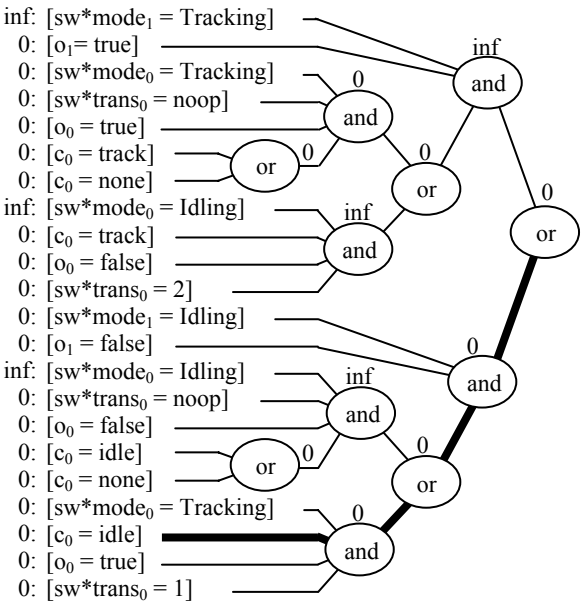


Figure 5: Evaluating a 2 level DNNF structure to compute a reconfiguration plan.

While a need to keep this example simple motivating not tagging transitions with costs, such tags reflect the likelihood of a transition once its preconditions are met. Thus, many transitions can have consistent preconditions and the underlying evaluation will actually adjust the preconditions to maximize the likelihood that the triggered transitions will result in attaining the target conditions. This implies that the planning algorithm finds *n* step solutions to probabilistic planning problems like those of BURIDAN [Kushmerick *et al*., 1994]. From this vantage point MEXEC's compiled internal structure can be viewed as a limited policy for solving POMDP problems if a solution can be found in *n-1* steps, but this perspective has yet to be fully explored.

5 Implementation and Experiments

The system is currently implemented in Allegro Common LISP with under 500 lines to compute a device's CNF equation, under 500 lines to compute a CNF equation's associated DNNF, and less than 80 lines to evaluate a DNNF equation to find all minimal cost satisfactions.

In addition to testing MEXEC on various switching circuit examples, there has been some work on developing and experimenting with models of a Space Interferometer Mission Test Bed 3 (STB-3) model [Ingham *et al*., 2001]

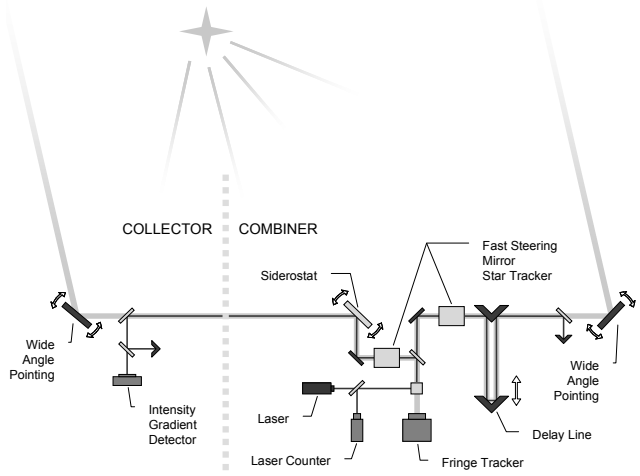


Figure 6: A simplified schematic of the Formation Interferometer Testbed (FIT). The left side of the dotted line represents the collector spacecraft and the right side of the dotted line represents the combiner spacecraft.

as well as the Formation Interferometer Test Bed (FIT) model, which is an extension on the STB-3 model. While STB-3 represents a single spacecraft interferometer, FIT represents a separated spacecraft interferometer. As illustrated in Figure 6, FIT is composed of combiner (right) and collector (left) spacecraft. The collector spacecraft precisely points at a star and reflects the starlight beam to the combiner spacecraft. While the combiner spacecraft also points at the star to collect the starlight, it also accurately points at the collector spacecraft in order to combine the starlight from the collector spacecraft with its own.

Device	C	V	S	$n = 1$	$n = 2$	$n = 3$	$n = 4$
STB-3	7	26	13	125	851	5191	36354
FIT	17	64	12	235	1488	7620	39705

Table 1: DNNF sizes (in nodes) of interferometers with C components, V variables, S sensors, and *n* levels.

Compiling these two models for instantaneous ($n = 1$) through three step ($n = 4$) DNNF structures results in the generation of Table 1. The initial message to pull out of this exercise is that instantaneous DNNF structures, for diagnosis only, tend to be extremely compact, but as *n* increases so does the DNNF size. Thus increasing n makes the system able to reason over longer durations when either diagnosing or planning, but the increased capability comes at the cost of longer evaluation times since DNNF equation evaluation is linear in equation size.

Still, work on planning [Barrett, 2004] and strict DNNF compilation [Darwiche, 2002] leads one to suspect that the scaling issue can be improved. In the case of our empirical experiments, the scaling issue already has been improved by an order of magnitude. The results reported in [Chung and Barrett, 2003] had a 4318 node DNNF structure for the FIT model with $n = 1$.

6 Related Work

While others have made the leap to applying compilation techniques to both simplify and accelerate embedded computation to determine a system's current mode of operation, they are more restricted than MEXEC. First, DNNF equation creation and evaluation was initially developed in a diagnosis application [Darwiche, 1998], but the resulting system restricted a component to only have one output and that there cannot be directed cycles (feedback) between components. MEXEC makes neither of these restrictions. The Mimi-ME system [Chung *et al*., 2001] similarly avoided making these restrictions, but it lacks hard real-time guarantees by virtue of having to solve an NP-complete problem, called MIN-SAT, when converting observations into mode estimates. MEXEC supports hard real-time performance guarantees.

The closest related work on real-time reconfiguration planning comes from the Burton reconfiguration planner used on DS-1 [Williams and Nayak, 1997] and other research on planning via symbolic model checking [Cimatti and Roveri, 1999]. In the case of Burton our system improves on that work by relaxing a number of restricting assumptions. For instance, Burton required the absence of causal cycles (feedback), no two transitions within a component can be simultaneously enabled, and that each transition must have a control variable in its precondition. MEXEC has none of these restrictions. On the other hand, our system can only plan *n* steps ahead where Burton did not have that limitation. Similarly, the work using symbolic model checking lacked the *n*-step restriction, but it compiled out a universal plan for a particular target state. Our system uses the same compiled structure to determine how to reach any target state within *n* steps of the current state.

7 Conclusions

This paper presented the MEXEC system, a knowledge-compilation based approach to implementing an offline domain compiler that enables embedded hard real-time diagnosis and reconfiguration planning for more robust system commanding. This system also enables feedback reasoning by virtue of global analysis during compilation.

Acknowledgements

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The author would also like to thank Alan Oursland, Seung Chung, Adnan Darwiche, Daniel Dvorak, and Mitch Ingham for discussions contributing to this effort

References

[Barrett, 2004] Anthony Barrett. Domain Compilation for Embedded Real-Time Planning. In *Proceedings of the Fourteenth International Conference on Automated Planning & Scheduling*, Whistler, British Columbia, Canada, June 2004. AAAI Press.

[Chung *et al*., 2001] Seung Chung, John Van Eepoel, and Brian Williams. Improving Model-based Mode Estimation through Offline Compilation. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, St-Hubert, Canada, June 2001.

[Chung and Barrett, 2003] Seung Chung and Anthony Barrett. Distributed Real-time Model-based Diagnosis. In Proceedings of the 2003 IEEE Aerospace Conference, Big Sky, MT. March 2003.

[Cimatti and Roveri, 1999] Alessandro Cimatti and Marco Roveri. Conformant Planning via Model Checking. In *Recent Advances in AI Planning, 5th European Conference on Planning*, Durham, UK: Springer

[Darwiche, 1998] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165-222.

[Darwiche, 2000] Adnan Darwiche. Model-based diagnosis under real-world constraints. AI Magazine, Summer 2000.

[Darwiche, 2002] Adnan Darwiche. A Compiler for Deterministic Decomposable Negation Normal Form. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*. 627-634. Edmonton, Alberta, Canada: AAAI Press.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research* 17:229-264.

[Ingham *et al*., 2001] Michel Ingham, Brian Williams, Thomas Lockhart, Amalaye Oyake, Micah Clarke, and Abdullah Aljabri. Autonomous Sequencing and Model-based Fault Protection for Space Interferometry, In *Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, St-Hubert, Canada, June 2001.

[Kushmerick *et al*., 1994] Nicholas Kushmerick, Steve Hanks, and Daniel Weld. An Algorithm for Probabilistic Least-Commitment Planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA: AAAI Press. 1994.

[Wagner and Klimmek, 1996] Frank Wagner and Regina Klimmek. A Simple Hypergraph Min Cut Algorithm. Technical Report, b 96-02, Inst. Of Computer Science, Freie Universität Berlin. 1996.

[Williams and Nayak, 1996] Brian Williams and P. Pandurang Nayak. A Model-based Approach to Reactive Self-Configuring Systems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR: AAAI Press. 1996.

[Williams and Nayak, 1997] Brian Williams and P. Pandurang Nayak. A Reactive Planner for a Model-based Executive. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan: Morgan Kaufmann. 1997.

Proactive Algorithms for Scheduling with Probabilistic Durations*

J. Christopher Beck
Department of Mechanical & Industrial Engineering
University of Toronto, Canada
jcb@mie.utoronto.ca

Nic Wilson
Cork Constraint Computation Centre
Department of Computer Science
University College Cork, Ireland
n.wilson@4c.ucc.ie

Abstract

Proactive scheduling seeks to generate high quality solutions despite execution time uncertainty. Building on work in [Beck and Wilson, 2004], we conduct an empirical study of a number of algorithms for the job shop scheduling problem with probabilistic durations. The main contributions of this paper are: the introduction and empirical analysis of a novel constraint-based search technique that can be applied beyond probabilistic scheduling problems, the introduction and empirical analysis of a number of deterministic filtering algorithms for probabilistic job shop scheduling, and the identification of a number of problem characteristics that contribute to algorithm performance.

1 Introduction

Classical models of scheduling assume all information is known and static. When faced with uncertainty, *proactive* scheduling techniques seek to incorporate models of the uncertainty in the off-line problem solving and to build solutions that can achieve a satisfactory level of performance at execution time. [Beck and Wilson, 2004] addressed the problem of job shop scheduling with probabilistic durations with two styles of algorithm both using Monte Carlo simulation to evaluate the probabilistic makespan of solutions. This paper builds on that work, making the following contributions: (i) A novel constraint-based search technique is introduced that performs a series of complete branch-and-bound searches on highly constrained problem models. The models are gradually relaxed with the cost of the best solution model found in the previous searches being used as an upper bound. Experimental results show that the algorithm performs as well as the existing branch-and-bound style of search on small problems and significantly out-performs it on larger problems. (ii) A number of novel deterministic filtering algorithms are presented. On larger problems such algorithms out-perform the other algorithms tested. Their performance is affected by the following two factors: the quality of the deterministic solutions found and the correlation between the deterministic and probabilistic makespan of solutions. (iii) It is demonstrated that the method introduced by Beck & Wilson for the incorporation of uncertainty in deterministic problems can increase the correlation between deterministic and probabilistic makespan, providing an explanation for its strong performance.

In the next section we briefly review previous work. The six search algorithms investigated in this paper are defined in Section 3 and Section 4 presents our experiments and results. In Section 5, we present our conclusions.

*This material is based upon works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075 and by ILOG, SA.

2 Background

The *job shop scheduling problem* (JSP) consists of a set $\mathcal{A}$ of activities each with a positive integer-valued duration, d_i. $\mathcal{A}$ is partitioned into *jobs*, and with each job is associated a total ordering on that set of activities. Each activity specifies a resource on which it must execute without interruption. No activities that require the same resource can overlap in their execution. We represent this formally by a partition of $\mathcal{A}$ into *resource sets*. A *solution* consists of a total ordering on each resource set such that the union of the resource and job orderings is an acyclic relation on $\mathcal{A}$. The *makespan* of a solution is the difference between the minimum start time and the maximum end time of all activities. Finding a solution with minimum makespan is NP-hard [Garey and Johnson, 1979]. An *independent probabilistic job shop scheduling problem* is a JSP where the duration $\mathbf{d}_i$ associated with an activity $A_i \in \mathcal{A}$ is a positive integer-valued random variable. These random variables are fully independent. $\mathbf{d}_i$ has expected value $\mu_i = \mathrm{E}[\mathbf{d}_i]$ and variance $\sigma_i^2 = \mathrm{Var}[\mathbf{d}_i]$. The makespan $\mathbf{make}(s)$ of solution s is therefore also a random variable. We fix a degree of confidence p, e.g., we set $p = 0.95$ in the experiments. The minimum probabilistic makespan, $D(s)$, of a solution s is the smallest value D such that the probability that all jobs will finish before D is at least p. That is, $D(s) = \min\{D : \Pr(\mathbf{make}(s) \leq D) \geq p\}$. An optimal probabilistic makespan is the minimum $D(s)$ over all solutions s.

Beck & Wilson introduce an analytical model and develop two main results. First, it is demonstrated that Monte Carlo simulation can be used to find the probabilistic makespan for a solution and to find a lower bound on the probabilistic makespan of a partial solution. Second, it is shown that a deterministic JSP instance can be constructed from a probabilis-

tic JSP instance by setting the duration d_i of each activity to $\mu_i + q\sigma_i$ and that, for the appropriate choice of non-negative q, the optimal makespan of the deterministic instance is a lower bound on the optimal probabilistic makespan of the corresponding probabilistic JSP. Three algorithms are defined: a branch-and-bound approach where Monte Carlo simulation is used at each node to find a lower bound (at internal nodes) or an upper bound (at a leaf node) on the probabilistic makespan, and two deterministic filtering algorithm consisting of using either constraint-based search or tabu search to find a series of increasingly good deterministic solutions, each of which is simulated. Empirical results indicated that the first algorithm was able to find optimal solutions only for very small problems, that for medium-sized problems the constraint-based filtering algorithm found the best solutions, and for the largest problems the tabu-based filtering algorithm performed best. Choosing $q > 0$ led to stronger performance of the filtering algorithms. No empirically founded explanations were provided to explain the differences in problem solving performance.

3 Algorithms for Probabilistic JSP

In this section, we describe six algorithms for solving the probabilistic job shop scheduling problem.

3.1 Approximately Complete Branch-and-Bound

We implemented a branch-and-bound algorithm identical to that of Beck & Wilson except that we use stronger heuristics that make decisions on the sequence of activities in each resource set (see Section 3.5). We refer to this algorithm as the *B&B-N* as it performs *B*ranch-and-*B*ound with simulation at each *N*ode.

3.2 Iterative Descending Bound Search

By setting activity durations based on a q value that ensures a lower bound on the probabilistic makespan, we can use standard constraint propagation on the deterministic durations to implicitly calculate a lower bound at each internal node. At each leaf node, simulation is used as in B&B-N to find the probabilistic makespan. Rather than parameterize the algorithm with a fixed q value, we perform repeated tree searches with a descending q value. Starting with a high q value, we begin a tree search. When the first leaf, e, is reached, simulation is used to find $D(s_e)$. It is likely that $D_q(s_e)$, the minimum makespan of s_e in the corresponding deterministic problem, will be larger than $D(s_e)$. Since we enforce $D_q(s_e) \leq D(s_e)$, estimating $D(s_e)$ through simulation causes the search to backtrack to a interior node, i, very high in the tree. At node i, $D_q(S_i) \leq D(s_e)$: S_i is the set of solutions in the subtree below node i and $D_q(S_i)$ is a lower bound on the deterministic makespan of those solutions based on standard constraint propagation. With high q values, we commonly observe that there are very few nodes that meet this criterion and the search space is quickly exhausted. We then decrement the q value (e.g., by 0.05) and start a new tree search. Eventually, and often very quickly, we reach a q value such that there exists a full solution where $D_q(s_e) \leq D(s_e)$. The probabilistic makespan of the best solution that has been found by any point in the overall search is used, as in B&B-N, as an upper bound on subsequent search.

B&B-DQ-L():
Returns the solution with lowest probabilistic makespan

1 $(s^*, D^*) \leftarrow$ findFirstB&BSimLeaves(∞, 0)
2 $q \leftarrow q_0$
3 **while** $q \geq 0$ *AND not timed-out* **do**
4 (s, D) $\leftarrow$ findOptB&BSimLeaves(D^*, q)
5 **if** $s \neq NIL$ **then**
6 $s^* \leftarrow s$; $D^* \leftarrow D$
end
7 $q \leftarrow q - q_{dec}$
end
8 return s^*

Algorithm 1: B&B-DQ-L

Algorithm 1 presents pseudocode for the basic algorithm. We make use of two functions not defined using pseudocode: *findFirstB&BSimLeaves(c, q)* creates a JSP with activity durations defined based on the q value and conducts a branch-and-bound search where Monte Carlo simulation is used for each leaf node and standard constraint propagation is used at interior nodes. The first solution that is found whose probabilistic makespan is less than c is returned. *findOptB&BSimLeaves(c, q)* does the same as findFirstB&BSimLeaves(c, q) except the solution with lowest probabilistic makespan is returned rather than the first one. If no solution is found, a NIL value is returned. Unless the q value is low enough that the deterministic makespan is a lower bound on the probabilistic makespan, this function does not necessarily return the globally optimal solution. We find a starting solution with $q = 0$ to serve as an initial upper bound on the probabilistic makespan. In practice, B&B-DQ-L is run with a maximum CPU time. If $q = 0$ is reached within the time limit, this algorithm is approximately complete.

We refer to this algorithm as *B&B-DQ-L* as it does a series of *B*ranch-and-*B*ound searches with *D*escending *Q* values and with simulation is used at the *L*eaves of the tree.

3.3 Branch-and-Bound Filtering Algorithms

Preliminary experiments with the branch-and-bound filtering algorithm presented by Beck & Wilson showed that a significant amount of simulation was being done early in the search when the upper bound on the deterministic makespan was relatively high. Such solutions would seem to have little chance of being the optimal probabilistic solution as the deterministic solution is very poor. We present two constraint-based filtering algorithms that seek to avoid these wasted simulations.

Branch-and-Bound Timed Better Solution A simple method is to spend a fixed amount of CPU time, $t_{initial}$ to find a solution, s^*, with a low deterministic makespan $D_{initial}$ using a fixed q value and constraint-based search. Then search can be restarted using the same q value and using $D_{initial}$ as an upper bound on the deterministic makespan. Whenever a solution, s, is found such that $D_q(s) \leq D_{initial}$, a simulation is run to evaluate the probabilistic makespan. The solution

B&B-TBS(q):
Returns the solution with lowest probabilistic makespan

```
1  (s*, D_initial) ← findOptB&B(∞, q, t_initial)
2  D* ← ∞
3  while termination criteria unmet AND not timed-out do
4      (s, D) ← findNextB&B(D_initial, q, t_remaining)
5      D' ← simulate(s)
6      if D' < D* then
7          s* ← s; D* ← D'
       end
   end
8  return s*
```

Algorithm 2: B&B-TBS

with the lowest probabilistic makespan is returned when either the entire tree has been searched or when the maximum allowed CPU time has expired. Algorithm 2 contains pseudocode for this approach.

As above, we make use of a number of functions not defined with pseudocode: *findOptB&B(c, q, t)* creates a JSP with activity durations defined based on q and conducts a deterministic branch-and-bound search for up to t CPU seconds using c as an upper bound on the makespan. When the search tree is exhausted or the time-limit is reached, the best solution found, together with its makespan, are returned. No simulation is done. *findNextB&B(c, q, t)* produces a sequence of solutions (one solution each time it is called) whose deterministic makespan is less than or equal to c. The problem is defined using the q value and t is the CPU time limit. The solutions produced are the leaves of the B&B search tree in the order encountered by the algorithm. The c value does not change. Given enough CPU time, the algorithm will evaluate the probabilistic makespan of all solutions whose deterministic makespan is less than or equal to $D_{initial}$. Finally, *simulate(s)* runs Monte Carlo simulation on solution s and the probabilistic makespan is returned.

The algorithm is called *B&B-TBS* for *B*ranch-and-*B*ound *T*imed *B*etter *S*olution. The algorithm is not complete, as there is no guarantee that the optimal probabilistic solution will have a deterministic makespan less than $D_{initial}$.

Branch-and-Bound Iterative Best Solution A more extreme filtering algorithm first finds an optimal deterministic solution and uses it as a filter for choosing the solutions to simulate. Using a fixed q value, a solution with an optimal deterministic makespan, D_q^*, is found and simulated. Then we execute a series of iterations beginning with $i = 0$. For each iteration, all solutions, s_e, with deterministic makespan $D_q(s_e) \leq D_q^* \times (1 + i/100)$ are simulated and the one with the lowest probabilistic makespan is returned. If an optimal deterministic solution cannot be found within the CPU time limit, the best deterministic solution found is simulated and returned (i.e., only one simulation is done). Algorithm 3 presents the pseudocode.

The algorithm is theoretically complete. When i is large enough that the cost bound is greater than the deterministic makespan of *all* activity permutations, they will all be simulated. However, i would have to grow unreasonably large and therefore we treat this algorithm as, practically, incomplete. We refer to this algorithm as *B&B-I-BS* for *B*ranch-and-*B*ound-*I*terative-*B*est *S*olution.

B&B-I-BS(q):
Returns the solution with smallest probabilistic makespan

```
1  (s*, D_initial) ← findOptB&B(∞, q, t_0 − 1)
2  D* ← simulate(s*)
3  i ← 0
4  while not timed-out do
5      while termination criteria unmet do
6          (s, D_q) ← findNextB&B(
           D_initial × (1 + i/100), q, t_remaining)
7          D ← simulate(s)
8          if D < D* then
9              s* ← s; D* ← D
           end
       end
10     i ← i + 1
   end
11 return s*
```

Algorithm 3: B&B-I-BS

3.4 Local Search Filtering Algorithms

We also experiment with two local search filtering algorithms analogous to the constraint-based algorithms above except that tabu search is used in place of constraint-based search.

Tabu Timed Better Solution For a fixed q and for a fixed CPU time, $t_{initial}$, a solution with the lowest deterministic makespan, $D_{initial}$, possible is sought. Search is then restarted and whenever a solution, s, is found that has a deterministic makespan $D_q(s) \leq D_{initial}$, simulation is used to evaluate the probabilistic makespan. The solution with the lowest probabilistic makespan is returned. The pseudocode for Tabu-TBS is the same as that for B&B-TBS (Algorithm 2) with the following changes. The function findNextTabu(c, q, t) replaces findNextB&B(c, q, t); it produces a sequence of solutions whose deterministic makespan is less than c. The findBestTabu(c, q, t) function replaces findOptB&B(c, q, t); tabu search is run for up to t CPU seconds and the solution with the lowest deterministic makespan that is less than c is returned. We call this algorithm *Tabu-TBS* for *Tabu-T*imed *B*etter *S*olution.

Tabu Iterative Best Solution The core tabu search implementation does not necessarily use the entire CPU time. We can therefore create an iterative tabu-based solver for the probabilistic JSP similar to B&B-I-BS. In the first phase, using the overall time limit less one second, tabu search is used to find a good deterministic solution, based on a fixed q value. That solution is then simulated. Any remaining time is spent generating solutions with a deterministic makespan within a fixed percentage of the initial solution's deterministic makespan. As with B&B-I-BS, iterations are run with increasing i value starting with $i = 0$. In each iteration, solutions found by the tabu search whose deterministic makespan

are within $1 + i/100$ of the initial deterministic makespan are simulated. The solution with the lowest probabilistic makespan is returned. The algorithm is termed *Tabu-I-BS* for *Tabu-I*terative-*B*est *S*earch. The pseudocode for this algorithm is the same as that for B&B-I-BS (Algorithm 3) with the same substitutions that were made for Tabu-TBS.

3.5 Core Algorithm Details

The branch-and-bound algorithms described use texture-based heuristics [Beck and Fox, 2000] to decide on the the pair of activities to sequence and which sequence to try first. When constraint propagation is used (i.e., all B&B algorithms except B&B-N), we use temporal propagation, timetables [Le Pape *et al.*, 1994], edge-finder [Nuijten, 1994], and the balance constraint [Laborie, 2003].

The tabu search is the same one used by Beck & Wilson: the TSAB algorithm of [Nowicki and Smutnicki, 1996].

4 Empirical Investigations

Our empirical investigations aim to evaluate and investigate the performance of the algorithms. In particular, we are interested in algorithm scaling with problem size and uncertainty level and with the usefulness of representing uncertainty information using q values.

4.1 Experimental Details

We use four sets of probabilistic JSPs of size $\{4\times4, 6\times6, 10\times10, 20\times20\}$ with three uncertainty levels $u_j \in \{0.1, 0.5, 1\}$. A deterministic problem is generated using an existing generator [Watson *et al.*, 2002] with integer durations drawn uniformly from the interval [1, 99]. Three probabilistic instances are then produced by setting the mean durations μ_i to be the deterministic durations and by uniformly drawing the standard deviation σ_i from the interval $[0, u_j\mu_i]$. The distribution of each duration is approximately normal. For each problem size, we generate 10 deterministic problems which are transformed into 30 probabilistic instances.

Given the stochastic nature of the simulation and the tabu search algorithm, each algorithm is run 10 times on each problem instance with different random seeds. Each run has a time limit of 600 CPU seconds. Each Monte Carlo simulation uses 1000 trials. For B&B-TBS and Tabu-TBS, $t_{initial}$ is 60 CPU seconds.

For the heuristic techniques, a deterministic duration is assigned to each of the activities based on the q value. We experiment with the same four q values used by Beck & Wilson as displayed in Table 1. The B&B-DQ-L algorithm requires an initial value of q, q_0, and a decrement, q_{dec}. For all except the largest problems, we used $q_0 = 1.25$ and $q_{dec} = 0.05$. Preliminary experiments with the 20×20 problems indicated that $q_0 = 1.25$ resulted in long runs with no solutions due to a large search space with few if any solutions. To avoid such runs, we used $q_0 = 0.9$, for which solutions could be found.

The probabilistic makespans are based on confidence level $p = 0.95$. Our primary evaluation criterion is the mean normalized probabilistic makespan (*MNPM*) that each algorithm achieves: $\mathit{MNPM}(a, L) = \frac{1}{|L|}\sum_{l\in L}\frac{D(a,l)}{D_{lb}(l)}$. L is a set of problem instances, $D(a, l)$ is the mean probabilistic makespan found by algorithm a on l over 10 runs, $D_{lb}(l)$ is the lower bound on the probabilistic makespan for l. For all problems except 20×20, the D_{lb} is found by solving the deterministic problems to optimality using the q_1 durations (see Section 3 of [Beck and Wilson, 2004]). For the 20×20 problems, the optimal solutions could not be found and so the D_{lb} values represent the best deterministic solutions found. These values, therefore, are not a true lower bound.

q_0	q_1	q_2	q_3
0	$\frac{B}{\sqrt{2n}}$	$\frac{q_1+q_3}{2}$	$\frac{B}{\sqrt{n}}\frac{\sqrt{\text{Average}_{A_i\in\pi}\sigma_i^2}}{\text{Average}_{A_i\in\pi}\sigma_i}$

Table 1: The q-values used, following Beck & Wilson. n is the number of jobs and $B = 1.645$.

The hardware used for the experiments is a 1.8GHz Pentium 4 with 512 M of main memory running Linux RedHat 9. All algorithms were implemented using ILOG Scheduler 5.3.

4.2 Results

Table 2 presents the results of our experiment. Each cell is the mean over 10 independent runs of 10 problems. The observed mean standard deviations for each cell are small: the maximum over all cells is less than 10.3% of the corresponding mean makespan, while the mean over all cells is less than 0.8%. We did not observe a large difference among the q values provided $q > 0$ and therefore present the results for q_2.

Problem Size	Unc. Level	Algorithms					
		B&B Complete		B&B Heuristic		tabu	
		N	DQ-L	TBS	I-BS	TBS	I-BS
4×4	0.1	1.027*	**1.023***	1.026	1.026	1.027	**1.023**
	0.5	1.060*	1.049*	1.064	1.059	1.061	**1.046**
	1	1.151*	1.129	1.154	1.149	1.150	**1.128**
6×6	0.1	1.034	**1.021**	1.022	1.022	1.025	1.023
	0.5	1.113	**1.073**	1.083	1.077	1.089	1.074
	1	1.226	1.170	1.178	1.174	1.182	**1.168**
10×10	0.1	1.185	1.028	**1.024**	**1.024**	1.035	1.028
	0.5	1.241	1.115	**1.101**	**1.101**	1.121	1.112
	1	1.346	1.234	**1.215**	**1.215**	1.244	1.223
20×20†	0.1	1.256	1.142	1.077	1.071	1.029	**1.027**
	0.5	1.326	1.233	1.177	1.181	**1.136**	1.137
	1	1.482	1.388	1.334	1.338	**1.297**	1.307

Table 2: The mean normalized probabilistic makespans for each algorithm. '*' indicates a set of runs for which we have, with high confidence, found close-to-optimal makespans. '†' indicates problem sets for which normalization was done with approximate lower bounds. The lowest *MNPM* found for each problem set are shown in bold.

For the 4×4 problems, a number of algorithms find lower mean probabilistic makespans than B&B-N despite the fact that B&B-N terminates before the limit on CPU time and therefore finds approximately optimal solutions. This is an artifact resulting from algorithms that simulate the same solution multiple times. In B&B-N, a particular solution is only simulated once. In other algorithms, the same solution may be simulated multiple times leading to a bias to lower probabilistic makespan values. Based on the theoretic model of Beck & Wilson, these values still correspond to approximately optimal solutions.

4.3 Analysis

Overall the empirical results agree with those presented by Beck & Wilson. Here, we will focus on investigations of B&B-DQ-L and of the behavior of the heuristic algorithms.

Branch-and-Bound Descending-Q Leaves

B&B-DQ-L out-performs B&B-N across all problem sets even when it is not able to reach $q = 0$. As an indication of the number of iterations, the mean minimum values q for each problem size are: $4 \times 4 : 0.007$, $6 \times 6 : 0.56$, $10 \times 10 : 1.00$, $20 \times 20 : 0.9$.

For B&B-DQ-L, the deterministic durations defined by the q value serve to guide and prune the search for each iteration and, therefore, as with the heuristic algorithms (see below), the search is heuristically guided to the extent that solutions with low deterministic makespans also have low probabilistic makespans. However, the characteristics of the solutions found by the search are unclear. Ideally, we would like the search with high q to find solutions with very good probabilistic makespans both because we wish to find good solutions quickly and because the probabilistic makespan values are used to prune subsequent search. Analysis indicates that the first solution found by B&B-DQ-L is consistently better than the first solution found by B&B-N and for the larger problems (10×10 and 20×20) the relative increase in solution quality over subsequent iterations is greater for B&B-DQ-L.

Heuristic Algorithms

To investigate the performance of the heuristic algorithms, we look at the performance differences coming from using the q values, and the extent to which the performance of the heuristic algorithms depends on two factors: their ability to find good deterministic solutions and the underlying correlation between deterministic and probabilistic makespans in the experimental problems.

The Effect of the q Values Using a non-zero q value almost always results in better performance. This appears to be particularly true when either the uncertainty or the problem size increases. Table 3 shows the difference between mean normalized probabilistic makespan when using q_0 and q_2. On a few problem sets, negative values indicate that the runs with q_0 deliver a better mean solution. However, by far the majority of the values are greater than 0 and their magnitude increases with both uncertainty and problem size, indicating increasingly good relative performance on the q_2 problems. We show below that one explanation for this performance is that problem instances with $q > 0$ have a greater correlation between deterministic and probabilistic makespans.

Finding Good Deterministic Makespans We hypothesize that the performance of the heuristic techniques (and B&B-DQ-L) is partially dependent upon the ability to find solutions with low deterministic makespans. We looked at two measures of the performance of B&B-I-BS and Tabu-I-BS: the quality of the best deterministic solutions found and the number of iterations. Because B&B-I-BS performs systematic search of all deterministic solutions with makespan below a given threshold, we hypothesize that on problem sets where it outperforms Tabu-I-BS, it will also have found better deterministic solutions and/or be able to systematically search solutions up to a higher threshold. In contrast, for problem sets where Tabu-I-BS outperforms B&B-I-BS, it will be due to finding better deterministic solutions.

Problem Size	Unc. Level	B&B		Tabu	
		TBS	I-BS	TBS	I-BS
4×4	0.1	0.001	0	0	0
	0.5	0	0.001	0.001	0.001
	1	0.02	-0.001	0.021	0.017
6×6	0.1	0.002	0	0.001	0.001
	0.5	-0.002	0	0.005	0.004
	1	0.027	0.001	0.041	0.009
10×10	0.1	0	0	-0.001	0.002
	0.5	0.004	0.004	0.006	0.002
	1	0.015	0.016	0.017	0.009
20×20†	0.1	0.009	0.010	0.011	0.015
	0.5	0.010	0.005	0.012	0.028
	1	0.025	0.030	0.029	0.032

Table 3: The difference between the mean normalized probabilistic makespans for runs using q_0 and using q_2. The higher the value, the more the q_2 runs out-performed the q_0 runs.

Table 4 presents data using $q = q_2$. The mean normalized deterministic makespan (*MNDM*) is the calculated as follows: $\mathit{MNDM}(a, L) = \frac{1}{|L|} \sum_{l \in L} \frac{D_q(a,l)}{D_{q,min}(l, B\&B-I-BS)}$. The notation is as above with the addition of $D_{q,min}(l, \mathit{B\&B\text{-}I\text{-}BS})$, the lowest deterministic makespan found by the B&B-I-BS algorithm over all runs on problem l. MNDM, therefore, provides a relative measure of the mean deterministic makespans from the two algorithms: the higher the value, the worse is the average makespan found relative to B&B-I-BS. The second measure (*Iters*) is the mean number of iterations performed by each algorithm.

Problem Size	Unc. Level	B&B-I-BS		Tabu-I-BS	
		MNDM	Iters	MNDM	Iters
4×4	0.1	1.000	100	1.000	76
	0.5	1.000	100	1.000	78
	1	1.000	100	1.007	83
6×6	0.1	1.000	10	1.000	24
	0.5	1.000	10	1.001	31
	1	1.000	11	1.000	21
10×10	0.1	1.000	1	1.002	5
	0.5	1.000	1	1.004	5
	1	1.000	1	1.004	5
20×20	0.1	1.045	0	1.002	0
	0.5	1.041	0	0.998	0
	1	1.037	0	1.002	0

Table 4: The mean normalized deterministic makespan (MNDM) and number of iterations (Iters) for B&B-I-BS and Tabu-I-BS.

Table 4 is consistent with our hypotheses. For those problems sets with large performance differences (i.e., 10×10 and 20×20) in probabilistic makespan, the better performing algorithm does find better deterministic makespans.

The Correlation Between Deterministic and Probabilistic Makespan The explanatory power of an algorithm's ability to find good deterministic makespans is, by itself, insufficient unless there is a correlation between the good deterministic and good probabilistic makespans. It is reasonable to expect that the level of uncertainty has an impact on any correlation: at low uncertainty, the variations in duration are small, so we can expect the probabilistic makespan to be relatively close to the deterministic makespan. When the uncertainty level is high, the distribution of probabilistic makespans for a single solution will be wider, resulting in a lower correlation. We hypothesize that this impact of uncertainty level contributes to the observed performance degradation (see Table 2) of the heuristic techniques with higher uncertainty levels.

We generated 100 new 10×10 deterministic JSP problem instances with the same generator and parameters used above. The standard deviation for the duration of each activity in the 100 instances were generated independently for each of five uncertainty levels $u_j \in \{0.1, 0.5, 1, 2, 3\}$ resulting in a total of 500 problem instances. For each instance and each of the four q values above, we randomly generated and simulated 100 deterministic solutions. Using R [R Development Core Team, 2004], we measured the correlation coefficient, r, for each problem set.

Unc. Level	q_0	q_1	q_2	q_3
0.1	0.9990	0.9996	0.9996	0.9995
0.5	0.9767	0.9912	0.9917	0.9909
1	0.9176	0.9740	0.9751	0.9736
2	0.8240	0.9451	0.9507	0.9517
3	0.7381	0.9362	0.9418	0.9423

Table 5: The correlation coefficient comparing deterministic and probabilistic makespans for a set of 10×10 probabilistic JSPs. Each cell represents the correlation coefficient for 10000 deterministic, probabilistic pairs.

Table 5 supports our hypothesis. As the uncertainty level increases, the correlation between the deterministic makespan and corresponding probabilistic makespan lessens. The strength of the correlation is somewhat surprising: even for the highest uncertainty level, r is more than 0.94 for q_2 and q_3. This suggests that the heuristic algorithms can be successfully applied to problem with higher uncertainty levels provided the appropriate q value can be found.

The correlation results in Table 5 also provide an explanation for the effect of the q values: they increase the correlation between deterministic and probabilistic makespans.

5 Conclusion

In this paper, we presented and conducted empirical analysis of a number of algorithms to proactively solve the job shop scheduling problem with probabilistic durations. Our empirical studies have demonstrated that a novel algorithm, B&B-DQ-L, based on iteratively reducing a parameter that determines the validity of the lower bound results in equal performance on small problems and much better performance on larger problems when compared to the previous branch-and-bound technique. However, highest performance was observed in heuristic algorithms based on using the deterministic makespan to filter the solutions which are to be simulated. It was argued, using detailed analysis of the experimental data and results from an ancillary experiment, that the performance of these algorithms is affected by their ability to find good deterministic makespans and the correlation in the problems between the quality of deterministic and probabilistic makespans. The correlation data on problems with higher uncertainty suggest that such algorithms may scale well to such problems. Central to the success of the heuristic algorithms was the use of a q value that governs the extent to which uncertainty (i.e., the variance) was represented in the durations of activities in deterministic problems. It was shown that such an incorporation of uncertainty data leads to a stronger correlation between deterministic and probabilistic makespans, and suggests a corresponding ability to find better probabilistic makespans.

References

[Beck and Fox, 2000] J. C. Beck and M. S. Fox. Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics. *Artificial Intelligence*, 117(1):31–81, 2000.

[Beck and Wilson, 2004] J.C. Beck and N. Wilson. Job shop scheduling with probabilistic durations. In *Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI04)*, pages 652–656, 2004.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[Laborie, 2003] P. Laborie. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, 143:151–188, January 2003.

[Le Pape *et al.*, 1994] C. Le Pape, P. Couronné, D. Vergamini, and V. Gosselin. Time-versus-capacity compromises in project scheduling. In *Proceedings of the Thirteenth Workshop of the UK Planning Special Interest Group*, 1994.

[Nowicki and Smutnicki, 1996] E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

[Nuijten, 1994] W. P. M. Nuijten. *Time and resource constrained scheduling: a constraint satisfaction approach*. PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1994.

[R Development Core Team, 2004] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. ISBN 3-900051-07-0.

[Watson *et al.*, 2002] J.-P. Watson, L. Barbulescu, L.D. Whitley, and A.E. Howe. Contrasting structured and random permutation flow-shop scheduling problems: search-space topology and algorithm performance. *INFORMS Journal on Computing*, 14(1), 2002.

Over-Subscription Planning with Numeric Goals

J. Benton
Computer Sci. & Eng. Dept.
Arizona State University
Tempe, AZ 85287-5406
j.benton@asu.edu

Minh B. Do
Embedded Reasoning Area
Palo Alto Research Center
Palo Alto, CA 94304
minh.do@parc.com

Subbarao Kambhampati*
Computer Sci. & Eng. Dept.
Arizona State University
Tempe, AZ 85287-5406
rao@asu.edu

Abstract

By relaxing the hard-goal constraints from classical planning and associating them with reward values, over-subscription planning allows users to concentrate on presenting what they want and leaves the task of deciding the best goals to achieve to the planner. In this paper, we extend the over-subscription planning problem and its limited goal specification to allow numeric goals with continuous utility values and goals with mixed hard and soft constraints. Together they considerably extend the modeling power of goal specification and allow the user to express goal constraints that were not possible before. To handle these new goal constraints, we extend the *Sapa*^{ps} planner's planning graph based techniques to help it choose the best beneficial subset of goals that can include both hard or soft logical and numeric goals. We also provide empirical results in several benchmark domains to demonstrate that our technique helps return quality plans.

1 Introduction

In classical planning, a given set of conjunctive goals must all be achieved together for the plan to be a success. As planning problems get more complicated with temporal and resource constraints, it is harder to predict which goals are achievable. Recently, motivated by the requirements of different NASA planning problems such as planning for airborne, telescopes such as Hubble and SIRTF, and planning for a Mars Rover mission, Smith [2003; 2004] introduced the over-subscription planning problem. In this type of problem, the planner is not required to achieve all the goals but instead achieve the *best* subset of goals given the resource limitations.

While the ability to represent goals as soft constraints with (fixed) utility values makes it much easier to model goals in many applications, the restriction to only support soft logical goals has limitations in metric temporal planning domains. In this paper, we extend the framework to handle numeric goals having variable utility values and mixed goal sets containing both hard and soft goal constraints. We first present a motivating example before discussing our techniques to handle the new extensions.

Mars Rover: A Rover is sent to Mars to collect soil/rock samples and take pictures at different locations. The rover should first land safely, roll off the lander and then go to different locations to gather scientific data. While the objective is to collect as much data as possible given energy and time limitations, only a subset of the data can be gathered. The quality of the plan will be measured by the amount of samples collected at different locations and the amount of picture data transferred back to earth. Obviously, more samples and picture data give better values and thus it is more natural to model the goals of collecting them as numeric goals with the utility given according to the amount actually collected.

The example above brings up two issues: (1) there are certain types of goals that are better represented as numeric goals with a range of utility values (e.g. amount of sample/data); (2) there are goals that are critical to the success of the plans (e.g. landing safely) and there are goals that are not critical but beneficial (e.g. collecting samples). Besides the rover domain, those types of goals are also common in many other real-world planning problems such as transportation or travel planning. We extend the goal structure by allowing numeric goals (e.g. constraints on some continuous quantities) with continuous utility values that depend on the actual value achieved by the final plan. We also strike a middle ground between classical planning and over-subscription goal constraints by supporting both hard (critical) and soft (beneficial) goals. Those additional types of goal constraints complement traditional logical goals and allow the user more freedom in expressing what they desire from the final plan.

While extending the current framework of hard and soft goals to support a mixture of them is not overly challenging, effective handling of the numeric goals with continuous utility values does pose several challenges. In particular, we need to reason about the *degree of satisfaction* of goals. To illustrate, if the goal is $Sample \geq 5\ grams$, it can be satisfied by $Sample = 6$ as well as $Sample = 10$ at different degrees. We need techniques to:

1. Assign utilities to different degrees of satisfaction.
2. Track costs of achieving goals of different degrees of satisfaction.
3. Use the achievement costs and utilities in combination

*This research is supported in part by the NSF grant IIS-0308139 and IBM Faculty Award.

to estimate the final plan that maximizes the overall net benefit achievable from a given state.

We present *Sapa*$^{\mathcal{M}ps}$ an extension of *Sapa*^{ps} [Do & Kambhampati, 2004; van den Briel *et al.*, 2004] to support both numeric goals and mixed soft/hard goal sets. *Sapa*$^{\mathcal{M}ps}$ significantly extends the relaxed plan extraction procedure used in *Sapa*^{ps} to support numeric goals.

We first briefly discuss the search framework used in *Sapa*^{ps} in Section 2. We then show how to extend the cost propagation and relaxed plan extraction routines to handle metric quantities and the combination of soft and hard goals in Section 3. We present empirical results on extended versions of some well-known planning benchmark domains in Section 4 to show that the new techniques help find larger and better quality plans. We conclude the paper with the related work and discussion.

2 Background

We start with the formal definition of the over-subscription (aka partial-satisfaction) planning problem. We then proceed by describing the *Sapa*^{ps} [Do & Kambhampati, 2004; van den Briel *et al.*, 2004] planner and its framework that heuristically searches for good quality plans for the problem.

2.1 Over-Subscription Planning (OSP)

To formally define the over-subscription planning problems [Smith, 2004; van den Briel *et al.*, 2004], the following notations will be used: $\mathcal{F}$ is a finite set of fluents and $\mathcal{A}$ is a finite set of actions, where each action consists of a list of preconditions and a list of add and delete effects. $\mathcal{I} \subseteq \mathcal{F}$ is the set of fluents describing the initial state and $\mathcal{G} \subseteq \mathcal{F}$ is the set of goals. Hence we define a planning problem as a tuple $\mathbf{P} = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$. The OSP problem is defined as follows:

Definition OSP NET BENEFIT: Given a planning problem $\mathbf{P} = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ and, for each action $a \in \mathcal{A}$ a "cost" $c_a \geq 0$ and, for each goal specification $g \in \mathcal{G}$ a "utility" $u_g \geq 0$: find a finite sequence of actions $\Delta = \langle a_1, ..., a_n \rangle$ starting from $\mathcal{I}$ that leads to a state[1] $\mathcal{S}$ maximizing the net benefit value $\sum_{g \in \mathcal{S}_\mathcal{G}} u_g - \sum_{a \in \Delta} c_a$, where $\mathcal{S}_\mathcal{G}$ is the set of goals satisfied in $\mathcal{S}$.

Thus, in contrast to the classical planning problems, in OSP no goal achievement is needed to qualify the plan as valid (any sequence of actions is a valid plan). We thus put emphasis on finding *good quality* plans where the objective function is to maximize the tradeoff between total achieved goal utility and total incurred action cost (both additive).

2.2 *Sapa*^{ps} : Heuristic search for OSP

Sapa^{ps} adapts the heuristic (progression) search framework to solve OSP problems. The quality of each search node S visited by the A* search algorithm in a forward planner is ranked using the value $f(S) = g(S) + h(S)$ with: $g(S) = \sum_{g \in \mathcal{S}_\mathcal{G}} u_g - \sum_{a \in \Delta} c_a$. Starting from the initial state $\mathcal{I}$ with the net benefit value $g(\mathcal{I})$, *Sapa*^{ps} keeps track of all the visited states S that have better $g(S)$ values than the best state visited before S. Thus, instead of finding a single plan, the algorithm keeps searching for incrementally better plans in terms of the achieved net benefit ($g(S)$) value. The heuristic value $h(S)$ is calculated by extending the cost propagation over the planning graph routine. It is followed by the relaxed plan extraction process in *Sapa* [Do & Kambhampati, 2003]. The search stops when the first node in the queue has value $h(S) = 0$ (i.e. $f(S) = g(S)$). For the rest of this section, we will discuss the three steps in estimating the $h(S)$ value.

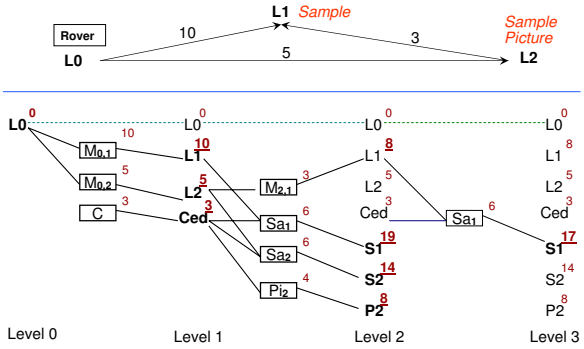


Figure 1: Rover example and the planning graph

Example: Figure 1 shows an example in the Mars Rover domain along with the complete planning graph structure for this example[2]. The rover initially resides at location L_0 and needs to collect samples at L_1, and to take a picture of L_2. The rover can: (1) move between two locations (M_{L_x,L_y}); (2) calibrate the equipment before taking sample/picture (C); (3) collect a sample at location l (Sa_l); and (4) take pictures (Pi_l). Action execution costs are depicted next to each action in the complete planning graph for the example shown in Figure 1. The first action layer of the graph contains three actions with their respective execution costs: $C_{M_{0,1}} = 10$, $C_{M_{0,2}} = 5$, and $C_C = 3$.

Cost-propagation over the planning graph: The planning graph propagates the achievement cost of each predicate and execution cost of each action forward starting at the initial level/time-point until fix-point. To simplify the discussion for this section, we will assume that all actions have unit duration.

Intuitively, the heuristic needs to realize that to be able to take the picture at L_2 we first need to have the camera calibrated and be at L_2. For each action a, its execution cost C_a is static and different from the cost to enable its execution $cost(a)$, which is the cost to achieve all of its preconditions. Thus, $cost(Pi_{L_2}) = cost(Ced) + cost(L_2) = C_C + C_{M_{0,2}} = 5 + 3$. We also want the heuristic to capture the fact that the cost to have a picture involves the cost to enable execution of Pi_{L_2} and the cost to actually carry out that action. Thus: $cost(P_{L_2}) = cost(Pi_{L_2}) + C_{Pi_{L_2}} = 8 + 4$. The propagation rules are:

1. Initially, at $t = 0$: (1) $\forall p \in \mathcal{I} : cost(p) = 0$; (2) $\forall p \notin \mathcal{I} : cost(p) = \infty$; (3) $\forall a \in \mathcal{A} : cost(a) = \infty$.
2. At level l: (1) $\forall a \in \mathcal{A} : cost(a_l) = \sum_{p \in Precond(a)} cost(p_{l-1})$; (2) $\forall p \in \mathcal{F} : cost(p_l) =$

[1] States are represented just as in the Sapa planner [Do & Kambhampati, 2003].

[2] We assume that the readers are familiar with the planning graph, which was originally described in [Blum & Furst, 1997].

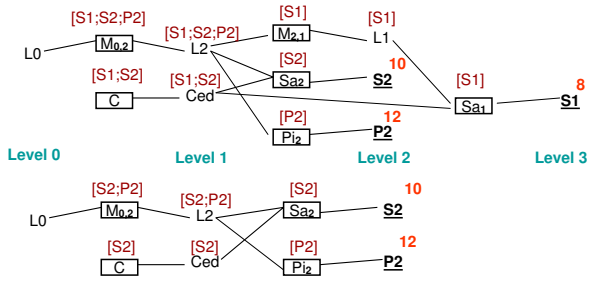


Figure 2: The relaxed plan

$min_{p \in Effect(a)} cost(a_{l-1})$

Our heuristic is inadmissible due to the use of sum propagation in calculating the cost to execute each action. As we grow the graph, new actions and facts are introduced and the cost to achieve facts ($cost(p)$) and execute actions ($cost(a)$) decreases due to new ways to achieve and support them. In Figure 1, we highlight the new facts and actions introduced at each level and the new achievement costs for each fact. For example, L_1 first appears at level 1 with $cost(L_1) = 10$ (achievable by $M_{0,1}$). The value of $cost(L_1)$ decreases to 8 at level 2 due to the new action $M_{2,1}$ in level 1. While the set of achieved facts and supporting actions in level 2 and level 3 are the same, we did not stop growing the graph at level 2. This is because $cost(S_1)$ decreases at level 3 due to the reduction in $cost(L_1)$ at level 2 (which leads to the decrease in $cost(Sa_1)$).

Extracting the relaxed plan: After terminating the cost-propagation, the cost values can be used to extract the relaxed plan RP, starting from all achieved goals G, as follows:

1. Starting from the last level, $\forall g \in G$ at level l select action a at the action level $l-1$ that supports g with the lowest cost.
2. When action a is selected at level l, then all preconditions p of a at the previous level will be added to G.
3. Stop when: $G \subset \mathcal{I}$.

All the collected actions a and the causal links between them make up the relaxed plan RP.

Refining the relaxed plan: For each goal g, we build the goal supporting set GS for each proposition p and action a by going backward using the extracted relaxed plan as follows:

- $\forall g \in G : GS(g) = \{g\}$
- $GS(a) = \bigcup GS(p) : p \in Effect(a)$
- $GS(p) = \bigcup GS(a) : p \in Precond(a)$

Intuitively, for each action a, $GS(a)$ is the set of goals that a supports. Thus, the achievement of any goal in $GS(a)$ depends on the inclusion of a in the relaxed plan while for any goal $g \notin GS(a)$, g will still be achievable without a. In Figure 2, we show the goal supporting sets for all actions and related propositions (e.g. $GS(C) = \{S_1, S_2\}$) and the corresponding utility values (e.g. $Util(S_2) = 10$) of the three goals in the relaxed plan. For each set $S_G \subset G$, let $A_{S_G} = a : GS(a) \subset S_G$ be the set of actions supporting only goals in S_G. We will remove S_G along with A_{S_G} from the relaxed plan if $\Sigma_{a \in A_{S_G}} cost(a) > \Sigma_{g \in S_G} Util(g)$ (i.e. cost > utility).

In our ongoing example, only the goal set $S = \{S_1\}$ can be removed because $cost(A_{S_1}) = cost(M_{2,1}) + cost(Sa_1) = 3 + 6 > Util(S_1) = 8$. Figure 2 shows the relaxed plan before and after refinement.

The net benefit of the final relaxed plan is: $Util(\{S_2, P_2\}) - cost(\{M_{0,2}, C, Sa_2, Pi_2\}) = (10 + 12) - (5 + 3 + 6 + 4) = 4$. This is used as the heuristic value $h(S)$ to guide the A* search algorithm. In general, we define $RP(S)$ to be the relaxed plan found for state S and $U(Apply(RP(S), S))$ to be the utility achieved by applying all actions in the relaxed plan to state S. The heuristic value is then calculated as: $h(S) = (U(Apply(RP(S), S)) - U(S)) - cost(RP(S))$ where $RP(S)$ is the final relaxed plan.

3 Handling Numeric Goals with Utility

Sapa^{ps} only supports logical achievement goals of the form $g = True$ (e.g. $HasPicture(L_2)$). However, if we have the goal to collect at least m grams of a Mars soil sample at a given location l (i.e. $sample(l) > m$), we can more naturally represent it as a numeric goal.

3.1 Numeric Goal Representation

Unlike logical goals that only have true/false values, there are an arbitrarily large number of values that can satisfy a given numeric goal. We assign a range of *continuous utility values* for numeric goals to represent a *degree of satisfaction*. Specifically, numeric goals and their utility values are set up as follows:

Definition NUMERIC GOAL: A numeric goal is a relation $f \in D$ in which f is a mathematical formula involving an arbitrary number of numeric variables and $D = [l, u]$ is an interval open or closed at either end and is bounded by two real values $l \leq u$ (l, u can be infinity).

For example, the goal of keeping travel cost between \$100 and \$500 can be represented as: $Hotel + AirTicket \in [100, 500)$.

For each numeric goal $g \in D$, the utility value $u(g)$ is specified by a linear function. For example, if the goal is to collect at least 10 grams of Mars sample (i.e. $f = Sample > 10$), then the utility of this goal can be $u(f) = 10^6 * Sample$ (i.e. it is worth 1 million dollars for each gram of Mars' soil if we have at least 10 grams, but 0 dollars otherwise).

3.2 Cost Propagation with Numeric Goals

To incorporate numeric goals into our current heuristic framework, we first have to be able to estimate the cost to achieve them. Unlike logical goals, there are multiple degrees of satisfaction for a numeric goal g. Therefore, the procedure that tracks the achievement cost for g is necessarily more complicated. Specifically:

- For logical values we track the time point t_p at which a proposition p can first be achieved ($p = True$) and the achievement cost for p at time points $t \geq t_p$. For numeric values we need to track a *range* of possible values $[\mathsf{L}_j^{v_i}, \mathsf{U}_k^{v_i}]$ for a numeric variable v_i at each j^{th} update to the lower bound value and k^{th} update to the upper bound value.

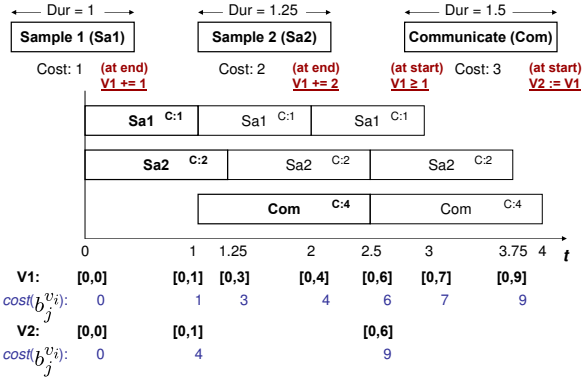


Figure 3: The RTPG for our example. Our actions are defined above it.

- In tracking costs to achieve logical literals, actions are only *re-applied* (e.g. Sa_1 at action level 2 in Figure 1) when the cost to execute (i.e. cost to achieve their preconditions) them decreases. However, actions having numeric effects on v_i need to be applied whenever possible (e.g. Sa_1 in Figure 3) because their effects continue to change the upper ($\mathsf{U}_k^{v_i}$) or lower ($\mathsf{L}_j^{v_i}$) bound values of the quantity v_i.

Example: To illustrate the techniques to track achievement costs for numeric goals, we will use a variation of our ongoing Mars Rover example. We solely concentrate on metric quantities in this example (illustrated in Figure 3). There are two sample-collecting actions: *sample1* (Sa_1) collects a single gram of soil sample; *sample2* (Sa_2) collects 2 grams of soil sample. The effects of these actions occur at the end of execution. The third action, *communicate* (Com), communicates the sample information to a lander at the start of execution. We use two continuous variables; v_1 to track the weight of the collected soil sample in the Rover's store and v_2 to track the total amount of communicated sample. The goal g is to achieve $v_2 > 5$ and the goal utility formula is $u(g) = v_2 * 3$ (i.e. if $v_2 \leq 5$ we get a utility of zero, otherwise the utility is found using $u(g)$).

While the connection between time and numeric goals is not obvious in OSP problems, one important component of action cost is the amount of time consumed by each action. Goal utilities also normally depend on the time the goals are achieved. Like *Sapa*^{ps} , the *Sapa*$^{\mathcal{M}ps}$ planner handles actions with different durations and thus we do not make the assumption that all actions have uniform duration (as in previous section).

Tracking the upper and lower bounds: The first step in estimating the achievement costs for numeric goals is to track the possible values for numeric variables at different time points. Previous work in tracking upper/lower bounds for numeric variables using the planning graph was done for non-temporal planning [Koehler, 1998; Hoffman, 2003; Sanchez & Mali, 2003]. There, actions are either always executed serially or marked mutually exclusive of one another if their numeric effects give varying results when ordered differently. In *Sapa*$^{\mathcal{M}ps}$, the semantics of the planner disallows interacting actions to be concurrent, whereas the relaxed temporal planning graph (RTPG) allows this. In our context, this means that an action in the RTPG can be applied concurrently with itself an arbitrary number of times, causing the number of time points in the graph to increase significantly. To avoid this problem, we disallow such concurrency as a practical compromise.

Figure 3 shows a RTPG for our example. We re-apply the numeric effects of actions directly after their duration completes. At time $t = 1$, action Sa_1 completes and we add the upper bound value $\mathsf{U}_1^{v_1} = 1$ of v_1 to indicate that the collected weight of the soil sample has increased by 1. Also at this time point, the precondition of Com can be satisfied by the bound $\mathsf{U}_1^{v_1} = 1$, so we apply the numeric effects of the action, adding the upper bound $\mathsf{U}_1^{v_2} = 1$ on the *weight* of the soil sample. At $t = 1.25$, Sa_2 completes and we add the upper bound value $\mathsf{U}_2^{v_1} = 1 + 2 = 3$ increasing the previous upper bound of v_1 the second time by 2 according to the numeric effect of Sa_2. This continues until we reach our numeric goal $v_2 > 5$ (when $(\mathsf{U}_2^{v_2} = 6) > 5$). In Figure 3, we show the upper and lower bound values for v_1 and v_2 as we grow the graph. Because we do not have actions that decrease the values of v_1 or v_2 in this simple example, the lower bound values of those two variables remain unchanged.

The RTPG handles numeric expressions in effects and preconditions by applying the formulas to each bound. For instance, if the bounds of three variables v_1, v_2, v_3 are $v_1 : [0, 20]$, $v_2 : [1, 5]$ and $v_3 : [-1, 3]$ and we want to calculate $f = v_3 + v_1 * v_2$, we first find $v_1 * v_2 = [min(0 * 1, 0 * 5, 20 * 1, 20 * 5), max(0 * 1, 0 * 5, 20 * 1, 20 * 5)] = [0, 100]$ then $f = v_3 + [0, 100] = [0 + -1, 100 + 3] = [-1, 103]$.

Tracking achievement costs: We let $b_j^{v_i}$ refer the j^{th} upper or lower bound of v_i in the RTPG. The RTPG associates for each bound value[3], $b_j^{v_i}$, a propagated cost, $cost(b_j^{v_i})$. The cost value estimates how costly it is to achieve a certain numeric value. The idea is that for each value n that satisfies some numeric goal, the tradeoff between the cost of achieving n and the utility that n incurs will be used as heuristic guidance.

Cost propagation is not trivial in the presence of numeric expressions. Before turning to this case, let us concentrate our discussion on simple numeric effects (i.e. effects using only constant values). Numeric updates (e.g. increases and decreases) will generate a new bound value with respect to the previous one. Because of this, we base the cost of each bound on the previous bound's cost, $cost(b_{j-1}^{v_i})$. Specifically, when an action a adds an upper or lower bound for a variable v_i using an increase (+=) or decrease (-=) numeric effect, the propagated cost of the bound is $cost(b_j^{v_i}) = cost(a) + cost(b_{j-1}^{v_i})$. This lets us track the cost of executing several actions that may be required to reach a numeric goal or precondition. Bounds found using an assign (:=) numeric effect only depend upon the action itself. So, the propagated cost is $cost(b_j^{v_i}) = cost(a)$. In our example shown in Figure 3, the cost of the new bound found by the numeric effect of Sa_2 at $t = 1.25$ is $cost(\mathsf{U}_2^{v_1}) = cost(Sa_2) + cost(\mathsf{U}_1^{v_1}) = 3$. When the numeric effect of Sa_1 is re-applied at $t = 2$ and causes

[3]The RTPG can find multiple upper and lower bound values for a variable v_i at the same time point. We index the bounds in this manner so that we may refer to previously found bounds without regard to the time point that they may appear in the RTPG.

the third update on the value of v_1 we have $cost(\mathsf{U}_3^{v_1}) = cost(Sa_1) + cost(\mathsf{U}_2^{v_1}) = 4$.

In our ongoing example, each numeric effect and precondition involves only two or fewer variables. However, in a more general scenario, numeric goals, action preconditions and action effects can be a formula involving an arbitrary number of numeric variables. These inter-dependencies between variables further complicates cost propagation on bounds. That is, when we calculate new bound values from expressions, we need to find the cost of the new bound based upon the costs of the values involved in the expression. To do this, we define for each variable v_i, a set $\mathcal{B}_{v_i}$ of all bound values involved in computing a new bound for variable v_i. For example, we have an expression $f = v_3 + v_1 * v_2$ and define an effect v_4 += f. When applying this effect, we track each of the bounds used to generate the minimum and maximum values from this formula (in this case [-1, 103]). We apply the effect using the resulting bounds. For the new upper bound of v_4, we have the set $\mathcal{B}_{v_4} = \{\mathsf{U}^{v_1} = 20, \mathsf{U}^{v_2} = 5, \mathsf{U}^{v_3} = 3\}$ to indicate the dependencies between the upper bound of v_4 and the particular bound values of v_1, v_2, and v_3 used to achieve this new upper bound.

During cost propagation, bounds used to satisfy a precondition are included in the cost of an action. For each variable used in expressions, we allow only a single bound of that variable to be included when calculating costs of new bounds (i.e. avoid including more than one bound for each variable when a variable is used both to support a precondition and as part of an expression in a numeric effect). To do this, we let $\mathcal{P}_a$ be the set of all bound values used to satisfy the numeric preconditions of an action a. In our example, $\mathcal{P}_{Com} = \{\mathsf{U}^{v_1} = 1\}$ starting at $t = 1$. We also define $\mathcal{P}_a \overline{\bigcap} \mathcal{B}_{v_i}$ to be a set operation over the variables represented by the bounds in $\mathcal{P}_a$ and $\mathcal{B}_{v_i}$, where the result gives us only the bounds in $\mathcal{P}_a$ that are not equal to the bounds in $\mathcal{B}_{v_i}$. More formally, $\mathcal{A} \overline{\bigcap} \mathcal{B} = \{b_j^{v_i} : b_j^{v_i} \in \mathcal{A} \wedge \exists b_{k \neq j}^{v_i} \in \mathcal{B}\}$.

For instance, if we have the sets $\mathcal{P}_a = \{\mathsf{L}_1^{v_1} = 1, \mathsf{U}_1^{v_2} = 2\}$ and $\mathcal{B}_{v_i} = \{\mathsf{U}_3^{v_1} = 6, \mathsf{U}_1^{v_2} = 2, \mathsf{U}_1^{v_3} = 1\}$ then $\mathcal{P}_a \overline{\bigcap} \mathcal{B}_{v_i} = \{\mathsf{L}_1^{v_1} = 1\}$. This result provides the means for removing the costs of bounds in $\mathcal{P}_a$ that are already present in an action, when we also use them in $\mathcal{B}_{v_i}$ to calculate bound cost. Thus, the cost of a bound value of variable v_i that is changed by an increase (+=) or decrease (-=) effect of an action a is $cost(b_j^{v_i}) = cost(a) + cost(b_{j-1}^{v_i}) + \sum_{d \in \mathcal{B}_{v_i} \setminus \mathcal{P}_a} cost(d) - \sum_{e \in \mathcal{P}_a \overline{\bigcap} \mathcal{B}_{v_i}} cost(e)$. It follows that the cost of a bound found by an assign effect (:=) on action a is $cost(b_j^{v_i}) = cost(a) + \sum_{d \in \mathcal{B}_{v_i} \setminus \mathcal{P}_a} cost(d) - \sum_{e \in \mathcal{P}_a \overline{\bigcap} \mathcal{B}_{v_i}} cost(e)$.

As shown in Figure 3, at $t = 0$ both actions Sa_1 and Sa_2 are added to the RTPG. The delayed numeric effect of Sa_1 increases v_1 by 1 at $t = 1$ and incurs $cost(\mathsf{U}_1^{v_1} = 1) = 1$. At this time point, the precondition $v_1 \geq 1$ of Com is satisfiable and we have $\mathcal{P}_{Com} = \{\mathsf{U}_1^{v_1} = 1\}$. The $cost(\mathsf{U}_1^{v_1} = 1)$ is included to find $cost(Com)$. So, when Com is put in the graph at $t = 1$, we have $cost(Com) = 4$. Its instantaneous effect $v_2 := v_1$ leads to a new bound for v_2 with $cost(\mathsf{U}_1^{v_2} = 1) = cost(Com) = 4$. At $t = 1.25$, Sa_2's delayed numeric effect is activated and increases v_1 by 2 to a new bound with cost $cost(\mathsf{U}_2^{v_1} = 3) = 3$. As we increase the time to $t = 2$ and $t = 2.5$, we keep updating the upper bound of v_1 and v_2 to values as shown in Figure 3. When activating Com at $t = 2.5$, we have the set $\mathcal{B}_{v_i} = \{\mathsf{U}_4^{v_1} = 6\}$ representing the bounds used for the expression $v_2 := v_1$. We get a new bound that satisfies the numeric goal with $cost(\mathsf{U}_2^{v_2} = 6) = cost(Com) + cost(\mathsf{U}_4^{v_1} = 6) - cost(\mathsf{U}_1^{v_1} = 1) = 9$. At $t = 3$ the update of Sa_1 completes, giving $cost(\mathsf{U}_5^{v_1} = 7) = 7$ and at $t = 3.75$, the numeric effect of Sa_2 gives $cost(\mathsf{U}_6^{v_1} = 9) = 9$.

Notice that even after numeric goals $f \in [L, U]$ are satisfied by the bound values on f at a given time point t (i.e. $[L, U] \bigcap [l^f, u^f] \neq \emptyset$ at t), we allow the RTPG to continue to expand until fix-point.

Relaxed Plan Extraction with Numeric Goals: After doing cost-propagation over the RTPG, the cost information can be used to extract a relaxed plan using an approach similar to that discussed in Section 2.2. The challenge here is in deciding for each numeric goal $f \in [l, u]$, how to select the most beneficial value v_f of f that is achievable through the planning graph and extract the action that supports that v_f value. When selecting an action a we add all of its logical and numeric preconditions into the goal set G. Also, we ensure that the cost of numeric bounds used to satisfy the numeric goal constraints are included with the relaxed plan. This is so we can accurately determine the achievement cost for each bound.

To handle relaxed plan extraction for numeric goals, we choose the bound values that provide the best tradeoff between goal utility value[4] with the achievement cost. Thus, for each achievable value $v_g \in [\mathsf{L}_j^{v_g}, \mathsf{U}_k^{v_g}]$ that satisfies the goal constraint on g, we select the one that gives the greatest $U(v_g) - cost(v_g)$ value. The action a that supports v_g is then selected and added to the relaxed plan. If given a goal interval $v_i \in [l, u]$ we never find a value $v \in [l, u]$ while expanding the RTPG, but do find two values $v_l < l$ and $v_u > u$, we say the goal is subsumed. In this case, we allow the first subsuming value to support the goal. However, since we cannot estimate the utility on this bound, we let its utility be 0.

Heuristic estimation: For each numeric goal supporting bound that we select, we include the cost to support it and its utility value in the net benefit calculation. In our ongoing example the net benefit of the relaxed plan would be $U(\{u(v_2 = 6)\}) - (C(Sa_2) + C(Com) + C(\mathsf{U}_4^{v_1} = 6) - C(\mathsf{U}_1^{v_1} = 1)) = (6 * 3) - (2 + 4 + 6 - 1) = 7$.

3.3 Combining Hard & Soft Goals

In our work, we support hard and soft goals for both traditional logical goals and the numeric goals discussed in the previous section. In the case of numeric goals, a single goal may involve both hard and soft constraints. For example the goal of having the Rover collect between 5 to 10 grams of Mars soil ($SC = SoilCollected \in [5, 10]$), can be modeled so that $SC > 5$ is a hard constraint (i.e. should collect at least 5 grams) and $SC < 10$ is a soft constraint (i.e. 10 is enough but more than that is not harmful).

To support both hard and soft goals in the best first search framework for over-subscription problem, we need to change

[4]Recall that the utility values are given by a linear formula over the goal variable, g.

both the search-termination criterion and the heuristic estimation routine in the search framework discussed in Section 2.2. Specifically:

Search node evaluation: When all goals are soft, any visited node encodes a legal plan. In the presence of hard goals, only nodes that satisfy all hard goals can be represented a valid plan.

Heuristic estimation:

- For a given node, if we cannot achieve some hard goal when building the planning graph until fix-point for cost propagation, then the node is a deadend.
- When refining the relaxed plan, we only remove soft goals and actions (solely) supporting them.

Notice that with the mixed soft/hard goal combination, it's possible to return plans that have negative net benefit (total action cost is higher than the total achieved goal utility). Those plans with negative net benefit will not be found if all the goals are soft constraints because an empty plan with zero net benefit is a valid plan in that case.

4 Empirical Evaluation

We have implemented the support for numeric goals on top of the *Sapa*^{ps} planner and also extended its best-first-search framework to support the combination of hard and soft goals. We call the new planner *Sapa*$^{\mathcal{M}ps}$. To test the implementation, we have created a test suite by extending the problem sets of the *Mars Rover* and *Logistic* domains. Both were used in previous planning competitions (IPC2002 and IPC2000). The experiments aim at testing whether or not the cost-propagation techniques for numeric goals can help *Sapa*$^{\mathcal{M}ps}$ find reasonable size plans with good quality for the extended OSP problems.

4.1 Configuration of Testing Domains

All test suites involve durative actions consuming metric resources. Action costs are added to the action representation and appear as a function of the time and resources consumed by that action (e.g. $cost(travel) = HotelRate * Duration(travel) + AirTicket$). Utilities of goals are randomly generated within some reasonable bounds. Logical goals have fixed utilities while utilities of numeric goals are represented using linear functions. The goals have the same probability to be either *hard* or *soft* (applied to both logical and numeric goals). Specific types of goals in the domains are created with various probabilities. In general the potential number of numeric goals increases with the problem number, with a minimum of at least one numeric goal.

The *Rovers* domain is extended and includes the weight of collected soil and rock samples. The goals are to send analyzed data about samples at different locations back to the lander. Utility of a sample is proportional to the weight collected for the sample. Rover's store has a fixed weight capacity and each time a sample action is executed, a single gram is stored. If the Rover keeps collecting sample, then the cost of sampling outweighs the overall utility of the collected sample at some certain time.

In the *Logistics* domain, we add numeric goals to deliver a certain number of packages of the same type to its destination using trucks and airplanes. The initial state is extended

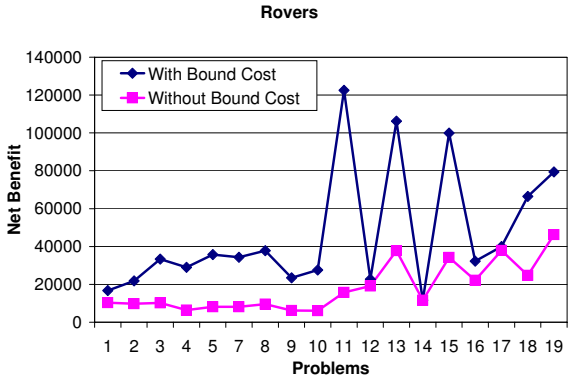


Figure 4: Comparison of utilities for our rovers domain

to represent numeric quantities for the number of packages at each location and the maximum capacity of each vehicle. Goals are then intervals of the number of packages we want at a particular location. For example, we may start with 10 packages at location l_1 and want to deliver between 5 and 7 packages to l_2 and at least 2 packages to l_3. Delivering a package to a certain location may be more beneficial than to the other (e.g. delivering 5 packages to l_2 gives more utility than delivering the same number of packages to l_3). The cost of delivering packages varies depending upon the distance between locations.

4.2 Results

All results were collected using a 2.67Ghz P4 machine with, 1 GB of RAM and 512 MB allocated to the planner. Given that *Sapa*$^{\mathcal{M}ps}$ employs an anytime search algorithm and continues to look for better quality plans as it is given more time, we set the time limit of 1200 seconds for all the tests[5]. We compare the final plan quality (i.e. the cumulative utility of goals achieved minus the cumulative action cost) on two heuristic approaches: (1) based on the cost-propagation as described in this paper; (2) only propagate the upper/lower bounds on numeric variables to check for numeric goal satisfaction, but do not propagate the goal achievement cost. The plan qualities are measured by the total achieved goal utility minus the total action cost. We were unable to make comparison with external planners because we are not aware of any other planner that is capable of solving the same problems.

Figure 4 shows the comparison results between two approaches listed above for the Rovers domain. The results clearly show that the cost-propagation helps *Sapa*$^{\mathcal{M}ps}$ return better quality solutions in all but 3 problems (with one of lower quality). The average improvement on the solution quality is 3.06 times better. The plans found are of reasonable size with a average/longest plan length in terms of number of actions is 30.5/77 (with cost propagation) and 20.9/67 (without cost propagation).

Figure 5 shows the results for the Logistics domain. Within the time limit, *Sapa*$^{\mathcal{M}ps}$ solves 19 of 20 problems with cost-propagation while only 10 (all with lower quality) without cost-propagation. Among the 10 problems that both approaches can solve, the average improvement in plan qual-

[5] *Sapa*$^{\mathcal{M}ps}$ usually finds the first solution with positive benefit values in very short time

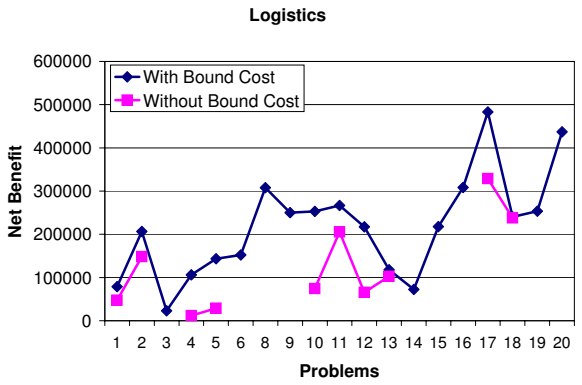


Figure 5: Comparison of utilities for our logistics domain

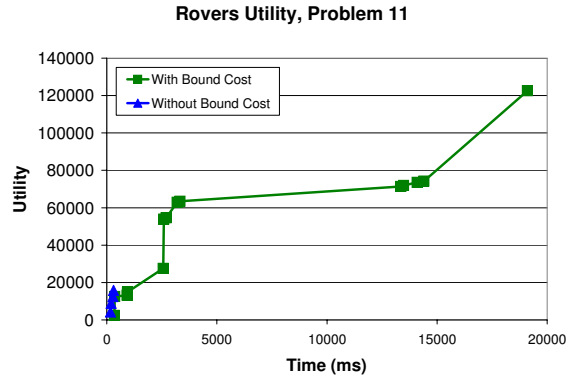


Figure 6: Plan net benefit as a function of time.

ity is 2.88 times. The average/longest plan length in terms of number of actions is 71.7/284 (with cost propagation) and 50.3/157 (without cost propagation) in this domain.

The results in these two domains confirm that the cost-propagation technique is indeed useful in helping $Sapa^{\mathcal{M}ps}$ find better quality and larger plans, compared to only doing bound propagation.

Figure 6 shows how quickly the utility of a typical problem is found (in this case, problem instance 11 of rovers). Since we are using an anytime A* search framework, the utility increases gradually during search. The cost propagation heuristic over numeric bounds enables the planner to find greater utility plans.

5 Related Work

Our primary focus in this paper was to develop effective heuristics for handling numeric goals in the context of over-subscription planning. As we mentioned in the introduction, the recent interest in over-subscription planning has lead to the development of several algorithms [van den Briel *et al.*, 2004; Smith, 2004]. None of these approaches can handle numeric goals, and thus do not address the challenges involved in deciding the degree of satisfaction of a goal. Earlier work on the PYRRHUS planning system [Williamson & Hanks, 1994] did consider the issues of handling goals that allow different degrees of satisfaction in terms of achievement time. However, PYRRHUS did not have effective heuristics for directing its search.

As discussed in the paper, our work is also related to existing efforts on cost propagation over planning graphs (c.f. [Do & Kambhampati, 2003]) and propagating reachability information over numeric goals (c.f. [Sanchez & Mali, 2003; Hoffman, 2003]). Our method of propagating the upper and lower bounds of numeric variables over the RTPG has its roots in work done by Koehler in the Metric-IPP planner [1998].

6 Conclusion

Many real-world planning scenarios are over-subscribed, and require that the planner carefully balance the utility of the achieved goals and the cost of actions in the plan. In this paper we focused on extending over-subscription planning to handle numeric goals as well as a mix of hard and soft goals. Our primary technical contributions involved effective approaches for reasoning about the expected net benefit of a partial plan in the presence of numeric goals that allow different degrees of satisfaction (with noncomitant utilities). We have empirically demonstrated the effectiveness of our heuristics in leading the planner to solutions with higher net benefit. For the future, we are investigating extensions to our heuristic to handle delayed satisfaction of goals (i.e. goals whose utility depend upon achievement time).

References

[Blum & Furst, 1997] Blum, A., and Furst, M. 1997. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence* 90:281–300.

[Boutilier et. al., 1999] Boutilier, C., Dean, T., and Hanks, S. 1999. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. In *JAIR 11 (p.1-94)*

[Do & Kambhampati, 2003] Do, M. and Kambhampati, S. 2003. Sapa: a multi-objective metric temporal planer. In *JAIR 20 (p.155-194)*

[Do & Kambhampati, 2004] Do, M. and Kambhampati, S. 2004. Partial Satisfaction (Over-Subscription) Planning as Heuristic Search. In *Proc. of KBCS-2004.*

[Hoffman, 2003] Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. In *JAIR 20 (p.291-341)*

[Koehler, 1998] Koehler, J. 1998. Planning Under Resource Constraints. In *Proc. of ECAI-98 (p.489-493)*

[Sanchez & Mali, 2003] Sanchez, J. and Mali, A. 2003. S-MEP: A Planner for Numeric Goals. In *Proc. of ICTAI*

[Smith, 2003] Smith, D. 2003. The Mystery Talk. *Plannet Summer School*

[Smith, 2004] Smith, D. 2004. Choosing Objectives in Over-Subscription Planning. In *Proc. of ICAPS-04.*

[van den Briel *et al.*, 2004] van den Briel, M., Sanchez, R., Do, M, and Kambhampati, S. 2004. Effective Approaches for Partial Satisfaction (Over-Subscription) Planning. In *Proc. of AAAI-04*

[Williamson & Hanks, 1994] Williamson, M., and Hanks, S. 1994. Optimal Planning with a Goal-Directed Utility Model. In *Proc. of AIPS-94.*

Robust Planning with (L)RTDP

Olivier Buffet and **Douglas Aberdeen**
National ICT Australia &
The Australian National University
{olivier.buffet,douglas.aberdeen}@nicta.com.au

Abstract

Stochastic Shortest Path problems (SSPs), a subclass of Markov Decision Problems (MDPs), can be efficiently dealt with using *Real-Time Dynamic Programming* (RTDP). Yet, MDP models are often uncertain (obtained through statistics or guessing). The usual approach is robust planning: searching for the best policy under the worst model. This paper shows how RTDP can be made robust in the common case where transition probabilities are known to lie in a given interval.

1 Introduction

In decision-theoretic planning, Markov Decision Problems [Bertsekas and Tsitsiklis, 1996] are of major interest when a probabilistic model of the domain is available. A number of algorithms make it possible to find plans (policies) optimizing the expected long-term utility. Yet, optimal policy convergence results all depend on the assumption that the probabilistic model of the domain is accurate.

Unfortunately, a large number of MDP models are based on uncertain probabilities (and rewards). Many rely on statistical models of physical or natural systems, such as plant control or animal behavior analysis. These statistical models are sometimes based on simulations (themselves being mathematical models), observations of a real system or human expertise.

Working with uncertain models first requires answering two closely related questions: 1– how to model this uncertainty, and 2– how to use the resulting model. Existing work shows that uncertainty is sometimes represented as a set of possible models, each assigned a model probability [Munos, 2001]. The simplest example is a set of possible models that are assumed equally probable [Bagnell *et al.*, 2001; Nilim and Ghaoui, 2004]. Rather than construct a possibly infinite set of models we represent model uncertainty by allowing each probability in a single model to lie in an interval [Givan *et al.*, 2000; Hosaka *et al.*, 2001].

Uncertain probabilities have been investigated in resource allocation problems [Munos, 2001] — investigating efficient exploration [Strehl and Littman, 2004] and state aggregation [Givan *et al.*, 2000] — and policy robustness [Bagnell *et al.*, 2001; Hosaka *et al.*, 2001; Nilim and Ghaoui, 2004]. We focus on the later, considering a two-player game where the opponent chooses from the possible models to reduce the long-term utility.

Our principal aim is to develop an efficient planner for a common sub-class of MDPs for which some policies are guaranteed to eventually terminate in a goal state: Stochastic Shortest Path (SSP) problems. The greedy *Real-Time Dynamic Programming* algorithm (RTDP) [Barto *et al.*, 1995] is particularly suitable for SSPs, as it finds good policies quickly and does not require complete exploration of the state space.

This paper shows that RTDP can be made robust. In Section 2, we present SSPs, RTDP and robustness. Then Sec. 3 explains how RTDP can be turned into a robust algorithm. Finally, experiments are presented to analyse the behavior of the algorithm, before a discussion and conclusion.[1]

2 Background

2.1 Stochastic Shortest Path Problems

A Stochastic Shortest Path problem [Bertsekas and Tsitsiklis, 1996] is defined here as a tuple $\langle S, s_0, G, A, T, c\rangle$. It describes a control problem where S is the finite set of **states** of the system, $s_0 \in S$ is a starting state, and $G \subseteq S$ is a set of goal states. A is the finite set of possible **actions** a. Actions control transitions from one state s to another state s' according to the system's probabilistic dynamics, described by the **transition function** T defined as $T(s,a,s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$. The aim is to optimize a performance measure based on the **cost function** $c : S \times A \times S \to \mathbb{R}^+$.[2]

SSPs assume a goal state is reachable from any state in S, at least for the optimal policy, so that one cannot get stuck in a looping subset of states. An algorithm solving an SSP has to find a **policy** that maps states to probability distributions over actions $\pi : S \to \Pi(A)$ which optimizes the **long-term cost** J defined as the expected sum of *costs* to a goal state.

In this paper, we only consider SSPs for planning purposes, with full knowledge of tuple defining the problem: $\langle S, s_0, G, A, T, c\rangle$. In this framework, well-known stochastic dynamic programming algorithms such as *Value Iteration* (VI) make it possible to find a deterministic policy that corresponds to the minimal expected long-term cost J^*. Value Iteration works by computing the function $J^*(s)$ that gives

[1] Work presented in more details in [Buffet and Aberdeen, 2004].

[2] As the model is not certain, we do not make the usual assumption $c(s,a) = \mathbb{E}_{s'}[c(s,a,s')]$.

the expected sum of costs of the optimal policies. It is the unique solution of the fixed point Bellman equation:

$$J(s) \quad = \quad \min_{a \in A} \sum_{s' \in S} T(s,a,s') \left[c(s,a,s') + J(s') \right]. \quad (1)$$

Updating J with this formula leads to asymptotic convergence to J^*. For convenience, we also introduce the Q-value: $Q(s,a) = \sum_{s' \in S} T(s,a,s')[c(s,a,s') + V(s')]$.

SSPs can easily be viewed as shortest path problems where choosing a path only probabilistically leads you to the expected destination. They can represent a useful sub set of MDPs, as they are essentially finite-horizon MDPs with no discount factor.

2.2 RTDP

Trial based[3] *Real-Time Dynamic Programming* (RTDP), introduced in [Barto *et al.*, 1995], uses the fact that the SSP costs are positive and the additional assumption that every trial will reach a goal state with probability 1. Thus, with a zero initialization of the long-term cost function J, both J and Q-values monotonically increase during their iterative computation.

The idea behind RTDP (Algorithm 1) is to follow paths from the start state s_0, always greedily choosing actions with the lowest long-term cost and updating $Q(s,a)$ as states s are encountered. In other words, the action chosen is the one expected to lead to the lowest future costs, until the iterative computations show that another action may do better.

Algorithm 1 RTDP algorithm for SSPs

```
RTDP(s:state) // s = s_0
repeat
   RTDPTRIAL(s)
until // no termination condition

RTDPTRIAL(s:state)
while ¬GOAL(s) do
   a =GREEDYACTION(s)
   J(s)=QVALUE(s, a)
   s =PICKNEXTSTATE(s, a)
end while
```

RTDP has the advantage of quickly avoiding plans that lead to high costs. Thus, the exploration looks mainly at a promising subset of the state space. Yet, because it follows paths by simulating the system's dynamics, rare transitions are only rarely taken into account. The use of a simulation makes it possible to get good policies early, but at the expense of a slow convergence, because of the bad update frequency of rare transitions.

2.3 Robust Value Iteration

Pessimism and Optimism — We now turn to the problem of taking the model's uncertainty into account when looking for a "best" policy. The (possibly infinite) set of alternative models is denoted $\mathcal{M}$.

[3]We always assume a trial based RTDP implementation.

A simplistic approach computes the average model over $\mathcal{M}$, or the most probable model in $\mathcal{M}$, then uses standard SSP optimization methods. Such approaches guarantee nothing about the long-term cost of the policy if the true model differs from the one chosen for optimization.

We follow the approach described in [Bagnell *et al.*, 2001], that consists of finding a policy that behaves well under the worst possible model. This amounts to considering a two-player zero-sum game where a player's gain is its opponent's loss. The player chooses a policy over actions while its "disturber" opponent simultaneously chooses a policy over models (as this is a simultaneous game, optimal policies can be stochastic). This results in a max-min-like algorithm:

$$\max_{\pi_{\mathcal{M}} \in \Pi_{\mathcal{M}}} \min_{\pi_A \in \Pi_A} J_{\pi_{\mathcal{M}}, \pi_A}(s_0).$$

In this SSP game, Value iteration does converge to a fixed solution [Patek and Bertsekas, 1999].

It is also possible to be optimistic, considering that both players collaborate (as they endure the same costs), which turns the $\max$ into a $\min$ in previous formula. This second case is equivalent to a classical SSP where a decision consists of choosing an action and a local model.

Locality — Such a max-min algorithm would be particularly expensive to implement. Even restricting the search to a deterministic policy over models, it requires computing the optimal long-term cost function for each model before picking the worst model found and the optimal policy associated with it. However, a simpler process may be used to compute J while looking simultaneously for the worst model. It requires the hypothesis that next-state distributions $T(s,a,\cdot)$ are independent from one state-action pair (s,a) to another. This assumption does not always hold. However, this makes things easier for the opponent because it now has larger set of models from which to choose. This consequence is conservative for producing robust policies.

Because we assume independence at the state-action level (not only at the state level), it is equivalent to a situation where the second player makes a decision depending on the current state and the first player's action. This situation amounts to a *sequential* game where the previous players move is known to the next player: both players can act in a deterministic way without loss of efficiency.

The result of this assumption is that the worst model can be chosen locally when Q is updated for a given state-action pair. As can be seen from Algorithm 2, the worst local model m_s^a may change while Q-values evolve. Previous updates of reachable states' long-term costs may change their relative ordering, changing which outcomes are considered to be worst.

The key contribution of this paper is to show that RTDP can be made *robust*, allowing for planning in very large and uncertain domains, retaining worst (or best) case behaviour guarantees.

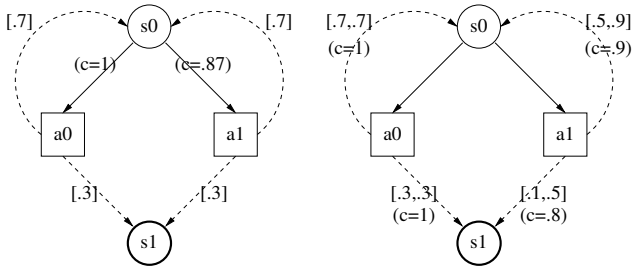
3 Robust RTDP

From now on, we consider **interval-based** uncertain SSPs, where $T(s,a,s')$ is known to be in an interval $[Pr^{\min}(s'|s,a), Pr^{\max}(s'|s,a)]$. Figure 1 is an example. We

Algorithm 2 Robust Value Iteration (for an SSP)

1: Initialize J to 0.
2: **repeat**
3: **for all** s: state **do**
4: **for all** a: action **do**
5: $Q_{\max}(s,a) \leftarrow \max_{m_s^a \in \mathcal{M}_s^a} \sum_{s' \in S} T_{m_s^a}(s,a,s') \left[J(s') + c_{m_s^a}(s,a,s') \right]$
6: **end for**
7: $J(i) \leftarrow \min_{a \in A} Q_{\max}(s,a)$
8: **end for**
9: **until** J converges

discuss the pessimistic approach, the optimistic one leading to similar results.



a) certain SSP b) uncertain SSP

Figure 1: Two views of one SSP, depending on whether model uncertainty is taken into account (costs in parenthesis). In the uncertain SSP, action a_0 will be prefered as it quickly reaches the goal s_1.

For a given state-action pair (s,a), there is a list $R = (s'_1, \cdots, s'_k)$ of reachable states. For each reachable state $T(s,a,s'_i) \in I_i = [p_i^{\min}, p_i^{\max}]$. Thus, possible models are the ones that comply with these interval constraints while ensuring $\sum_i T(s,a,s'_i) = 1$. Fig. 2 illustrates this with three reachable states.

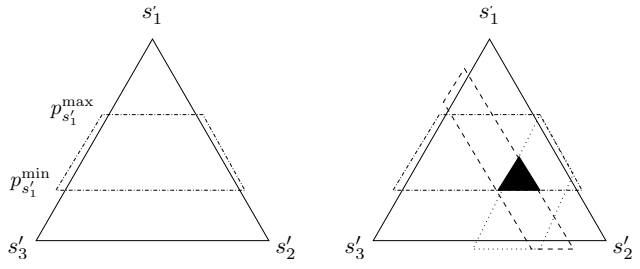


Figure 2: A triangle is a probability simplex representing all possible probability distributions with three different outcomes ($Pr(s'_i) = 1$ at the s'_i vertex). On the left triangle is the trapezium showing the interval constraint for s'_1. The right triangle shows possible models at the intersection of the three interval constraints.

Worst Local Models — The maximisation step to compute $Q(s,a)$ in Alg. 2 is done by giving the highest probability to the worst outcome. This requires firstly sorting reachable states in decreasing order of the values: $c(s,a,s'_1) + J(s'_1) \geq c(s,a,s'_2) + J(s'_2) \geq \cdots c(s,a,s'_k) + J(s'_k)$. Then, the worst distribution is the one giving the highest probability to the first state s'_1, then to s'_2, and so on up to s'_k. As pointed out in [Givan *et al.*, 2000], this is equivalent to finding the highest index $r \in [1..k]$ such that

$$\sum_{i=1}^{r-1} p_i^{\max} + \sum_{i=r}^{k} p_i^{\min} \leq 1.$$

The resulting transition probabilities are

$$Pr(s'_i) = \begin{cases} p_i^{\max} \text{ if } i < r \\ p_i^{\min} \text{ if } i > r \end{cases} \quad (2)$$

$$Pr(s'_r) = 1 - \sum_{i=1, i \neq r}^{k} Pr(s'_i). \quad (3)$$

Using the pre-computed bound $B_{\min} = \sum_{i=1}^{k} p_i^{\min}$, Alg. 3 gives a complete implementation. The *insertion sort* algorithm[4] is chosen as the list will usually be ordered from previous J updates.

Algorithm 3 Worst Model for State-Action Pair (s,a)

WORSTMODEL(s: state, a: action)
$R = (s'_1, \cdots, s'_k) =$ REACHABLESTATES(s,a)
SORT(R)
$i = 1$, $bound = B_{\min}$
while $(bound - p_i^{\min} + p_i^{\max} < 1)$ **do**
$bound \leftarrow bound - p_i^{\min} + p_i^{\max}$
$Pr(s'_i) \leftarrow p_i^{\max}$
$i \leftarrow i + 1$
end while
$r = i$
$Pr(s'_r) \leftarrow 1 - (bound - p_r^{\min})$
for all $i \in \{r+1, \ldots, k\}$ **do**
$Pr(s'_i) \leftarrow p_i^{\min}$
end for
return $(R, Pr(\cdot))$

To summarise, Robust VI on an interval-based SSP consists of applying normal Value Iteration with the transition probabilities updated through Alg. 3.

We need only a single worst model to compute the corresponding Q-value. Yet, because several reachable states s'_i may have the same value $c(s,a,s'_i) + J(s'_i)$ as s'_r (we call this set of states S'_r), there may be an infinite number of equivalent worst local models. Any model differing only in how the probability mass is distributed among the equally bad states of S'_r is also a worst local model.

Worst Global Models — Contrary to VI, RTDP does not necessarily visit the complete state-space. This is why [Barto *et al.*, 1995] introduces the notion of *relevant* states, which we extend to the uncertain case: a state s is *relevant* for $\mathcal{M}$ if

[4] `http://en.wikipedia.org/wiki/Insertion_sort`

there exists a start state s_0, a model $m \in \mathcal{M}$ and an optimal policy π under that model such that s can be reached from state s_0 when the controller uses π under m.

This notion is important because two equally worst local models on a given state-action pair may forbid different states, so that for two models m_1 and m_2, a state may be relevant in m_1 but not in m_2. Yet, RTDP should not find an optimal policy just for the relevant states of only one worst global model. Neither does the policy have to apply to all possible states. It should apply to all *reachable* states under *any* model (for optimal policies) i.e., for relevant states. But covering all relevant states in the worst model used for updating Q-values does not necessarily cover all relevant states in $\mathcal{M}$: it depends on the model used to choose the next state, i.e., to simulate the system's dynamics.

To avoid missing relevant states, each local model used for the simulation should ensure that all reachable states can be visited. As can be seen in Fig. 2, the set of possible local models for a state-action pair is an n-dimensional convex polytope. Any model *inside* this polytope, excluding the boundary, is therefore adequate because, for all s'_i, it ensures that $P(s'_i|s,a) > 0$.

So there exists a global model m_d that can be used to effectively simulate the system's dynamics without missing any potentially reachable state.

3.1 Robust (Trial-Based) RTDP

Robust RTDP differs from the original RTDP in that:

- Each time the algorithm is updating a state's evaluation, an opponent is looking for a worst local model which serves to compute the Q-values.
- For exploration purposes, the algorithm follows dynamics of the system that consider all possible transitions (using model m_d).
- "Relevant" states are now the states reachable by following any optimal policy under any possible model.

From this, we adapt to our context convergence Theorem 3 from [Barto *et al.*, 1995] and the corresponding proof, discussing mainly its changes.

Proposition 1. *In* **uncertain** *undiscounted stochastic shortest path problems,* **robust** *Trial-Based RTDP with the initial state of each trial restricted to a set of start states, converges (with probability one) to J^* on the set of relevant states, and the controller's policy converges to an optimal policy (possibly nonstationary) on the set of relevant states, under the same conditions as Theorem 3 in [Barto* et al.*, 1995].*

Proof. The proof outline is:

1. As shown in Sec. 2.3, robustness is achieved by considering a Stochastic Shortest Path *Game* (SSPG). In particular, this one is a *sequential* SSPG, for which Bertsekas and Tsitsiklis [1996] establish value iteration and Q-learning algorithms. Provided all states are visited infinitely often, *asynchronous* value iteration can also be used to solve sequential SSPGs (AVI-SSPG).
2. We adapt the proof of Barto *et al.* [1995] to transform AVI-SSPG to an RTDP algorithm (RTDP-SSPG). The transformation ensures: 1- the additional $\max_{m\in\mathcal{M}}$ in the update formula (Alg. 2, Line 5) does not break the requirement that J_t is increasing and non-overestimating; and 2- *relevant* states are visited infinitely often, guaranteeing a complete and optimal policy.
3. RTDP-SSPG and robust RTDP differ because RTDP-SSPG assumes an optimal opponent. In the robust case we wish to plan for opponents that may choose the wrong model. Thus robust RTDP is RTDP-SSPG following model m_d — for simulation only — that allows states to be visited that RTDP-SSPG optimal opponents would forbid. Under m_d (described in Sec. 3), no state reachable under any model in $\mathcal{M}$ is excluded from the search. Thus we have redefined an expanded set of relevant states for added robustness. All relevant states are still visited infinitely often.

□

Our use of the model m_d for simulation ensures that the policy will cover all states *any* opponent could lead us to, but assumes the worst model will apply afterwards.

4 Experiments

Labelled RTDP [Bonet and Geffner, 2003] is a modified version of RTDP which can be made robust in a similar way. The experiments conducted illustrate the behavior of *robust LRTDP*. To that end, it is compared to Bagnell *et al.*'s *Robust Value Iteration*, as well as *LRTDP*. In all cases, the convergence criteria is $\epsilon = 10^{-3}$ for LRTDP, and for VI we stop when the maximum change in a state long-term cost over one iteration is less than 10^{-3}.

4.1 Heart

In this first experiment, we compare a non-robust optimal policy with a robust one on the small example of Fig. 1-b. Table 1 shows the theoretical expected long-term costs of each policy on the normal (most probable) model, as well as the pessimistic and optimistic ones. The robust policy is largely better in the pessimistic case.

Table 1: Theoretical evaluation of robust and non-robust policies on various models, which match empirical evaluation.

	Normal	Pessimistic	Optimistic
Non-robust	2.90	8.90	1.70
Robust	3.33	3.33	3.33

4.2 Mountain-Car

We use here the mountain-car problem as defined in [Sutton and Barto, 1998]: starting from the bottom of a valley, a car has to get enough momentum to reach the top of a mountain (see Fig. 3). The same dynamics as described in the mountain car software[5] have been employed. The objective is to minimize the number of time steps to reach goal.

[5] `http://www.cs.ualberta.ca/~sutton/...MountainCar/MountainCar.html`

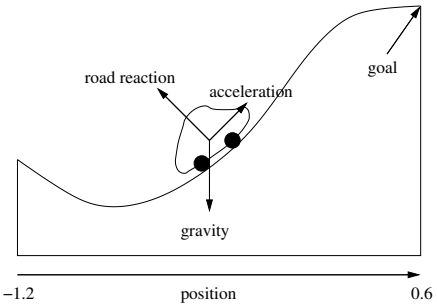


Figure 3: The mountain-car problem.

The continuous state-space is discretized (32×32 grid) and the corresponding uncertain model of transitions is obtained by sampling 1000 transitions from each state-action pair (s, a). For each transition, we computed intervals in which the true model lies with 95% confidence.

Results — Preliminary remark: simulating a path generally shows a car oscillating several times before leaving the valley. This has two main explanations: 1- the speed gathering is just sufficient to reach the summit, but not over it; and 2- the discretized model is not accurate enough and applying the policy obtained on the true mathematical model (instead of the discretized one) should be much better.

Fig. 4 shows the long-term cost function obtained by using *value iteration*, LRTDP, and their robust counterparts on the mountain-car problem. The x and y axes are the car's position and speed. The z axis is the expected cost to the goal. On the surface is an example path from the start state to a goal state: it follows the greedy policy under the average model.

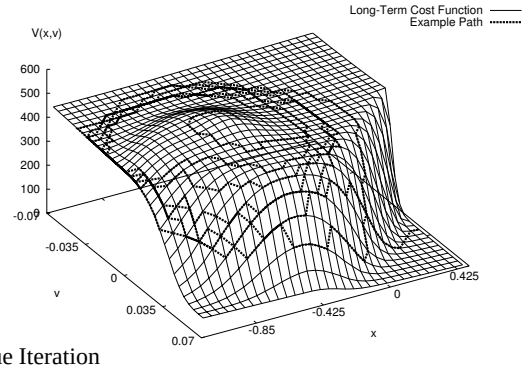
The general shape of the surface obtained is always the same, with some unexplored parts of the state-space in LRTDP and Robust LRTDP (as expected). The vertical scales are much larger in robust cases. This reflects the fact that reaching the goal is much more time-consuming under a pessimistic model. Because J can here be interpreted as the average time to the goal, these graphs show how a small uncertainty can lead to longer policies. Here the times are multiplied by more than 2.5.

While executing the four different algorithms, an evaluation of the current greedy policy was made every $10 * nStates = 10\,240$ Q-value updates. The result appears in Fig. 5, the y axis being the expected cost to the goal from the start state. On both sub-figures, LRTDP-based algorithms obtain good policies quickly, but have slow convergence times of VI=2.83×10^6 updates, LRTDP=6.76×10^6, rVI=8.31×10^6, rLRTDP=11.06×10^6.

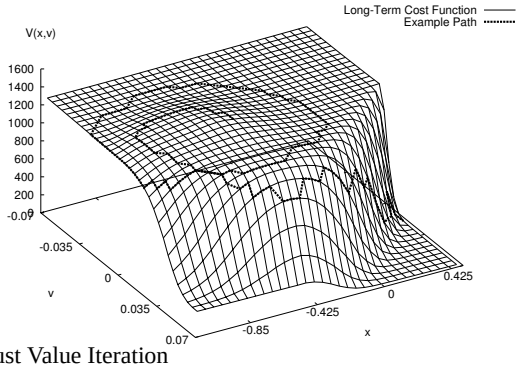
Other experiments [Buffet and Aberdeen, 2004] also confirm these results, with one example showing that LRTDP can have a faster convergence than VI, and another one illustrating this approach on a temporal planning problem.

5 Discussion and Conclusion

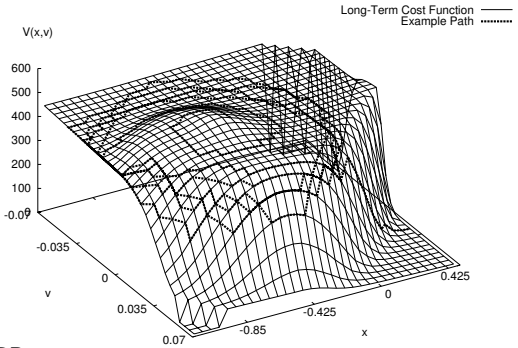
A simple extension to this work, suggested by Hosaka *et al.* [2001], is to find the set of worst models as we do in this



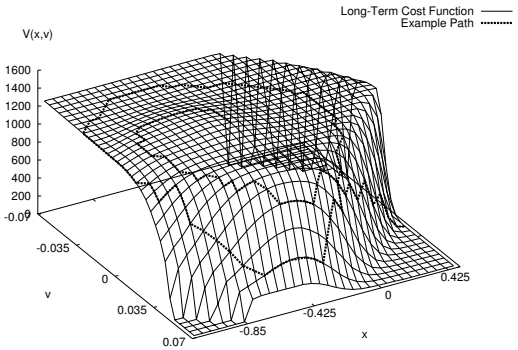
a) Value Iteration



b) Robust Value Iteration

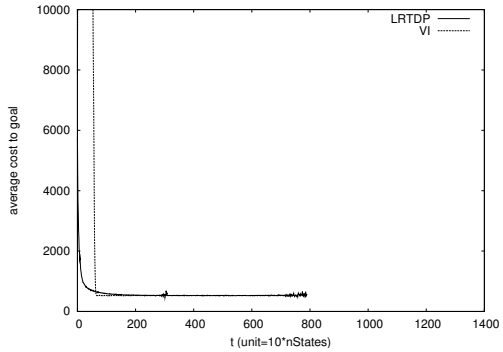


c) LRTDP

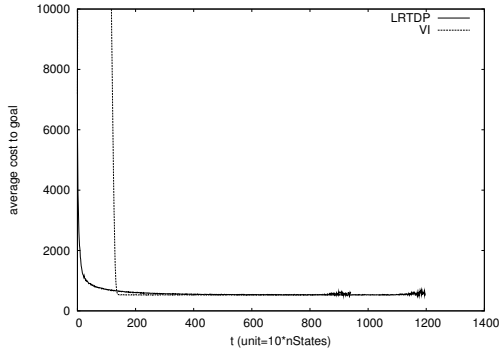


d) Robust LRTDP

Figure 4: Long-term cost functions for the mountain-car problem. In each case, an example path is generated based on the most likely model.



a) Non-Robust Algorithms



b) Robust Algorithms

Figure 5: Average cost to goal for the mountain-car problem, measured every $10 * nStates$ updates of Q-values.

paper, but then to choose from these the model that has the lowest cost under an *optimistic model*. This improvement is presented in [Buffet and Aberdeen, 2004].

The approach to robustness adopted in this paper considers the knowledge of a set of possible models. An open question is whether it is possible to use more information from our uncertain model, by taking the probability distribution over possible models into account.

Model uncertainty has similarly been considered in model learning while planning [Strehl and Littman, 2004]. The algorithm proposed is optimistic, but does not seem to adapt well to our framework as an evolving model can break the "no-overestimation" assumption: $\forall s \in S,\ t \geq 0,\ J_t(s) \leq J^*(s)$. It is still important to notice that *robust* RTDP would not suffer on-line, as the real dynamics can be employed to pick a next state (the worst model appearing only in the long-term cost update formula).

Finally, a crucial assumption in RTDP is that a goal state must be reachable from any state. We present an algorithm tackling this problem in [Buffet, 2004].

Conclusion — Recent works show that model uncertainty is a major issue in decision-theoretic planning. We have proposed a modification of the RTDP algorithm enabling it to compute robust policies efficiently in large and uncertain domains. Model uncertainty is represented through confidence intervals on transition probabilities. The convergence of the resulting algorithm is sketched. We demonstrate robust LRTDP on a domain where statistics are used to estimate appropriate intervals.

Acknowledgments

National ICT Australia is funded by the Australian Government. This work was also supported by the Australian Defence Science and Technology Organisation.

References

[Bagnell *et al.*, 2001] J.A. Bagnell, A. Y. Ng, and J. Schneider. Solving uncertain markov decision problems. Technical Report CMU-RI-TR-01-25, Robotics Institute, Carnegie Mellon U., 2001.

[Barto *et al.*, 1995] A.G. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72, 1995.

[Bertsekas and Tsitsiklis, 1996] D.P. Bertsekas and J.N. Tsitsiklis. *Neurodynamic Programming*. Athena Scientific, 1996.

[Bonet and Geffner, 2003] B. Bonet and H. Geffner. Labeled rtdp: Improving the convergence of real time dynamic programming. In *Proc. of the 13th Int. Conf. on Automated Planning and Scheduling (ICAPS'03)*, 2003.

[Buffet and Aberdeen, 2004] O. Buffet and D. Aberdeen. Planning with robust (l)rtdp. Technical report, National ICT Australia, 2004.

[Buffet, 2004] O. Buffet. Robust (l)rtdp: Reachability analysis. Technical report, National ICT Australia, 2004.

[Givan *et al.*, 2000] R. Givan, S. Leach, and T. Dean. Bounded parameter markov decision processes. *Artificial Intelligence*, 122(1-2):71–109, 2000.

[Hosaka *et al.*, 2001] M. Hosaka, M. Horiguchi, and M. Kurano. Controlled markov set-chains under average criteria. *Applied Mathematics and Computation*, 120(1-3):195–209, 2001.

[Munos, 2001] R. Munos. Efficient resources allocation for markov decision processes. In *Advances in Neural Information Processing Systems 13 (NIPS'01)*, 2001.

[Nilim and Ghaoui, 2004] A. Nilim and L. El Ghaoui. Robustness in markov decision problems with uncertain transition matrices. In *Advances in Neural Information Processing Systems 16 (NIPS'03)*, 2004.

[Patek and Bertsekas, 1999] S. D. Patek and D. P. Bertsekas. Stochastic shortest path games. *SIAM J. on Control and Optimization*, 36:804–824, 1999.

[Strehl and Littman, 2004] A. L. Strehl and M. L. Littman. An empirical evaluation of interval estimation for markov decision processes. In *Proc. of the 16th Int. Conf. on Tools with Artificial Intelligence (ICTAI'04)*, 2004.

[Sutton and Barto, 1998] R. Sutton and G. Barto. *Reinforcement Learning: an introduction*. Bradford Book, MIT Press, Cambridge, MA, 1998.

Abstraction-based Action Ordering in Planning

Maria Fox and **Derek Long** and **Julie Porteous**
Department of Computer and Information Sciences
University of Strathclyde, Glasgow, UK

Abstract

Many planning problems contain collections of symmetric objects, actions and structures which render them difficult to solve efficiently. It has been shown that the detection and exploitation of symmetric structure in planning problems can dramatically reduce the size of the search space and the time taken to find a solution. We present the idea of using an abstraction of the problem domain to reveal symmetric structure and guide the navigation of the search space. We show that this is effective even in domains in which there is little accessible symmetric structure available for pruning. Proactive exploitation represents a flexible and powerful alternative to the symmetry-breaking strategies exploited in earlier work in planning and CSPs. The notion of *almost symmetry* is defined and results are presented showing that proactive exploitation of almost symmetry can improve the performance of a heuristic forward search planner.

1 Introduction

Symmetries frequently occur in search problems such as planning problems [Joslin and Roy, 1997; Fox and Long, 1999; Rintanen, 2003], CSPs [Roy and Pachet, 1998; Gent and Smith, 2000; Roney-Dougal *et al.*, 2004] and model checking [Ip and Dill, 1996; Audemard and Benhamou, 2002]. Many techniques have been developed for symmetry-breaking to improve search performance, with emphasis on using symmetries to prune search spaces rather than to suggest branches to pursue. In this paper we propose the use of symmetries to direct the forward search of an FF-style planner in a positive way. We show that the proactive use of symmetries can lead to significant performance improvements across a variety of planning domains.

The automatic identification of all of the symmetries of a problem is NP-hard. Because of the complexity of the identification problem most researchers working in symmetry have broken symmetries by hand by expressing specific symmetry-breaking constraints in the modelling of the problem [Gent and Smith, 2000; Ip and Dill, 1996; Roney-Dougal *et al.*, 2004]. Planning domain models are structured in a way that often makes some of the underlying symmetric structure accessible and a subset of the available symmetries can be efficiently identified using automatic techniques [Fox and Long, 1999; 2002]. However, it has been observed that the most directly accessible symmetries are often not the ones that would be most useful to exploit.

It happens that structures within a planning problem are often *almost*, but not quite, symmetric, and that treating them as symmetric would increase the efficiency of the search. *Almost symmetries* are revealed by the application of an appropriate abstraction to the domain. As we will see, the term *almost* refers to the fact that the associated symmetry belongs to the abstracted domain and might be unsound with respect to the original domain. Such symmetries can only be exploited in a positive way (to suggest how best to develop the plan) because using them for pruning would compromise completeness. In this paper we consider how the application of a certain simple abstraction, which we call the *property-based abstraction*, can reveal a form of almost symmetry which we then show can be effectively exploited to solve the original problem. We begin with a simple motivating example, then we present the definitions and examples that support the following discussion. We then describe the extension of FF that enables the proactive exploitation of symmetric problem structure and discuss a collection of results obtained from STRIPS domains used in the 2002 planning competition.

2 Identifying Almost Symmetry

Consider the problem of transporting a number of crates from one location to another (as in the Depots domain [Fox and Long, 2003]). The crates are stacked in several different piles in the initial state and must be moved into new configurations. Any solution plan will involve unstacking the crates, loading them onto transport and then delivering and unloading them. At an abstract level at which the precise locations of the crates are ignored, the crates can be treated as symmetric because the same sub-plans are required to get all of the crates into their goal configurations.

The fact that the cargoes start stacked in different piles means that they cannot be automatically identified as functionally equivalent [Fox and Long, 1999]). Furthermore, since their initial configurations are different they cannot be identified as symmetric in the sense exploited in [Joslin and Roy, 1997]). Nevertheless, there is a high degree of symmetry

in the structure of the problem even though it is not immediately accessible to automatic analysis. The crates are *almost symmetric* because they can be made symmetric by abstracting out the specific domain objects to which they are related (eg: crate1 is on *something*, but it doesn't matter what), without losing sight of the properties (being *on* something as opposed to having something else *on top*) that distinguish them. When a collection of objects (in this case crates) are almost symmetrical we reason that we may well need to perform the same operations on these objects during the course of any eventual solution plan.

We say that two actions are almost symmetric if they are two different instances of the same operator in which objects in the corresponding argument positions are symmetric in the abstracted domain. The stategy we describe below favours the selection of actions that are almost symmetric to actions that have already been applied in the plan.

3 Symmetries in Planning Domains

In this section we develop a formal framework for defining symmetries in planning problems. We begin with familiar definitions of a planning domain, a planning problem and a plan. Symmetries have been studied for decades by mathematicians as *groups*. We therefore adopt a group-theoretic characterisation, defining each symmetry as a permutation that preserves the equivalence of the domain configurations to which it is applied. The elements of our groups act on the objects in a domain.

Definition 3.1 *A* planning problem*, P, is a tuple (O,I,G,C), where O is a set of operator schemas, I is a set of initial state facts, G is a set of goal propositions and C is a set containing all of the constants appearing in I∪G.*

Definition 3.2 *A* plan*, p, for a planning problem, (O,I,G,C), is a sequence of operator schemas from O each instantiated with constants from C such that each instantiated operator schema can be applied from the preceding state, starting at I, and the final state satisfies G.*

The first symmetry we consider is the form explored by Joslin and Roy [Joslin and Roy, 1997] and subsequently considered by Rintanen [Rintanen, 2003].

Definition 3.3 *A* configuration symmetry *of a planning problem P = (O,I,G,C) is a group, S, acting on the constants in C, such that under the action of the elements of S, the initial and goal states are invariant and every plan for P remains a valid plan for P.*

This definition requires that a plan should remain valid under the action of elements of the symmetry S, but it does not require that the plan should be invariant. For example, consider the simple blocks problem illustrated in figure 1. The group $\{id, (CE)(DF)\}$[1] is a configuration symmetry. However, although valid plans will remain valid, plans will not be invariant under the transpositions of C with E and D with F, since this will rearrange the steps of a plan.

[1]The identity element is denoted by *id*. The elements are expressed in the standard cycle notation for groups.

A special subclass of configuration symmetries contains those symmetries that have proved to be amongst the most fruitful in existing exploitations of symmetries [Fox and Long, 1999; 2002]. This is the class that is generated by all pairwise transpositions of a set of symmetric objects:

Definition 3.4 *A configuration symmetry, S, of P = (O,I,G,C), is a* functional symmetry *if it is a permutation group, S_n, acting on a subset of n of the constants in C.*

3.1 The Property-based Abstraction

All symmetries can be seen as a consequence of some form of abstraction. The abstraction renders certain groups of actions, objects or structures equivalent by not specifying the details of the relationships that differentiate them. In some problems, applying an abstraction to remove the distinctions between objects can lead to the discovery of new potential symmetries that are not present in the base domain. It is this observation that leads us to make the following definition:

Definition 3.5 *An* abstraction *of a planning problem, P, is a mapping, f, from P onto a new planning problem, Q.*

This definition is deliberately permissive, since we do not wish to exclude any possible abstractions. In general we will be considering compositional mappings in which the components of Q are constructed by applying an abstraction independently to the different components of P. The following definition formalises the property-based abstraction described earlier. Properties were first defined in [Fox and Long, 1998]. A property of an argument to a predicate is the pair containing the predicate name and the argument position index (this is denoted below by subscripting the predicate name with the argument position).

Definition 3.6 *The following mapping is the* property-based *abstraction for planning problem P.*

For each proposition in a planning problem, P, formed from predicate p applied to k arguments, $x_1, \ldots, x_k$, $f(p(x_1, \ldots, x_k)) = \{p_1(x_1), \ldots, p_k(x_k)\}$. The k unary target predicates, $p_1, \ldots, p_k$ are called properties*.*

The property-based abstraction simplifies the structure of a problem by removing the linkage between pairs of objects and considering the problem as the union of separate projections of the original problem for each constant. This has the effect of abstracting out which objects play which roles with respect to other objects.

If we apply an abstraction to a planning problem and then identify a symmetry in the abstracted problem, this symmetry can have a useful relationship to the original problem:

Definition 3.7 *Given a planning problem, P, and an abstraction, f, mapping P into the planning problem Q, a symmetry of Q is called an* almost symmetry *for P with respect to the abstraction f.*

Abstractions can throw away so much of the structure of a problem that the abstracted domain has no useful connection with the original domain. In practice we are interested in abstractions that preserve significant structure from the original problem. One way in which structure might be preserved is in the behaviour of plans under an abstraction:

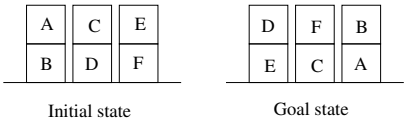


Figure 1: A simple example blocks problem

Definition 3.8 *An abstraction, f, of planning problem P is* plan preserving *if, when p is a plan for P, $f(p)$ is a plan for $f(P)$.*

The property-based abstraction of a problem is not necessarily plan preserving because each property is available only once in the abstracted domain and once it is deleted it cannot be replaced. Thus, in cases where a set of propositions collapses to a single property there may be no solution to the abstracted problem.

Definition 3.9 *An almost symmetry of planning problem P with respect to the abstraction f is a* strong (weak) almost symmetry *if f is (is not) plan preserving.*

In the case where f is the identity, any symmetry on P with respect to f is, trivially, a strong almost symmetry.

The property-based abstraction is just one abstraction function that reveals exploitable almost symmetries. In general, the abstraction process can remove important constraints from the original problem, so can only act as a guide to the solution of the original problem: indeed, even for use as a guide, an abstraction must be chosen with care. For this reason, almost symmetries must be handled cautiously: pruning a search space using an almost symmetry is very likely to compromise completeness.

Our starting point for identifying almost symmetries between objects is the method used by Joslin and Roy [Joslin and Roy, 1997]. For a given planning problem we build a graph representing the object relationships in the initial and goal states of the problem and then use NAUTY [McKay, 1990], the graph automorphism discovery tool, to identify automorphisms in the graph. The key difference in our approach is the way in which the *graph* is constructed. Given a planning problem, our approach identifies a subset of the almost symmetries of the domain. We construct a coloured graph that represents an abstraction of the problem under the property-based abstraction. This construction is performed by the *makeGraph* algorithm shown in figure 2. NAUTY is then used to find the symmetries in the abstracted problem. The colouring of the nodes distinguishes between the properties that arise in the property-based abstraction described in definition 3.6. Thus, NAUTY finds the almost symmetries of the problem with respect to the property-based abstraction.

The following theorem guarantees that *makeGraph* generates an almost functional symmetry with respect to the property-based abstraction. The proof is omitted for brevity.

Theorem 3.10 *The automorphisms of the graph makeGraph(P), for planning problem P, restricted to the constants in P, form an* almost functional symmetry *with respect to the property-based abstraction of P.*

```
1:  makeGraph(P)
2:  input: planning problem P = (O, I, G, C)
3:  output: coloured graph N
4:  initialise an empty graph N.
5:  for all c ∈ C do
6:      create vertex v_c with colour equal to its type
7:      add v_c to N
8:      initialise an empty set of propositions, Q_I
9:      for all propositions p ∈ I do
10:         if p contains c then
11:             add p to Q_I
12:     let props = the bag of properties of c in Q_I
13:     create a vertex v_{Q_I} with colour equal to (|Q_I|,props)
14:     {the names of the properties in props identify the arities of
        the predicates and the types of their arguments}
15:     add v_{Q_I} to N
16:     create an edge e between v_c and v_{Q_I}
17:     add e to N
18:     initialise an empty set of propositions, Q_G
19:     for all propositions p ∈ G do
20:         if p contains c then
21:             add p to Q_G
22:     let props = the bag of properties of c in Q_G
23:     create a vertex v_{Q_G} with colour equal to (|Q_G|,props)
24:     add v_{Q_G} to N
25:     create an edge e between v_c and v_{Q_G}
26:     add e to N
27: return N
```

Figure 2: The construction of the coloured graph.

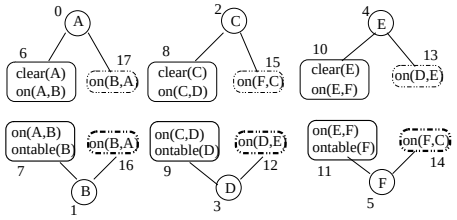


Figure 3: The graph constructed by makeGraph for the blocks example. Nodes with the same shape and outline have the same colour and are numbered 0-17.

Having identified the almost symmetries that exist at the object level in the domain we can construct the almost symmetric relation on actions, as described above. The application of a property-based abstraction, as described in definition 3.6, allows the identification of the symmetry in figure 1 between the blocks on the tops of the stacks and between the blocks on the bottoms of the stacks. In such an interpretation it can be seen that the blocks A, C and E, which are *on* other blocks and therefore at the tops of the three stacks, will move to being at the bases of the three new stacks in the goal. Similarly, B, D and F will move from being beneath other blocks to being at the tops of the new stacks. By abstraction, the fact that these groups of blocks start in different positions has been made irrelevant and the symmetry in their behaviour becomes apparent.

The coloured graph that we construct for this problem is shown in figure 3. The round nodes are the 6 blocks in the problem. These nodes have the same colour because the

blocks are all of the same type (so can be considered to share the same unary type predicate). Each of these nodes is connected to the set of propositions that mention the corresponding block in the initial state, and the set of propositions that mention the block in the goal state. It can be seen that when the properties obtained from the propositions in these sets are different the corresponding nodes have different colours. For example, nodes 16 and 17 have different colours because, although they contain the same proposition, the colour of node 16 contains the property on with respect to block A (the property on_2) while the colour of node 17 contains the property on with respect to block B (the property on_1).

The output from NAUTY is the generators for the graph automorphism group. When restricted to domain objects, the output for this problem is: vertices 0, 2, 4 or $\{A, C, E\}$ and vertices 1, 3, 5 or $\{B, D, F\}$

The almost symmetry that is obtained by first applying the property-based abstraction to this blocks world problem, and then applying NAUTY to the resulting graph, belongs to the abstracted problem in which the detailed relationships between particular blocks have been removed. It can be seen that the blocks A, C and E are interchangeable in the property-based abstraction, as are the blocks B, D and F. Interestingly, in this particular simple example these collections of blocks are interchangeable in the plan as well, so that the property-based abstraction is plan preserving in this case and the almost symmetry is strong (as defined in definition 3.9).

4 Proactive Symmetry Exploitation

In most existing work exploiting symmetry in search problems, the exploitation has been *negative*, in the sense that it is used to prune the search space in order to avoid searching symmetrically equivalent parts of the search space. This is useful when one part of the space has been searched fruitlessly, since the symmetry implies that the symmetrically equivalent parts of the space will also prove fruitless. Various approaches can be used to prune the search space, including the introduction of symmetry breaking constraints into the problem encoding itself (this technique is used in the planning context in [Joslin and Roy, 1997; Rintanen, 2003]), by forcing an ordering between otherwise symmetric choices for example, and monitoring choices in the search machinery itself, pruning equivalent choices as they arise (the approach followed in [Fox and Long, 1999; 2002; Long and Fox, 2003]). However, in the context of certain heuristic search strategies the utility of these approaches is significantly reduced. In particular, the forward search approach of FF effectively breaks symmetries by not backtracking over action choices. Although some benefit might be obtained by avoiding the consideration of symmetric action choices during a breadth-first search of a plateau, pruning symmetric branches appears likely to be less useful in heuristic search planners than in systematic search planners.

In order to exploit symmetry in heuristic search planners, we propose an alternative strategy to the pruning approach: in cases where action choices are available that are symmetric with choices that have already been made and adopted, we propose to encourage those choices.

```
1:  actionSelect(h,G,As,H)
2:  input: The heuristic value of the current state h, a symmetry G,
    a set of proposed actions As, a plan head H
3:  output: An extended plan head H'
4:  initialise a vector vs of counts, one for each action in As.
5:  for all actions a in As do
6:     for all actions b in H do
7:        if a uses object c and b uses object d and (b d) ∈ G then
8:           increment vs[a]
9:  sort As according to vs
10: for all a in As do
11:    if heuristic value of a > h then
12:       H' = H+a
13:       return H'
```

Figure 4: A modified version of FF favouring symmetric action choices.

5 Exploitation of symmetry information

In the work described in this paper we have explored a simple strategy where the symmetry information was used as a *heuristic guide* for selection between proposed action choices. During plan generation FF performs a forward state space search and at each stage proposes actions to apply at the current state. We amend the action selection strategy of FF to favour actions that are almost symmetric to actions selected earlier in the plan. This is a positive exploitation of symmetry because we use the symmetric structure of the problem to propose rather than to prune action choices.

The heuristic prefers actions that use objects that are symmetrical to objects that have appeared in *the same* actions earlier in the plan. This exploits the observation that where symmetric objects each require treatment in a plan they are likely to require symmetric treatment.

This heuristic is straightforward to implement in FF. At each stage during plan generation we have a current plan which consists of the actions applied so far to get from the initial state to the current state. At this stage a set of possible next actions are proposed by FF. These are the helpful actions, all of which occur in the first layer of the relaxed plan from the current state and are immediately applicable in the current state. FF normally chooses between these by applying its relaxed distance measure to the states that they produce and choosing the first helpful action that produces the state closest to the goal. We modify the strategy that FF uses for selecting between these by ordering the helpful actions so that those that use objects that are symmetrical to any that have appeared in actions in the plan so far are visited first in the selection strategy. The consequence of this is that only actions that improve the heuristic distance will be selected by the modified strategy. We do not modify the heuristic estimate itself. To order the actions we record, for each action, a count of the number of symmetric objects it uses and sort the actions into descending order of this count. Any ties are broken using FF's standard action selection strategy. The necessary modifications to the action selection step are shown in figure 4.

To illustrate the use of the symmetry information in FF version 2.3 (ff-v2.3), consider a simple Depots example (pfile3 in the IPC3 archive). The following sets of objects are found

Action	Symmetric args	Score
(a)	(distributor1,depot0)	1
(b)	(distributor1,depot0), (depot0,distributor1)	2
(c)	(hoist2,hoist0)	1
(d)		0
(e)	(hoist1,hoist0), (crate4,crate5)	2

Table 1: Scoring helpful actions for symmetric arguments.

to be symmetric according to their properties in the initial and goal states of the problem: {*pallet0, pallet1, pallet2*}, {*crate4,crate5*}, {*crate0, crate2*}, {*truck1, truck2*}, {*depot0, distributor1*} and {*hoist0, hoist1, hoist2*} At some intermediate point during plan generation the plan contains the following actions:

drive(truck1,distributor0,depot0)
drive(truck1,depot0,distributor1)
drive(truck0,depot0,distributor1)
lift(hoist0,crate5,crate2,distributor1)
load(hoist1,crate5,truck1,distributor1)

The set of helpful actions proposed by FF at this point in the plan contains:

(a) *drive(truck1,distributor1, distributor0)*
(b) *drive(truck1,distributor1,depot0)*
(c) *lift(hoist2,crate2,pallet2,distributor1)*
(d) *lift(hoist0,crate1,pallet0,depot0)*
(e) *lift(hoist1,crate4,crate3,distributor0)*

The scores associated with these actions are obtained by counting the number of corresponding pairs of symmetric arguments between each action and an occurence of the *same action* in the plan so far. In scoring, we do not increment the score in the case where the argument in the helpful action is identical to the corresponding argument in the earlier action as we want to encourage acting on *other* objects in the same symmetric set.

We now consider the helpful actions in turn, scoring them, as shown in figure 1. The heuristic must choose between the two actions weighted 2. In this case the lift action is selected (it is likely that the competing drive action was not highly valued by the distance heuristic because it undoes the effect of an immediately preceding step).

It can be seen that our strategy is forcing the planner to commit earlier to actions that have already proved successful for symmetrical objects (we needed to lift up $crate5$ and, since $crate4$ and $crate5$ are symmetric it may be a good idea to do the same thing with $crate4$).

6 Experimental Results

In this section we report results of experiments that compare the performance of the planner ff-v2.3 [Hoffmann and Nebel, 2001; Hoffmann, 2002] with a version of the same planner that uses symmetry information as a positive heuristic in deciding which action to apply next. We refer to this planner as *ff-v2.3+symm*. We consider six domains: two of these are artificially constructed to contain a high degree of directly exploitable symmetry, whilst the remaining four are more natural domains, taken from the IPC3 competition bench marks. In the latter group (Depots, Driverlog, Rovers and Freecell), there is very little directly exploitable symmetry — existing

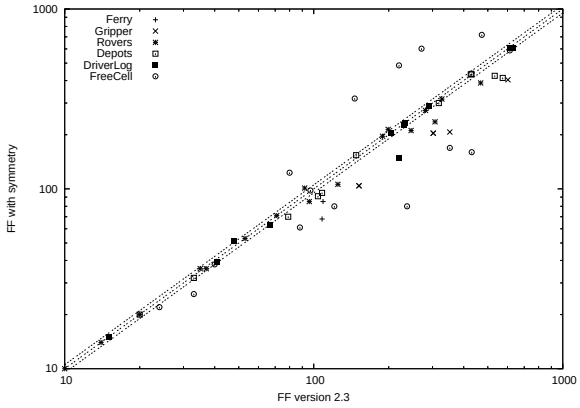


Figure 5: Comparison of states visited.

techniques discover at most 4 symmetric objects in any of these problems. Our results show that we obtain significant advantages in both the artificial and the natural domains. The Gripper results we present are for randomly generated problems in which the number of balls and the number of grippers both vary. The Ferry problems are randomly generated with varying numbers of ferries and cars. Increasing the numbers of grippers and ferries in these problems reduces the number of times that a *move* or a *sail* is the only helpful action and therefore increases the relative density of choicepoints at which symmetric actions are available. We begin by presenting graphs of the results comparing numbers of states explored and time taken (which includes the time required to carry out the symmetry analysis itself).

Figure 5 shows the comparison of the performances of *ff-v2.3* and *ff-v2.3+symm* in terms of the number of states visited in the search. The plot is log-scaled and the middle of the three lines represents equal performance. The other lines represent a 10% performance difference either side of equal. Again, points below the line represent a performance advantage for *ff-v2.3+symm*. It can be seen that five of the Freecell problems and one Rovers problem (six problems in total) were solved at least 10% more efficiently by *ff-v2.3*. By contrast, twenty eight problems were solved at least 10% more efficiently by *ff-v2.3+symm*.

Figure 6 shows the comparative time performance on problems taken from the six domains. In this plot, and in the statistical reported below, we exclude instances solved in under one second. There are two reasons for this: firstly, noise effects in the measurements are more severe in values this small and, secondly, the implementation of our symmetry analysis is not optimised for speed, making its overhead a relatively distorted penalty for these small problems. The line represents equal performance and points below the line represent better performance by *ff-v2.3+symm*. It can be seen that *ff-v2.3* outperforms *ff-v2.3+symm* in only four cases.

To confirm statistical significance of our results, we have used the Wilcoxon-Mann-Whitney matched pairs signed ranks test. This test is more appropriate than a matched pairs t-test, since the distribution of the differences in performance

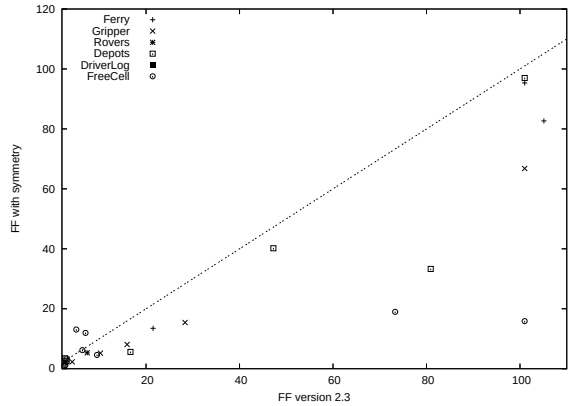


Figure 6: Comparison of time performance.

		Time			States visited		
Test	orig	spl	Z	p	spl	Z	p
6 doms	82	26	3.43	0.0006	65	4.14	$<$0.0004
4 doms	69	17	2.18	0.0292	55	3.12	0.0018
Symm	11	9	2.72	$<$0.05	10	2.80	$<$0.05

Table 2: Results from matched pairs ranked signs tests showing sample size, Z-value, and significance (all less than 5%). Tests are for all domains (6 doms), 4 IPC3 domains (4 doms) and Gripper+Ferry (Symm).

is certainly not normal: where problems are solved with little search the potential advantage is restricted, while it can be larger in harder problems. The matched pairs signed ranks test is more sensitive and more robust in cases where the underlying distribution is not known. We show, in figure 2, that in both sets of domains the performance of *ff-v2.3+symm* is statistically significantly better than that of *ff-v2.3*, both in terms of time taken and in terms of number of states visited. For each of the tests we performed we provide the sample size, the Z-value obtained (using the Wilcoxon-Mann-Whitney test) and the corresponding p-value. Since equal performance cases are discarded in performing the test, we also report the original sample size (including equal performance cases), for comparison. Since the Gripper and Ferry domains contain artificially high degrees of symmetry, we perform the tests both including and excluding these domains (although note that only 10 problems were considered for these domains). We do not report results for individual domains, but it is interesting to note that these are not significant, indicating that the effects of almost symmetry across the whole problem set are not the consequence of strong performance in a subset of the domains, but are distributed across all of them.

7 Conclusions and Further Work

In this paper we have introduced a method for extracting almost symmetries from a planning problem and exploiting them proactively in a forward search planner. We have presented results showing that proactive symmetry exploitation can improve the search performance of a planner and sometimes result in the discovery of higher quality plans. Specifically, using symmetry information to inform the heuristic selection of the next action can reduce the number of states expanded during search, the number of steps added to the plan and also the overall time taken to generate a solution plan.

A useful aspect of the proactive approach is that it can be combined with negative symmetry-breaking techniques. It is feasible to combine the action choice heuristic with a completeness preserving pruning strategy such as has been exploited by Rintanen [Rintanen, 2003] and Fox and Long [Fox and Long, 1999], since these approaches are not in any way mutually exclusive.

References

[Audemard and Benhamou, 2002] G. Audemard and B. Benhamou. Reasoning by symmetry and function ordering in finite model generation. In *Proc. 18th Int. Conf. on Automated Deduction (CADE-18)*, volume 2392 of *LNCS*, 2002.

[Fox and Long, 1998] M. Fox and D. Long. The automatic inference of state invariants in TIM. *JAIR*, 9, 1998.

[Fox and Long, 1999] M. Fox and D. Long. The Detection and Exploitation of Symmetry in Planning. In *Proc. of the 16th Int. Joint Conf. on AI (IJCAI)*, 1999.

[Fox and Long, 2002] M. Fox and D. Long. Extending the exploitation of symmetries in planning. In *Proc. of AIPS'02*, 2002.

[Fox and Long, 2003] M. Fox and D. Long. An extension to PDDL for expressing temporal planning domains. *Journal of AI Research*, 20, 2003.

[Gent and Smith, 2000] I. P. Gent and B. Smith. Symmetry breaking during search in constraint programming. In *Proc. of ECAI*, 2000.

[Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14, 2001.

[Hoffmann, 2002] J. Hoffmann, 2002. The FF-v2.3 planner is available to download from http://www.informatik.uni-freiburg.de/∼hoffmann/ff.html.

[Ip and Dill, 1996] C. Norris Ip and David L. Dill. Better verification through symmetry. *Formal Methods in System Design*, 9, 1996.

[Joslin and Roy, 1997] D. Joslin and A. Roy. Exploiting symmetry in lifted CSPs. In *Proc. of 14th National Conf. on AI (AAAI-97)*, 1997.

[Long and Fox, 2003] D. Long and M. Fox. Plan permutation symmetries as a source of planner inefficiency. In *Proc. of UK Workshop on Planning and Scheduling*, 2003.

[McKay, 1990] B. McKay. *Nauty* Users Guide 1.5. Technical Report TR-CS-90-02, Australian National University, 1990.

[Rintanen, 2003] J. Rintanen. Symmetry reduction for SAT representations of transition systems. In *Proc. of the 13th Int. Conf. on Planning and Scheduling*, 2003.

[Roney-Dougal *et al.*, 2004] C.M. Roney-Dougal, I.P. Gent, T. Kelsey, and S. Linton. Tractable symmetry breaking using restricted search trees. In *Proceedings of ECAI*, 2004.

[Roy and Pachet, 1998] P. Roy and F. Pachet. Using symmetry of global constraints to speed up the resolution of CSPs. In *Workshop on Non-binary Constraints, ECAI*, 1998.

Integrating Planning and Temporal Reasoning for Domains with Durations and Time Windows

Alfonso Gerevini[+] **Alessandro Saetti**[+] **Ivan Serina**[*]
[+] Dip. Elettronica per l'Automazione, Università di Brescia, via Branze 38, 25123 Brescia, Italy
[*] Dept. of Computer and Information Systems, University of Strathclyde, Glasgow, UK
E-mail: {gerevini,saetti}@ing.unibs.it ivan.serina@cis.strath.ac.uk

Abstract

The treatment of exogenous events in planning is practically important in many domains. In this paper we focus on planning with exogenous events that happen at known times, and affect the plan actions by imposing that the execution of certain plan actions must be during some time windows. When actions have durations, handling such constraints adds an extra difficulty to planning, which we address by integrating temporal reasoning into planning. We propose a new approach to planning in domains with durations and time windows, combining graph-based planning and disjunctive constraint-based temporal reasoning. Our techniques are implemented in a planner that took part in the 4th International Planning Competition showing very good performance in many benchmark problems.

1 Introduction

In many real-world planning domains, the execution of certain actions can only occur during some predefined time windows where one or more necessary conditions hold. For instance, we can refuel a car at a gas station only during specific period(s) of the day (when the gas station is open). The truth of these conditions is determined by some exogenous events that happen at known times, and that cannot be influenced by the actions available to the planning agent (e.g., the closing of the fuel station).

Several frameworks supporting durations and time windows have been proposed (e.g., [Vere, 1983; Muscettola, 1994; Laborie & Ghallab, 1995; Schwartz & Pollack, 2004]). However, most of them are domain-dependent systems or are not fast enough on large-scale problems. In this paper, we propose a new approach to planning with these temporal features, that combines graph-based planning and constraint-based temporal reasoning.

The last two versions of the language of the International planning competition, PDDL2.1 and PDDL2.2, support planning with action durations and deterministic exogenous events [Fox & Long, 2003; 2004; Edelkamp & Hoffmann, 2004]. In particular, in PDDL2.2, deterministic exogenous events can be represented by *timed initial literals*, one of the new PDDL features on which the 2004 competition focused. Timed initial literals are stated in the description of the initial state of the planning problem through assertions of the form "`(at t L)`", where `t` is a real number, and `L` is a ground literal whose predicate does not appear in the effects of any domain action. The obvious meaning of `(at t L)` is that `L` is true from time `t`. A set of these assertions involving the same ground predicate defines a sequence of disjoint time windows over which the timed predicate holds. An example in the known benchmark domain " Zenotravel" is

```
(at 8 (open-fuelstation city1))
(at 12 (not (open-fuelstation city1)))
(at 15 (open-fuelstation city1))
(at 20 (not (open-fuelstation city1)))
```

These assertions define two time windows over which `(open-fuelstation city1)` is true. A timed initial literal is relevant to the planning process when it is a precondition of a domain action, which we call a *timed precondition* of the action. Each timed precondition of an action can be seen as a temporal scheduling constraint for the action, defining the feasible time window(s) when the action can be executed.

When actions in a plan have durations and timed preconditions, finding a valid plan is a complex task that requires integrating planning and reasoning about time, to check whether the execution of the planned actions can satisfy their scheduling constraints. If an action in the plan cannot be scheduled, the plan is not valid, and it must be revised.

The main contributions and organization of this work are: (i) a new representation of temporal plans with action durations and timed preconditions, integrating disjunctive constraint-based temporal reasoning into a recent graph-based approach to planning (Section 2); (ii) a polynomial method for solving the disjunctive temporal reasoning problems that arise in our context (Section 2); (iii) some new local search heuristics to guide the planning process using our representation (Section 3); (iv) an experimental analysis evaluating an implementation of our approach, showing good performance with respect to other recent domain-independent temporal planners (Section 4).

2 Temporally Disjunctive Action Graph

In our approach, we represent a (partial) plan for a domain with timed initial literals through an extension of the linear action graph representation [Gerevini, et al., 2003], which we call *Temporally-Disjunctive Action Graph* (TDA-graph).

2.1 Background: Linear Action Graph

A linear action graph (LA-graph) $\mathcal{A}$ for a planning problem Π is a directed acyclic leveled graph alternating a *fact level*, and an *action level*. Fact levels contain *fact nodes*, each of which is labeled by a ground predicate of Π. Each fact node

f at a level l is associated with a *no-op* action node at level l representing a dummy action having the predicate of f as its only precondition and effect. Each action level contains one action node labeled by the name of a domain action that it represents, and the no-op nodes corresponding to that level.

An action node labeled a at a level l is connected by incoming edges from the fact nodes at level l representing the preconditions of a (*precondition nodes*), and by outgoing edges to the fact nodes at level $l+1$ representing the effects of a (*effect nodes*). The initial level contains the special action node a_{start}, and the last level the special action node a_{end}. The effect nodes of a_{start} represent the positive facts of the initial state of Π, and the precondition nodes of a_{end} the goals of Π.

A pair of action nodes (possibly no-op nodes) can be constrained by a *persistent mutex relation*, i.e., a mutually exclusive relation holding at every level of the graph, imposing that the involved actions can never occur in parallel in a valid plan. Such relations can be efficiently precomputed using an algorithm given in [Gerevini, et al., 2003].

An LA-graph $\mathcal{A}$ also contains a set of *ordering constraints* between actions in the (partial) plan represented by the graph. These constraints are (i) constraints imposed during search to deal with mutually exclusive actions: if an action a at level l of $\mathcal{A}$ is mutex with an action b at a level after l, then a is constrained to finish before the start of b; (ii) constraints between actions implied by the causal structure of the plan: if an action a is used to achieve a precondition of an action b, then a is constrained to finish before the start of b.

The effects of an action node can be automatically propagated to the next levels of the graph through the corresponding no-ops, until there is an *interfering action* "blocking" the propagation, or the last level of the graph has been reached.

2.2 Augmenting the LA-graph with Disjunctive Temporal Constraints

Let p be a timed precondition over a set $W(p)$ of time windows. In the following, x^- and x^+ indicate the start time and end time of x, respectively, where x is either a time window or an action. We will describe our techniques focusing on action preconditions that must hold during the whole execution of the action (except at the end of the action, as for PDDL2.1 "over all" conditions), and on operator effects that hold at the end of the action execution.[1]

In order to represent plans where actions have durations and time windows for their possible scheduling, we augment the ordering constraints of an LA graph with (i) action *duration constraints* and (ii) action *scheduling constraints*. Duration constraints have form $a^+ - a^- = Dur(a)$, where $Dur(a)$ denotes the duration of an action a.[2] Duration constraints are supported by the representation presented in [Gerevini, et al., 2003], while the representation and treatment of scheduling constraints are a major contribution of this work.

Let π be the plan represented by an LA-graph $\mathcal{A}$. It is easy to see that the set $\mathcal{C}$ of the ordering constraints in $\mathcal{A}$, extended with the duration constraints of the actions in π, can be encoded into a *Simple Temporal Problem* (STP) [Dechter, et al., 1991], i.e., a set of constraints of form $y - x \leq t$, where y and x are point variables and t is a real number. For instance, if $a_i \in \pi$ is used to support a precondition node of a_j, then $a_i^+ - a_j^- \leq 0$ is in $\mathcal{C}$; if a_i and a_j are two mutex actions in π and a_i is ordered before a_j, then $a_i^+ - a_j^- \leq 0$ is in $\mathcal{C}$. Moreover, for every action $a \in \pi$, the following STP-constraints are in $\mathcal{C}$:

$$a^+ - a^- \leq Dur(a), \;\; a^- - a^+ \leq -Dur(a).$$

A scheduling constraint imposes that the execution of an action must occur during the time windows associated with a timed precondition of the action. Syntactically, it is a disjunctive constraint $c_1 \vee \cdots \vee c_n$, where c_i is of the form

$$(y_i^1 - x_i^1 \leq k_i^1) \wedge (y_i^2 - x_i^2 \leq k_i^2),$$

$y_i^1, x_i^1, y_i^2, x_i^2$ are action start times or action end times, and $k_i^1, k_i^2 \in \mathbb{R}$. For every action $a \in \pi$ with a timed precondition p, the following disjunctive constraint is added to $\mathcal{C}$:

$$\bigvee_{w \in W(p)} \big((a_{start}^+ - a^- \leq -w^-) \wedge (a^+ - a_{start}^+ \leq w^+)\big).$$

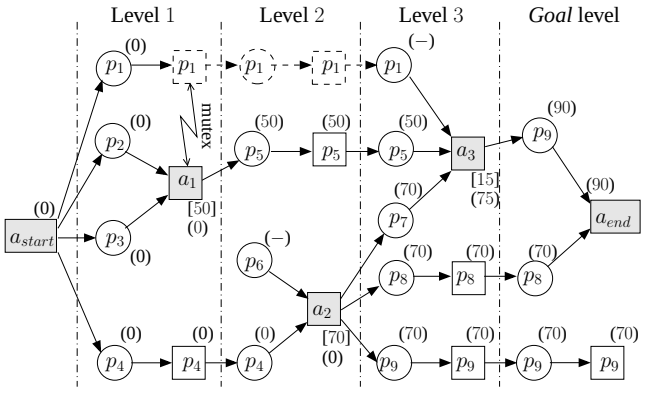


Figure 1: An example of LA-graph with nodes labeled by $\mathcal{T}$-values. Square nodes are action nodes; circle nodes are fact nodes. Dashed edges form chains of no-ops blocked by mutex actions. The $\mathcal{T}$-values are the numbers inside round brackets. The action durations are the numbers inside square brackets. Unsupported precondition nodes are labeled "(–)".

Definition 1 *A* **temporally disjunctive action graph** (TDA-graph) *is a 4-tuple $\langle \mathcal{A}, \mathcal{T}, \mathcal{P}, \mathcal{C} \rangle$ where*

- *$\mathcal{A}$ is a linear action graph;*
- *$\mathcal{T}$ is an assignment of real values to the nodes of $\mathcal{A}$;*
- *$\mathcal{P}$ is the set of time point variables corresponding to the start times and end times of the actions labeling the action nodes of $\mathcal{A}$;*
- *$\mathcal{C}$ is a set of ordering constraints, duration constraints and scheduling constraints involving variables in $\mathcal{P}$.*

A TDA-graph $\langle \mathcal{A}, \mathcal{T}, \mathcal{P}, \mathcal{C} \rangle$ represents the (partial) plan formed by the actions labeling the action nodes of $\mathcal{A}$ with start times assigned by $\mathcal{T}$. Figure 1 gives the LA-graph and $\mathcal{T}$-values of a simple TDA-graph. The ordering constraints and duration constraints in $\mathcal{C}$ are:[3]

[1] Our methods and planner support all the types of operator condition and effect that can be specified in PDDL 2.1 and 2.2.

[2] The duration of a_{start} and a_{end} is 0, $a_{start}^- = a_{start}^+$ and $a_{end}^- = a_{end}^+$.

[3] For brevity, in our examples we omit the constraints $a_{start}^+ - a_i^- \leq 0$ and $a_i^+ - a_{end}^- \leq 0$, for each action a_i.

$a_1^+ - a_3^- \leq 0,\ a_2^+ - a_3^- \leq 0,$
$a_1^+ - a_1^- = 50,\ a_2^+ - a_2^- = 70,\ a_3^+ - a_3^- = 15.$

Assuming that p is a timed precondition of a_3 with windows $[25, 50)$ and $[75, 100)$, the only scheduling constraint in $\mathcal{C}$ is:

$(a_{start}^+ - a_3^- \leq -25 \wedge a_3^+ - a_{start}^+ \leq 50) \vee$
$(a_{start}^+ - a_3^- \leq -75 \wedge a_3^+ - a_{start}^+ \leq 100).$

The pair $\langle \mathcal{P}, \mathcal{C} \rangle$ defines a *Disjunctive Temporal Problem* $\mathcal{D}$ (DTP) [Stergiou & Koubarakis, 2000; Tsamardinos & Pollack, 2003].[4] Let $\mathcal{D}_s$ be the set of scheduling constraints in $\mathcal{D}$. We have that $\mathcal{D}$ represents a set Θ of STPs, each of which consists of the constraints in $\mathcal{D} - \mathcal{D}_s$ and one disjunct (pair of STP-constraints) for each disjunction in a *subset* of $\mathcal{D}_s$. We call a consistent STP in Θ an *induced STP* of $\mathcal{D}$. If an induced STP contains a disjunct for every disjunction in $\mathcal{D}_s$, we say that such a (consistent) STP is a *complete induced STP* of $\mathcal{D}$.

An STP is *consistent* iff it has a solution. A *solution* of an STP is an assignment of real values to the variables of the STP that is consistent with every constraint in the STP. Given an induced STP, we can compute in $O(n \cdot c)$ time a solution where each variable has the shortest possible distance from a_{start}^+ [Dechter, et al., 1991; Gerevini & Cristani, 1997], for n variables and c constraints. We call such a solution an *optimal solution* for the induced STP under consideration.

The values assigned by $\mathcal{T}$ to the action nodes of $\mathcal{A}$ are the action start times corresponding to an optimal solution of an induced STP. We call these start times a *schedule* of the actions in $\mathcal{A}$. The $\mathcal{T}$ value labeling a fact node f of $\mathcal{A}$ is the earliest time $t = \mathcal{T}_a + Dur(a)$ such that a supports f in $\mathcal{A}$, and a starts at $\mathcal{T}_a$.

If the induced STP from which we derive a schedule is incomplete, $\mathcal{T}$ may violate the scheduling constraint of some action nodes, that we say are *unscheduled* in $\mathcal{T}$.

The following definition gives a notion of optimality over the complete induced STPs of a DTP that will be used in the next section.

Definition 2 *Given a DTP $\mathcal{D}$ with a point variable p, a complete induced STP of $\mathcal{D}$ is an* **optimal induced STP** *of $\mathcal{D}$ for p, iff it has a solution assigning to p a value that is less than or equal to the value assigned to p by every solution of every other complete induced STP of $\mathcal{D}$.*

An *optimal schedule* is an optimal solution of an optimal induced STP for a_{end}^-. Note that an optimal solution minimizes the makespan of the represented (possibly partial) plan.

2.3 Solving the DTP of a TDA-graph

In general, computing a complete induced STP of a DTP (if it exists) is an NP-hard problem that can be solved by a backtracking algorithm [Stergiou & Koubarakis, 2000; Tsamardinos & Pollack, 2003]. However, given the particular structure of the temporal constraints forming a TDA-graph, we show that this task can be accomplished in polynomial time with a backtrack-free algorithm. Moreover, the algorithm computes an optimal induced STP for a_{end}^-.

[4] The disjunctive constraints in $\mathcal{C}$ are not exactly in DTP-form, i.e., a disjunction $c_1 \vee \cdots \vee c_n$, where c_i is of form $y_i - x_i \leq k_i$, x_i and y_i are time points, and k_i is a real number. However, it is easy to see that every disjunctive constraint in $\mathcal{C}$ can be translated into an equivalent conjunction of constraints in exact DTP-form. We use our more compact notation for clarity and efficiency reasons.

Solve-DTP(X, S)

1. **if** $X = \emptyset$ **then stop** and **return** S;
2. $x \leftarrow$ SelectVariable(X); $X' \leftarrow X - \{x\}$;
3. **while** $D(x) \neq \emptyset$ **do**
4. $d \leftarrow$ SelectValue($D(x)$); $D(x) \leftarrow D(x) - \{d\}$;
5. $S' \leftarrow S \cup \{x \leftarrow d\}$;
6. $D'(x) \leftarrow D(x)$; /* Saving the domain values */
7. **if** ForwardChecking-DTP(X', S') **then**
8. Solve-DTP(X', S');
9. $D(x) \leftarrow D'(x)$; /* Restoring the domain values */
10. **return** *fail*; /* backtracking */

ForwardChecking-DTP(X, S)

1. **forall** $x \in X$ **do**
2. **forall** $d \in D(x)$ **do**
3. **if** *not* Consistency-STP($S \cup \{x \leftarrow d\}$) **then**
4. $D(x) \leftarrow D(x) - \{d\}$;
5. **if** $D(x) = \emptyset$ **then return** *false*; /* dead-end */
6. **return** *true*.

Figure 2: Basic algorithm for solving a DTP. The input is the set X of the meta-variables in the meta CSP of the DTP, and a (partial) solution S of the meta CSP. $D(x)$ is a global variable storing the current domain of the meta variable x.

Without loss of generality, we can assume that each action has at most one timed precondition. It is easy to see that we can replace a set of timed preconditions of an action a with a single equivalent timed precondition, whose time windows are obtained by intersecting the windows forming the different original timed preconditions of a.

As observed in [Stergiou & Koubarakis, 2000; Tsamardinos & Pollack, 2003], a DTP can be seen as a "meta CSP", where the variables are the constraints, and the values of the meta-variables are the disjuncts forming the constraints. The constraints of the meta CSP are not explicitly stated. Instead, they are implicitly defined as follows: an assignment θ of values to the meta-variables satisfies the constraints of the meta CSP iff θ forms a consistent STP (an induced STP of the DTP). A solution of the meta CSP is a complete induced STP of the DTP.

Figure 2 shows an algorithm for solving the meta CSP of a DTP [Tsamardinos & Pollack, 2003], which is a variant of the forward-checking backtracking algorithm for solving general CSPs. By appropriately choosing the next meta-variable to handle (function SelectVariable) and its value (function SelectValue), we can show that the algorithm finds a solution (if one exists) with *no backtracking*. Moreover, by a simple modification of the basic algorithm, we can derive an algorithm that is backtrack free even when the meta CSP has no solution. This can be achieved by exploiting the information in the LA-graph $\mathcal{A}$ of the TDA-graph for decomposing its DTP $\mathcal{D}$ into a sequence of "growing DTPs". I.e.,

$$\mathcal{D} = \mathcal{D}_{last} \supset \mathcal{D}_{last-1} \supset ... \supset \mathcal{D}_1,$$

where (i) *last* is the number of the levels in $\mathcal{A}$, (ii) the variables V_i of $\mathcal{D}_i$ ($i = 1..last$) are all the variables of $\mathcal{D}$ corresponding to the action nodes in $\mathcal{A}$ up to level i, and (iii) the constraints of $\mathcal{D}_i$ are all the constraints of $\mathcal{D}$ involving only

variables in V_i. From the decomposed DTP, we can derive an ordered partition of the set of meta-variables X in the meta CSP of the original DTP

$$X = X_1 \cup X_2 \cup ... \cup X_{last},$$

where X_i is the set of the meta-variables corresponding to the constraints in $\mathcal{D}_i - \mathcal{D}_{i-1}$, if $i > 1$, and in $\mathcal{D}_1$ otherwise.

This ordered partition is used to define the order in which SelectVariable chooses the next variable to handle, which is crucial to avoid backtrack: every variable with a single domain value (i.e., an ordering constraint or duration constraint) is selected before every variable with more than one possible value (i.e., a scheduling constraint with more than one time window); if $x_i \in X_i$, $x_j \in X_j$ and $i < j$, then x_i is selected before x_j.

Also the order in which SelectValue chooses the value for a meta-variable is important: given a meta-variable with more than one value, we choose the value corresponding to the earliest available time window. E.g., if the current domain of the meta-variable is

$$\bigcup_{i=1..m} \{(a^+_{start} - a^- \leq -k^-_i) \wedge (a^+ - a^+_{start} \leq k^+_i)\},$$

then SelectValue chooses the j-th value (time window) such that $|k^-_j| < |k^-_h|$, for every $h \in \{1, ..., m\}$, $h \neq j$.

By using these techniques for selecting the next variable to handle and its domain value in the algorithm of Figure 2, we can derive the following result.[5]

Theorem 1 *Given a DTP $\mathcal{D}$ for a TDA-graph, if the meta CSP $\mathcal{X}$ of $\mathcal{D}$ is solvable, then* Solve-DTP *finds a solution of $\mathcal{X}$ with no backtracking. Moreover, this solution is an optimal induced STP of $\mathcal{D}$ for a^-_{end}.*

As a consequence of the previous theorem, we have that, if Solve-DTP performs backtracking (step 10), then the DTP under consideration has no solution. Thus, we can obtain a backtrack free algorithm by replacing step 10 with

10. **stop** and **return** *fail*.

It is easy to see that in the modified algorithm, called Solve-DTP$^+$, every variable is instantiated at most once with the same value. It follows that, under the assumptions that we have a constant maximum number of action preconditions and, for every scheduling constraint, a constant maximum number of windows, the total runtime complexity of Solve-DTP$^+$ is polynomial.

Theorem 2 *Given a TDA-graph $\mathcal{G}$ with DTP $\mathcal{D}$,* Solve-DTP$^+$ *processes the meta CSP corresponding to $\mathcal{D}$ in polynomial time with respect to the number of action nodes in $\mathcal{G}$.*

The actual algorithm that we developed for our planner to find an induced STP for the DTP of a TDA-graph contains some improvements making it more efficient. For lack of space and simplicity of presentation, we omit a detailed description of the improved algorithm, and we indicate only the main differences, which are the following ones:

- the consistency of the STP formed by the values of all the variables of the meta CSP $\mathcal{X}$ with single-valued domains can be checked at the beginning of Solve-DTP, using a single-source shortest-path algorithm: if such an STP is inconsistent, then $\mathcal{X}$ has no solution;
- forward checking is performed only once for each meta-variable: if the first value chosen by SelectValue is not feasible (i.e., ForwardChecking-DTP returns false), then $\mathcal{X}$ has no solution, and thus we can stop the algorithm;
- finally, the improved algorithm is incremental since, as we will see in the next section, at each search step the DTP of the TDA-graph is updated as a consequence of adding a new action node to the graph, or removing an existing one.

Moreover, in order to use the local search techniques described in the next section, we need another change to the basic algorithm: when the algorithm detects that $\mathcal{X}$ has no solution, instead of returning failure, (i) it processes the next meta variables, and (ii) when it terminates, it returns the (partial) induced STP $\mathcal{S}$ formed by the STP-constraints of the DTP and the values assigned to the meta-variables. The optimal solution of $\mathcal{S}$ defines the $\mathcal{T}$-assignment of the TDA-graph.

In the next section, $\mathcal{S}_\mathcal{G}$ denotes the induced STP for the DTP of a TDA-graph $\mathcal{G}$ computed by our method.

3 Local Search Techniques for TDA-Graphs

A TDA-graph $\langle \mathcal{A}, \mathcal{T}, \mathcal{P}, \mathcal{C} \rangle$ may contain two types of *flaw*: unsupported precondition nodes in $\mathcal{A}$ (*propositional flaw*s), action nodes in $\mathcal{A}$ that are unscheduled in $\mathcal{T}$ (*temporal flaw*s). If a level of $\mathcal{A}$ contains a flaw, we say that this level is flawed. A TDA-graph with no flawed level represents a valid plan, and it is called a *solution graph*.

In this section, we present new heuristics for searching a solution graph in the space of TDA-graphs. These heuristics are used to guide a local search procedure, called Walkplan, that was originally proposed in [Gerevini, et al., 2003], and that is the heart of search engine of our planner.

The initial TDA-graph contains only a_{start} and a_{end}. Each search step identifies the neighborhood $N(\mathcal{G})$ (successor states) of the current TDA-graph $\mathcal{G}$ (search state), which is a set of TDA-graphs obtained from $\mathcal{G}$ by adding a *helpful action node* or removing a *harmful action node* in the attempt to repair the *earliest* flawed level of $\mathcal{G}$.[6] In the following, a_i denotes an action node a at level i of $\mathcal{A}$, and l_a the level of a.

Given a flawed level l of $\mathcal{G}$, an action node a_i is *helpful* for l if its insertion into $\mathcal{G}$ at a level $i \leq l$ removes a propositional flaw at l; a_i is *harmful* for l if its *removal* from a level $i \leq l$ of $\mathcal{G}$ (i) would remove a propositional flaw at l, or (ii) would decrease the $\mathcal{T}$-value of a_l, if a_l is unscheduled (intuitively, a_l is unscheduled if $\mathcal{C}$ forces it to start "too late").

The addition/removal of an action node a requires us to update the DTP of $\mathcal{G}$ by adding/removing some ordering constraints between a and other actions in the LA-graph of $\mathcal{G}$, the duration constraints of a, and the scheduling constraint of a (if any). From the updated DTP $\mathcal{D}$, we can use the method described in the previous section to revise $\mathcal{T}$, and to compute a possibly new schedule of the actions in $\mathcal{G}$ (i.e., an optimal solution of $\mathcal{S}_\mathcal{G}$).

The elements in $N(\mathcal{G})$ are evaluated using a *heuristic evaluation function* E consisting of two weighted terms, estimating the additional *search cost* and *temporal cost* of the el-

[5] For lack of space, the proofs are omitted; they are available in an extended version of this paper [Gerevini, et al., 2005].

[6] When we add an action node, the graph is extended by one level, and when we remove an action node, it is "shrunk" by one level. More details in [Gerevini, et al., 2003].

ements (i.e., the number of search steps required to find a solution graph and the plan makespan, respectively). An element with the lowest cost is then selected from $N(\mathcal{G})$ using a "noise parameter" randomizing the search to escape from local minima [Gerevini, et al., 2003]. For lack of space, in the rest of this section we focus only on the search cost term of E.

The search cost of adding an helpful action node a to $\mathcal{G}$ is estimated by constructing a *temporal relaxed plan* π achieving (1) the unsupported precondition nodes of a, (2) the propositional flaws remaining at l after adding a, and (3) the supported precondition nodes of other action nodes in $\mathcal{G}$ that would become *un*supported by adding a. Moreover, we count the number of: (4) action nodes that would become *un*scheduled by adding a to $\mathcal{G}$, (5) unsatisfied timed preconditions of a, (6) actions of π with a scheduling constraint that we estimate cannot be satisfied in the context of $\mathcal{G}$. The search cost of adding a to $\mathcal{G}$ is the number of actions in π plus (4), (5) and (6).

The evaluation of a TDA-graph derived by *removing* an harmful action node a is similar, with π achieving the precondition nodes supported by a that would become *un*supported by removing a and, when l_a precedes the flawed level l under reparation, the unsupported precondition nodes at level l that would not become supported by removing a.

π is constructed using a polymomial backward process similar to the algorithm proposed in [Gerevini, et al., 2003], giving in output two values: a set of actions forming a (sub)relaxed plan, and its estimated earliest finishing time. The initial state I is the state obtained by applying the actions of $\mathcal{G}$ up to level $l_a - 1$, ordered according to their levels.

The main difference in the extended algorithm concerns the choice of the actions forming the relaxed plan. The action b chosen to achieve a (sub)goal g is an action minimizing the sum of (i) the estimated minimum number of additional actions required to support its propositional preconditions, (ii) the number of supported precondition nodes in the LA-graph that would become unsupported by adding b to $\mathcal{G}$, (iii) the number of timed preconditions of b that we estimate would be unsatisfied in $\mathcal{G}$ extended with π ($TimedPre(b)$); and (iv) the number of action nodes scheduled in $\mathcal{T}$ that we estimated would become unscheduled by adding b to $\mathcal{G}$ ($TimeThreats(b)$). (i)-(ii) are computed as described in [Gerevini, et al., 2003]; (iii)-(iv) are new components of the action selection method, and they are computed as follows.

In order to compute $TimedPre(b)$, we estimate the earliest start time of b ($Est(b)$) and the earliest finishing time of b ($Eft(b)$). Using these values, we count the number of the timed preconditions of b that cannot be satisfied. $Eft(b)$ is $Est(b) + Dur(b)$, while $Est(b)$ is the maximum over

- the lowest earliest start time of b computed by an extension of the reachability analysis algorithm given in [Gerevini, et al., 2003], which derives a lower bound on the start time of each domain action;
- the $\mathcal{T}$-values of the action nodes c_i, with $i < l_a$, that are mutex with b (because the addition of b to $\mathcal{G}$ would determine the addition of $c_i^+ - b^- \leq 0$ to $\mathcal{G}$);
- the maximum over an estimated lower bound on the time when all the preconditions of b are achieved in relaxed plan (this estimate is computed from the causal structure of the relaxed plan, the duration and scheduling constraints of its actions, and the $\mathcal{T}$-values of the facts in the initial state I).

In order to compute *TimeThreats*(b), we use a notion of *time slack* between action nodes.

Definition 3 *Given two action nodes $a1$ and $a2$ of a TDA-graph $\langle \mathcal{A}, \mathcal{T}, \mathcal{P}, \mathcal{C} \rangle$ such that $\mathcal{C} \models a1^+ < a2^-$, $slack(a1, a2)$ is the maximum time by which the $\mathcal{T}$-value of $a1^-$ can be consistently increased in $\mathcal{S}_\mathcal{G}$ without violating the time window chosen for scheduling $a2$.*

To estimate whether b is a time threat for an action node a_k ($l \leq k$), we check if $\Delta(\pi_b, a_l) > Slack(a_l, a_k)$ holds, where π_b is the portion of the relaxed plan computed so far, and $\Delta(\pi_b, a_l)$ estimates the delay of the start time of a_l that the addition of the actions in π_b to $\mathcal{G}$ would determine.

4 Experimental Results

We have implemented our approach in a planner called LPG-td, which obtained the 2nd prize in the suboptimal metric-temporal track of the 4th International Planning Competition (IPC-4). LPG-td performed especially well in the domain variants with timed initial literals, in terms of both CPU-time to find a plan and quality of the best plan computed with a CPU-time limit of 30 minutes (LPG-td is an incremental planner finding a succession of valid plans). In this section, we present some experimental results using the test problems of IPC-4.[7] The problems in the `Airport` domain specify at most 6 time windows for each timed precondition, the problems in the `Satellite` domain at most 3 windows, while those in the other domains only one time window. Additional results are available from the web sites of our planner and of IPC-4, and in a technical report including an experimental analysis on solving problems with many windows associated with the timed preconditions [Gerevini, et al., 2005].

Figure 3 shows the CPU-time of LPG-td in three IPC-4 domains with respect to the best among the other three planners of IPC-4 that support timed initial literals: SGPLAN, P-MEP, and TILSAPA.[8] In these domains, LPG-td is generally faster than the other planners and solves more problems.

Table 1 gives a summary of the results for all the IPC-4 domain variants with timed initial literals (252 test problems in total). We compare LPG-td's results with the best results over the corresponding results of *all* the other IPC-4 planners ("AllOthers"). In general, LPG-td solves more problems than AllOthers; the percentage of problems in which it is faster is higher than the one in which it is slower; and the percentage in which it produces better quality plans is much higher.

Finally, it is worth noting that, if in the CPU-time comparison we consider only problems where LPG-td is at least one order of magnitude faster (slower) than AllOthers, then the results in the 3rd column of Table 1 are even more favorable to LPG-td. LPG-td is faster in 31% of the problems, and it is slower in 13% of the problems.

[7] All tests were conducted on an Intel Xeon(tm) 3 GHz, 1 Gbytes of RAM. For a description and formalization of the IPC-4 benchmark problems and domains, see `http://ls5-www.cs.uni-dortmund.de/~edelkamp/ipc-4/index.html`.

[8] An abstract of every IPC-4 planner is available in [Edelkamp, et al., 2004]). LPG-td and TILSAPA are the only planners of IPC-4 that addressed the variant of `PipesWorld` with timed initial literals; TILSAPA did not address `UMTS-flaw` with timed initial literals.

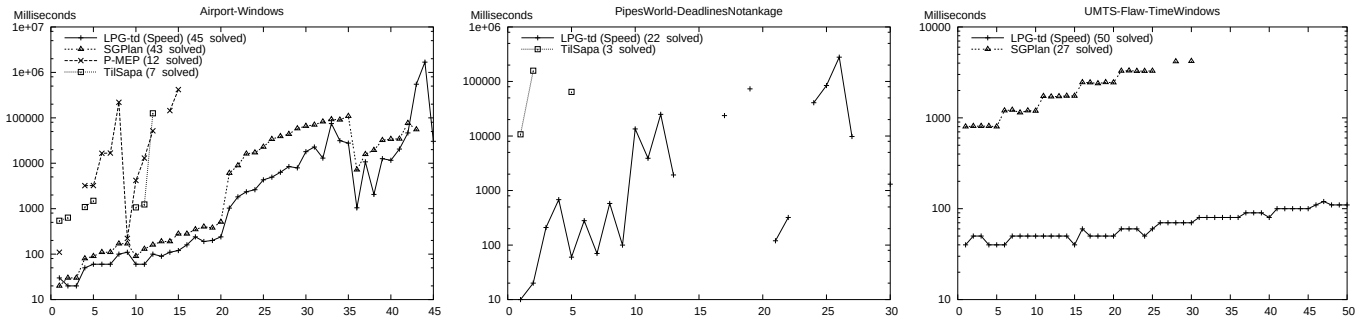


Figure 3: CPU-times of LPG-td, P-MEP, SGPLAN, and TILSAPA in three IPC-4 test domains with timed initial literals. On the x-axis we have the problem names simplified by numbers. On the y-axis, we have the CPU-times in logarithmic scale.

IPC-4 domain	Problems solved	CPU-time better (worse)	Plan quality better (worse)
Airport	90 (86)	86 (4)	58 (5)
PipesWorld	73 (10)	73 (0)	67 (0)
Sat-Complex	53 (67)	6 (67)	71 (12)
Sat-Time	53 (67)	6 (64)	59 (35)
UMTS-Flaw	100 (54)	100 (0)	82 (0)
UMTS	100 (100)	0 (88)	100 (0)
Total	81.3 (67.8)	47.2 (39.3)	73.2 (6.4)

Table 1: A comparison of LPG-td and the best over the results of *all* the other IPC-4 planners. Summary results in terms of: % of problems solved by LPG and AllOthers (in brackets); % of problems in which LPG-td is faster (slower in brackets); % of problems in which LPG-td produces a plan with shorter makespan (longer in brackets).

5 Conclusions

We have presented an approach to temporal planning for domains where actions have durations and must be executed during certain time windows. This allows us to deal with deterministic exogenous events, which is important in many real-world planning domains. Our approach combines constraint-based temporal reasoning and a recent graph-based method for planning. We propose a new plan representation and search space, a polynomial algorithm for temporal constraint reasoning during search, and some local search techniques for planning that exploit temporal information. An analysis of the IPC-4 results show that our planner performs very well compared to other recent temporal planners. We believe that our temporal reasoning results can be exploited also in the context of other approaches to planning.

Like our planner, SAPA uses a relaxed plan heuristic to guide the search [Do et al., 2004]. However, SAPA uses a time slack analysis for selecting these actions that is limited to the actions of the relaxed plan, while our heuristics consider also the actions of the "real" plan under construction. Other very recent planners supporting time windows include DT-POP and MIPS. Edelkamp proposes a method for handling timed preconditions in MIPS with only one time window [Edelkamp, 2004]. DT-POP extends POP-planning with DTPs [Schwartz & Pollack, 2004]. DT-POP supports more temporal features than LPG-td, but it is less efficient, and it does not exploit the plan representation that we use for achieving tractable temporal reasoning during planning. For a more detailed discussion of related work see [Gerevini, et al., 2005].

References

[Blum & Furst, 1997] Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90.

[Do et al., 2004] Do, M., B., Kambhampati, S., and Zimmerman, T. 2004. Planning - Scheduling Connections through Exogenous Events. In *Proc. of WIPIS-04*.

[Dechter, et al., 1991] Dechter, R., Meiri, I., and Pearl, J., 1991. Temporal Constraint Networks, *Artificial Intelligence* 49:61–95.

[Edelkamp, 2004] Edelkamp, S. 2004. Extended Critical Paths in Temporal Planning. In *Proceedings of the ICAPS-04 Workshop on Integrating Planning into Scheduling*.

[Edelkamp & Hoffmann, 2004] Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The Language for the Classic Part of the 4th International Planning Competition. Technical Report 195, Institut für Informatik, Freiburg, Germany.

[Edelkamp, et al., 2004] In Edelkamp, S., Hoffmann, J., Littman, M., Younes, H. (Eds.) 2004. *In Abstract Booklet of the Competing Planners of ICAPS-04*.

[Fox & Long, 2003] Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *JAIR* 20:61–124.

[Fox & Long, 2004] Fox, M. and Long, D. and Halsey, K., 2004. An Investigation into the Expressive Power of PDDL2.1. In *Proc. of ECAI-04*.

[Gerevini & Cristani, 1997] Gerevini, A., and Cristani, M., 1997. On Finding Solutions in Temporal Constraint Networks. In *Proc. of IJCAI-97*.

[Gerevini, et al., 2003] Gerevini, A., Saetti, A., and Serina, I. 2003. Planning through Stochastic Local Search and Temporal Action Graphs. *JAIR* 20:239–290.

[Gerevini, et al., 2005] Gerevini, A., Saetti, A., and Serina, I. 2005. An Approach to Temporal Planning in Domains with Deterministic Exogenous Events. Technical Report RT 2005-06-45, DEA, Universitá di Brescia, Italy.

[Laborie & Ghallab, 1995] Laborie, P. and Ghallab, M. 1995. Planning with Sharable Resource Constraints. *Proc. of IJCAI-95*

[Muscettola, 1994] Muscettola, N. 1994 HSTS: Integrating Planning and Scheduling. *Intelligent Scheduling*. Morgan Kaufmann.

[Schwartz & Pollack, 2004] Schwartz, P., J. and Pollack, M., E. 2004. Planning with Disjunctive Temporal Constraints. In *Proc. of WIPIS-04*.

[Stergiou & Koubarakis, 2000] Stergiou, K., Koubarakis, M. 2000. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence* 120:81–117.

[Tsamardinos & Pollack, 2003] Tsamardinos, I., Pollack, M., E. 2003. Efficient solution tecniques for Disjunctive Temporal Reasoning Problems. *Artificial Intelligence* 151:43–90.

[Vere, 1983] Vere, S. A. 1983 Planning in time: Windows and durations for activities and goals. *IEEE Transactions on PAMI* 5(3):246–267.

Real-Time Path Planning for Humanoid Robot Navigation

Jens-Steffen Gutmann **Masaki Fukuchi** **Masahiro Fujita**
Intelligent Systems Research Laboratory
Sony Corporation, Tokyo, Japan
{steffen,fukuchi,mfujita}@pdp.crl.sony.co.jp

Abstract

We present a data structure and an algorithm for real-time path planning of a humanoid robot. Due to the many degrees of freedom, the robots shape and available actions are approximated for finding solutions efficiently. The resulting 3 dimensional configuration space is searched by the A* algorithm finding solutions in tenths of a second on low-performance, embedded hardware. Experimental results demonstrate our solution for a robot in a world containing obstacles with different heights, stairs and a higher-level platform.

1 Introduction

Path-planning is one of the fundamental problems in mobile robot navigation. It has been shown long before that the problem of moving an object through space is *PSPACE-hard* with a time complexity exponential in the degrees of freedom of the object [Latombe, 1991]. In mobile robots the degrees of freedom are usually small (three or less) which opens the application of a range of search techniques specifically tailored to the problem. If one can neglect orientation, e.g. a cylindrical holonomic system, the resulting 2 dimensional configuration space can be searched efficiently by A*, D* [Stentz, 1995], or dynamic programming [Buhmann *et al.*, 1995].

Humanoid robots while being mobile allow for more than the usual three degrees of freedom. Their feet can be placed with a greater choice and the change of body posture allows to overcome obstacles where wheeled robots fail in passing through. This enables humanoids to step onto objects [Kuffner *et al.*, 2002], climb up and down stairs [Hirai *et al.*, 1998; Gutmann *et al.*, 2004], step over obstacles [Guan *et al.*, 2004], or crawl underneath objects [Shiller *et al.*, 2001; Kanehiro *et al.*, 2004].

The higher degrees of freedom can be tackled e.g. by probabilistic roadmaps [Kavraki *et al.*, 1996]. However, if one can employ a non-probabilistic approach by finding a suitable approximation then the solution can be seen preferable due to its deterministic nature. One possible approximation is to model the robot as a cylinder or by its convex hull which allows for efficient collision checking [Okada *et al.*, 2003].

Care should be taken though not to over-simplify the problem. Humanoid robots are non-holonomic, i.e. they cannot turn in place without requiring additional space. The trajectory of the body center describes a curve similar to a car-like robot, although the turn radius is generally quite small. It is possible to account for the extra turning space in a cylinder model of the robot enlarged by twice the turn radius [Li *et al.*, 2003; Sabe *et al.*, 2004]. However such an approximation is overly pessimistic and prevents the robot from passing through narrow space as we will show.

Discretizing the possible actions of the robot into a small set of well-chosen actions, e.g. different foot placements, leads to another way for finding solutions efficiently. The induced search graph grows exponentially with the number of steps and thus, only a small number of foot steps can be explored in a real-time system [Lorch *et al.*, 2002]. Such a foot-step planning system is nevertheless a valuable tool in a hybrid system where a higher level planner provides waypoints for this sub-system [Chestnutt and Kuffner, 2004].

Shiller *et al.* [2001] discretize the actions and orientations of a digital figure that can walk forward, sideways, turn and crawl. They maintain an obstacle space for each combination of action and orientation and find solutions efficiently for trajectories of several ten steps.

We compute a trajectory for the body-center of a humanoid robot by approximating the shape of the robot using multiple cylinders and discretizing configurations into a set of orientations. When following this trajectory, the next few foot steps are computed using an analytical method. This enables a humanoid robot to find paths of up to a few meters in real-time and includes actions such as sideways walking through narrow space and climbing up and down stairs.

Our work differs from Shiller's in that we do not plan behaviors but select the behavior depending on the environment when following a found path, e.g. stair-climbing over a sill. We also maintain only one obstacle space and show how to perform collision checking efficiently. Last but not least, our system is demonstrated on a real robot where stereo range data incrementally updates a representation of the world.

Our approximations of configuration space and actions are presented in the next section. Section 3 briefly describes our approach for generating a navigation map for collision checking. Section 4 details our application of the A* algorithm for path searching. Experimental results on Sony's humanoid robot QRIO are described in Section 5. Finally, we discuss limitations and extensions of our approach in Section 6.

2 Configuration Space and Actions

The robot moves in a 2 dimensional world. Obstacles, different floor levels and stairs are projected onto one plane. We do not neglect the orientation of the non-holonomic robot, thus the configurations space C is 3 dimensional. For allowing efficient search we discretize C into a regular grid of evenly spaced cells using 8 different orientations:

$$\begin{aligned} C &= X \times Y \times \Phi \\ X, Y &= \{i \cdot cs \mid 0 \leq i < n\} \\ \Phi &= \{i \cdot 45^\circ \mid 0 \leq i < 8\} \end{aligned} \quad (1)$$

where cs defines the cell size and n the number of cells.

We restrict the movement of the robot's body center to the following six actions $A = \{$ *forward, backward, turn-left, turn-right, side-left, side-right*$\}$. For each action the transition in C is given for orientations 0° and 45° in Figure 1. For other orientations the transitions are analogous.

When concatenating turn-left (or turn-right) actions, the resulting trajectory describes a curve, see Figure 2(a). The turn actions thus model the non-holonomic constraint of a humanoid where changing the orientation requires extra space for turning. If we approximate this curve with an arc of a circle, we find that the cell size cs coincides with the turn radius of the robot and therefore cs should be chosen as such.

The shape of the robot is approximated by two cylinders sharing the same axis as shown in Figure 2(b). Cylinders allow for fast collision checking and the smaller cylinder for the legs enables the robot to pass close to low obstacles where upper body and arms are above the obstacle.

The cylinders are a tight fit to the shape of the robot standing still. When the robot moves, however, additional space is needed for the transition in C as body and legs are swinging while walking. Figure 3 shows snapshots of the robot when moving forward, turning left and stepping sideways. We account for the additional space by enlarging the cylinders at the start and end configurations depending on the action. Both, upper and lower, cylinders are enlarged by the same amount. Only when walking sideways, no additional space is needed. As we will see in Section 5 this enables the robot to pass through narrow space by choosing the sideways walk.

It is worth noting that the humanoid robot does not necessarily need to be able to exactly perform the 6 actions defined in this section. For example, the robot could well use several foot steps for performing the 45° rotation in our turn actions. The only requirement is that the enlarged cylinders cover all the space the robot requires for completing the 45° turn.

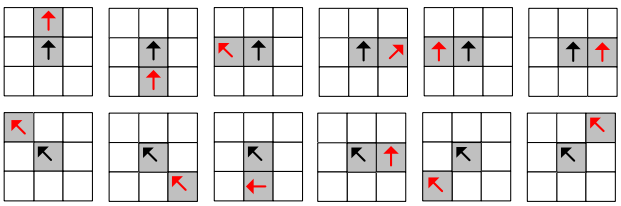


Figure 1: Change in configuration for our 6 actions.

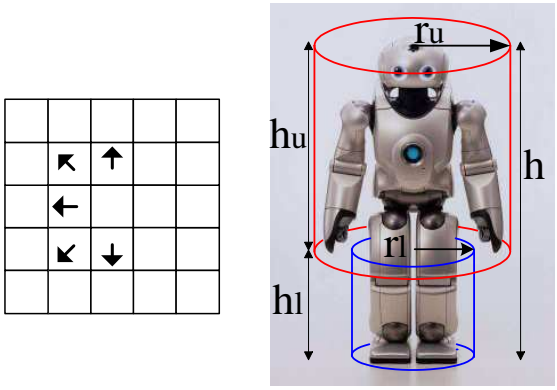


Figure 2: (a) Turn around by concatenating 4 turn-left actions. (b) Multi-cylinder model for approximating robot shape.

One might think of our approach to be specifically tailored to humanoids as the proposed actions and the cell size matching the turn radius are probably only found for this kind of robot. While this seems to be true, the contribution of this work is to demonstrate the abilities of such a system in a complex real-world environment. We will discuss the limitations and possible extensions of our approach in Section 6.

3 Navigation Map and Collision Checking

Collision checking is one of the crucial factors in path planning. Cylinder models of robots allow to perform this check in constant time if the distance to the nearest obstacle is known; the distance is simply compared to the radius of the cylinder. Our two cylinder model allows to check for collision in the same amount of time.

For efficient collision checking we make the following assumptions about the world the robot is navigating in:

1. The world is partitioned into floor and obstacles.
2. Floor is planar and horizontal (else it is an obstacle).
3. There are no multiple floor levels at the same location.
4. The robot is able to distinguish between floor and obstacles and can can estimate their relative position and height using its sensors.

Assumption 1 is general, 2 rules out inclined surfaces and treats them like obstacles, 3 is true for most but a few environments and enables the use of a 2.5D map for representing

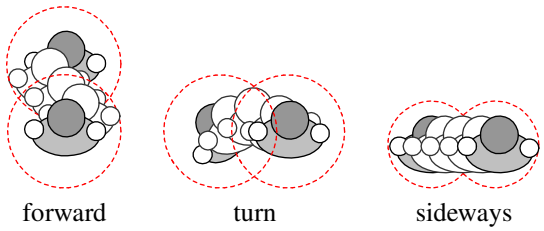


Figure 3: Space required for performing an action. Enlarged upper cylinders approximate the additional space needed.

the world. Finally, assumption 4 is a practical consideration and can be achieved e.g. by plane extraction from range data.[1]

Under these assumptions, the robot is able to build a floor and obstacle height map FOG of its environment. We employ a combination of a 3D occupancy grid with a 2.5D floor height map where range data from stereo vision is segmented into planes and integrated in a probabilistic way [Gutmann *et al.*, 2005]. Each cell in FOG holds a type and a height:

$$FOG : (x, y) \quad \mapsto \quad (t, h), \tag{2}$$

with $x \in X,\ y \in Y,\ t \in T = \{floor, obstacle, unknown\}$, and $h \in R$ is the height of the floor or obstacle.

From this map a navigation grid *NAV* is computed storing refined floor types and distances to nearest obstacles:

$$NAV : (x, y) \quad \mapsto \quad (t', d) \tag{3}$$

where $t' \in T' = T \cup \{stairs, border\}$ and $d \in R$ is the clearance, i.e. the distance to the closest obstacle for $t' \in \{floor, stairs\}$. *NAV* refines the type of $FOG(x, y) = (t, h)$ for $t = floor$ in the following way. Let $U(x, y)$ be a neighborhood around position (x, y) and $\Delta\hat{h}$ be the maximum absolute difference in floor height to cells in $U(x, y)$:

$$\Delta\hat{h} = \max_{(x',y') \in U(x,y), FOG(x',y') = (floor, h')} |h - h'|. \tag{4}$$

The refined type is then computed as:

$$refine(floor) = \begin{cases} floor, & \Delta\hat{h} \le d_{floor} \\ stairs, & d_{floor} < \Delta\hat{h} \le d_{stairs} \\ border, & d_{stairs} < \Delta\hat{h} \end{cases} \tag{5}$$
$$refine(t) = t, \qquad t \neq floor$$

where d_{floor} is the maximum step height for regular walk and d_{stairs} the corresponding one for climbing up or down stairs.

The refinement of floor cells allows to treat stairs and borders differently from regular floor. We will make use of stairs in path-planning by assigning a different cost for traversal. Borders are treated similar to obstacles and as such prevent the robot from getting close to or over the end of a floor layer.

The clearance of a cell is defined in the following way. For a cell (x, y) with $FOG(x, y) = (floor, h)$ we compute the difference in height $\Delta h(x', y')$ to cell (x', y') with $FOG(x', y') = (t', h'), refine(t') \in \{obstacle, border\}$:

$$\Delta h'(x', y') \quad = \quad h' - h. \tag{6}$$

This height difference then decides which cylinder of our two-cylinder model comes into effect for computing the distance $d'(x', y')$ to this obstacle, or if it can be ignored:

$$d'(x', y') \quad = \quad \begin{cases} e - r_l, & 0 \le \Delta h'(x', y') < h_l \\ e - r_u, & h_l \le \Delta h'(x', y') \le h \\ \infty, & \text{otherwise} \end{cases} \tag{7}$$

where $e = \sqrt{(x' - x)^2 + (y' - y)^2}$ is the Euclidean distance between the cells, and r_l, r_u and h_l, h_u are the radii and heights of the two cylinders as depicted in Figure 2 (b).

[1] It should be possible to relax these assumptions and allow for inclined surfaces. However, the model for representing the world would become more complicated which is why we do not consider them in this paper.

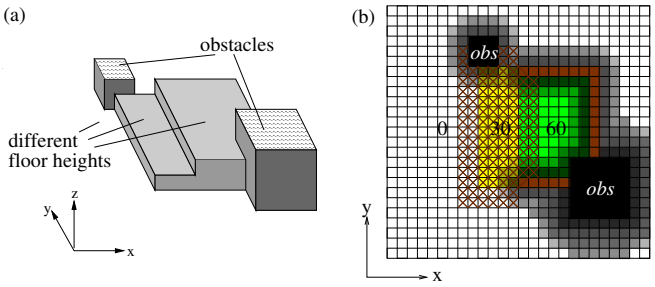


Figure 4: Example navigation map for a simulated world.

The clearance $d(floor)$ of cell (x, y) is then defined as the minimum distance to all obstacles and borders.

$$d(floor) = \min_{(x',y') \in X \times Y, NAV(x',y') = (t', \cdot), t' \in \{obstacle, border\}} d'(x', y') \tag{8}$$

The computation of this value can be implemented efficiently by *region growing* of obstacles and borders until a preset maximum distance is reached.

Note that (8) only applies for cells where $NAV(x, y) = (floor, \cdot)$. For other types $t \in T$ the clearances are set to:

$$d(obstacle) = -r_l, \qquad d(unknown) = \infty. \tag{9}$$

For the robot being able to stand at a given position (x, y) with $NAV(x, y) = (\cdot, d)$, the clearance d must be positive in order not to collide with an obstacle. For unknown terrain we are optimistic in that there could be a floor height well above all obstacles and thus ignore obstacles.

We can define the free configuration space C_{free} as

$$C_{free} = \{(x, y, \theta) \in C \mid NAV(x, y) = (\cdot, d), d > 0\}. \tag{10}$$

As an example Figure 4(a) shows a simulated world fully observed by an ideal sensor. The resulting *NAV* grid is shown in Figure 4(b). Floor heights are drawn in white, yellow and green, stairs are marked with brown crosses, the border is filled in brown, and obstacles are black. Different shades of gray show the clearance of cells where brighter cells indicate a larger clearance than darker ones. Note how the taller obstacle in the bottom right *grows* more to the bottom and right of the map (where the distance to the floor is larger than h_l) and less to the higher level floor.

4 Path-Planning

We have now the basic tools for defining the path-planning problem in our domain. Let $c, c' \in C$ be configurations. The transition $c' = succ(c, a)$ for a single action $a \in A$ is given in Figure 1. We define $succ$ for a sequence of actions $a_1 a_2 \ldots a_m$ in a recursive way as

$$succ(c, a_1 a_2 \ldots a_m) = succ(succ(c, a_1), a_2 \ldots a_m). \tag{11}$$

In order to test whether action a is *applicable* in configuration c, we check the type and clearance of c in *NAV*:

$$app(c, a) = allow(c, a) \wedge \tag{12}$$
$$clear(c, r_a) \wedge clear(succ(c, a), r_a)$$

where $allow(c, a)$ forbids certain actions depending on the type t' in *NAV* at the position of c (e.g. we do not allow backward and sideways walk in unknown terrain and only allow

forward walk on stairs), r_a is additional space required for execution of a (see Figure 3), and

$$clear(c, r) = d > r, \quad c = (x, y, \cdot), NAV(x, y) = (\cdot, d). \quad (13)$$

We also define *app* on a sequence of actions $a_1 a_2 \dots a_m$

$$app(c, a_1 a_2 \dots a_m) = app(c, a_1) \wedge app(succ(c, a_1), a_2 \dots a_m). \quad (14)$$

and call the sequence a *path* if (14) holds.

The problem of path-planning for given start and goal configurations $c_{start}, c_{goal} \in C_{free}$ is then to find a path $\pi = a_1 a_2 \dots a_m$ such that

$$c_{goal} = succ(c_{start}, \pi). \quad (15)$$

Problems for which a path exists are called *solvable*. In general there can be more than one solution for a given solvable problem. We are interested in an *optimal* path, i.e. a path $\hat{\pi}$ that minimizes a cost function g over the path:

$$\hat{\pi} = \underset{\pi}{\operatorname{argmin}}\ g(c_{start}, \pi). \quad (16)$$

In our system we found the following cost function valuable:

$$g(c, a_1) = g_a(a_1) + g_{t'}(c') + g_o(c') \quad (17)$$

$$g(c, a_1 a_2 \dots a_m) = g(c, a_1) + g_c(a_1, a_2) + g(c', a_2 \dots a_m) \quad (18)$$

where $c' = succ(c, a_1)$, g_a is a cost depending on the action (e.g. forward walk has less cost than sideways which takes more time), $g_{t'}$ a cost depending on the type t' in the *NAV* grid ($floor$ is preferred over $stairs$ and $unknown$), g_c a constant for changes in action (the robot might have to do additional foot steps), and g_o considers the clearance of c' in *NAV* in order to prefer safer paths more distant from obstacles:

$$g_o(c) = \hat{g}_o \max(d_{max} - d, 0). \quad (19)$$

Here d_{max} is the maximum clearance above which no further improvement in safety is expected, d is defined as in (13), and $\hat{g}_o$ is a predefined constant.

We employ the well-known A* algorithm for finding solutions. A* maintains a *priority queue* where candidates are ranked according to an evaluation function f composed of the cost function g and an estimated *heuristic* h to c_{goal}:

$$f(c_{start}, \pi) = g(c_{start}, \pi) + h(succ(c_{start}, \pi)). \quad (20)$$

For our chosen discretization of C we can employ the following heuristic which is *optimistic* and a larger cost estimate than the Euclidean distance:

$$h(c) = \hat{g}_a \cdot \begin{cases} \Delta x - \Delta y + \sqrt{2}\Delta y, & \Delta x > \Delta y \\ \Delta y - \Delta x + \sqrt{2}\Delta x, & \Delta x \le \Delta y \end{cases} \quad (21)$$

where $\Delta x = |x - x_g|$ and $\Delta y = |y - y_g|$ are absolute coordinate differences from $c = (x, y, \cdot)$ to $c_{goal} = (x_g, y_g, \cdot)$ and $\hat{g}_a$ is the minimum action cost.

From the properties of the A* algorithm and our chosen cost and heuristic functions it follows that A* finds the cost optimal path $\hat{\pi}$ according to (16) if a solution exists.

An important factor for the efficiency of A* is to detect whether a configuration has already been visited before. By using a regular grid, we can pre-compute the search graph (but not the optimal paths since the environment is unknown) and check for duplicates in constant time. This is an advantage over methods that build the tree at the time of search as checking for duplicates needs further processing e.g. by applying nearest-neighbor methods.

Robot specifications and grid dimensions

r_l = 60mm	h_l = 100mm	d_{floor} = 15mm	cs = 40mm
r_u =140mm	r_h = 500mm	d_{stairs} = 50mm	n = 100

Allowed actions depending on *NAV* type

$allow(c, a)$	$floor$	$unknown$	$stairs$	$border$	$obstacle$
$forward$	**true**	**true**	**true**	false	false
$turn$	**true**	**true**	false	false	false
$sideways$	**true**	false	false	false	false
$backward$	**true**	false	false	false	false

Additional clearance r_a depending on action

a	$forward$	$turn$	$sideways$	$backward$
r_a	60mm	80mm	0	60mm

Cost of action $g_a(a)$ depending on orientation

$g_a(a)$	$forward$	$turn$	$sideways$	$backward$
0°	1	1.1	1.3	2
45°	$\sqrt{2}$	1.1	$1.3\sqrt{2}$	$2\sqrt{2}$

Cost of action $a_{t'}$ depending on *NAV* type

	$floor$	$unknown$	$stairs$	$border$	$obstacle$
$g_{t'}(c)$	0	0.5	1	∞	∞

Other constants for cost function

$g_c(a, a)$ = 0	$g_c(a_1, a_2)$ = 0.25,	$a_1 \neq a_2$
d_{max} = 200mm	$\hat{g}_o = 3/d_{max}$	$\hat{g}_a = 1$

Table 1: Parameters used in our implementation

5 Results

We implemented our approach on QRIO, Sony's small humanoid robot with 38 degrees of freedom [Fujita *et al.*, 2003]. The robot is equipped with 3 MIPS R5000 CPUs clocked at 400MHz and a stereo camera system with a 45° field of view providing disparity images at 12.5 fps. This stereo data is segmented into planes from which our FOG and *NAV* grid representations are updated in real-time.

Table 1 lists parameters used in our implementation for all variables introduced in the previous sections. For stairs we restrict actions to forward walk only. Thus, the robot only finds paths on straight staircases unless there is a larger floor area where turns are possible. We forbid sideways and backward walk in unknown terrain because the robot is not able to see into its moving direction. We set a high cost for backward motion since it is generally more unsafe. Costs are added for walking in unknown terrain and on stairs which focuses the algorithm to search for a path on known floor even if the distance traveled is larger than when moving on other terrain.

For our experiments we built a 4 meters long and 1 meter wide stage where several obstacles, a 4cm high sill, a staircase with 4 steps each 3cm high, and a higher-level platform were arranged as shown in Figure 5.

We placed QRIO at the left end of the stage and let it observe the environment by rotating on the spot. We then commanded the robot to find and follow a path to a position 3 meters ahead (the center of the higher-level platform). Our path-planning system continuously evaluates the current *NAV*



Figure 5: Experimental setup. Several obstacles and a sill are placed on a stage. A staircase leads to a higher-level platform.

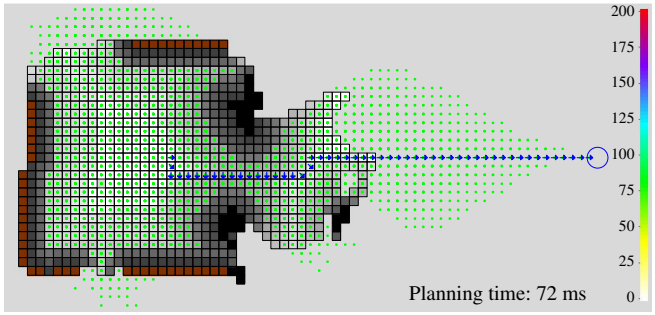


Figure 6: Path-planning in narrow environment: the robot decides to use sideways walk for passing through the obstacles.

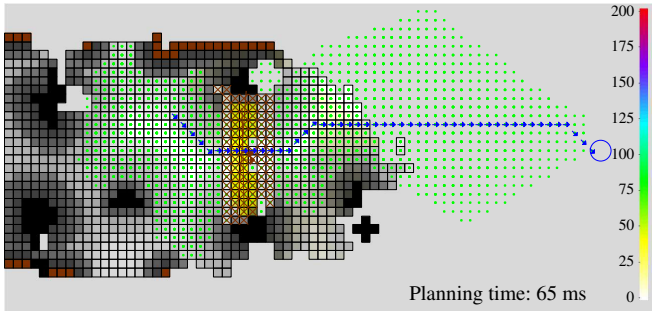


Figure 7: Path-planning over a sill: the robot finds a straight path over the sill and then moves around the next obstacle.

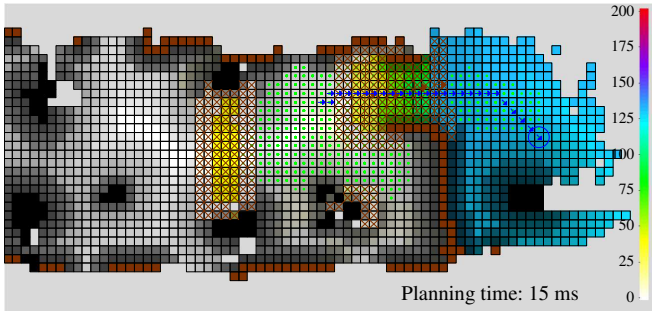


Figure 8: Path-planning to the higher platform: first the robot aligns with the staircase and then climbs up in a straight way.

grid and sends paths to a *path-following* system which computes the next two foot steps and feeds them to the robot's motion control system. Figures 6–8 show 3 snapshots of the robot while moving on the stage. We use the same encoding for floor, border, stairs and obstacles as before. The color indicator on the right shows the floor height as estimated by the robot. Unknown space is left blank. Green dots mark the search space of the A* algorithm while the path is shown with blue arrows. Brown bars on the path indicate stair transitions.

In the beginning the system has to find a path leading between two close obstacles (see Figure 6). Since space between the obstacles is narrow, forward walk is not possible due to the lack of the required clearance $r_{forward}$. Therefore, the system decides to walk sideways between the obstacles.

After passing through the narrow passage, the robot plans a path over the sill in the center of the stage (see Figure 7). For overcoming the sill, the system employs a separately developed *stair-climbing* module [Gutmann *et al.*, 2004].

Finally, as the robot walks closer to the higher platform, it finds a path on the staircase to the target location (Figure 8).

The time required for finding a path on our fixed-size configuration space depends on the environment and the path length. In the examples above the time for planning was always below 100 ms on the robot's embedded CPU. In order to investigate the run-time dependency on path length we randomly selected goal configurations in the final map after the robot reached its target. If a path to the goal could be found, we recorded the length of the path and the time needed for computation. Figure 9 shows the runtime versus path length measured on a Pentium III, 1.4GHz computer. Although the curve shows a slope at least quadratic in path length, for our chosen dimensions of C the runtime increases only slowly and does not exceed 40 ms even for longer paths.

The worst case run-time occurs when the goal is placed at an unreachable location, e.g. a free position completely surrounded by obstacles. In this case all but a few of the $8n^2$ = 80,000 configurations have to be examined. The robot's embedded processor needs close to 900 ms for exploring this search space (110 ms on a Pentium III, 1.4GHz).

6 Conclusion

We presented and evaluated a path-planning algorithm that provides collision free trajectories of up to a few meters for a humanoid robot in a layered environment in real time.

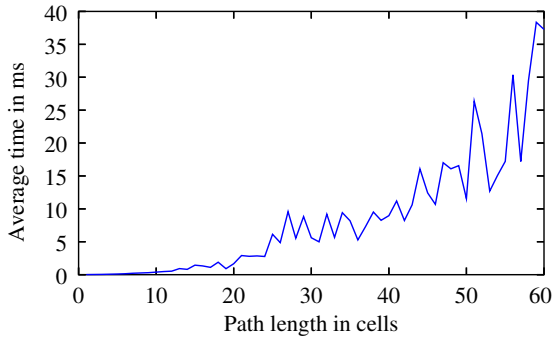


Figure 9: Time for path-planning depending on path length.

In our approach unknown terrain is treated in an optimistic way, i.e. their clearing is assumed arbitrary large. As the robot moves closer to unknown area and observes the situation, the navigation map is refined and a new path is planned automatically. This allows navigation in unknown environments.

Our perception system distinguishes between floor, border, stairs and obstacles and interprets them in a meaningful way for path planning. We believe our approach advances previous methods on simulated worlds, and real robot systems with mainly reactive, short distance navigation capabilities.

The limitations of our approach are various. Our discretization onto a grid is an approximate cell decomposition and as such is only complete up to the resolution of the grid. Furthermore, using only 8 orientations can prevent finding paths in certain environments, for example in a narrow corridor oriented at an angle of 20°.

Our approach may be extended to 2^m orientations $(m > 3)$ by changing cell size and actions such that repeated application of a turn action describes a curve with the minimum turn radius of the robot. Other actions might move the robot over more than one cell in order to align with the grid.

Our set of actions and the navigation map do not consider the possibility of stepping over an obstacle. We believe that for such a capability, a foot step planner using precise 3D information is of importance. However, our approach could be combined with such a system by marking obstacles the robot can step over similar to as we mark stairs.

At current, only straight paths are allowed on stairs. By introducing new actions *left-forward* and *right-forward* we can also find paths on spiral staircases at the cost of a higher runtime. In practice, we observed an increase of about 25% in run-time for the worst case scenario described in Section 5 when adding these two actions.

We think it is also possible to extend our approach to find areas where the robot can crawl underneath an obstacle. Our current floor and obstacle map and the navigation grid, however, do not support such a detection yet and have to be extended first. We will follow this direction in the future.

Acknowledgment

We would like to thank Stefan Edelkamp, Thilo Weigel and the anonymous reviewers for their valuable comments. We also thank all developers of the Sony QRIO robot for making this work possible.

References

[Buhmann *et al.*, 1995] J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The mobile robot RHINO. *AI Magazine*, 16(2):31–38, 1995.

[Chestnutt and Kuffner, 2004] J. Chestnutt and J.J. Kuffner. A tiered planning strategy for biped navigation. In *Int. Conf. on Humanoid Robotics (Humanoids)*, 2004.

[Fujita *et al.*, 2003] M. Fujita, Y. Kuroki, T. Ishida, and T.T. Doi. A small humanoid robot SDR-4X for entertainment applications. In *Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, Kobe, Japan, 2003.

[Guan *et al.*, 2004] Y. Guan, K. Yokoi, N.E. Sian, and K. Tanie. Feasibility of humanoid robots stepping over obstacles. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[Gutmann *et al.*, 2004] J.-S. Gutmann, M. Fukuchi, and M. Fujita. Stair climbing for humanoid robots using stereo vision. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[Gutmann *et al.*, 2005] J.-S. Gutmann, M. Fukuchi, and M. Fujita. A floor and obstacle height map for 3D navigation of a humanoid robot. In *Int. Conf. on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[Hirai *et al.*, 1998] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of Honda humanoid robot. In *Int. Conf. on Robotics and Automation (ICRA)*, 1998.

[Kanehiro *et al.*, 2004] F. Kanehiro, H. Hirukawa, K. Kaneko, S. Kajita, K. Fujiwara, K. Harada, and K. Yokoi. Loclomotion planning of humanoid robots to pass through narrow spaces. In *Int. Conf. on Robotics and Automation (ICRA)*, New Orleans, 2004.

[Kavraki *et al.*, 1996] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), 1996.

[Kuffner *et al.*, 2002] J.J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

[Latombe, 1991] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[Li *et al.*, 2003] T.-Y. Li, P.-F. Chen, and P.-Z. Huang. Motion planning for humanoid walking in a layered environment. In *Int. Conf. Robotics and Automation (ICRA)*, 2003.

[Lorch *et al.*, 2002] O. Lorch, A. Albert, J. Denk, M. Gerecke, R. Cupec, J. F. Seara, W. Gerth, and G. Schmidt. Experiments in vision-guided biped walking. In *Int. Conf. Intelligent Robots and Systems (IROS)*, 2002.

[Okada *et al.*, 2003] K. Okada, M. Inaba, and H. Inoue. Walking navigation system of humanoid robot using stereo vision based floor recognition and path planning with multi-layered body image. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, October 2003.

[Sabe *et al.*, 2004] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *Int. Conf. on Robotics and Automation (ICRA)*, New Orleans, 2004.

[Shiller *et al.*, 2001] Z. Shiller, K. Yamane, and Y. Nakamura. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. In *Int. Conf. on Robotics and Automation (ICRA)*, Seoul, Korea, 2001.

[Stentz, 1995] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. *Int. Journal of Robotics and Automation*, 10(3), 1995.

LRTA*(k) *

Carlos Hernández and **Pedro Meseguer**
Institut d'Investigació en Intel.ligència Artificial, CSIC
Campus UAB, 08193 Bellaterra, Spain
{chernan,pedro}@iiia.csic.es

Abstract

LRTA* is a real-time heuristic search algorithm widely used. In each iteration it updates the heuristic estimate of the current state. Here we present LRTA*(k), a new LRTA*-based algorithm that is able to update the heuristic estimates of up to k states, not necessarily distinct. Based on bounded propagation, this updating strategy maintains heuristic admissibility, so the new algorithm keeps the good theoretical properties of LRTA*. Experimentally, we show that LRTA*(k) produces better solutions in the first trial and converges faster when compared with other state-of-the-art algorithms on benchmarks for real-time search.

1 Introduction

LRTA* [Korf, 1990] is a real-time heuristic search algorithm that interleaves planning and action execution in an on-line manner. This allows LRTA* to face search problems on unknown or changing environments, tasks that are not easily performed by off-line search. This algorithm works on a search space where every state x has a heuristic estimate $h(x)$ of the cost from x to a goal. It is complete under a set of reasonable assumptions. In addition, LRTA* improves its performance over successive trials on the same problem, by recording better heuristic estimates. If $h(x)$ is admissible, after a number of trials $h(x)$ converges to their exact values along every optimal path. At this point, only optimal paths are traversed by LRTA*. Before convergence improvement is not monotonic: after a near optimal path LRTA* could find a worse path [Shimbo and Ishida, 2003].

In this paper we present LRTA*(k), an algorithm based on LRTA* with its same structure. The only difference lies in the updating strategy. LRTA* updates the heuristic estimate of a single state per iteration. LRTA*(k) updates the heuristic estimate of up to k, not necessarily distinct, states per iteration following a bounded propagation strategy. This updating maintains heuristic admissibility, so LRTA*(k) converges to exact heuristic values in optimal paths, just like LRTA*. In fact, LRTA* is a particular case of LRTA*(k) with $k = 1$.

*Partially supported by the Spanish REPLI project TIC-2002-04470-C03-03.

LRTA*(k) propagates the cost discovered by lookahead up to k states. Because of that, better heuristic estimates are propagated, causing the following benefits:

- First solution. The quality of the first solution is important for real-time search algorithms. If the first solution would involve no cycles, LRTA*(k) would behave like LRTA*. However, this rarely happens. When LRTA*(k) revisits a state, it finds a better heuristic estimate than the one LRTA* would have found, selecting a better action. Experimentally, LRTA*(k) finds shorter solutions in less computation time than LRTA*.
- Convergence. LRTA*(k) records heuristic estimates that are closer to their exact values than those recorded by LRTA*. This causes LRTA*(k) to converge faster than LRTA* (in number of steps, in number of trials and also in total CPU time) in the benchmarks tested. This also happens for other algorithms (FALCONS).
- Solution stability. Getting generally better quality solutions causes less differences between solutions and increases stability. This was not initially pursued (we were not bounding how bad a solution could be) but it has appeared as a side-effect of improving solution quality.

These benefits come at the cost of extra computation per planning step, required for bounded propagation. Limiting computation up to k updates bounds this extra cost, which could be adjusted (varying k) to the requirements of each application. In our experiments, the time per planning step has remained reasonable when compared with LRTA*.

The structure of the paper is as follows. In Section 2 we summarize related approaches. In Section 3 we present the basic elements of LRTA*. In Section 4 we describe LRTA*(k) and its new updating strategy, using the concepts of bounded propagation and support. We provide experimental results in Section 5, showing the benefits of our approach on classical real-time search benchmarks. Finally, Section 6 contains some conclusions and lines for further research.

2 Related Work

In his seminal work, Korf proposed LRTA* and RTA* (an algorithm with a different updating strategy, able to find better solutions in the first trial but without converging to optimal routes) [Korf, 1990]. After that, several approaches have

been made to to improve LRTA* on the quality of the first solution, convergence and stability. HLRTA* [Thorpe, 1994] is a hybrid between RTA* and LRTA*. It finds better solutions than LRTA* in the first trial and converges to optimal paths. As RTA*, it avoids the ping-pong effect [Edelkamp and Eckerle, 1997] but requires more memory. The weighted and bounded versions of LRTA* [Shimbo and Ishida, 2003] speed up convergence and improve solution stability, but sacrifice optimality. FALCONS [Furcy and Koenig, 2000] accelerates convergence and keeps optimality by using $g(x) + h(x)$ as heuristic function, where $g(x)$ is the cost from the start state to x. FALCONS improves convergence but it may perform a large amount of exploration in earlier trials. eFALCONS [Furcy and Koenig, 2001] is a hybrid between HLRTA* and FALCONS. It converges as FALCONS, performing a smaller amount of actions in earlier trials, although it may be greater than LRTA*. A new version of LRTA* [Koenig, 2004] improves convergence by increasing lookahead depth, causing to increase the planning time per step.$\gamma-$Trap [Bulitko, 2004] uses adaptive lookahead depth and offers control on the exploration vs. exploitation trade-off, but sacrifices optimality. Other approaches consider search with moving target [Ishida and Korf, 1991] and cooperative agents [Knight, 1993], [Goldenberg *et al.*, 2003].

3 LRTA*

We assume a state space defined by the tuple (X, A, c, s, G), where (X, A) is a finite graph, $c : A \mapsto [0, \infty)$ is a cost function that associates each arc with a finite cost, s is the start state, and $G \subset X$ is the set of goal states. X is a finite set of states, and $A \subset X \times X - \{(x, x)|x \in X\}$ is a finite set of arcs. Each arc (v, w) represents an action whose execution causes the agent to move from v to w. The state space is undirected, that is, for any action $(x, y) \in A$ there exists its inverse $(y, x) \in A$ with the same cost $c(x, y) = c(y, x)$. The successors of a state x are $Succ(x) = \{y|(x, y) \in A\}$. A path $(x_0, x_1, x_2, \ldots)$ is a sequence of states such that every pair $(x_i, x_{i+1}) \in A$. The cost of a path is the sum of costs of the actions in that path.

A heuristic function $h : X \mapsto [0, \infty)$ associates with each state x an approximation $h(x)$ of the cost of a path from x to a goal. $h(x)$ is called the heuristic estimate of x. The exact cost $h^*(x)$ is the minimum cost to go from x to a goal. If $\forall x \in X$, $h(x) \leq h^*(x)$ then h is admissible. A path $(x_0, x_1, \ldots, x_n)$ such that $h(x_i) = h^*(x_i)$, $0 \leq i \leq n$ is called optimal.

The LRTA* algorithm with lookahead at depth 1 (the case considered in this paper) and converging to optimal paths (with h admissible) appears in Figure 1. Like in [Korf, 1990], we assume the existence of $Succ$ and h_0 functions, which when applied to a state x generate its set of successors and its initial heuristic estimate, respectively. Procedure `LRTA*` initializes the heuristic estimate of every state using the function h_0, and repeats the execution of `LRTA*-trial` until convergence (h does not change). At this point, an optimal path has been found. Procedure `LRTA*-trial` initializes the current state x with the start s. Then, the following loop is executed until a goal is found. First, function `LookaheadUpdate1` is executed ignoring its result. It

```
procedure LRTA*(X, A, c, s, G)
  for each x ∈ X do h(x) ← h0(x);          /* initialization */
  repeat
    LRTA*-trial(X, A, c, s, G);
    until h does not change;

procedure LRTA*-trial(X, A, c, s, G)
  x ← s;
  while x ∉ G do
    dummy ← LookaheadUpdate1(x);           /* look+upt */
    y ← argmin_{w∈Succ(x)}[c(x, w) + h(w)]; /* next state */
    execute(a ∈ A such that a = (x, y)); /* action exec */
    x ← y;

function LookaheadUpdate1(x): boolean;
  y ← argmin_{v∈Succ(x)}[c(x, v) + h(v)];  /* lookahead */
  if h(x) < c(x, y) + h(y) then
    h(x) ← c(x, y) + h(y); return true;
  else return false;
```

Figure 1: The LRTA* algorithm.

performs lookahead from x at depth 1, updating its heuristic estimate accordingly. Second, the state y of $Succ(x)$ with minimum value of $c(x, y) + h(y)$ is selected as next state (breaking ties randomly). Third, an action that passes from x to y is executed. After it, y is the new current state and the loop iterates.

Function `LookaheadUpdate1` performs lookahead from x at depth 1, and updates $h(x)$ if it is lower than the minimum cost of moving from x to one of its successors y plus its heuristic estimate $h(y)$. If $h(x)$ changes it returns $true$ otherwise returns $false$.

In a state space like the one assumed here (finite, positive costs, finite heuristic estimates) where from every state there is a path to a goal it has been proved that LRTA* is complete. In addition, if h is admissible, over repeated trials the heuristic estimates eventually converge to their exact values along every optimal path [Korf, 1990].

4 LTRA*(k)

4.1 Propagation of Changes of Heuristic Estimates

LRTA*(k) differs from LRTA* in the *propagation* of changes of heuristic estimates, which is explained in the following. For ease of presentation, first we describe unbounded propagation, and second we present bounded propagation, which is the one used in LRTA*(k).

Unbounded propagation works as follows. If it happens that the current state heuristic is updated by the effect of lookahead, LRTA* selects the next state and executes the action that moves to that state, but it does not perform further updates. The current iteration stops and future iterations will be in charge of propagating that change (if needed). Unbounded propagation propagates this change to the successors of the current state. Each successor is taken as the basis for a new lookahead, and its heuristic is updated accordingly. If this propagation causes further changes, the successors of the states whose heuristic have changed will be considered for further propagation, and so forth. The process iterates until no further changes are performed.

To illustrate unbounded propagation, let us consider the example of a 4×4 grid that appear in Figure 2, where m is the start state and p is the goal. Initial heuristic estimates are computed by Manhattan distance, and obstacles are represented by "-". From state m, LRTA* performs lookahead, updates $h(m)$ to 5 (assuming that costs to move to successor states are equal to 1), and moves to i, terminating the current iteration. Unbounded propagation performs further propagation of the $h(m)$ change. The successor of m is i, from which lookahead is performed and $h(i)$ is updated to 6. Again, this change causes to reconsider the successors of i, e and m. $h(e)$ is updated to 7, which causes to reconsider its successors a and i. $h(m)$ is updated to 7, which causes to reconsider its successor i. a is reconsidered but its heuristic estimate does not change. $h(i)$ is updated to 8, which causes to reconsider its successors e and m. $h(e)$ does not change. $h(m)$ is updated to 9, which causes to reconsider its successor i. $h(i)$ does not change. Unbounded propagation stops. This propagation presents some drawbacks for real-time search,

1. Move in bounded time [Koenig, 2001][Koenig, 2004]. The number of states involved in unbounded propagation may differ in consecutive steps, so the computational effort required may change between steps. This is against the requirement that real-time search must perform individual moves in bounded (appr. constant) time.
2. Acting on vicinity. Unbounded propagation can act very far from the current state. This violates a basic assumption of real-time search, that lookahead and update operations can only be done in the vicinity of the current state [Shimbo and Ishida, 2003].

To solve these drawbacks, we propose to limit unbounded propagation. This new version, called *bounded* propagation, limits to k the maximum number of states that can be updated in one step. In this way, the computational effort devoted to propagation is bounded, and individual moves are performed within that bound. This is acceptable as far as the agent could perform the extra computation required between successive actions without disturbing its activity.

In addition, we require another condition: updating is limited to previously expanded states. In other words, only states belonging to the path from the start to the current state can be updated. This condition tries to prevent that heuristic updating to be scattered over the search space. Combined with the condition of bounded propagation, the vicinity for updating is limited to k previously visited states.

To illustrate bounded propagation updating expanded states, the example of Figure 2 is executed with $k = 5$. State m is updated and $h(m)$ changes to 5. Since there are no previous states in the path, no propagation is made. In this iteration, 1 state (m) has been considered for update. The agent moves to i, which is updated, $h(i)$ changes to 6 and state m is reconsidered (the only successor in the path). m is updated and $h(m)$ changes to 7. This causes to reconsider state i, also in the path, but this causes no changes. In this iteration, 3 states (i, m, i) have been considered for update. The agent moves to e, which is updated, $h(e)$ changes to 7 and state i is reconsidered. $h(i)$ changes to 8, and its successors in the path, e and m are reconsidered. e causes no changes. m is

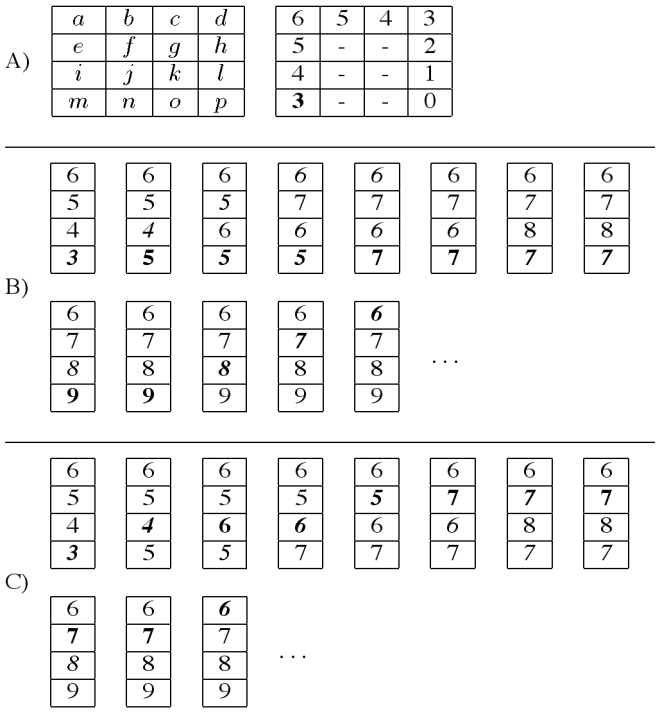


Figure 2: A 4×4 grid. (A) States (m is the start and p is the goal) and initial heuristic estimates (Manhattan distance). Obstacles are represented by "-". A number in **bold** indicates the heuristic estimate of the state where the agent is located, while numbers in *italics* represent states whose heuristic estimates should be reconsidered. (B) LRTA* with unbounded propagation in the leftmost column, until the agent reaches state a. (C) LRTA* with bounded propagation ($k = 5$) and updating previously visited states in the leftmost column, until the agent reaches state a.

updated, $h(m)$ changes to 9 and causes its successor i in the path to be reconsidered. i causes no change. In this iteration, 5 states (e, i, e, m, i) have been considered for update. At this point, agent moves to a and continues towards the goal p.

Bounded propagation depends on k, the maximum number of previous states that can be updated. We believe that bounded propagation is a suitable strategy for real-time search. This is confirmed by the experiments of Section 5 on different problems with several k values.

4.2 The Algorithm

The LRTA*(k) algorithm appears in Figure 3. Procedure `LRTA*(k)-trial` is very similar to procedure `LRTA*-trial` of Figure 1. They only differ in two points: `LRTA*(k)-trial` records the sequence of expanded states in $path$, and it executes the procedure `LookaheadUpdateK` instead of the function `LookaheadUpdate1`. Procedure `LookaheadUpdateK` performs bounded propagation as follows. It maintains a sequence Q of states candidates to update their heuristic estimates. Q is initialized with the current state. At most, k states will be entered in Q. This is controlled by the counter $cont$, initialized to $k-1$. Then, the following loop is executed until

```
procedure LRTA*(k)(X, A, c, s, G, k)
  for each x ∈ X do h(x) ← h_0(x);
  repeat
    LRTA*(k)-trial(X, A, c, s, G, k);
    until h does not change;

procedure LRTA*(k)-trial(X, A, c, s, G, k)
  x ← s; path ← ⟨s⟩;
  while x ∉ G do
    LookaheadUpdateK(x, k, path);
    y ← argmin_{w∈Succ(x)}[c(x, w) + h(w)];
    execute(a ∈ A such that a = (x, y));
    path ← add-last(path, y); x ← y;

procedure LookaheadUpdateK(x, k, path)
  Q ← ⟨x⟩; cont ← k − 1;
  while Q ≠ ∅ do
    v ← extract-first(Q);
    if LookaheadUpdate1(v) then
      for each w ∈ Succ(v) do
        if w ∈ path ∧ cont > 0 then
          Q ← add-last(Q, w); cont ← cont − 1;

function LookaheadUpdate1(x): boolean;
  y ← argmin_{v∈Succ(x)}[c(x, v) + h(v)];
  if h(x) < c(x, y) + h(y) then
    h(x) ← c(x, y) + h(y); return true;
  else return false;
```

Figure 3: The LRTA*(k) algorithm.

Q contains no states. The first state v in Q is extracted, from which lookahead is performed and it is updated accordingly. If $h(v)$ changes (`LookaheadUpdate1` of Figure 1 returns $true$), this is propagated over its successors as follows. Any w successor of v that belongs to $path$ enters Q in the last position, provided that there is still room in Q (the limit of k states has not been exhausted during the current execution of the procedure). If $h(v)$ does not change, the loop iterates processing the next state of Q. The admissibility of h after bounded propagation is guaranteed by the following lemma.

Lemma 1 *(Lemma 2 of [Edelkamp and Eckerle, 1997]). Let $x \in X - G$ and h admissible. Updating $h(x) \leftarrow max(h(x), min_{v \in Succ(x)}(c(x, v) + h(v))$ implies that $h(x) \leq h^*(x)$.*

Proof. If $h(x) \geq min_{v \in Succ(x)}(c(x, v) + h(v))$ there is nothing to prove. Otherwise, there is an optimal path from x to a goal that passes through a successor w. Then, $h^*(x) = c(x, w) + h^*(w) \geq c(x, w) + h(w)$. In particular, this is true for the minimum $c(x, v) + h(v)$ among successors of x, that is, $h^*(x) \geq min_{v \in Succ(x)}(c(x, v) + h(v))$. So $h(x) \leq h^*(x)$.

From this result, convergence of LRTA*(k) to optimal paths is guaranteed in the same terms as LRTA*, because Theorem 3 of [Korf, 1990] is also valid for LRTA*(k). The new algorithm inherits the good properties of LRTA*.

4.3 Supports

Algorithm LRTA*(k) can easily be improved using the notion of support of an heuristic estimate. Imagine the situation depicted in Figure 5, where each state has four successors (north, south, east, west) and the cost to move to any

```
procedure LRTA*(k)-trial(X, A, c, s, G, k)
  for each x ∈ X do supp(x) ← null;
  x ← s; path ← ⟨s⟩;
  while x ∉ G do
    LookaheadUpdateK(x, k, path);
    y ← argmin_{w∈Succ(x)}[c(x, w) + h(w)];
    execute(a ∈ A such that a = (x, y));
    path ← add-last(path, y); x ← y;

procedure LookaheadUpdateK(x, k, path)
  Q ← ⟨x⟩; cont ← k − 1;
  while Q ≠ ∅ do
    v ← extract-first(Q);
    if LookaheadUpdate1(v) then
      for each w ∈ Succ(v) do
        if w ∈ path ∧ cont > 0 ∧ v = supp(w) then
          Q ← add-last(Q, w); cont ← cont − 1;

function LookaheadUpdate1(x): boolean;
  y ← argmin_{v∈Succ(x)}[c(x, v) + h(v)]; supp(x) ← y;
  if h(x) < c(x, y) + h(y) then
    h(x) ← c(x, y) + h(y); return true;
  else return false;
```

Figure 4: Adding supports to LRTA*(k).

of them is 1. e is the current state and the path is the sequence $(\ldots, b, c, e)$. If it happens that $h(e)$ changes from 3 to 5, LRTA*$(k > 1)$ will reconsider the heuristic estimate of state c. However, no matter the change of $h(e)$, $h(c)$ will not change because the minimum of its successors, state b, has not changed its heuristic estimate. The value of $h(b)$ justifies the current value of $h(c)$. State b is called a support for $h(c)$.

Formally, state y is *support* of $h(x)$, written $y = sup(x)$, iff $y = argmin_{v \in Succ(x)}(c(x, v) + h(v))$. The previous example illustrates a simple property of bounded propagation: if state y changes its heuristic estimate, only those states x successors of y such that y is their support could change its heuristic estimate. A successor state z not supported by y will not change: z is supported by other state and as far this state does not change its heuristic estimate, z will not change.

LRTA*(k) can benefit from this property, by entering in Q only those states which are supported by the state that has changed its heuristic estimate. This requires minor modifications that appear in Figure 5. In `LookaheadUpdate1`, each time a state is expanded its support is recorded. In `LookaheadUpdateK`, only those states supported by the state who has changed are considered for inclusion in Q. The use of supports requires a table that records for each expanded state its corresponding support. This duplicates the memory requirements of LRTA*.

	a	
b	c	d
	e	

	6	
3	4	5
	$3 \rightarrow 5$	

Figure 5: A state space (left) and its heuristic estimates (right). The path followed by the agent is $(\ldots, b, c, e)$, e is the current state and $h(e)$ changes from 3 to 5. State c does not change because its support (state b) has not changed.

Grid35			
k	Cost%	States%	Time/Step%
	68755.6=100%	7014.3=100%	0.00034=100%
1 (LRTA*)	100%	100%	100%
6	24%	89%	181%
15	17%	100%	237%
51	15%	117%	324%
500	**14%**	133%	523%
∞	**14%**	86%	754%
RTA*	45%	**66%**	**91%**
FALCONS	1402%	183%	126%
Grid70			
k	Cost%	States%	Time/Step%
	146382.3=100%	1447.1=100%	0.00034=100%
1 (LRTA*)	100%	100%	100%
6	34%	**99%**	169%
15	18%	100%	222%
51	9%	102%	328%
500	4%	104%	708%
∞	**2%**	108%	4668%
RTA*	29%	101%	**97%**
FALCONS	53%	103%	105%
Maze			
k	Cost%	States%	Time/Step%
	588001.0=100%	8218.7=100%	0.00035=100%
1 (LRTA*)	100%	100%	100%
6	29%	93%	171%
15	16%	90%	231%
51	8%	**88%**	387%
500	5%	89%	914%
∞	**2%**	89%	7683%
RTA*	6%	**88%**	**96%**
FALCONS	15%	129%	106%

Table 1: Results for the first trial on Grid35, Grid70 and Maze benchmarks, averaged over 1000 instances.

Grid35				
k	Cost%	Trials%	MaxStates%	Time/Step%
	6493803.3=100%	2561.4=100%	7014.3=100%	0.00036=100%
1 (LRTA*)	100%	100%	100%	**100%**
6	48%	63%	**89%**	163%
15	36%	48%	100%	189%
51	29%	43%	117%	205%
500	26%	**42%**	133%	216%
∞	**25%**	**42%**	136%	223%
FALCONS	62%	26%	245%	122%
Grid70				
k	Cost%	Trials%	MaxStates%	Time/Step%
	790030.4=100%	306.8=100%	1480.7=100%	0.00037=100%
1 (LRTA*)	100%	100%	**100%**	**100%**
6	44%	57%	**100%**	148%
15	27%	37%	103%	179%
51	18%	26%	105%	219%
500	10%	17%	107%	302%
∞	**2%**	**5%**	111%	906%
FALCONS	81%	55%	104%	103%
Maze				
k	Cost%	Trials%	MaxStates%	Time/Step%
	27767147.9=100%	1584.3=100%	11360.7=100%	0.00035=100%
1 (LRTA*)	100%	100%	**100%**	**100%**
6	39%	48%	120%	169%
15	23%	24%	128%	232%
51	11%	11%	128%	360%
500	5%	7%	120%	663%
∞	**2%**	**6%**	116%	1792%
FALCONS	78%	50%	134%	105%

Table 2: Results for convergence on Grid35, Grid70 and Maze benchmarks, averaged over 1000 instances.

5 Experimental Results

We compare the performance of LRTA*(k), for different values of k, with RTA* (first trial only), LRTA* and FALCONS (first trial, convergence and stability). As benchmarks we use four-connected grids where an agent can move one cell north, south, east or west. We use the following benchmarks:

1. Grid35. Grids of size 301x301 with a 35% of obstacles placed randomly. In this type of grid heuristics tend to be only slightly misleading.
2. Grid70. Grids of size 301x301 with a 70% obstacles placed randomly. In this type of grid heuristics could be misleading.
3. Maze. Acyclic mazes of size 181x181 whose corridor structure was generated with depth-first search. In this type of grid heuristics could be very misleading.

In each case, results are averaged over 1000 different instances. In grids of size 301x301 the start and goal state are chosen randomly with the restriction that there is a path from the start state to the goal state. In mazes, the start state is (0,0), and the goal state is (180,180). As initial heuristic between two states we use the Manhattan distance.

Table 1 contains the results for the first trial, in terms of solution cost, number of expanded states and time per step (in milliseconds). The smallest solution cost is obtained by LRTA*(k) with $k = \infty$, that is, propagation runs without limit of states, updating states which are in the path connecting the start with the current state. Comparing with LRTA*, our algorithm produces better solutions for all values of k tested. Because of bounded propagation, LRTA*(k) requires more computation than LRTA* per step. This can be adjusted using the k parameter. If the agent has enough planning time per step, it can use a high k because this largely increases the solution quality. Otherwise, low values of k improve solution quality and differences with LRTA* in planning time per step remain reasonable. With k between 6 and 15, LRTA*(k) uses approximately twice the planning time per step required by LRTA*, decreasing solution cost by a factor between 3 and 5. Comparing with RTA*, our algorithm finds better solutions from $k = 6$ on in Grid35, $k = 15$ on in Grid70 and $k = 500$ on in Maze. Comparing with FALCONS, our algorithm always produces better solutions in Grid35 and Grid70. In Maze, it works better from $k = 51$ on.

We compare the learning process of LRTA*(k) with LRTA* and FALCONS (RTA* is excluded, it does not converge to optimal routes). Table 2 contains the results for convergence to optimal paths, in terms of total cost, number of trials, maximum number of expanded states and planning time per step. Considering solution cost, LRTA*(k) obtains better results than LRTA* and FALCONS for the values of k tested on the three benchmarks. Solution cost decreases monotonically as k increases. We observe that the worse the heuristic information is, the better LRTA*(k) behaves with respect to its competitors (results are better in Maze than in Grid70, and in Grid70 than in Grid35). Considering trials to convergence, LRTA*(k) requires many fewer trials than LRTA*. The number of trials decreases steadily as k increases, from approximately half of the trials with $k = 6$. Comparing with FALCONS, it does not perform better for Grid35, but gets better results in Grid70 and Maze. The good results of LRTA*(k) come at the cost of extra computation,

Grid35					
k	IAE $\times 10^6$	ISE $\times 10^9$	ITAE $\times 10^8$	ITSE $\times 10^{11}$	SOD $\times 10^5$
1 (LRTA*)	5.2	56.5	38.1	130.9	15.9
6	2.3	10.6	11.7	26.3	6.8
15	1.7	7.6	6.5	14.2	5.1
51	1.3	5.8	4.5	8.6	4.1
500	1.2	4.3	3.9	6.5	3.6
∞	1.1	3.7	3.8	6.2	3.4
FALCONS	3.7	3898.1	6.3	88.3	10.6
Grid70					
k	IAE $\times 10^3$	ISE $\times 10^6$	ITAE $\times 10^4$	ITSE $\times 10^7$	SOD $\times 10^2$
1 (LRTA*)	509.1	24227.7	2029.2	9145.2	1174.8
6	188.4	2931.4	702.6	1265.5	592.0
15	112.9	950.8	301.7	501.4	379.7
51	67.0	327.5	119.7	166.0	210.8
500	26.5	75.2	25.0	26.8	65.3
∞	4.0	5.9	2.7	1.1	5.8
FALCONS	480.9	17268.5	4088.3	46665.2	1244.5
Maze					
k	IAE $\times 10^5$	ISE $\times 10^9$	ITAE $\times 10^6$	ITSE $\times 10^{10}$	SOD $\times 10^4$
1 (LRTA*)	221.6	5338.3	6952.1	120297.1	1331.1
6	82.0	403.8	2109.0	4490.6	412.4
15	50.7	200.3	659.2	1242.9	188.8
51	25.1	93.4	157.0	340.7	104.0
500	9.6	28.4	34.8	66.6	48.6
∞	1.5	1.6	4.2	1.6	7.0
FALCONS	189.4	1136.2	7072.6	43695.6	680.8

Table 3: Results for solution stability on Grid35, Grid70 and Maze benchmarks, averaged over 1000 instances.

that is, longer planning time. It is worth noticing that low values of k generate large improvements in solution cost and number of trials, with a limited effect in planning time per step. For instance, with $k = 6$, the total cost to converge to optimal path is divided by a factor a bit higher than 2, the number of trials is divided by a factor of appr. 2, at the cost of increasing the time per step by a factor around 1.6.

In these algorithms the number of expanded states determines the amount of memory used. LRTA*(k) records for each expanded state its heuristic estimate and its support, while LRTA* records its heuristic estimate only. This means that LRTA*(k) requires twice the memory used by LRTA* per expanded state. FALCONS records for each expanded state the values of g and h, so it has similar requirements as LRTA*(k) per expanded state. On the benchmarks tested, all the algorithms expand a relatively close number of states (except FALCONS in Grid35, that requires a larger amount). Roughly speaking, we can say that LRTA*(k) requires from two to two and a half times the memory required by LRTA* on the same instances, and a similar (and in some cases larger) amount of the memory is required by FALCONS.

To measure solution stability we computed the indices IAE, ISE, ITAE, ITSE, and SOD [Shimbo and Ishida, 2003]. IAE provides the sum of the error in the convergence. ISE provides the square of the sum of the error in the convergence, it penalizes large overshoots. ITAE and ITSE are two time-weighted versions of IAE and ISE, that impose large penalties on sustained errors. SOD sums up the difference in solution costs between two consecutive trials when the solution worsens. If SOD is equal to 0 convergence is monotonic.

Table 3 contains the stability indices for LRTA*(k), LRTA* and FALCONS for each grid. LRTA*(k) outperforms LRTA* for all k values tested in all indices for the three benchmarks. Something similar happens when comparing with FALCONS, except for index ITAE on Grid35 for $k = 6, 15$, where FALCONS obtains better results. In Grid35 FALCONS have some advantage with respect to ITAE because it performs a great amount of moves in earlier trials (see Tables 1 and 2), which are less penalized by the index. Solution stability improves monotonically as k increases. Again, results of Table 3 suggest that LRTA*(k) is more robust than its competitors in presence of misleading heuristics.

6 Conclusions

LRTA*(k) is a real-time search algorithm that converges to optimal routes. Based on LRTA*, it performs bounded propagation of heuristic changes up to k states, which causes longer planning steps. Experimentally, LRTA*(k) shows a substantial performance improvement with respect to LRTA* and FALCONS, in terms of first trial, convergence and solution stability. Improvements depend on k: the higher k, the better results at the cost of longer planning steps. Small k values cause moderate planning times but large benefits. LRTA*(k) can be very useful when the agent has enough time between actions to accommodate the planning time required. Future research includes the setting of k and the combination with other heuristic techniques.

References

[Bulitko, 2004] V. Bulitko. Learning for adaptive real-time search. *The Computing Research Rep. (CoRR): cs.DC/0407017*, 2004.

[Edelkamp and Eckerle, 1997] S. Edelkamp and J. Eckerle. New strategies in learning real time heuristic search. In *Proc. AAAI Workshop on On-Line Search*, pages 30–35, 1997.

[Furcy and Koenig, 2000] D. Furcy and S. Koenig. Speeding up the convergence of real-time search. In *Proc. AAAI*, pages 891–897, 2000.

[Furcy and Koenig, 2001] D. Furcy and S. Koenig. Combining two fast-learning real-time search algorithms yields even faster learning. In *Proc. 6th European Conference on Planning*, 2001.

[Goldenberg *et al.*, 2003] M. Goldenberg, A. Kovarksy, X. Wu, and J. Schaeffer. Multiple agents moving target search. In *Proc. 18th IJCAI*, pages 1536–1538, 2003.

[Ishida and Korf, 1991] T. Ishida and R. E. Korf. Moving target search. In *Proc 12th IJCAI*, pages 204–211, 1991.

[Knight, 1993] K. Knight. Are many reactive agents better than a few deliberative ones? In *Proc. 13th IJCAI*, pages 432–437, 1993.

[Koenig, 2001] S. Koenig. Agent-centered search. *Artificial Intelligence Magazine*, 22(4):109–131, 2001.

[Koenig, 2004] S. Koenig. A comparison of fast search methods for real-time situated agents. In *Proc. 3rd AAMAS*, pages 864–871, 2004.

[Korf, 1990] R. E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

[Shimbo and Ishida, 2003] M. Shimbo and T. Ishida. Controlling the learning process of real-time heuristic search. *Artificial Intelligence*, 146(1):1–41, 2003.

[Thorpe, 1994] P. E. Thorpe. A hybrid learning real-time search algorithm. Master's thesis, Computer Science Dep., UCLA, 1994.

Planning with Continuous Resources in Stochastic Domains

Mausam
Dept. of Computer Science and Engineering
University of Washington
Seattle, WA 981952350
mausam@cs.washington.edu

Emmanuel Benazera[*] **Ronen Brafman**[†]**, Nicolas Meuleau**[†]
NASA Ames Research Center
Mail Stop 269-3
Moffet Field, CA 94035-1000
{ebenazer, brafman, nmeuleau}
@email.arc.nasa.gov

Eric A. Hansen
Dept. of Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
hansen@cse.msstate.edu

Abstract

We consider the problem of optimal planning in stochastic domains with resource constraints, where resources are continuous and the choice of action at each step may depend on the current resource level. Our principal contribution is the HAO* algorithm, a generalization of the AO* algorithm that performs search in a hybrid state space that is modeled using both discrete and continuous state variables. The search algorithm leverages knowledge of the starting state to focus computational effort on the relevant parts of the state space. We claim that this approach is especially effective when resource limitations contribute to reachability constraints. Experimental results show its effectiveness in the domain that motivates our research – automated planning for planetary exploration rovers.

1 Introduction

Control of planetary exploration rovers presents several important challenges for research in automated planning. Because of difficulties inherent in communicating with devices on other planets, remote rovers must operate autonomously over substantial periods of time [Bresina *et al.*, 2002]. The planetary surfaces on which they operate are very uncertain environments: there is a great deal of uncertainty about the duration, energy consumption, and outcome of a rover's actions. Currently, instructions sent to planetary rovers are in the form of a simple plan for attaining a single goal (e.g., photographing some interesting rock). The rover attempts to carry this out, and, when done, remains idle. If it fails early on, it makes no attempt to recover and possibly achieve an alternative goal. This may have a serious impact on missions. For example, it has been estimated that the 1997 Mars Pathfinder rover spent between 40% and 75% of its time doing nothing because plans did not execute as expected. The current MER rovers (*aka* Spirit and Opportunity) require an average of 3 days to visit a single rock, but in future missions, multiple rock visits in a single communication cycle will be possible [Pedersen *et al.*, 2005]. As a result, it is expected that space scientists will request a large number of potential tasks for future rovers to perform, more than may be feasible, presenting an oversubscribed planning problem.

Working in this application domain, our goal is to provide a planning algorithm that can generate reliable contingent plans that respond to different events and action outcomes. Such plans must optimize the expected value of the experiments conducted by the rover, while being aware of its time, energy, and memory constraints. In particular, we must pay attention to the fact that given any initial state, there are multiple locations the rover could reach, and many experiments the rover could conduct, *most combinations of which* are infeasible due to resource constraints. To address this problem we need a faithful model of the rover's domain, and an algorithm that can generate optimal or near-optimal plans for such domains. General features of our problem include: (1) a concrete starting state; (2) continuous resources (including time) with stochastic consumption; (3) uncertain action effects; (4) several possible one-time-rewards, only a subset of which are achievable in a single run. This type of problem is of general interest, and includes a large class of (stochastic) logistics problems, among others.

Past work has dealt with some features of this problem. Related work on MDPs with resource constraints includes the model of constrained MDPs developed in the OR community [Altman, 1999]. A constrained MDP is solved by a linear program that includes constraints on resource consumption, and finds the best feasible policy, given an initial state and resource allocation. A drawback of the constrained MDP model is that it does not include resources in the state space, and thus, a policy cannot be conditioned on resource availability. Moreover, it does not model stochastic resource consumption. In the area of decision-theoretic planning, several techniques have been proposed to handle uncertain continuous variables (e.g. [Feng *et al.*, 2004; Younes and Simmons, 2004; Guestrin *et al.*, 2004]). Smith 2004 and van den Briel et *al.* 2004 consider the problem of over-subscription planning, i.e., planning with a large set of goals which is not entirely achievable. They provide techniques for selecting a subset of goals for which to plan, but they deal only with deterministic domains. Finally, Meuleau *et al.* 2004 present preliminary experiments towards scaling up decision-theoretic approaches to planetary rover problems.

[*] Research Institute for Advanced Computer Science.
[†] QSS Group Inc.

Our contribution in this paper is an implemented algorithm, Hybrid AO* (HAO*), that handles all of these problems together: oversubscription planning, uncertainty, and limited continuous resources. Of these, the most essential features of our algorithm are its ability to handle hybrid state-spaces and to utilize the fact that many states are unreachable due to resource constraints.

In our approach, resources are included in the state description. This allows decisions to be made based on resource availability, and it allows a stochastic resource consumption model (as opposed to constrained MDPs). Although this increases the size of the state space, we assume that the value functions may be represented compactly. We use the work of Feng et *al.* (2004) on piecewise constant and linear approximations of dynamic programming (DP) in our implementation. However, standard DP does not exploit the fact that the reachable state space is much smaller than the complete state space, especially in the presence of resource constraints. Our contribution is to show how to use the forward heuristic search algorithm called AO* [Pearl, 1984; Hansen and Zilberstein, 2001] to solve MDPs with resource constraints and continuous resource variables. Unlike DP, forward search keeps track of the trajectory from the start state to each reachable state, and thus it can check whether the trajectory is feasible or violates a resource constraint. This allows heuristic search to prune infeasible trajectories and can dramatically reduce the number of states that must be considered to find an optimal policy. This is particularly important in our domain where the discrete state space is huge (exponential in the number of goals), yet the portion reachable from any initial state is relatively small because of the resource constraints. It is well-known that heuristic search can be more efficient than DP because it leverages a search heuristic and reachability constraints to focus computation on the relevant parts of the state space. We show that for problems with resource constraints, this advantage can be even greater than usual because resource constraints further limit reachability.

The paper is structured as follows: In Section 2 we describe the basic action and goal model. In Section 3 we explain our planning algorithm, HAO*. Initial experimental results are described in Section 4, and we conclude in Section 5.

2 Problem Definition and Solution Approach

2.1 Problem Formulation

We consider a Markov decision process (MDP) with both continuous and discrete state variables (also called a *hybrid MDP* [Guestrin *et al.*, 2004] or *Generalized State MDP* [Younes and Simmons, 2004]). Each state corresponds to an assignment to a set of state variables. These variables may be discrete or continuous. Continuous variables typically represent resources, where one possible type of resource is time. Discrete variables model other aspects of the state, including (in our application) the set of goals achieved so far by the rover. (Keeping track of already-achieved goals ensures a Markovian reward structure, since we reward achievement of a goal only if it was not achieved in the past.) Although our models typically contain multiple discrete variables, this plays no role in the description of our algorithm, and so, for notational convenience, we model the discrete component as a single variable n.

A *Markov state* $s \in S$ is a pair $(n, \mathbf{x})$ where $n \in N$ is the discrete variable, and $\mathbf{x} = (x_i)$ is a vector of continuous variables. The domain of each x_i is an interval X_i of the real line, and $\mathbf{X} = \bigotimes_i X_i$ is the hypercube over which the continuous variables are defined. We assume an explicit *initial state*, denoted $(n_0, \mathbf{x}_0)$, and one or more absorbing *terminal states*. One terminal state corresponds to the situation in which all goals have been achieved. Others model situations in which resources have been exhausted or an action has resulted in some error condition that requires executing a safe sequence by the rover and terminating plan execution.

Actions can have executability constraints. For example, an action cannot be executed in a state that does not have its minimum resource requirements. $A_n(\mathbf{x})$ denotes the set of actions executable in state $(n, \mathbf{x})$.

State transition probabilities are given by the function $\Pr(s' \mid s, a)$, where $s = (n, \mathbf{x})$ denotes the state before action a and $s' = (n', \mathbf{x}')$ denotes the state after action a, also called the arrival state. Following [Feng *et al.*, 2004], the probabilities are decomposed into:

- the discrete marginals $\Pr(n'|n, \mathbf{x}, a)$. For all $(n, \mathbf{x}, a)$, $\sum_{n' \in N} \Pr(n'|n, \mathbf{x}, a) = 1$;
- the continuous conditionals $\Pr(\mathbf{x}'|n, \mathbf{x}, a, n')$. For all $(n, \mathbf{x}, a, n')$, $\int_{x' \in \mathbf{X}} \Pr(\mathbf{x}'|n, \mathbf{x}, a, n') d\mathbf{x}' = 1$.

Any transition that results in negative value for some continuous variable is viewed as a transition into a terminal state.

The *reward* of a transition is a function of the arrival state only. More complex dependencies are possible, but this is sufficient for our goal-based domain models. We let $R_n(\mathbf{x}) \geq 0$ denote the *reward* associated with a transition to state $(n, \mathbf{x})$.

In our application domain, continuous variables model non-replenishable resources. This translates into the general assumption that the value of the continuous variables is non-increasing. Moreover, we assume that each action has some minimum positive consumption of at least one resource. We do not utilize this assumption directly. However, it has two implications upon which the correctness of our approach depends: (1) the values of the continuous variables are a-priori bounded, and (2) the number of possible steps in any execution of a plan is bounded, which we refer to by saying the problem has a *bounded horizon*. Note that the actual number of steps until termination can vary depending on actual resource consumption.

Given an initial state $(n_0, \mathbf{x_0})$, the objective is to find a policy that maximizes expected cumulative reward.[1] In our application, this is equal to the sum of the rewards for the goals achieved before running out of a resource. Note that there is no direct incentive to save resources: an optimal solution would save resources only if this allows achieving more goals. Therefore, we stay in a standard decision-theoretic framework. This problem is solved by solving Bellman's op-

[1] Our algorithm can easily be extended to deal with an uncertain starting state, as long as its probability distribution is known.

timality equation, which takes the following form:

$$V_n^0(\mathbf{x}) = 0 \ ,$$

$$V_n^{t+1}(\mathbf{x}) = \max_{a \in A_n(\mathbf{x})} \left[\sum_{n' \in N} \Pr(n' \mid, n, \mathbf{x}, a) \right. \quad (1)$$

$$\left. \int_{\mathbf{x}'} \Pr(\mathbf{x}' \mid n, \mathbf{x}, a, n') \left(R_{n'}(\mathbf{x}') + V_{n'}^t(\mathbf{x}') \right) d\mathbf{x}' \right] \ .$$

Note that the index t represents the iteration or *time-step* of DP, and does not necessarily correspond to time in the planning problem. The duration of actions is one of the biggest sources of uncertainty in our rover problems, and we typically model time as one of the continuous resources x_i.

2.2 Solution Approach

Feng *et al.* describe a dynamic programming (DP) algorithm that solves this Bellman optimality equation. In particular, they show that the continuous integral over $\mathbf{x}'$ can be computed exactly, as long as the transition function satisfies certain conditions. This algorithm is rather involved, so we will treat it as a black-box in our algorithm. In fact, it can be replaced by any other method for carrying out this computation. This also simplifies the description of our algorithm in the next section and allows us to focus on our contribution. We do explain the ideas and the assumptions behind the algorithm of Feng *et al.* in Section 3.3.

The difficulty we address in this paper is the potentially *huge* size of the state space, which makes DP infeasible. One reason for this size is the existence of continuous variables. But even if we only consider the discrete component of the state space, the size of the state space is exponential in the number of propositional variables comprising the discrete component. To address this issue, we use forward heuristic search in the form of a novel variant of the AO* algorithm. Recall that AO* is an algorithm for searching AND/OR graphs [Pearl, 1984; Hansen and Zilberstein, 2001]. Such graphs arise in problems where there are choices (the OR components), and each choice can have multiple consequences (the AND component), as is the case in planning under uncertainty. AO* can be very effective in solving such planning problems when there is a large state space. One reason for this is that AO* only considers states that are reachable from an initial state. Another reason is that given an informative heuristic function, AO* focuses on states that are reachable in the course of executing a good plan. As a result, AO* often finds an optimal plan by exploring a small fraction of the entire state space.

The challenge we face in applying AO* to this problem is the challenge of performing state-space search in a continuous state space. Our solution is to search in an *aggregate state space* that is represented by a search graph in which there is a node for each distinct value of the discrete component of the state. In other words, each node of our search graph represents a region of the continuous state space in which the discrete value is the same. In this approach, different actions may be optimal for different Markov states in the aggregate state associated with a search node, especially since the best action is likely to depend on how much energy or time is remaining. To address this problem and still find an optimal solution, we associate a value estimate with each of the Markov states in an aggregate. That is, we attach to each search node a value function (function of the continuous variables) instead of the simple scalar value used by standard AO*. Following the approach of [Feng *et al.*, 2004], this value function can be represented and computed efficiently due to the continuous nature of these states and the simplifying assumptions made about the transition functions. Using these value estimates, we can associate different actions with different Markov states within the aggregate state corresponding to a search node.

In order to select which node on the fringe of the search graph to expand, we also need to associate a scalar value with each search node. Thus, we maintain for a search node both a heuristic estimate of the value function (which is used to make action selections), and a heuristic estimate of the priority which is used to decide which search node to expand next. Details are given in the following section.

We note that LAO*, a generalization of AO*, allows for policies that contain "loops" in order to specify behavior over an infinite horizon [Hansen and Zilberstein, 2001]. We could use similar ideas to extend LAO* to our setting. However, we need not consider loops for two reasons: (1) our problems have a bounded horizon; (2) an optimal policy will not contain any intentional loop because returning to the same discrete state with fewer resources cannot buy us anything. Our current implementation assumes any loop is intentional and discards actions that create such a loop.

3 Hybrid AO*

A simple way of understanding HAO* is as an AO* variant where states with identical discrete component are expanded in unison. HAO* works with two graphs:

- The *explicit graph* describes all the states that have been generated so far and the AND/OR edges that connect them. The nodes of the explicit graph are stored in two lists: OPEN and CLOSED.
- The *greedy policy* (or partial solution) graph, denoted GREEDY in the algorithms, is a sub-graph of the explicit graph describing the current optimal policy.

In standard AO*, a single action will be associated with each node in the greedy graph. However, as described before, multiple actions can be associated with each node, because different actions may be optimal for different Markov states represented by an aggregate state.

3.1 Data Structures

The main data structure represents a search node n. It contains:

- The value of the discrete state. In our application these are the discrete state variables and set of goals achieved.
- Pointers to its parents and children in the explicit and greedy policy graphs.
- $P_n(\cdot)$ – a probability distribution on the continuous variables in node n. For each $\mathbf{x} \in \mathbf{X}$, $P_n(\mathbf{x})$ is an estimate

of the probability density of passing through state $(n, \mathbf{x})$ under the current greedy policy. It is obtained by *progressing* the initial state forward through the optimal actions of the greedy policy. With each P_n, we maintain the probability of passing through n under the greedy policy:

$$M(P_n) = \int_{\mathbf{x} \in \mathbf{X}} P_n(\mathbf{x}) d\mathbf{x} \ .$$

- $H_n(\cdot)$ – the heuristic function. For each $\mathbf{x} \in \mathbf{X}$, $H_n(\mathbf{x})$ is a heuristic estimate of the optimal expected reward from state $(n, \mathbf{x})$.
- $V_n(\cdot)$ – the value function. At the leaf nodes of the explicit graph, $V_n = H_n$. At the non-leaf nodes of the explicit graph, V_n is obtained by backing up the H functions from the descendant leaves. If the heuristic function $H_{n'}$ is admissible in all leaf nodes n', then $V_n(\mathbf{x})$ is an upper bound on the optimal reward to come from $(n, \mathbf{x})$ for all $\mathbf{x}$ reachable under the greedy policy.
- g_n – a heuristic estimate of the increase in value of the greedy policy that we would get by expanding node n. If H_n is admissible then g_n represents an upper bound on the gain in expected reward. The gain g_n is used to determine the priority of nodes in the OPEN list ($g_n = 0$ if n is in CLOSED), and to bound the error of the greedy solution at each iteration of the algorithm.

Note that some of this information is redundant. Nevertheless, it is convenient to maintain all of it so that the algorithm can easily access it. HAO* uses the customary OPEN and CLOSED lists maintained by AO*. They encode the explicit graph and the current greedy policy. CLOSED contains expanded nodes, and OPEN contains unexpanded nodes and nodes that need to be re-expanded.

```
 1: Create the root node n_0 which represents the initial state.
 2: P_{n_0} = initial distribution on resources.
 3: V_{n_0} = 0 everywhere in X.
 4: g_{n_0} = 0.
 5: OPEN = GREEDY = {n_0}.
 6: CLOSED = ∅.
 7: while OPEN ∩ GREEDY ≠ ∅ do
 8:    n = arg max_{n' ∈ OPEN ∩ GREEDY} (g_{n'}).
 9:    Move n from OPEN to CLOSED.
10:    for all (a, n') ∈ A × N not expanded yet in n and
       reachable under P_n do
11:       if n' ∉ OPEN ∪ CLOSED then
12:          Create the data structure to represent n' and add
             the transition (n, a, n') to the explicit graph.
13:          Get H_{n'}.
14:          V_{n'} = H_{n'} everywhere in X.
15:          if n' is terminal: then
16:             Add n' to CLOSED.
17:          else
18:             Add n' to OPEN.
19:       else if n' is not an ancestor of n in the explicit graph
          then
20:          Add the transition (n, a, n') to the explicit graph.
21:    if some pair (a, n') was expanded at previous step (10)
       then
22:       Update V_n for the expanded node n and some of its
          ancestors in the explicit graph, with Algorithm 2.
23:    Update P_{n'} and g_{n'} using Algorithm 3 for the nodes
       n' that are children of the expanded node or of a node
       where the optimal decision changed at the previous
       step (22). Move every node n' ∈ CLOSED where
       P changed back into OPEN.
```

Algorithm 1: Hybrid AO*

3.2 The HAO* Algorithm

Algorithm 1 presents the main procedure. The crucial steps are described in detail below.

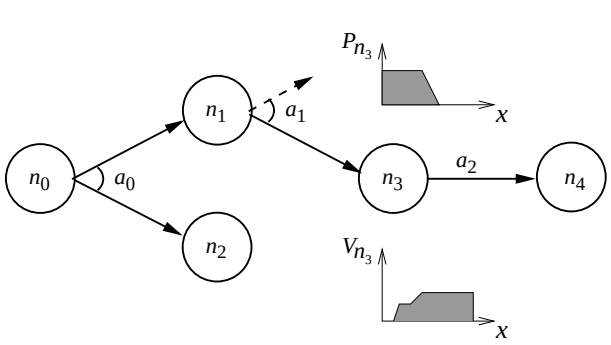
Expanding a node (lines 10 to 20): At each iteration, HAO* expands the open node n with the highest priority g_n in the greedy graph. An important distinction between AO* and HAO* is that in the latter, nodes are often only partially expanded (i.e., not all Markov states associated with a discrete node are considered). Thus, nodes in the CLOSED list are sometimes put back in OPEN (line 23). The reason for this is that a Markov state associated with this node, that was previously considered unreachable, may now be reachable. Technically, what happens is that as a result of finding a new path to a node, the probability distribution over it is updated (line 23), possibly increasing the probability of some Markov state from 0 to some positive value. This process is illustrated in Figure 1. Thus, while standard AO* expands only tip nodes, HAO* sometimes expands nodes that were moved from CLOSED to OPEN and are "in the middle of" the greedy policy subgraph.

Next, HAO* considers all possible successors (a, n') of n given the state distribution P_n. Typically, when n is expanded for the first time, we enumerate all actions a possible in $(n, \mathbf{x})$ ($a \in A_n(\mathbf{x})$) for some reachable $\mathbf{x}$ ($P_n(\mathbf{x}) > 0$), and all arrival states n' that can result from such a transition ($\Pr(n' \mid n, \mathbf{x}, a) > 0$).[2] If n was previously expanded (i.e. it has been put back in OPEN), only actions and arrival nodes not yet expanded are considered. In line 11, we check whether a node has already been generated. This is not necessary if the graph is a tree (i.e., there is only one way to get to each discrete state).[3] In line 15, a node n' is terminal if no action is executable in it (because of lack of resources). In our application domain each goal pays only once, thus the nodes in which all goals of the problem have been achieved are also terminal. Finally, the test in line 19 prevents loops in the explicit graph. As discussed earlier, such loops are always suboptimal.

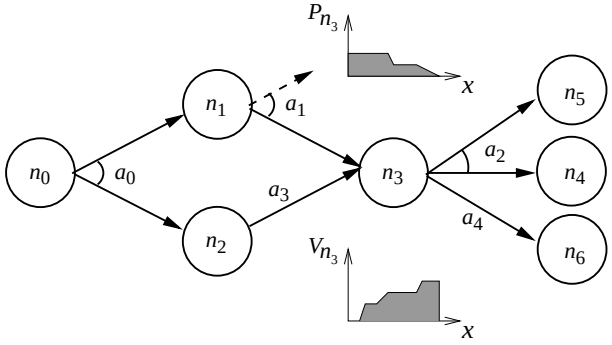
Updating the value functions (lines 22 to 23): As in standard AO*, the value of a newly expanded node must be updated. This consists of recomputing its value function with Bellman's equations (Eqn. 1), based on the value functions of all children of n in the explicit graph. Note that these backups

[2] We assume that performing an action in a state where it is not allowed is an error that ends execution with zero or constant reward.

[3] Sometimes it is beneficial to use the tree implementation of AO* when the problem graph is *almost* a tree, by duplicating nodes that represents the same (discrete) state reached through different paths.



(a) Initial GREEDY graph. Actions have multiple possible discrete effects (e.g., a_0 has two possible effects in n_0). The curves represent the current probability distribution P and value function V over x values for n_3. n_2 is a fringe node.



(b) GREEDY graph with n_2 expanded. Since the path (n_0, n_2, n_3) is optimal for some resource levels in n_0, P_{n_3} has changed. As a consequence, n_3 has been re-expanded , showing that node n_5 is now reachable from n_3 under a_2, and action a_4 has become do-able in n_3.

Figure 1: Node re-expansion.

involve all continuous states $\mathbf{x} \in \mathbf{X}$ for each node, *not* just the reachable values of $\mathbf{x}$. However, they consider only actions and arrival nodes that are reachable according to P_n. Once the value of a state is updated, its new value must be propagated backward in the explicit graph. The backward propagation stops at nodes where the value function is not modified, and/or at the root node. The whole process is performed by applying Algorithm 2 to the newly expanded node.

1: $Z = \{n\}$ // n is the newly expanded node.
2: **while** $Z \neq \emptyset$ **do**
3: Choose a node $n' \in Z$ that has no descendant in Z.
4: Remove n' from Z.
5: Update $V_{n'}$ following Eqn. 1.
6: **if** $V_{n'}$ was modified at the previous step **then**
7: Add all parents of n' in the explicit graph to Z.
8: **if** optimal decision changes for some $(n', \mathbf{x})$, $P_{n'}(\mathbf{x}) > 0$ **then**
9: Update the greedy subgraph (GREEDY) at n' if necessary.
10: Mark n' for use at line 23 of Algorithm 1.

Algorithm 2: Updating the value functions V_n.

Updating the state distributions (line 23): P_n's represent the state distribution *under the greedy policy*, and they need to be updated after recomputing the greedy policy. More precisely, P needs to be updated in each descendant of a node where the optimal decision changed. To update a node n, we consider all its parents n' in the greedy policy graph, and all the actions a that can lead from one of the parents to n. The probability of getting to n with a continuous component $\mathbf{x}$ is the sum over all (n', a) and all possible values of $\mathbf{x}'$ of the continuous component over the the probability of arriving from n' and $\mathbf{x}'$ under a. This can be expressed as:

$$P_n(\mathbf{x}) = \sum_{(n',a)\in\Omega_n} \int_{\mathbf{X}'} P_{n'}(\mathbf{x}') \Pr(n \mid n', \mathbf{x}', a) \Pr(\mathbf{x} \mid n', \mathbf{x}', a, n) d\mathbf{x}' \ . \quad (2)$$

Here, $\mathbf{X}'$ is the domain of possible values for $\mathbf{x}'$, and Ω_n is the set of pairs (n', a) where a is the greedy action in n' for some reachable resource level:

$$\Omega_n = \{(n', a) \in N \times A : \exists \mathbf{x} \in \mathbf{X}, P_{n'}(\mathbf{x}) > 0,\ \mu^*_{n'}(\mathbf{x}) = a,\ \Pr(n \mid n', \mathbf{x}, a) > 0\} \ ,$$

where $\mu^*_n(\mathbf{x}) \in A$ is the greedy action in $(n, \mathbf{x})$. Clearly, we can restrict our attention to state-action pairs in Ω_n, only. Note that this operation may induce a loss of total probability mass ($P_n < \sum_{n'} P_{n'}$) because we can run out of a resource during the transition and end up in a sink state.

When the distribution P_n of a node n in the OPEN list is updated, its priority g_n is recomputed using the following equation (the priority of nodes in CLOSED is maintained as 0):

$$g_n = \int_{\mathbf{x}\in S(P_n)-\mathbf{X}_n^{\text{old}}} P_n(\mathbf{x}) H_n(\mathbf{x}) d\mathbf{x} \ ; \quad (3)$$

where $S(P)$ is the support of P: $S(P) = \{\mathbf{x} \in \mathbf{X} : P(\mathbf{x}) > 0\}$, and $\mathbf{X}_n^{\text{old}}$ contains all $\mathbf{x} \in \mathbf{X}$ such that the state $(n, \mathbf{x})$ has already been expanded before ($\mathbf{X}_n^{\text{old}} = \emptyset$ if n has never been expanded). The techniques used to represent the continuous probability distributions P_n and compute the continuous integrals are discussed in the next sub-section. Algorithm 3 presents the state distribution updates. It applies to the set of nodes where the greedy decision changed during value updates (including the newly expanded node, i.e. n in HAO* – Algorithm 1).

3.3 Handling Continuous Variables

Computationally, the most challenging aspect of HAO* is the handling of continuous state variables, and particularly the

1: Z = children of nodes where the optimal decision changed when updating value functions in Algorithm 1.
2: **while** $Z \neq \emptyset$ **do**
3: Choose a node $n \in Z$ that has no ancestor in Z.
4: Remove n from Z.
5: Update P_n following Eqn. 2.
6: **if** P_n was modified at step 5 **then**
7: Move n from CLOSED to OPEN.
8: Update the greedy subgraph (GREEDY) at n if necessary.
9: Update g_n following Eqn. 3.

Algorithm 3: Updating the state distributions P_n.

computation of the continuous integral in Bellman backups and Eqns. 2 and 3. We approach this problem using the ideas developed in [Feng *et al.*, 2004] for the same application domain. However, we note that HAO* could also be used with other models of uncertainty and continuous variables, as long as the value functions can be computed exactly in finite time. The approach of [Feng *et al.*, 2004] exploits the structure in the continuous value functions of the type of problems we are addressing. These value functions typically appear as collections of humps and plateaus, each of which corresponds to a region in the state space where similar goals are pursued by the optimal policy (see Fig. 3). The sharpness of the hump or the edge of a plateau reflects uncertainty of achieving these goals. Constraints imposing minimal resource levels before attempting risky actions introduce sharp cuts in the regions. Such structure is exploited by grouping states that belong to the same plateau, while reserving a fine discretization for the regions of the state space where it is the most useful (such as the edges of plateaus).

To adapt the approach of [Feng *et al.*, 2004], we make some assumptions that imply that our value functions can be represented as piece-wise constant or linear. Specifically, we assume that the continuous state space induced by every discrete state can be divided into hyper-rectangles in each of which the following holds: (i) The same actions are applicable. (ii) The reward function is piece-wise constant or linear. (iii) The distribution of discrete effects of each action are identical. (iv) The set of arrival values or value variations for the continuous variables is discrete and constant. Assumptions (i-iii) follow from the hypotheses made in our domain models. Assumption (iv) comes down to discretizing the actions' resource consumptions, which is an approximation. It contrasts with the naive approach that consists of discretizing the state space regardless of the relevance of the partition introduced. Instead, we discretize the action outcomes first, and then deduce a partition of the state space from it. The state-space partition is kept as coarse as possible, so that only the relevant distinctions between (continuous) states are taken into account. Given the above conditions, it can be shown (see [Feng *et al.*, 2004]) that for any finite horizon, for any discrete state, there exists a partition of the continuous space into hyper-rectangles over which the optimal value function is piece-wise constant or linear. The implementation represents the value functions as kd-trees, using a fast algorithm to intersect kd-trees [Friedman *et al.*, 1977], and merging adjacent pieces of the value function based on their value. We augmented this approach by representing the continuous state distributions P_n as piecewise constant functions of the continuous variables. Under the set of hypotheses above, if the initial probability distribution on the continuous variables is piecewise constant, then the probability distribution after any finite number of actions is too, and Eqn. 2 may always be computed in finite time.[4]

3.4 Properties

As for standard AO*, it can be shown that if the heuristic functions H_n are admissible (optimistic), the actions have positive resource consumptions, and *the continuous backups are computed exactly*, then: (i) at each step of HAO*, $V_n(\mathbf{x})$ is an upper-bound on the optimal expected return in $(n, \mathbf{x})$, for all $(n, \mathbf{x})$ expanded by HAO*; (ii) HAO* terminates after a finite number of iterations; (iii) after termination, $V_n(\mathbf{x})$ is equal to the optimal expected return in $(n, \mathbf{x})$, for all $(n, \mathbf{x})$ reachable under the greedy policy ($P_n(\mathbf{x}) > 0$). Moreover, if we assume that, in each state, there is a *done* action that terminates execution with zero reward (in a rover problem, we would then start a safe sequence), then we can evaluate the greedy policy at each step of the algorithm by assuming that execution ends each time we reach a leaf of the greedy subgraph. Under the same hypotheses, the error of the greedy policy at each step of the algorithm is bounded by $\sum_{n \in \mathrm{GREEDY} \cap \mathrm{OPEN}} g_n$. This property allows trading computation time for accuracy by stopping the algorithm early.

3.5 Heuristic Functions

The heuristic function H_n helps focus the search on truly useful reachable states. It is essential for tackling real-size problems. Our heuristic function is obtained by solving a relaxed problem. The relaxation is very simple: we assume deterministic transitions for the continuous variables, i.e., $Pr(\mathbf{x}'|n, \mathbf{x}, a, n') \in \{0, 1\}$. If we assume the actions consume the minimum amount of each resource, we obtain an admissible heuristic function. A non-admissible, but probably more informative heuristic function is obtained by using the mean resource consumption.

The central idea is to use *the same algorithm* to solve both the relaxed and the original problem. Unlike classical approaches where a relaxed plan is generated for every search state, we generate a "relaxed" search-graph using our HAO* algorithm *once* with a deterministic-consumption model and a trivial heuristic. The value function V_n of a node in the relaxed graph represents the heuristic function H_n of the associated node in the original problem graph. Solving the relaxed problem with HAO* is considerably easier, because the structure and the updates of the value functions V_n and of the probabilities P_n are much simpler than in the original domain. However, we run into the following problem: deterministic consumption implies that the number of reachable states for any given initial state is very small (because only one continuous assignment is possible). This means that in a single expansion, we obtain information about a small number of

[4] A deterministic starting state $\mathbf{x_0}$ is represented by a uniform distribution with very small rectangular support centered in $\mathbf{x_0}$.

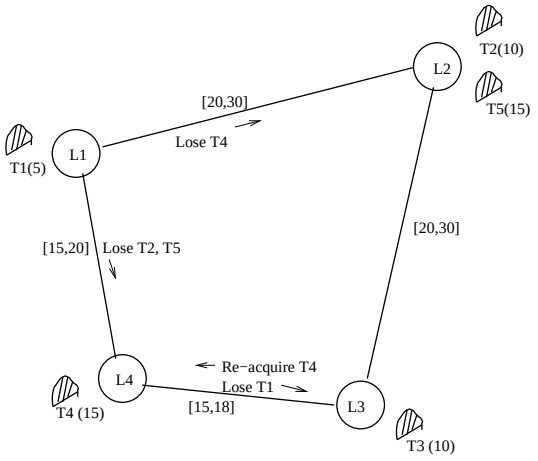


Figure 2: Case study: the rover navigates around five target rocks (T1 to T5). The number with each rock is the reward received on testing that rock.

states. To address this problem, instead of starting with the initial resource values, we assume a uniform distribution over the possible range of resource values. Because it is relatively easy to work with a uniform distribution, the computation is simple relative to the real problem, but we obtain an estimate for many more states. It is still likely that we reach states for which no heuristic estimate was obtained using these initial values. In that case, we simply recompute starting with this initial state.

4 Experimental Evaluation

We tested our algorithm on a slightly simplified variant of the rover domain model used for NASA Ames October 2004 Intelligent Systems demo [Pedersen *et al.*, 2005]. In this domain, a planetary rover moves in a planar graph made of locations and paths, sets up instruments at different rocks, and performs experiments on the rocks. Actions may fail, and their energy and time consumption are uncertain. Resource consumptions are drawn from two type of distributions: uniform and normal, and then discretized. The problem instance used in our preliminary experiments is illustrated in figure 2. It contains 5 target rocks (T1 to T5) to be tested. To take a picture of a target rock, this target must be tracked. To track a target, we must register it before doing the first move. Later, different targets can be lost and re-acquired when navigating along different paths. These changes are modeled as action effects in the discrete state. Overall, the problem contains 43 propositional state variables and 37 actions. Therefore, there are 2^{48} different discrete states, which is far beyond the reach of a flat DP algorithm.

The results presented here were obtained using a preliminary implementation of the piecewise constant DP approximations described in [Feng *et al.*, 2004] based on a flat representation of state partitions instead of kd-trees. This is considerably slower than an optimal implementation. To compensate, our domain features a single abstract continuous resource, while the original domain contains two resources

A	B	C	D	E	F	G	H
30	0.1	39	39	38	9	1	239
40	0.4	176	163	159	9	1	1378
50	1.8	475	456	442	12	1	4855
60	7.6	930	909	860	32	2	12888
70	13.4	1548	1399	1263	22	2	25205
80	32.4	2293	2148	2004	33	2	42853
90	87.3	3127	3020	2840	32	2	65252
100	119.4	4673	4139	3737	17	2	102689
110	151.0	6594	5983	5446	69	3	155733
120	213.3	12564	11284	9237	39	3	268962
130	423.2	19470	17684	14341	41	3	445107
140	843.1	28828	27946	24227	22	3	17113
150	1318.9	36504	36001	32997	22	3	1055056

Table 1: Performance of the algorithm for different initial resource levels. A: initial resource (abstract unit). B: execution time (s). C: # reachable discrete states. D: # nodes created by AO*. E: # nodes expanded by AO*. F: # nodes in the optimal policy graph. G: # goals achieved in the longest branch of the optimal solution. H: # reachable Markov states.

(time and energy). Another difference in our implementation is in the number of nodes expanded at each iteration. We adapt the findings of [Hansen and Zilberstein, 2001] that overall convergence speeds up if all the nodes in OPEN are expanded at once, instead of prioritizing them based on g_n values and changing the value functions after each expansion.[5] Finally, these preliminary experiments do not use the sophisticated heuristics presented earlier, but the following simple admissible heuristic: H_n is the constant function equal to the sum of the utilities of all the goals not achieved in n.

We varied the initial amount of resource available to the rover. As available resource increases, more nodes are reachable and more reward can be gained. The performance of the algorithm is presented in Table 1. We see that the number of reachable discrete states is much smaller than the total number of states (2^{48}) and the number of nodes in an optimal policy is surprisingly small. This indicates that AO* is particularly well suited to our rover problems. However, the number of nodes expanded is quite close to the number of reachable discrete states. Thus, our current simple heuristic is only slightly effective in reducing the search space, and reachability makes the largest difference. This suggests that much progress can be obtained by using better heuristics. The last column measures the total number of reachable Markov states, after discretizing the action consumptions as in [Feng *et al.*, 2004]. This is the space that a forward search algorithm manipulating Markov states, instead of discrete states, would have to tackle. In most cases, it would be impossible to explore such space with poor quality heuristics such as ours. This indicates that our algorithm is quite effective in scaling up to very large problems by exploiting the structure presented by continuous resources.

Figure 3 shows the converged value function of the initial state of the problem. The value function is comprised of several plateaus, where different sets of goals are achieved. The first plateau (until resource level 23) corresponds to the

[5] In this implementation, we do not have to maintain exact probability distributions P_n. We just need to keep track of the supports of these distributions, which can be approximated by lower and upper bounds on each continuous variable.

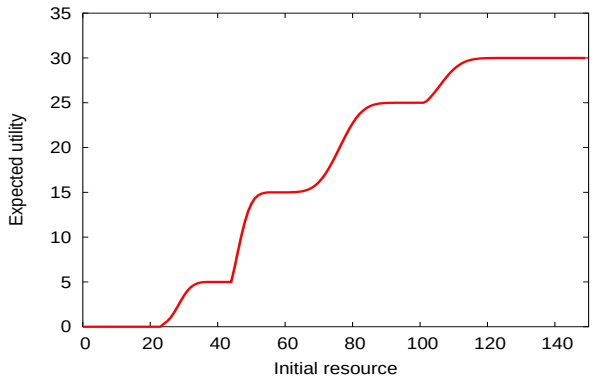


Figure 3: Value function of the initial state.

Initial resource	ε	Execution time	# nodes created by AO*	# nodes expanded by AO*
130	0.00	426.8	17684	14341
130	0.50	371.9	17570	14018
130	1.00	331.9	17486	13786
130	1.50	328.4	17462	13740
130	2.00	330.0	17462	13740
130	2.50	320.0	17417	13684
130	3.00	322.1	17417	13684
130	3.50	318.3	17404	13668
130	4.00	319.3	17404	13668
130	4.50	319.3	17404	13668
130	5.00	318.5	17404	13668
130	5.50	320.4	17404	13668
130	6.00	315.5	17356	13628

Table 2: Complexity of computing an ε-optimal policy. The optimal return for an initial resource of 130 is 30.

case where the resource level is insufficient for any goal to be achieved. The next plateau (until 44) depicts the region in which the target T1 is tested. The remaining resources are still not enough to move to a new location and generate additional rewards. In the region between 44 and 61 the rover decides to move to L4 and test T4. Note that the location L2 is farther from L4 and so the rover does not attempt to move to L2, yet. The next plateau corresponds to the region in which the optimal strategy is to move to L2 and test both T2 and T5.The last region (beyond 101) is in which three goals T1, T2 and T5 are tested and reward of 30 is obtained.

When H_n is admissible, we can bound the error of the current greedy graph by summing g_n over fringe nodes. In Table 2 we describe the time/value tradeoff we found for this domain. On the one hand, we see that even a large compromise in quality leads to no more than 25% reduction in time. On the other hand, we see that much of this reduction is obtained with a very small price ($\epsilon = 0.5$). Additional experiments are required to learn if this is a general phenomenon.

5 Conclusions

We presented a variant of the AO* algorithm that, to the best of our knowledge, is the first algorithm to deal with: limited continuous resources, uncertainty, and oversubscription planning. We developed a sophisticated reachability analysis involving continuous variables that could be useful for heuristic search algorithms at large. Our preliminary implementation of this algorithm shows very promising results on a domain of practical importance. We are able to handle problems with 2^{48} discrete states, as well as a continuous component.

In the near future, we hope to report on a more mature version of the algorithm, which we are currently implementing. It includes: (1) a full implementation of the techniques described in [Feng *et al.*, 2004]; (2) a rover model with two continuous variables; (3) a more informed heuristic function, as discussed in Section 3.5.

Acknowledgements

This work was funded by the NASA Intelligent Systems program. Eric Hansen was supported in part by NSF grant IIS-9984952, NASA grant NAG-2-1463 and a NASA Summer Faculty Fellowship.

References

[Altman, 1999] E. Altman. *Constrained Markov Decision Processes*. Chapman and HALL/CRC, 1999.

[Bresina *et al.*, 2002] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 77–84, 2002.

[Feng *et al.*, 2004] Z. Feng, R. Dearden, N. Meuleau, and R. Washington. Dynamic programming for structured continuous Markov decision problems. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 154–161, 2004.

[Friedman *et al.*, 1977] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Mathematical Software*, 3(3):209–226, 1977.

[Guestrin *et al.*, 2004] C. Guestrin, M. Hauskrecht, and B. Kveton. Solving factored MDPs with continuous and discrete variables. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 235–242, 2004.

[Hansen and Zilberstein, 2001] E. Hansen and S. Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129:35–62, 2001.

[Meuleau *et al.*, 2004] N. Meuleau, R. Dearden, and R. Washington. Scaling up decision theoretic planning to planetary rover problems. In *AAAI-04: Proceedings of the Workshop on Learning and Planning in Markov Processes Advances and Challenges*, pages 66–71, Technical Report WS-04-08, AAAI Press, Menlo Park, CA, 2004.

[Pearl, 1984] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[Pedersen *et al.*, 2005] L. Pedersen, D. Smith, M. Deans, R. Sargent, C. Kunz, D. Lees, and S.Rajagopalan. Mission planning and target tracking for autonomous instrument placement. In *Submitted to 2005 IEEE Aerospace Conference*, 2005.

[Smith, 2004] D. Smith. Choosing objectives in over-subscription planning. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, pages 393–401, 2004.

[van den Briel *et al.*, 2004] M. van den Briel, M.B. Do R. Sanchez and, and S. Kambhampati. Effective approaches for partial satisfation (over-subscription) planning. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 562–569, 2004.

[Younes and Simmons, 2004] H.L.S. Younes and R.G. Simmons. Solving generalized semi-Markov decision processes using continuous phase-type distributions. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 742–747, 2004.

Automated Composition of Web Services by Planning at the Knowledge Level

M. Pistore
University of Trento
Via Sommarive 14
38050 Povo (TN), Italy
pistore@dit.unitn.it

A. Marconi and **P. Bertoli** and **P. Traverso**
ITC-IRST
Via Sommarive 18
38050 Povo (TN), Italy
[marconi, bertoli, traverso]@itc.it

Abstract

In this paper, we address the problem of the automated composition of web services by planning on their "knowledge level" models. We start from descriptions of web services in standard process modeling and execution languages, like BPEL4WS, and automatically translate them into a planning domain that models the interactions among services at the knowledge level. This allows us to avoid the explosion of the search space due to the usually large and possibly infinite ranges of data values that are exchanged among services, and thus to scale up the applicability of state-of-the-art techniques for the automated composition of web services. We present the theoretical framework, implement it, and provide an experimental evaluation that shows the practical advantage of our approach w.r.t. techniques that are not based on a knowledge-level representation.

1 Introduction

Research in planning is more and more focusing on the problem of the automated composition of web services: given a set of services that are published on the Web, and given a goal, generate a composition of the available services that satisfies the goal (see, e.g., [Narayanan and McIlraith, 2002]). In spite of the fact that several approaches have been proposed so far (see, e.g., [McIlraith and Son, 2002; Wu *et al.*, 2003; Sheshagiri *et al.*, 2003; Traverso and Pistore, 2004]), solving this problem *in practice*, by scaling up to realistic descriptions of web services, is far from trivial. Indeed it is widely recognized that web services must be modeled with nondeterministic and partially observable behaviors [Koehler and Srivastava, 2003; Hull *et al.*, 2003; Berardi *et al.*, 2003; McIlraith and Fadel, 2002; Traverso and Pistore, 2004; Martinez and Lesperance, 2004], and thus planning algorithms must work with incomplete information and with actions with uncertain effects. Moreover, in several application domains, web services cannot be simply modeled as atomic components, but as stateful processes whose interactions are intrinsically asynchronous [Fu *et al.*, 2004; Foster *et al.*, 2003; Pistore *et al.*, 2005].

Recent works address the problem of the practicality of the proposed solutions for web service composition. For instance, [Traverso and Pistore, 2004; Pistore *et al.*, 2005] propose a framework where web services are modeled with stateful, nondeterministic, and partially observable behaviors, and planning techniques based on symbolic model checking are used to address the scalability problem. However, these techniques work under the rather unrealistic assumption that web services can exchange only a very small number of data values. For instance, as the experimental results reported in [Traverso and Pistore, 2004; Pistore *et al.*, 2005] show, reasonable performances are obtained for web services whose variables can contain only two values. This amounts to say that amazon.com could sell just two books!

Luckily enough, as already stated in [Pistore *et al.*, 2005], the composition solution should not depend on the actual data exchanged among web services, in the same way as the operations that one has to perform to buy a book do not depend on the precise book one wants to buy. The flow of operations and interactions depends instead on whether the desired book is available or not, on whether its cost is affordable, and so on. The relevant issues for interacting with amazon.com are whether one *knows whether* the book is available or not, whether one *knows the value of* its price, and so on. The hope here is to apply *planning techniques at the knowledge level*, in the style of those proposed in [Petrick and Bacchus, 2002], or used in [McIlraith and Son, 2002; Martinez and Lesperance, 2004]. This would make it possible to model only those features of the services which are relevant to compose them, and thus allow for an efficient automated composition.

Unfortunately, applying knowledge-level planning to solve the automated composition problem presents a major difficulty. While in [Petrick and Bacchus, 2002] the planning domain at the knowledge level is defined by hand, this is impractical for web service automated composition tasks. The knowledge level domain must be extracted automatically from the description of the services that are published, e.g., in standard languages, like BPEL4WS [Andrews *et al.*, 2003]. The problem is therefore to devise a proper knowledge level model, which is suited for the automated composition task, and which can be obtained automatically from the published descriptions of the web services.

In this paper we propose a novel approach to the problem

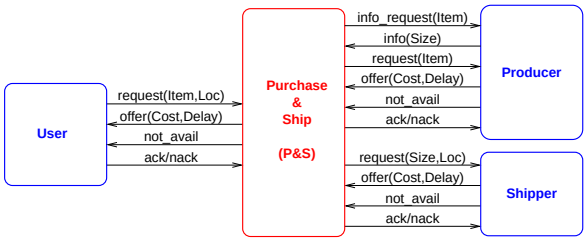


Figure 1: The Purchase & Ship example.

of automated composition that is based on planning at the knowledge level. We achieve this in the following steps:

- We formally define a knowledge level model of web services that can be obtained automatically from specifications in standard languages for modeling and implementing services. We show indeed how BPEL4WS processes can be automatically translated into their knowledge level models.
- We show how the automated composition problem can be described as a planning problem: from the knowledge level models of the available services and from the composition goal, we generate a planning problem such that a solution plan encodes the desired composition.
- We apply the planning techniques described in [Pistore *et al.*, 2005] to the knowledge level planning problem that we obtain from the previous steps. The key advantage w.r.t. [Pistore *et al.*, 2005] is that we do not have to deal with all the possible values of variables that are exchanged among services.
- We implement the proposed framework, and provide a preliminary experimental evaluation that clearly shows the benefits of our approach.

The paper is structured as follows. We first describe a simple example of composition of BPEL4WS processes, which we will use all along the paper (Section 2). We then briefly recall the framework for planning in asynchronous domains that has been first introduced in [Pistore *et al.*, 2005] (Section 3). In Section 4, we describe the knowledge level representation of web services, and the resulting planning framework at the knowledge level. We finally describe the experimental evaluation (Section 5), some conclusions, and related work.

2 Composition of BPEL processes

Our reference example is the Purchase and Ship (P&S hereafter) example introduced in [Pistore *et al.*, 2005; Traverso and Pistore, 2004].

Example 1 *The P&S example consists in providing a furniture purchase & ship service by combining two independent existing services, a furniture producer Producer and a delivery service Shipper. This way, the User, also described as a service, may directly ask the composite service P&S to purchase a given item and deliver it at a given place (for simplicity, we assume that the shipment origin is fixed and leave it implicit).*

The interactions with the existing services have to follow specific protocols. For instance, the interactions with the Shipper start with a request for transporting a product of a given size to a given location. This might not be possible, in which case the requester is notified, and the protocol terminates with failure. Otherwise, a cost and delivery time are computed and sent back to the requester. Then the Shipper waits for either an acceptance or a refusal of the offer from the invoker. In the former case, a delivery contract has been defined and the protocol terminates with success, while it terminates with failure in the latter case. Similar protocols are defined also for Producer and User. The messages exchanged among the involved services are described in Figure 1.

The P&S has the goal to sell home-delivered furniture (i.e., to reach the situation where the user has confirmed an order and the service has confirmed the corresponding suborders to producer and shipper), interacting with Shipper, Producer, and User according to their protocols. A typical interaction could be as follows:

1. *the User asks P&S for an article a, that he wants to be transported at location l;*
2. *P&S asks the Producer for the size, the cost, and how much time does it take to produce the article a;*
3. *P&S asks the Shipper for the price and time needed to transport an object of such a size to l;*
4. *P&S sends the User an offer which takes into account the overall cost (plus an added cost for P&S) and time to achieve its goal;*
5. *the User sends a confirmation of the order, which is dispatched by P&S to Shipper and Producer.*

This is however only the normal case, and other interactions should be considered, e.g., for the cases the producer and/or delivery services are not able to satisfy the request, or the user refuses the final offer.

With automated composition of web services we mean the generation of a new *composite service* (the P&S in our case) that interacts with a set of existing *component services* (Shipper, Producer, and User in our case) in order to satisfy a given *composition goal* (sell home-delivered furniture).

We assume that the interaction protocols of the component services, as well as the composite service, are described as BPEL4WS processes (see Figure 2 for a graphical representation of the Shipper BPEL4WS process). Notice that while existing services are described as abstract BPEL4WS processes (providing essentially the communication protocol), the synthesized service is an executable BPEL4WS process exporting all the details to be directly deployed and run. A BPEL4WS description specifies the types and internal variables of a service, and its input and output capabilities. The behavior of the process is described using input (`receive`) and output (`invoke`, `reply`) activities combined by standard constructs, such as sequences, loops, parallel executions, conditional choices, and nondeterministic choices.

From a BPEL4WS process, we can automatically extract a formal model of the interactions with the service, covering both the static aspects (e.g., its communication channels) and the behavioral aspects (defined in term of transition steps). For the moment, the translation is restricted to a

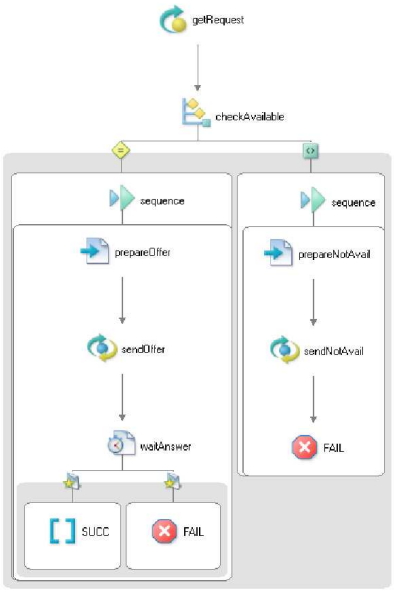


Figure 2: The Shipper BPEL4WS process.

```
PROCESS  Shipper;
TYPE
   Size;  Location;  Cost;  Delay;
INPUT
   request(Size, Location);  ack();  nack();
OUTPUT
   offer(Delay, Cost);  not_avail();
FUNC
   costOf(Size, Location) : Cost;  delayOf(Size, Location) : Delay;
VARIABLE
   pc: { START, getRequest, checkAvailable, end_checkAvailable, sequence1, sequence2,
       prepareOffer, sendOffer, waitAnswer, endWaitAnswer, empty1, prepareNotAvail,
       sendNotAvail, SUCC, FAIL };
   customer_size: Size;      customer_loc: Location;
   offer_delay: Delay;       offer_cost: Cost;
INIT
   pc := START;
TRANS
   pc = START  -[TAU]-> pc := getRequest;
   pc = getRequest  -[INPUT request(customer_size, customer_loc)]-> pc := checkAvailable,
   pc = checkAvailable  -[TAU]-> pc := sequence1;
   pc = checkAvailable  -[TAU]-> pc := sequence2;
   pc = sequence1  -[TAU]-> pc := prepareOffer;
   pc = prepareOffer  -[TAU]-> pc := sendOffer,
                                 offer_cost:=costOf(customer_size,customer_loc),
                                 offer_delay:=delayOf(customer_size,customer_loc);
   pc = sendOffer  -[OUTPUT offer(offer_cost, offer_delay)]-> pc := waitAnswer;
   pc = waitAnswer  -[INPUT nack]-> pc := FAIL;
   pc = waitAnswer  -[INPUT ack]-> pc := empty1;
   pc = empty1  -[TAU]-> pc := endWaitAnswer;
   pc = endWaitAnswer  -[TAU]-> pc := endCheckAvailable;
   pc = endCheckAvailable  -[TAU]-> pc := SUCC;
   pc = sequence2  -[TAU]-> pc := prepareNotAvail;
   pc = prepareNotAvail  -[TAU]-> pc := sendNotAvail;
   pc = sendNotAvail  -[OUTPUT not_avail]-> pc :=FAIL;
```

Figure 3: A formal model of the Shipper.

subset of BPEL4WS processes: we support all BPEL4WS *basic* and *structured activities*, like `invoke`, `reply`, `receive`, `sequence`, `switch`, `while`, `flow` (without `links`) and `pick`; moreover we support `assignments` and a limited form of `correlation`. Our next steps will be dealing with `scopes` and with `fault`, `event` and `compensation handlers`.

For lack of space we omit the formal definition of this translation (details can be found at `http://astroproject.org/`) but we illustrate it in the case of the Shipper process.

Example 2 *Figure 3 shows a formal model of the Shipper, automatically extracted from the* BPEL4WS *process of Figure 2. The Shipper process is characterized by a set of abstract types used in the interactions (e.g.,* `Size`, `Cost`*), by a set of inputs and outputs (e.g.,* `request` *and* `offer`*, respectively), and by a set of typed functions used to manipulate internal variables (e.g., function* `costOf`*, which is used to obtain the cost of a particular shipping request). Variables are used to store information on the state of the process. In Figure 3, variables of abstract types (e.g.,* `customer_size`*) store values used in the communications, while an additional variable* `pc` *implements a "program counter" that holds the current execution step of the service. The model describes the evolution of the process through a set of possible transitions, each corresponding to a "step" in the* BPEL4WS *process; each transition defines an applicability condition, a firing action, and an effect (defined as a list of assignment to variables). Possible actions are inputs, outputs, and a special action TAU which is used to model internal evolutions of the process, such as assignments and decision making (e.g., the transition from state* `prepareOffer`*, where the internal functions* `costOf` *and* `delayOf` *are used to compute the shipping price and delivery time).*

3 A Planning Framework for Service Composition

The work in [Pistore *et al.*, 2005] presents a formal framework for the automated composition of web services which is based on planning techniques: component services define the planning domain, composition requirements are formalized as a planning goal, and planning algorithms are used to generate the composite service. Due to the nature of web services, the resulting planning domain is nondeterministic and partially observable. The framework of [Pistore *et al.*, 2005] differs from other planning frameworks since it assumes an asynchronous, message-based interaction between the domain (encoding the component services) and the plan (encoding the composite service). More precisely, the planning domain is modeled as a *state transition system* (STS from now on) that can be in one of its possible states (a subset of which are *initial*) and can evolve to new states as a result of performing some actions. In particular, *input* actions represent messages sent to the component services, while *output* actions are messages received from the component services. *Private* actions are actions that the composite service can perform internally, without interacting with the component services[1], while the special action τ is used to model internal evolutions of the component services which are not visible to the service user. Finally, a labeling function associates to each state the set of properties $\mathcal{P}rop$ holding in that state.

[1] As we will see, private actions are used to model operations such as computing the total cost for the user from the costs received from the producer and from the shipper. Private actions are specifi c of the knowledge-level approach presented here and for this reason they do not appear in [Pistore *et al.*, 2005]. The extension of the theory of [Pistore *et al.*, 2005] to private actions is straightforward.

Definition 1 (state transition system (STS))
A state transition system Σ *is a tuple* $\langle \mathcal{S}, \mathcal{S}^0, \mathcal{I}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{L} \rangle$ *where:*

- $\mathcal{S}$ *is the finite set of states;*
- $\mathcal{S}^0 \subseteq \mathcal{S}$ *is the set of initial states;*
- $\mathcal{I}$ *is the finite set of input actions;*
- $\mathcal{O}$ *is the finite set of output actions;*
- $\mathcal{A}$ *is the finite set of private actions;*
- $\mathcal{R} \subseteq \mathcal{S} \times (\mathcal{I} \cup \mathcal{O} \cup \mathcal{A} \cup \{\tau\}) \times \mathcal{S}$ *is the transition relation;*
- $\mathcal{L} : \mathcal{S} \rightarrow 2^{\mathcal{P}rop}$ *is the labeling function.*

In a composition problem, the composite service is defined as a "controller" Σ_c (also described as a STS), which interacts with the domain Σ, orchestrating the component services by invoking their operations and handling results. We now recall the formal definition of the behavior of a STS Σ when controlled by Σ_c.

Definition 2 (controlled system)
Let $\Sigma = \langle \mathcal{S}, \mathcal{S}^0, \mathcal{I}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{L} \rangle$ *and* $\Sigma_c = \langle \mathcal{S}_c, \mathcal{S}_c^0, \mathcal{O}, \mathcal{I}, \mathcal{A}, \mathcal{R}_c, \mathcal{L}_\emptyset \rangle$ *be two state transition systems, where* $\mathcal{L}_\emptyset(s_c) = \emptyset$ *for all* $s_c \in \mathcal{S}_c$. *The STS* $\Sigma_c \triangleright \Sigma$, *describing the behaviors of system* Σ *when controlled by* Σ_c, *is defined as:*
$\Sigma_c \triangleright \Sigma = \langle \mathcal{S}_c \times \mathcal{S}, \mathcal{S}_c^0 \times \mathcal{S}^0, \mathcal{I}, \mathcal{O}, \mathcal{A}, \mathcal{R}_c \triangleright \mathcal{R}, \mathcal{L} \rangle$
where:

- $\langle (s_c, s), \tau, (s_c', s) \rangle \in (\mathcal{R}_c \triangleright \mathcal{R})$ *if* $\langle s_c, \tau, s_c' \rangle \in \mathcal{R}_c$;
- $\langle (s_c, s), \tau, (s_c, s') \rangle \in (\mathcal{R}_c \triangleright \mathcal{R})$ *if* $\langle s, \tau, s' \rangle \in \mathcal{R}$;
- $\langle (s_c, s), a, (s_c', s') \rangle \in (\mathcal{R}_c \triangleright \mathcal{R})$, *with* $a \neq \tau$, *if* $\langle s_c, a, s_c' \rangle \in \mathcal{R}_c$ *and* $\langle s, a, s' \rangle \in \mathcal{R}$.

Due to the asynchronous nature of web service interactions, and in order to guarantee a correct execution of the composite service, we need to rule out explicitly the cases where the sender is ready to send a message that the receiver is not able to accept. According to [Pistore *et al.*, 2005], a state s is able to accept a message a if there exists some successor s' of s, reachable from s through a (possibly empty) sequence of τ transitions, such that an input transition labeled with a can be performed in s'. This intuition is captured in the following definition, where we denote by τ-closure(s) the set of states reachable from s through a chain of τ transitions. For what concerns private actions, since they correspond to internal operations of the composite service, we simply assume that they are executable in the current state of Σ.

Definition 3 (deadlock-free controller)
Let $\Sigma = \langle \mathcal{S}, \mathcal{S}^0, \mathcal{I}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{L} \rangle$ *be a STS and* $\Sigma_c = \langle \mathcal{S}_c, \mathcal{S}_c^0, \mathcal{O}, \mathcal{I}, \mathcal{A}, \mathcal{R}_c, \mathcal{L}_\emptyset \rangle$ *be a controller for* Σ. Σ_c *is said to be* deadlock free *for* Σ *if all states* $(s_c, s) \in \mathcal{S}_c \times \mathcal{S}$ *that are reachable from the initial states of* $\Sigma_c \triangleright \Sigma$ *satisfy the following conditions:*

- *if* $\langle s, a, s' \rangle \in \mathcal{R}$ *with* $a \in \mathcal{O}$ *then there is some* $s_c' \in \tau$*-closure*(s_c) *such that* $\langle s_c', a, s_c'' \rangle \in \mathcal{R}_c$ *for some* $s_c'' \in \mathcal{S}_c$;
- *if* $\langle s_c, a, s_c' \rangle \in \mathcal{R}_c$ *with* $a \in \mathcal{I}$ *then there is some* $s' \in \tau$*-closure*(s) *such that* $\langle s', a, s'' \rangle \in \mathcal{R}$ *for some* $s'' \in \mathcal{S}$.
- *if* $\langle s_c, a, s_c' \rangle \in \mathcal{R}_c$ *with* $a \in \mathcal{A}$ *then* $\langle s, a, s' \rangle \in \mathcal{R}$ *for some* $s' \in \mathcal{S}$.

In [Pistore *et al.*, 2005], the composition problem for domain Σ and composition goal ρ is defined as the problem of finding a deadlock-free STS Σ_c such that $\Sigma_c \triangleright \Sigma$ satisfies ρ. Planning techniques, based on the *planning as model checking* framework, are used to solve this problem, and the experimental results show the effectiveness of this approach.

However, the approach of [Pistore *et al.*, 2005] requires a finite set of data values for all data types (e.g., only a finite number of articles, locations, costs... can be defined in the P&S domain). This assumption guarantees that the formal model of a BPEL4WS process such as the one in Figure 3 can be mapped into a finite-state STS. Moreover, the number of data values impact dramatically on the size of the generated STS, and very small sets of values need to be used to allow for an effective plan generation: for instance, all experiments in [Pistore *et al.*, 2005] assume only two values for each data type. In the following we show how to adapt the framework of [Pistore *et al.*, 2005] to a knowledge-level planning approach in order to remove this restriction and allow for automated composition with a realistic (or even infinite) number of data values.

4 Service Composition via Knowledge-Level Planning

The key aspect for extending the approach of [Pistore *et al.*, 2005] to the knowledge level is the definition of an appropriate model for providing a knowledge level description of the component services. In this section we formally define such a model in terms of a suitable knowledge base (from now on KB). Then we show how to construct the STS corresponding to the planning domain by composing the knowledge bases of the component services.

Definition 4 (Knowledge Base)
A knowledge base KB *is a set of propositions of the following form:*

- $K_V(x)$ *where* x *is a variable with an abstract type;*
- $K(x = v)$ *where x is an enumerative variable and v is one of its possible values;*
- $K(x = y)$ *where x and y are two variables with the same type;*
- $K(x = f(y_1, \ldots, y_n))$ *where* $x, y_1, \ldots, y_n$ *are variables with an abstract type and* f *is a function compatible with the types of* $x, y_1, \ldots, y_n$.

With $K(p)$ we mean that we know that proposition p is true and with $K_V(x)$ that we know the value of the variable x. This definition of knowledge base is very simple; still, our experiments show that it is powerful enough to model web service composition problems.

We say that a knowledge base KB is *consistent* if it does not contain contradictory knowledge propositions such as $K(x = v)$ and $K(x = v')$, with $v \neq v'$. We say that KB is *closed under deduction* if it contains all the propositions that can be deduced from the propositions in KB; for instance a

KB containing both $K_V(x)$ and $K(x = y)$ should contain also $K_V(y)$.

The knowledge base KB of a component service is obtained from the variables, functions and types of the service.

Example 3 *An example of* knowledge base *for the Shipper process in Figure 3 is:*

KB={K(pc = waitAnswer), K_V(customer_size), K_V(customer_loc),
K(offer_cost = costOf(customer_size, customer_loc)),
K(offer_delay = delayOf(customer_size, customer_loc)),
K_V(offer_cost), K_V(offer_delay)}.

In the following we describe when a transition can be executed in a knowledge base KB and how its execution affects KB.

We model a transition t, such as those presented in Figure 3, as a triple (C, a, E) where $C = (c_1 \wedge \ldots \wedge c_n)$ are its conditions, a is its firing action and $E = (e_1; \ldots; e_n)$ are its effects. We start by defining the auxiliary *restriction* and *update* operations.

The *restriction* of a knowledge base KB with a condition C, denoted with $restrict(KB, C)$, is performed adding to KB the knowledge obtained from C and closing under deduction; for instance:

restrict({K(x = y)}, y = z) =
{K(x = y), K(y = z), K(x = z)}.

The *update* of a knowledge base KB with an effect E, denoted with $update(KB, E)$, consists in performing the following steps: for each assignment in E, remove from KB the knowledge we had on the modified variable, add the knowledge derived from the assignment and close the KB under deduction. For instance:

update({K(x = y)}, z := x; x := w) =
update({K(x = y), K(z = x), K(z = y)}, x := w) =
{K(z = y), K(x = w)}.

We now give the definitions of *applicability* and *execution* which depend on a service transition t and on an action a_c performed by a peer interacting with the service; the firing action a in t and the peer action a_c correspond to the same action except that the former is instantiated on service variables, while the latter on variables of the peer. Consider for instance the following example where action request(s, l) performed by the peer corresponds to the Shipper request(customer_size, customer_loc).

Example 4 *Let's consider the input transition* t *of the Shipper*

pc = getRequest
-[INPUT request(customer_size, customer_loc)]->
pc := checkAvailable

and suppose that our current knowledge base is:

KB = { K(pc = getRequest), K_V(s), K_V(l) } .

where s:Size *and* l:Location *are additional variables. Action* request(s, l)*, corresponding to invoking the Shipper action using* s *and* l *as parameters, is applicable in* KB *since we know the values of* s *and* l *and the condition of transition* t *is obviously consistent with* KB*.*

The knowledge base obtained by executing on KB *transition* t *and action* request(s, l)*, is defined as follows:*

- *we restrict* KB *with the knowledge* K(pc = getRequest) *associated to the transition condition; in this specific case* KB *remains unchanged;*
- *then we update* KB *with the knowledge carried by the input action on the variables used as action parameters; this consists in removing from* KB *all the knowledge we had on* customer_size *and* customer_loc*, adding the new knowledge* K(customer_size = s), K(customer_loc = l) *and closing under deduction; we obtain:*

KB′ = { K(pc = getRequest), K_V(s),
K_V(l), K_V(customer_size), K_V(customer_loc),
K(customer_size = s), K(customer_loc = l) } ;

- *finally we update the knowledge base with the effects, removing from* KB′ *all the knowledge we had on the variables modified by the assignments, adding the new knowledge (in this case* K(pc := checkAvailable)*) and closing under deduction; we obtain:*

KB′′ = { K(pc = checkAvailable), K_V(s),
K_V(l), K_V(customer_size), K_V(customer_loc),
K(customer_size = s), K(customer_loc = l) } .

Definition 5 (KL Applicability and Execution) *A transition* $t{=}(C, a, E)$ *and a corresponding action* a_c *are* applicable *in* KB*, written* $applicable[t](KB, a_c)$ *if:*

- $a_c{=}i(x_1, \ldots, x_n)$ *and* $a{=}i(y_1, \ldots, y_n)$*, where* i *is an input,* $K_V(x_1), \ldots, K_V(x_n) \in KB$ *and* $restrict(KB, C)$ *is consistent; or*
- $a_c{=}o(x_1, \ldots, x_n)$ *and* $a{=}o(y_1, \ldots, y_n)$*, where* o *is an output, and* $restrict(KB, C)$ *is consistent; or*
- $a_c{=}a{=}\tau$ *and* $restrict(KB, C)$ *is consistent.*

If $applicable[t](KB, a_c)$*, then we denote with* $KB'{=}exec[t](KB, a_c)$ *the* execution *on* KB *of* t *and* a_c*, defined as follows:*

- *if* $a_c{=}\tau$ *and* $t{=}(C, \tau, E)$ *then* $KB'{=}update(KB'', E)$*, where* $KB''{=}restrict(KB, C)$*;*
- *if* $a_c{=}i(x_1, \ldots, x_n)$*, where* i *is an input, and* $t{=}(C, i(y_1, \ldots, y_n), E)$ *then* $KB'{=}update(KB'', y_1{:=}x_1; \ldots; y_n{:=}x_n; E)$*, where* $KB''{=}restrict(KB, C)$*;*
- *if* $a_c{=}o(x_1, \ldots, x_n)$*, where* o *is an output, and* $t{=}(C, o(y_1, \ldots, y_n), E)$ *then* $KB'{=}update(KB'', x_1{:=}y_1; \ldots; x_n{:=}y_n; E)$*, where* $KB''{=}restrict(KB, C) \cup \{K_V(y_1), \ldots, K_V(y_n)\}$.

Notice that the execution of a transition increases the knowledge in KB, not only with the information in the effects, but also with those in the condition. Indeed, if the transition is executed, this means that the condition is known to hold.

The planning domain at the knowledge level is constructed from the knowledge-level models of each component service and from a knowledge-level representation of the composition goal. The latter defines which are the variables and the functions of the composite service, like, for instance, the cost of the offer to the user, or a special function that adds a mark up to the sum of the costs of the shipper and producer. We

call these variables and functions, *goal variables* and *goal functions*.

Given a composition goal, we automatically generate its knowledge level representation that declares what the composite service must know and how goal variables and functions must be related with the variables and functions of the component services.

Example 5 *A possible (very simple) composition goal for our reference example is:*

TryReach

```
user.pc = SUCC ∧ producer.pc = SUCC ∧ shipper.pc = SUCC ∧
user.offer_cost =
  addCost(producer.costOf(user.article),
  shipper.costOf(producer.sizeOf(user.article), user.location))
```

The goal declares that we want all the services to reach the situation where the order has been confirmed. Moreover it states that the offered cost must be obtained by applying the function `addCost` *to the costs offered by the producer and the shipper. The operator* ***TryReach*** *is one of the modal operators provided by the* EAGLE *goal specification language. It requires that the plan reaches the goal condition whenever possible in the domain. For further details, see [Pistore* et al.*, 2005]*

To obtain the knowledge level goal, we flatten the functions introducing auxiliary variables until only basic propositions are left:

TryReach

```
user.pc = SUCC ∧ producer.pc = SUCC ∧ shipper.pc = SUCC ∧
user.offer_cost = goal.added_cost ∧
goal.added_cost = addCost(goal.prod_cost, goal.ship_cost) ∧
goal.prod_cost = producer.costOf(goal.user_art) ∧
goal.ship_cost = shipper.costOf(goal.prod_size, goal.user_art) ∧
goal.user_art = user.article ∧
goal.prod_size = producer.sizeOf(goal.user_art) ∧
goal.user_loc = user.location
```

From this flattened goal we can extract the goal variables: `goal.added_cost`, `goal.prod_cost`, `goal.ship_cost`, `goal.prod_size`, `goal.user_art`, *and* `goal.user_loc`*; and the goal function* `goal.addCost(Cost, Cost):Cost`.

The goal can then be automatically translated into its corresponding knowledge level goal:

TryReach

```
K(user.pc = SUCC) ∧ K(producer.pc = SUCC) ∧ K(shipper.pc = SUCC) ∧
K(user.offer_cost = goal.added_cost)∧
K(goal.added_cost = goal.addCost(goal.prod_cost, goal.ship_cost))∧
K(goal.prod_cost = producer.costOf(goal.user_art))∧
K(goal.ship_cost = shipper.costOf(goal.prod_size, goal.user_art))∧
K(goal.user_art = user.article)∧
K(goal.prod_size = producer.sizeOf(goal.user_art))∧
K(goal.user_loc = user.location)
```

The knowledge-level representation of the composition goal defines therefore a further knowledge base, that we call the *knowledge base of the goal*.

The knowledge-level planning domain Σ is obtained by combining the knowledge bases of the component services and the knowledge base of the goal, by instantiating the input and output actions of the component services on goal variables, and by adding the private actions obtained by applying goal functions to goal variables.

Definition 6 (Knowledge-level Planning Domain) *The planning domain Σ for a composition problem is an STS $\langle \mathcal{S}, \mathcal{S}^0, \mathcal{I}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{L} \rangle$ defined as follows:*

- *the set of states $\mathcal{S}$ are all the possible KB defined on the set of typed variables $X = \bigcup_{i=0\ldots n} X_i$ and on the set of typed functions $F = \bigcup_{i=0\ldots n} F_i$, where $X_1, \ldots, X_n$ and $F_1, \ldots, F_n$ are the variables and functions of the component services, while X_0 and F_0 are those of the composition goal;*
- *$\mathcal{S}^0 \subseteq \mathcal{S}$ is the set of initial states corresponding to the initial knowledge bases $KB_0^1, \ldots, KB_0^n$, obtained from the initial assignments of the component services;*
- *$\mathcal{I}$ is the set of input actions $i(x_1, \ldots, x_n)$ such that $i(y_1, \ldots, y_n)$ is an input action in a transition of a component service and $x_1, \ldots, x_n$ are goal variables with the same type of service variables $y_1, \ldots, y_n$;*
- *$\mathcal{O}$ is the set of output actions $o(x_1, \ldots, x_n)$ such that $o(y_1, \ldots, y_n)$ is an output action in a transition of a component service and $x_1, \ldots, x_n$ are goal variables with the same type of service variables $y_1, \ldots, y_n$;*
- *$\mathcal{A}$ is the set of private actions $x_0{:=}f(x_1, \ldots, x_n)$ where f is a goal function and $x_0, x_1, \ldots, x_n$ are goal variables compatible with the type of f;*
- *$\mathcal{R}$ is the set of transitions $r{=}\langle s, a_c, s' \rangle$, with $s, s' \in \mathcal{S}$, such that:*
  - *if a_c is an input, output or τ action, then there exists a $t{=}(C, a, E)$ in the sets of transitions of the component services such that $applicable[t](s, a_c)$ and $s'{=}exec[t](s, a_c)$;*
  - *if a_c is a private action of the form $x_0{:=}f(x_1, \ldots, x_n)$, then $K_V(x_1), \ldots, K_V(x_n) \in s$ and $s'{=}update(s, x_0{:=}f(x_1, \ldots, x_n))$;*
- *$\mathcal{L}$ is the trivial function associating to each state the set of propositions that hold in that state.*

Given the domain Σ described above, we can apply the approach presented in [Pistore *et al.*, 2005] and obtain a deadlock free Σ_c that controls Σ by satisfying the composition requirements ρ. Despite of the fact that the synthesized controller Σ_c is modelled at the knowledge level, its elementary actions model communication with the component services (sending and receiving of messages) and manipulation of goal variables; given this, it is straightforward to obtain the executable BPEL4WS composite service from Σ_c.

5 Experimental Evaluation

We have implemented the proposed approach, and used the planning as symbolic model checking techniques presented in [Pistore *et al.*, 2005] to perform an experimental evaluation. All the tests have run on a 3 GHz Xeon PC, limiting memory usage to 512MBytes, and with a CPU timeout of 1000 seconds.

We first considered the P&S example explained in the previous sections, which, in spite of the reduced number of components, requires a rather intricated protocol to be established for achieving the goal. Figure 4 shows the results of our

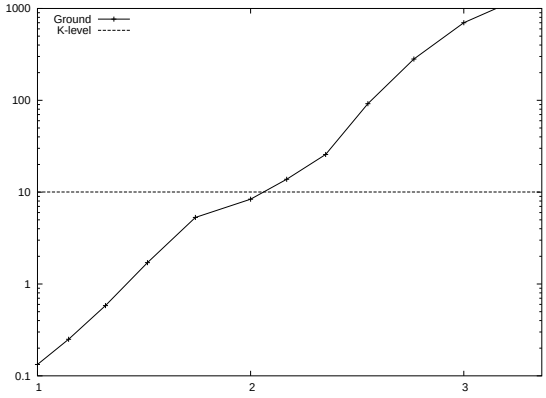


Figure 4: Experiments with P&S.

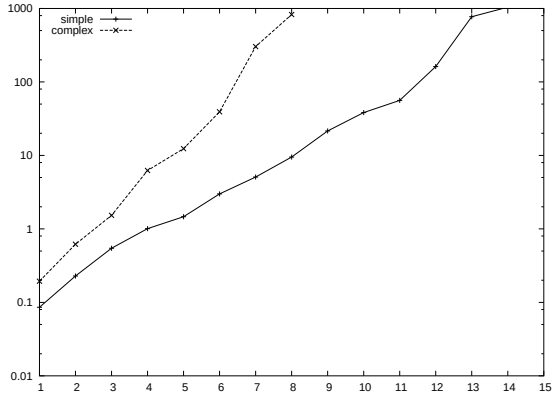


Figure 5: Experiments with parametrized domains.

experiments. The same planning engine is run to perform knowledge-level composition and ground level compositions, where different ranges of values can be produced and exchanged by the Shipper, Producer and User. The horizontal axis reports the cardinality $\overline{n}$ of the data types (i.e. Size, Location, Cost, Delay) handled by the services. We also consider intermediate cases where we have, for instance, two possible values for the Cost and only one value for the Size), reporting the average cardinality in the figure. On the vertical axis, we report the composition time. As expected, ground level composition is only feasible for the unrealistic cases where processes may exchange only data with 2 or at most 3 values. Indeed, the time for ground composition grows exponentially with the cardinality of the data types, and even the simple case where types have cardinality 4 is unmanageable. On the contrary, knowledge-level composition takes about 10 seconds to complete, with a performance similar to that of the ground level for $\overline{n} = 2$. This is a reasonable result, since, basically, binary variables at the ground level correspond to (binary) knowledge atoms at the knowledge-level.

To evaluate the scalability of the knowledge-level approach when the number of component services grows, we perform two sets of experiments, considering a generalization of the example domain that involves a set of component services. In our first set of experiments, each component is represented by a very simple abstract BPEL4WS process that is requested to provide a service and can respond either by performing the service, or by refusing. The composition requirement is that either all services end successfully, or a failure is reported to the invoker of the composed service. Figure 5 reports the knowledge-level composition times, for increasing number of services to be composed (indicated on the horizontal axis). The composition achieves results comparable to those reported in [Pistore *et al.*, 2005], where ground composition is performed only considering the case of types with range of cardinality 2. We manage to compose 14 services in 20 minutes.

In the second set of experiments, also reported in Figure 5, we make the protocol more complex, by requiring a higher degree of interleaving between components. Here, the interactions with each component are more complex than a single invoke-response step, and, to achieve the goal, it is necessary to carry out interactions with all components in an interleaved way. Such interleaving is common in the P&S example where, e.g., the P&S cannot confirm the order to the producer if shipping is not available or if the user does not accept the offer. The increased complexity reflects on the complexity of the composition; however, compositions of reasonable complexity (up to 8 services) can be achieved within 20 minutes. Even in this case, automated composition takes a rather low amount of time, surely faster than manual development of BPEL4WS code.

6 Conclusions and Related Work

In this paper, we have shown how automated composition can be obtained by translating automatically process-level descriptions of web services, e.g., BPEL4WS processes, to a planning problem that describes the interactions among services at the knowledge level. Our solution exploits the framework for planning in asynchronous domains presented in [Pistore *et al.*, 2005]. We show experimentally how the knowledge level approach increases significantly the practical applicability of this framework.

The idea of planning at the knowledge level is not new, and the solution that we propose is based on the idea presented in [Petrick and Bacchus, 2002]. Notice however that our work addresses an orthogonal problem, i.e., the problem of how a knowledge-level planning domain can be automatically generated from a set of BPEL4WS processes that describe web services. Moreover, at the technical level, our works differs from those proposed in [Petrick and Bacchus, 2002] in the kind of information that we represent and store in the knowledge bases, as well as in the knowledge-level planning domain that we automatically generate, and in the planning algorithm we exploit.

The work in [Martinez and Lesperance, 2004] shows how to use the knowledge level planner presented in [Petrick and Bacchus, 2002] to solve service composition problems. However they deal only with atomic services (services exporting a single operation instead of a protocol) and the knowledge level domain (e.g. domain action specification, update rules) is written by hand.

Different automated planning techniques have been pro-

posed to tackle the problem of service composition, see, e.g., [Wu *et al.*, 2003; Sheshagiri *et al.*, 2003]. However, most of them cannot deal with the problem that we address in this paper, since they cannot deal with conditional outputs, uncertain action effects, and partial observability. A remarkable exception is the work described in [Narayanan and McIlraith, 2002; McIlraith and Son, 2002; McIlraith and Fadel, 2002], which instead deals with sensing actions and knowledge level predicates. However, even in this work, the knowledge level domain is given by hand, and the problem of devising a knowledge base that can be generated automatically from BPEL4WS web services descriptions is not addressed. Moreover, the composition problem is limited to sequential compositions of services.

The work in [Hull *et al.*, 2003] presents a formal framework for composing e-services from behavioral descriptions given in terms of automata; they focus on the theoretical foundations, without providing practical implementations. Moreover, the considered e-composition problem is fundamentally different from our process-level composition, since it is seen as the problem of generating a set of rules coordinating the execution of the available services. No concrete and executable process is generated with that approach. This is the main difference also with the work described in [Berardi *et al.*, 2003].

In the future, we plan to extend the work to the automated composition of semantic web services, e.g., described in OWL-S [OWL-S, 2003], along the lines of the work in [Traverso and Pistore, 2004], which at the moment is not based on knowledge-level planning. We will also explore variants of our knowledge base, so to allow flexible ways to model different features of ground level domains.

Acknowledgements This work is partially funded by the MIUR-FIRB project RBNE0195K5, "Knowledge Level Automated Software Engineering", and by the MIUR-PRIN 2004 project "Advanced Artificial Intelligence Systems for Web Services". The authors want to thank all members of the Astro project (`http://astroproject.org/`) for their collaboration and their feedback.

References

[Andrews *et al.*, 2003] T. Andrews, F. Curbera, H. Dolakia, J. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weeravarana. Business Process Execution Language for Web Services (version 1.1), 2003.

[Berardi *et al.*, 2003] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic Composition of E-Services that Export their Behaviour. In *Proc. ICSOC'03*, 2003.

[Foster *et al.*, 2003] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based Verification of Web Service Compositions. In *Proc. ASE'03*, 2003.

[Fu *et al.*, 2004] X. Fu, T. Bultan, and J. Su. Analysis of Interacting BPEL Web Services. In *Proc. WWW'04*, 2004.

[Hull *et al.*, 2003] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-Services: A Look Behind the Curtain. In *Proc. PODS'03*, 2003.

[Koehler and Srivastava, 2003] J. Koehler and B. Srivastava. Web Service Composition: Current Solutions and Open Problems. In *Proc. of ICAPS'03 Workshop on Planning for Web Services*, 2003.

[Martinez and Lesperance, 2004] E. Martinez and Y. Lesperance. Web Service Composition as a Planning Task: Experiments using Knowledge-Based Planning. In *Proc. of ICAPS'04 Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[McIlraith and Fadel, 2002] S. McIlraith and R. Fadel. Planning with Complex Actions. In *Proc. NMR'02*, 2002.

[McIlraith and Son, 2002] S. McIlraith and S. Son. Adapting Golog for Composition of Semantic Web Services. In *Proc. KR'02*, 2002.

[Narayanan and McIlraith, 2002] S. Narayanan and S. McIlraith. Simulation, Verification and Automated Composition of Web Services. In *Proc. WWW'02*, 2002.

[OWL-S, 2003] OWL-S. OWL-S: Semantic Markup for Web Services (OWL-S version 1.0), 2003.

[Petrick and Bacchus, 2002] R. Petrick and F. Bacchus. A Knowledge-Based Approach to Planning with Incomplete Information and Sensing. In *Proc. AIPS'02*, 2002.

[Pistore *et al.*, 2005] M. Pistore, P. Traverso, and P. Bertoli. Automated Composition of Web Services by Planning in Asynchronous Domains. In *Proc. ICAPS'05*, 2005.

[Sheshagiri *et al.*, 2003] M. Sheshagiri, M. desJardins, and T. Finin. A Planner for Composing Services Described in DAML-S. In *Proc. AAMAS'03*, 2003.

[Traverso and Pistore, 2004] P. Traverso and M. Pistore. Automated Composition of Semantic Web Services into Executable Processes. In *Proc. ISWC'04*, 2004.

[Wu *et al.*, 2003] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S Web Services Composition using SHOP2. In *Proc. ISWC'03*, 2003.

Conditional Planning in the Discrete Belief Space

Jussi Rintanen*
Albert-Ludwigs-Universität Freiburg, Institut für Informatik
Georges-Köhler-Allee, 79110 Freiburg im Breisgau
Germany

Abstract

Probabilistic planning with observability restrictions, as formalized for example as partially observable Markov decision processes (POMDP), has a wide range of applications, but it is computationally extremely difficult. For POMDPs, the most general decision problems about existence of policies satisfying certain properties are undecidable.

We consider a computationally easier form of planning that ignores exact probabilities, and give an algorithm for a class of planning problems with partial observability. We show that the basic backup step in the algorithm is NP-complete. Then we proceed to give an algorithm for the backup step, and demonstrate how it can be used as a basis of an efficient algorithm for constructing plans.

1 Introduction

When the sequence of states that will be visited during plan execution cannot be exactly predicted, for example because of nondeterminism, it is necessary to produce plans that apply different actions depending on how the plan execution has proceeded so far. Such plans are called conditional plans.

Construction of conditional plans is particularly difficult when there is no full observability; that is, when during plan execution it is not possible to uniquely determine what the current state of the world is. Planning problems having this property are said to be partially observable, and their solution requires that the sets of possible current world states – the belief states – are (implicitly) maintained during plan execution and (implicitly) represented by a plan.

The earliest work on planning with partial observability was in the framework of partially observable Markov decision processes (POMDPs) [Smallwood and Sondik, 1973; Kaelbling *et al.*, 1998]. Planning with POMDPs is computationally difficult. For unbounded horizon lengths an unbounded number of probability distributions corresponding to belief states needs to be considered, and finding optimal plans is not in general solvable [Madani *et al.*, 1999]. A natural approach for easing the computational difficulty of POMDP planning is to consider horizons of a bounded length [Mundhenk *et al.*, 2000]. A second approach which has been pursued with algorithms for conditional planning [Weld *et al.*, 1998; Bonet and Geffner, 2000; Bertoli *et al.*, 2001], ignores probabilities and hence directly yields a finitary problem. Main decision problems related to non-probabilistic planning with partial observability are 2-EXP-complete [Rintanen, 2004a].

A main difference between POMDPs and corresponding non-probabilistic problems is that the latter do not use probabilistic notions like success probability or expected cost, and require that a plan must reach the goals with certainty. An implication of success probability 1 is that uncertainty about observations and sensing can be ignored: if an observation is correct with a probability strictly less than 1 then it is as good as no observation at all.

For this planning problem we present an iterative algorithm that has some resemblance to iterative algorithms for solving POMDPs. The algorithm maintains a data structure representing those belief states for which a conditional plan has been shown to exist. Initially this data structure represents those belief states consisting of goal states only. Then this data structure is repeatedly extended by performing search backwards from the goal belief states.

The structure of the paper is as follows. Section 2 defines the planning problem. Sections 3 and 4 respectively describe the formal framework and analyze its properties. Section 5 proposes a planning algorithm and Section 6 presents experimental results obtained with an implementation of the algorithm. Section 7 concludes the paper.

2 The Planning Problem

In this section we present a formalization of planning in which states are atomic objects without internal structure.

Definition 1 *A* problem instance *is* $\langle S, I, O, G, P\rangle$ *where* S *is the set of states,* $I \subseteq S$ *is the set of initial states,* O *is the set of actions* $o \subseteq S \times S$*,* $G \subseteq S$ *is the set of goal states and* $P = (C_1, \ldots, C_n)$ *is a partition of* S *into classes of observationally indistinguishable states satisfying* $\bigcup\{C_1, \ldots, C_n\} = S$ *and* $C_i \cap C_j = \emptyset$ *for all* i, j *such that* $1 \leq i < j \leq n$*.*

Making an observation tells which set C_i the current state belongs to. Distinguishing states in a given C_i is not possible.

*This research was partly supported by DFG grant RI 1177/2-1.

An action is a relation between states and their successor states. An action o is *applicable* in a state s if sos' for some $s' \in S$. Define the *image* of a set B of states with respect to an action o as $\text{img}_o(B) = \{s' \in S | s \in B, sos'\}$. The *preimage* is $\text{preimg}_o(B) = \{s \in S | \emptyset \neq \text{img}_o(\{s\}) \subseteq B\}$, consisting of those states from which o is guaranteed to reach a state in B. An action o is deterministic if it is a partial function.

Plans are directed graphs with two kinds of nodes: action nodes and observation nodes.

Definition 2 *Let $\langle S, I, O, G, (C_1, \ldots, C_n)\rangle$ be a problem instance. A plan is a triple $\langle N, b, l\rangle$ where*

- *N is a finite set of nodes,*
- *$b \in N$ is the initial node,*
- *$l : N \to (O \times N) \cup 2^{2^S \times N}$ is a function that assigns each node an action and a successor node $\langle o, n\rangle \in O \times N$ or a set of states and successor nodes $\langle C, n\rangle \in 2^{2^S \times N}$ where $C = \bigcup\{C'_1, \ldots, C'_m\}$ for some $\{C'_1, \ldots, C'_m\} \subseteq \{C_1, \ldots, C_n\}$. In the first case the node is an action node and in the second an observation node.*

  For all $n \in N$ and $\{\langle C, m\rangle, \langle C', m'\rangle\} \subseteq l(n)$ the observations C and C' may not intersect: $C \cap C' = \emptyset$.

  Nodes with $l(n) = \emptyset$ are terminal.

We restrict to acyclic plans. Acyclicity means that the graph $\langle N, E\rangle$, where $\langle n, n'\rangle \in E$ iff $l(n) = \langle o, n'\rangle$ for some o or $\langle C, n'\rangle \in l(n)$ for some C, is acyclic.[1]

Plan execution starts from the initial node b and any of the initial states. For an action node with label $\langle o, n\rangle$ in state s execute o and continue from n and a state in $\text{img}_o(s)$. For an observation node identify $\langle C, n\rangle$ in the node label so that $s \in C$, and then continue from n and s. A plan solves a problem instance if all of its executions terminate in a terminal node and a goal state. Execution of an acyclic plan can have at most as many steps as there are nodes in the plan.

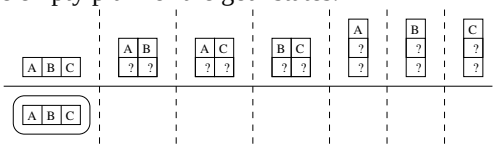
3 Problem Representation

Now we introduce the representation for sets of state sets for which a plan for reaching goal states exists.

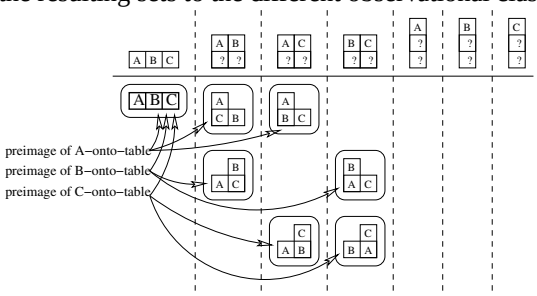
In the following example states are viewed as valuations of state variables, and the observational classes correspond to valuations of those state variables that are observable.

Example 3 Consider the blocks world with the state variables *clear(X)* observable, allowing to observe the topmost block of each stack. With three blocks there are 7 observational classes because there are 7 valuations of $\{\text{clear(A)}, \text{clear(B)}, \text{clear(C)}\}$ with at least one block clear.

Consider the problem of trying to reach the state in which all blocks are on the table. For each block there is an action for moving it onto the table from wherever it was before. If a block cannot be moved nothing happens. Initially we only have the empty plan for the goal states.

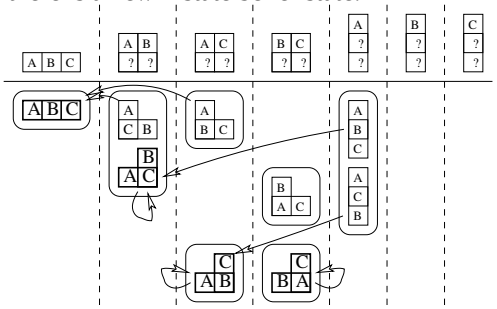


Then we compute the preimages of this set with actions that respectively put the blocks A, B and C onto the table, and split the resulting sets to the different observational classes.



Now for these 7 belief states we have a plan consisting of one or zero actions. But we also have plans for sets of states that are only represented implicitly. These involve branching. For example, we have a plan for the state set consisting of the four states in which respectively all blocks are on the table, A is on C, A is on B, and B is on A. This plan first makes observations and branches, and then executes the plan associated with the belief state obtained in each case. Because 3 observational classes each have 2 belief states, there are 2^3 maximal state sets with a branching plan. From each class only one belief state can be chosen because observations cannot distinguish between belief states in the same class.

We can find more belief states that have plans by computing preimages of existing belief states. Let us choose the belief states in which respectively all blocks are on the table, B is on C, C is on B, and C is on A, and compute their union's preimage with A-onto-table. The preimage intersected with the observational classes yields new belief states: for the class with A and B clear there is a new 2-state belief state covering both previous belief states in the class, and for the class with A clear there is a new 2-state belief state.



Computation of further preimages yields for each observational class a belief state covering all the states in that class, and hence a plan for every belief state. ■

The above example shows how the exponential number of state sets (corresponding to the Cartesian product of the observational classes) considered by Rintanen [2002] is represented only implicitly. The algorithm by Rintanen [2002] ex-

[1] Construction of cyclic plans requires looking at more global properties of transition graphs than what is needed for acyclic plans. The difficulties of cyclic plans in our framework are similar to those in the MDP/POMDP framework when using average rewards instead of finite horizons or discounted rewards [Puterman, 1994].

plicitly generates the state sets, the number of which in many cases is very high. With the new representation the computational complexity is shifted from the size of the representation to the time it takes to find a combination of belief states having a useful preimage. This shift is useful for two reasons. First, much of the space complexity (and the time complexity it implies) is traded to time complexity only: the state sets are not represented explicitly (except in the unobservable special case.) Second, the succinct representation allows much better control on which belief states to produce, and although finding one new belief state and plan still takes worst-case exponential time, this may be performed by clever algorithms and be further sped up by heuristics.

Next we formalize the framework in detail.

Definition 4 (Belief space) *Let $P = (C_1, \ldots, C_n)$ be a partition of the set of all states. Then* a belief space *is an n-tuple $\langle G_1, \ldots, G_n \rangle$ where $G_i \subseteq 2^{C_i}$ for all $i \in \{1, \ldots, n\}$ and $B \not\subseteq B'$ for all $i \in \{1, \ldots, n\}$ and $\{B, B'\} \subseteq G_i$.*

Notice that in each component of a belief space we only have set-inclusion maximal belief states. The simplest belief spaces are obtained from sets B of states as $\mathcal{B}(B) = \langle \{C_1 \cap B\}, \ldots, \{C_n \cap B\} \rangle$. A belief space is extended as follows.

Definition 5 (Extension) *Let $P = (C_1, \ldots, C_n)$ be the partition of all states, $G = \langle G_1, \ldots, G_n \rangle$ a belief space, and T a set of states. Define $G \oplus T$ as $\langle G_1 \Cup (T \cap C_1), \ldots, G_n \Cup (T \cap C_n) \rangle$ where the operation $\Cup$ adds the latter set of states to the former set of sets of states and eliminates sets that are not set-inclusion maximal, defined as $U \Cup V = \{R \in U \cup \{V\} | R \not\subseteq K$ for all $K \in U \cup \{V\}\}$.*

A belief space $G = \langle G_1, \ldots, G_n \rangle$ represents the set of sets of states $\text{flat}(G) = \{B_1 \cup \cdots \cup B_n | B_i \in G_i$ for all $i \in \{1, \ldots, n\}\}$ and its cardinality is $|G_1| \cdot |G_2| \cdot \ldots \cdot |G_n|$.

4 Complexity of Basic Operations

The basic operations on belief spaces needed in planning algorithms are testing the membership of a set of states in a belief space, and finding a set of states whose preimage with respect to an action is not contained in the belief space. Next we analyze the complexity of these operations.

Theorem 6 *For belief spaces G and state sets B, testing whether there is $B' \in$ flat(G) such that $B \subseteq B'$, and computing $G \oplus B$ takes polynomial time.*

Proof: Idea: A linear number of set-inclusion tests suffices. □

Our algorithm for extending belief spaces by computing the preimage of a set of states (Lemma 8) uses exhaustive search and runs in worst-case exponential time. This asymptotic worst-case complexity is very likely the best possible because the problem is NP-hard. Our proof for this fact is a reduction from SAT: represent each clause as the set of literals that are not in it, and then a satisfying assignment is a set of literals that is not included in any of the sets, corresponding to the same question about belief spaces.

Theorem 7 *Testing if for belief space G there is $R \in$ flat(G) such that preimg$_o(R) \not\subseteq R'$ for all $R' \in$ flat(G) is NP-complete. This holds even for deterministic actions o.*

Proof: Membership is easy: For $G = \langle G_1, \ldots, G_n \rangle$ choose nondeterministically $R_i \in G_i$ for every $i \in \{1, \ldots, n\}$, compute $R = \text{preimg}_o(R_1 \cup \cdots \cup R_n)$, and verify that $R \cap C_i \not\subseteq B$ for some $i \in \{1, \ldots, n\}$ and all $B \in G_i$. Each of these steps takes only polynomial time.

Let $T = \{c_1, \ldots, c_m\}$ be a set of clauses over propositions $A = \{a_1, \ldots, a_k\}$. We define a belief space based on states $\{a_1, \ldots, a_k, \hat{a}_1, \ldots, \hat{a}_k, z_1, \ldots, z_k, \hat{z}_1, \ldots, \hat{z}_k\}$. The states $\hat{a}$ represent negative literals. Define

$$\begin{array}{rcl} c'_i & = & (A \backslash c_i) \cup \{\hat{a} | a \in A, \neg a \notin c_i\} \text{ for } i \in \{1, \ldots, m\}, \\ G & = & \langle \{c'_1, \ldots, c'_m\}, \{\{z_1\}, \{\hat{z}_1\}\}, \ldots, \{\{z_k\}, \{\hat{z}_k\}\} \rangle, \\ o & = & \{\langle a_i, z_i \rangle | 1 \leq i \leq k\} \cup \{\langle \hat{a}_i, \hat{z}_i \rangle | 1 \leq i \leq k\}. \end{array}$$

We claim that T is satisfiable if and only if there is $B \in$ flat(G) such that preimg$_o(B) \not\subseteq B'$ for all $B' \in$ flat(G).

Assume T is satisfiable, that is, there is M such that $M \models T$. Define $M' = \{z_i | a_i \in A, M \models a_i\} \cup \{\hat{z}_i | a_i \in A, M \not\models a_i\}$. Now $M' \subseteq B$ for some $B \in$ flat(G) because from each class only one of $\{z_i\}$ or $\{\hat{z}_i\}$ is taken. Let $M'' = \text{preimg}_o(M') = \{a_i \in A | M \models a_i\} \cup \{\hat{a}_i | a_i \in A, M \not\models a_i\}$. We show that $M'' \not\subseteq B$ for all $B \in$ flat(G). Take any $i \in \{1, \ldots, m\}$. Because $M \models c_i$, there is $a_j \in c_i \cap A$ such that $M \models a_j$ (or $\neg a_j \in c_i$, for which the proof goes similarly.) Now $z_j \in M'$, and therefore $a_j \in M''$. Also, $a_j \notin c'_j$. As there is such an a_j (or $\neg a_j$) for every $i \in \{1, \ldots, m\}$, M'' is not a subset of any c'_i, and hence $M'' \not\subseteq B$ for all $B \in$ flat(G).

Assume there is $B \in$ flat(G) such that $D = \text{preimg}_o(B) \not\subseteq B'$ for all $B' \in$ flat(G). Now D is a subset of $A \cup \{\hat{a} | a \in A\}$ with at most one of a_i and $\hat{a}_i$ for any $i \in \{1, \ldots, k\}$. Define a model M such that for all $a \in A$, $M \models a$ if and only if $a \in D$. We show that $M \models T$. Take any $i \in \{1, \ldots, m\}$ (corresponding to a clause.) As $D \not\subseteq B$ for all $B \in$ flat(G), $D \not\subseteq c'_i$. Hence there is a_j or $\hat{a}_j$ in $D \backslash c'_i$. Consider the case with a_j ($\hat{a}_j$ goes similarly.) As $a_j \notin c'_i$, $a_j \in c_i$. By definition of M, $M \models a_j$ and hence $M \models c_i$. As this holds for all $i \in \{1, \ldots, m\}$, $M \models T$. □

5 Planning Algorithms

Based on the problem representation in the preceding section, we devise a planning algorithm that repeatedly identifies new belief states (and associated plans) until a plan covering the initial states is found. The algorithm in Figure 2 tests for plan existence; further book-keeping is needed for outputting a plan. The size of the plan is proportional to the number of iterations the algorithm performs, and outputting the plan takes polynomial time in the size of the plan. The algorithm uses the subprocedure *findnew* (Figure 1) for extending the belief space (this is the NP-hard subproblem from Theorem 7). Our implementation of the subprocedure orders sets $f_1, \ldots, f_m$ by cardinality in a decreasing order: bigger belief states are tried first. We also use a simple pruning technique for deterministic actions o: If preimg$_o(f_i) \subseteq$ preimg$_o(f_j)$ for some i and j such that $i > j$, then we may ignore f_i.

PROCEDURE findnew(o,A,F,H);
IF $F = \langle\rangle$ *AND* preimg$_o(A) \not\subseteq B$ for all $B \in$ flat(H)
THEN RETURN A;
IF $F = \langle\rangle$ *THEN RETURN* $\emptyset$;
F is $\langle\{f_1, \ldots, f_m\}, F_2, \ldots, F_k\rangle$ for some $k \geq 1$;
FOR i := 1 *TO* m *DO*
  B := findnew(o,$A \cup f_i$,$\langle F_2, \ldots, F_k\rangle$,$H$);
  IF $B \neq \emptyset$ *THEN RETURN* B;
END;
RETURN $\emptyset$

Figure 1: Algorithm for finding new belief states

PROCEDURE plan(I,O,G);
$H := \mathcal{B}(G)$;
progress := true;
WHILE progress and $I \not\subseteq I'$ for all $I' \in$ flat(H) *DO*
  progress := false;
  FOR EACH $o \in O$ *DO*
    B := findnew(o,$\emptyset$,H,H);
    IF $B \neq \emptyset$ *THEN*
      BEGIN
        $H := H \oplus$ preimg$_o(B)$;
        progress := true;
      END;
  END;
END;
IF $I \subseteq I'$ for some $I' \in$ flat(H) *THEN RETURN* true
ELSE RETURN false;

Figure 2: Algorithm for planning with partial observability

Lemma 8 *Let H be a belief space and o an action. The procedure call findnew(o,$\emptyset$,F, H) returns a set B' of states such that $B' =$ preimg$_o(B)$ for some $B \in$ flat(F) and $B' \not\subseteq B''$ for all $B'' \in$ flat(H), and if no such belief state exists it returns $\emptyset$.*

Proof: Sketch: The procedure goes through the elements $\langle B_1, \ldots, B_n\rangle$ of $F_1 \times \cdots \times F_n$ and tests whether preimg$_o(B_1 \cup \cdots \cup B_n)$ is in H. The sets $B_1 \cup \cdots \cup B_n$ are the elements of flat(F). The traversal through $F_1 \times \cdots \times F_n$ is by generating a search tree with elements of F_1 as children of the root node, elements of F_2 as children of every child of the root node, and so on, and testing whether the preimage is in H. The second parameter of the procedure represents the state set constructed so far from the belief space, the third parameter is the remaining belief space, and the last parameter is the belief space that is to be extended, that is, the new belief state may not belong to it. □

The correctness proof of the procedure *plan* consists of the following lemma and theorems. The first lemma simply says that extending a belief space H is monotonic in the sense that the members of flat(H) can only become bigger.

Lemma 9 *Assume T is any set of states and $B \in$ flat(H). Then there is $B' \in$ flat$(H \oplus T)$ so that $B \subseteq B'$.*

The second lemma says that if we have belief states in different observational classes such that each is included in a belief state of a belief space H, then there is a set in flat(H) that includes all these belief states.

Lemma 10 *Let $B_1, \ldots, B_n$ be sets of states so that for every $i \in \{1, \ldots, n\}$ there is $B_i' \in$ flat(H) such that $B_i \subseteq B_i'$, and there is no observational class C such that for some $\{i, j\} \subseteq \{1, \ldots, n\}$ both $i \neq j$ and $B_i \cap C \neq \emptyset$ and $B_j \cap C \neq \emptyset$. Then there is $B' \in$ flat(H) such that $B_1 \cup \cdots \cup B_n \subseteq B'$.*

The proof of the next theorem shows how the algorithm is capable of finding any plan by constructing it bottom up starting from the leaf nodes. The construction is based on first assigning a belief state to each node in the plan, and then showing that the algorithm reaches that belief state from the goal states by repeated computation of preimages.

Theorem 11 *Whenever there exists a finite acyclic plan for a problem instance, the algorithm in Figure 2 returns* true.

Proof: Assume there is a plan $\langle N, b, l\rangle$ for a problem instance $\langle S, I, O, G, P\rangle$. Label all nodes of the plan as follows. The root node b is labeled with I, that is, $Z(b) = I$. When all parent nodes of a node n have a label, we assign a label to n. Let $l(n_1) = \langle o_1, n\rangle, \ldots, l(n_m) = \langle o_m, n\rangle$ for action nodes $n_1, \ldots, n_m$ that have n as the child node, and let $l(n_1') = \{\langle C_1, n\rangle, \ldots\}, \ldots, l(n_k') = \{\langle C_k, n\rangle, \ldots\}$ for all observation nodes $n_1', \ldots, n_k'$ with n as one of the children. Then $Z(n) =$ img$_{o_1}(Z(n_1)) \cup \cdots \cup$ img$_{o_m}(Z(n_m)) \cup (Z(n_1') \cap C_1) \cup \cdots \cup (Z(n_k') \cap C_k)$. If the above labeling does not assign anything to a node n, then assign $Z(n) = \emptyset$. Each node is labeled with those states that are possible in that node on some execution.

We show that if plans for $Z(n_1), \ldots, Z(n_k)$ exist, where $n_1, \ldots, n_k$ are children of a node n in a possible plan, then the algorithm determines that a plan for $Z(n)$ exists as well.

Induction hypothesis: For each plan node n such that all paths to a terminal node have length i or less, its label $B = Z(n)$ is a subset of some $B' \in$ flat(H) where H is the value of the program variable H after the *while* loop exits and H could not be extended further.

Base case $i = 0$: Terminal nodes of the plan are labeled with subsets of G. By Lemma 9, there is G' such that $G \subseteq G'$ and $G' \in$ flat(H) because initially $H = \mathcal{B}(G)$ and thereafter it was repeatedly extended.

Inductive case $i \geq 1$: Let n be a plan node. By the induction hypothesis for all children n' of n, $Z(n') \subseteq B$ for some $B \in$ flat(H).

If n is an observation node with children $n_1, \ldots, n_k$ and respective observations $C_1, \ldots, C_k$, then $Z(n) \cap C_1$, …, $Z(n) \cap C_k$ all occupy disjoint observational classes and superset of $Z(n) \cap C_i$ for every $i \in \{1, \ldots, k\}$ is in flat(H). Hence by Lemma 10 $Z(n) \subseteq B$ for some $B \in$ flat(H).

If n is an action node with action o and child node n', then img$_o(Z(n)) \subseteq Z(n')$, and by the induction hypothesis $Z(n') \subseteq B'$ for some $B' \in$ flat(H). We have to show that $Z(n) \subseteq B''$ for some $B'' \in$ flat(H). Assume that there is no such B''. But now by Lemma 8 findnew(o,$\emptyset$,H,H) would return B''' such that preimg$_o(B''') \not\subseteq B$ for all $B \in$ flat(H),

and the *while* loop could not have exited with H, contrary to our assumption about H. □

Theorem 12 *Let $\Pi = \langle S, I, O, G, P\rangle$ be a problem instance. If plan(I,O,G) returns* true*, then Π has a solution plan.*

Proof: Let $H_0, H_1, \ldots$ be the sequence of belief spaces H produced by the algorithm. We show that for all $i \geq 0$, for every $B \in \text{flat}(H_i)$ there is a plan that reaches G.

Induction hypothesis: H_i contains only such state sets $B \in \text{flat}(H_i)$ for which a plan reaching G exists.

Base case $i = 0$: $H_0 = \mathcal{B}(G)$, and the only state set in H_0 is G. The empty plan reaches G from G.

Inductive case $i \geq 1$: H_{i+1} is obtained as $H_i \oplus \text{preimg}_o(B)$ where $B = \text{findnew}(o, \emptyset, H_i, H_i)$.

Because by Lemma 8 $B \in \text{flat}(H_i)$, by induction hypothesis there is a plan π for B. The plan that executes o followed by π reaches G from $\text{preimg}_o(B)$.

Let B be any member of $\text{flat}(H_{i+1})$. We show that for B there is a plan for reaching G. The plan for B starts by a branch[2]. We show that for every possible observation, corresponding to one observational class, there is a plan that reaches G. Let C_j be the jth observational class. When observing C_j, the current state is in $B_j = B \cap C_j$. Now for B_j there is $B'_j \in H_{i+1,j}$ with $B_j \subseteq B'_j$ where $H_{i+1,j}$ is the jth component of H_{i+1}. Now by induction hypothesis there is a plan for B'_j if $B'_j \in H_{i,j}$, and if $B'_j \in H_{i+1,j} \backslash H_{i,j}$, then for the branch corresponding to C_j we use the plan for $\text{preimg}_o(B)$, as B'_j must be $\text{preimg}_o(B) \cap C_j$. □

Until now we have used only one partition of the state space to observational classes. However, it is relatively straightforward to generalize the above definitions and algorithms to the case in which several partitions are used, each for a different set of actions. This means that the possible observations depend on the action that has last been taken.

6 Experimentation with an Implementation

We have implemented the algorithm from the previous section and call the resulting planning system BBSP. The only heuristic is the one described in the preceding sections: *findnew* chooses bigger belief states first. The implementation is based on representing sets of states and actions as BDDs [Burch *et al.*, 1994]. There is a small improvement in the belief space representation with BDDs: all components of a belief space consisting of one belief state only are represented by one BDD.

We carried out a comparison to the MBP planner [Bertoli *et al.*, 2001] which uses forward-search together with some heuristics for restricting branching. MBP starts search from the initial states, and proceeds forward by taking actions, leading to another set of states, or by using observations to split the current state set to several smaller ones. Different choices of actions and observations induce a search tree.

[2]Some branches might not be needed, and if the intersection of B with only one observational class is non-empty the plan could start with an action node instead of a degenerate observation node.

		runtime in seconds		iterations
problem	$\|S\|$	MBP	BBSP	BBSP
BTCS1601	98	**0.60**	1.21	32
BTCS1701	104	**0.79**	1.38	34
BTCS1801	110	**1.01**	2.08	36
BTCS1901	116	**1.22**	2.28	38
BTCS2001	122	**1.44**	2.52	40
medical18	148	7.34	**2.49**	21
medical20	164	24.13	**2.91**	23
medical22	180	60.53	**3.48**	25
medical24	196	$>$ 20 m	**4.11**	27
medical26	212	$>$ 20 m	**6.61**	29
emptyroom07	49	**0.09**	0.38	41
emptyroom08	64	**0.12**	0.63	53
emptyroom10	100	**0.16**	1.56	81
emptyroom15	225	**0.37**	9.60	198
emptyroom20	400	**0.62**	25.58	243
ring03	162	**0.09**	0.16	11
ring04	648	**0.38**	**0.44**	19
ring05	2430	1.99	**1.03**	23
ring06	8748	13.12	**1.63**	27
ring07	30618	94.73	**2.41**	31
ring08	104976	744.90	**3.34**	35
ring09	354294	$>$ 20 m	**5.55**	39
ring10	1180980	$>$ 20 m	**7.06**	43
BW03fo	13	32.64	**0.14**	9
BW04fo	73	$>$ 20 m	**0.67**	22
BW05fo	501	$>$ 20 m	**6.43**	46
BW06fo	4051	$>$ 20 m	**133.14**	64
BW03pfo	13	**0.13**	**0.12**	9
BW04pfo	73	90.15	**0.69**	22
BW05pfo	501	$>$ 20 m	**6.29**	46
BW06pfo	4051	$>$ 20 m	**133.88**	64
BW03po	13	**0.08**	**0.12**	9
BW04po	73	**0.71**	**0.64**	22
BW05po	501	13.49	**6.41**	46
BW06po	4051	394.21	**198.85**	64

Table 1: Runtime comparison BBSP vs. MBP

Some of the MBP runtimes given by Bertoli et al. [2001] are much better than given by us in this paper (specifically on the medical and ring problems) because the branching heuristic used by Bertoli et al. works well on their formulations of the benchmarks: branch only on one observable state variable if possible. We used a slightly different formulation where one many-valued observation is replaced by a small number of Boolean observations.

Runtimes of the planners are given in Table 1. The runs were on a 360 MHz Sun Sparcstation under Solaris. The problem instances are the same as in [Rintanen, 2002] where MBP was shown in almost all cases to be faster than GPT [Bonet and Geffner, 2000] and much faster than the YKÄ planner [Rintanen, 2002] on most of the problems except the blocks worlds problems with full or almost full observability. BTCS is the bomb-in-the-toilet problem with sensing. In the medical problems patients are cured by performing tests and medicating. The emptyroom problems are about navigating from an unknown location to the center of the room. The ring problems are about closing and locking all the windows of a building consisting of a cycle of rooms. BW is the blocks

world with increasing number of blocks with the goal to arrange them into one stack from any initial configuration under different degrees of observability: *fo* is full observability, *pfo* is with the *on* relation observable, and *po* is partial observability (*clear* and *ontable* are observable). The problems are solvable under partial observability because moving a block only requires that it is clear and a move action is applicable no matter where the block is.

The rightmost column gives the number of iterations BBSP needs for finding a plan. MBP runtimes in some cases grow faster because it performs more search. This is most obvious in some of the problems with more observations as the number of possible ways of branching becomes astronomical. In our algorithm, the dynamic programming character of plan generation better avoids this explosion in the number of belief states to be considered.

In forward search there is an inherent conflict between A) keeping the size of plans and search trees down by *not* branching on all possible observations and B) branching enough to be able to find a plan. In MBP a number of heuristics is used for controlling branching, and for these benchmarks the heuristics in most cases work very well. For backward plan construction no similar conflict exists, and plan construction by a form of dynamic programming leads to effective reuse of already constructed belief states and plans, and there is no separate problem of deciding how to branch.

7 Conclusions and Future Work

We have presented a novel framework for non-probabilistic conditional planning with partial observability, proposed a backward search algorithm for finding plans, and shown that the basic backup step of the algorithm is NP-hard. We have also demonstrated how an efficient implementation of the backup step leads to competitive planning.

For future work we propose considering the problem of finding plans with quality guarantees, for example, plans with optimal execution length, as well as planning under more general infinite-horizon executions. Also, the use of more sophisticated heuristics for driving the planning algorithm should be considered, for example the ones proposed by Rintanen [2004b] generalized to partially observable problems.

References

[Bertoli *et al.*, 2001] Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In Bernhard Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 473–478. Morgan Kaufmann Publishers, 2001.

[Bonet and Geffner, 2000] Blai Bonet and Héctor Geffner. Planning with incomplete information as heuristic search in belief space. In Steve Chien, Subbarao Kambhampati, and Craig A. Knoblock, editors, *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, pages 52–61. AAAI Press, 2000.

[Burch *et al.*, 1994] J. R. Burch, E. M. Clarke, D. E. Long, K. L. MacMillan, and D. L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, 1994.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[Madani *et al.*, 1999] Omid Madani, Steve Hanks, and Anne Condon. On the decidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99) and the 11th Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, pages 541–548. AAAI Press, 1999.

[Mundhenk *et al.*, 2000] Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM*, 47(4):681–720, 2000.

[Puterman, 1994] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

[Rintanen, 2002] Jussi Rintanen. Backward plan construction under partial observability. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems*, pages 173–182. AAAI Press, 2002.

[Rintanen, 2004a] Jussi Rintanen. Complexity of planning with partial observability. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, pages 345–354. AAAI Press, 2004.

[Rintanen, 2004b] Jussi Rintanen. Distance estimates for planning in the discrete belief space. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-2004) and the 13th Conference on Innovative Applications of Artificial Intelligence (IAAI-2004)*, pages 525–530. AAAI Press, 2004.

[Smallwood and Sondik, 1973] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[Weld *et al.*, 1998] Daniel S. Weld, Corin R. Anderson, and David E. Smith. Extending Graphplan to handle uncertainty and sensing actions. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 897–904. AAAI Press, 1998.

Probabilistic Reasoning for Plan Robustness *

Steve R. Schaffer, Bradley J. Clement, Steve A. Chien
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, M/S 126-347, Pasadena, CA 91109
{firstname.lastname}@jpl.nasa.gov

Abstract

A planning system must reason about the uncertainty of continuous variables in order to accurately project the possible system state over time. A method is devised for directly reasoning about the uncertainty in continuous activity duration and resource usage for planning problems. By representing random variables as parametric distributions, computing projected system state can be simplified. Common approximations and novel methods are compared for over-constrained and lightly constrained domains within an iterative repair planner. Results show improvements in robustness over the conventional non-probabilistic representation by reducing the number of constraint violations during execution. The improvement is more significant for larger problems and those with higher resource subscription levels but diminishes as the system is allowed to accept higher risk levels.

1 Introduction

Planning systems that reason about real world events must eventually deal with the inherent uncertainty of any real world mechanism. For example, actions may take longer or consume more resources than predicted. Even if it were possible to model every variable that affected a planned set of actions, doing so is impractical for realistically sized domains. Further, practical modeling abstractions themselves also introduce uncertainty into reasoning about a system.

The way a planning system deals with uncertainty in its actions and observations is critical to how well the system is able to perform in the real world. Clearly, systems that effectively reason about uncertainty can better avoid generating plans that are likely to violate execution constraints. But effective use of uncertainty can also improve the long-term efficiency of a plan by balancing acceptable risk levels against the inefficiencies incurred to avoid those risks. Finally, knowledge of uncertainty allows the system to better assess and report on the most risky plan segments.

One historical approach to dealing with uncertainty is to assume no uncertainty at the level of planning abstraction. To be used in a real world system, such systems are often augmented with some replanning mechanism for when predictions do not match results [Chien *et al.*, 2000; Koenig, 2001]. One step further is to depend on an execution system to handle any variations in plan execution. Effectively, the planner itself is abstracted from any knowledge that the real world does not behave as predicted.

There are many planning systems that reason more directly about uncertainty. Classifications and surveys of this work are given by Bresina *et al.* (2002), Blythe (1999), and Boutilier *et al.* (1999). Some techniques handle some level of temporal uncertainty [Puterman, 1994; Boyan & Littman, 2000] or continuous resources, e.g. [Bertsekas & Tsitsiklis, 1996; Smart & Kaelbling, 2000], and one can represent both complex temporal constraints and continuous states/resources [Dearden *et al.*, 2003]. This is important for domains where concurrent tasks interact in their effect on continuous resources. For example, a spacecraft can be slewing, operating instruments, and communicating at the same time. Metric resources such as power, energy, memory, and temperature are continuously affected and often require complex temporal constraints to balance safe operation with efficiency.

This paper outlines one possible approach for directly reasoning about the uncertainty in action timing and resource consumption. While Dearden *et al.* use a Monte Carlo approach to estimating the value of a plan (2003), we directly compute parametric probability distributions for time and resource variables based on a user-supplied model of activities and resources. The distributions are then combined during planning to determine the net probability distribution of a resource at any time point, which in turn may be integrated to yield the probability of violating any execution constraints on the resource. The key idea is to use this "probability of conflict" to score potential plans and to drive the planner's search toward low-risk actions. An output plan provides a balance between the user's risk aversion and other measures of plan optimality. This is a simple conformant planning approach—the planner does no contingency planning but also assumes no future state observability.

The present work deals only with durations and resource usages that can be modeled as normally distributed random

*This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

variables, though the techniques are more widely applicable. To gauge the effectiveness of our probabilistic system, batch-generated plans are executed in a stochastic simulator. A comparative evaluation of our technique versus some common probabilistic approximations is provided along with an analysis of its applicability to different kinds of planning problems.

2 Approach

Planning effort is directed to repairing areas of a plan that have unacceptable levels of risk, as determined by a user-specified risk tolerance on each resource as a function of time. Risk for any one timeline segment is assessed by computing the probability that the sum of all activity reservations that potentially overlap the segment would exceed one of the modeled system resource limits. This probability of resource constraint conflict is readily derived if the resources' net probability density functions are available. Our approach for maintaining each net resource distribution is to combine individual activity resource reservations parametrically.

Each activity in the plan is considered to make uncertain resource reservations that follow a known distribution. Further, each activity can also have a duration that is similarly uncertain. (For simplicity, all activities are considered to have certain start times - an assumption that holds for directly commanded actions, but may not apply for exogenous events.) In this paper, we only consider reservations and durations that are normally distributed random variables, though in practice other parametric distributions can also be used. The parametric representation for a normal distribution is very compact, requiring only the distribution mean (μ) and standard deviation (σ). In comparison, a particle filter [Gordon, Salmond, & Smith, 1993] requires a value and weight for each sample taken from a distribution. Conveniently, specified values can be also represented as normal random variables with a given μ but $\sigma = 0$.

In the case of activities that make persistent reservations on a resource (that is, they consume or produce a resource), the net resource distribution for a timeline segment is the sum of all current and preceding normally distributed reservations. Fortunately, the sum Σ of i independent normal reservations n is itself a normal, with parameters $\mu_\Sigma = \sum_i \mu_{n_i}$ and $\sigma_\Sigma = \sqrt{\sum_i {\sigma_{n_i}}^2}$. Notice that the uncertainty of the sum is greater than any single component, indicating that resultant uncertainty grows with the number of interacting reservations.

For actions that only have a transient reservation (lasting for their duration only), the same method can be applied to those reservations that are concurrent. In the simple case that each concurrent activity has a certain duration, the net resource distribution is computed by adding each local reservation. In the more complex case of concurrent activities with uncertain duration, the net resource distribution itself becomes a function of time.

For an activity A with start time t_s and duration $d = d_\mu \pm d_\sigma$, consider $\mathrm{P}[A](t)$ to be the probability that action A is executing at time t (see figure 1). As d is normally distributed, the end time t_e is also a normal, and we can express $\mathrm{P}[A](t)$

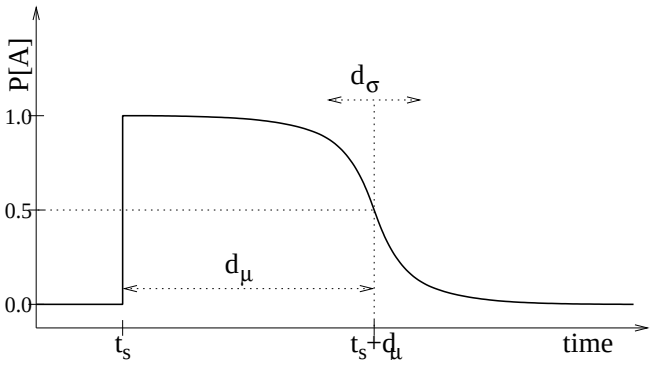


Figure 1: Probability of activity A with normally distributed duration d continuing after its start time t_s.

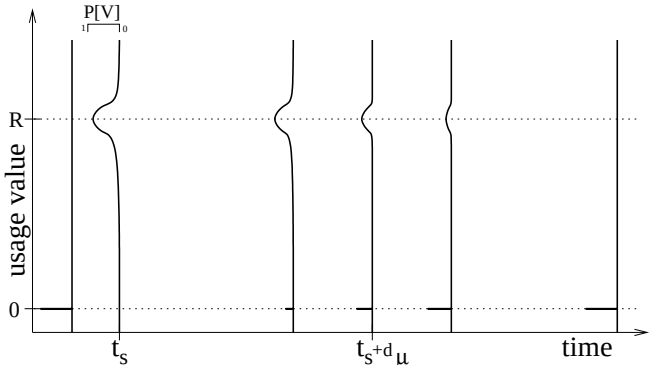


Figure 2: Transient resource usage distribution for activity A of uncertain duration, showing peaks at R when the activity is likely and 0 when the activity is unlikely.

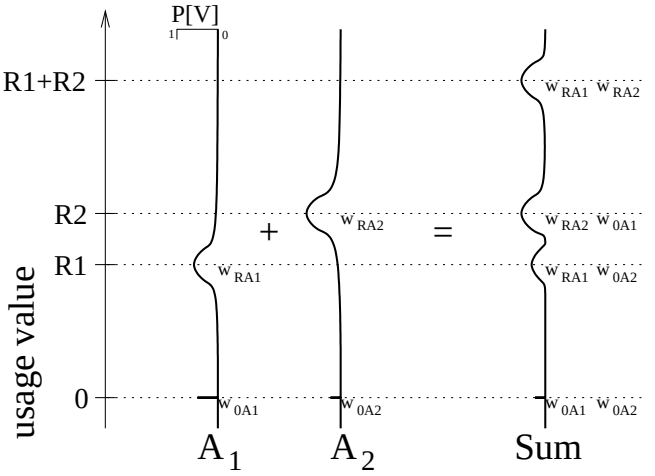


Figure 3: Computing the sum of two bimodal resource usage distributions results in a multi-modal distribution. Each resultant peak weight is the product of the component weights.

as:

$$\mathrm{P}[A](t) = \begin{cases} 0 & t < t_{sA} \\ 1 - \Phi_{\mu_{t_e},\sigma_{t_e}}(t) & t \geq t_{sA} \end{cases}$$

where $\Phi_{\mu,\sigma}(x)$ is the cumulative distribution function for a normal with mean μ and standard deviation σ. Strictly, normal distributions may yield negative samples, so we must truncate only the duration distributions to $[0, \infty)$, or in practice $[0, \mu + 3\sigma]$.

Each of A's resource reservations must reflect the gradual diminishment of the activity's probability. If A makes a reservation R when active, its *effective* reservation becomes a function of time, $R(t)$, as in figure 2). This distribution is bimodal: one peak at zero resource usage represents that the activity is not in effect (weighted $w_0 = 1 - \mathrm{P}[A](t)$), and the R peak represents A's transient reservation (weighted $w_{R_A} = \mathrm{P}[A](t)$). The peak at zero is a scaled Dirac delta function: it integrates to w_0, but has infinitesimal width.

The time-sensitive reservations seen in figure 2 are no longer simple normals, so the net resource distribution must also be more complex. In fact, the sum of $|A_i|$ different bimodal reservations results in a multi-modal distribution with $\mathrm{O}(2^{|A_i|})$ distinct peaks: one for each combination of activities that could be in effect (see figure 3).

With the net resource distribution $\mathrm{PDF}_R(x,t)$ in hand, Computing the probability of violating a system resource constraint during a timeline unit becomes a simple integral. For a timeline unit T with a random variable resource level R and constraints that $R \in [l_{min}, l_{max}]$, then the probability of violation is given by:

$$\begin{aligned}
\mathrm{P}[V_T](t) &= \mathrm{P}[R(t) < l_{min}] + \mathrm{P}[R(t) > l_{max}] \\
&= 1 - \mathrm{P}[l_{min} \leq R(t) \leq l_{max}] \\
&= 1 - (\mathrm{CDF}_R(l_{max}, t) - \mathrm{CDF}_R(l_{min}, t)) \\
&= 1 - \int_{l_{min}}^{l_{max}} \mathrm{PDF}_R(x,t)dx
\end{aligned}$$

Fortunately, this integral for normal distributions is fast to compute and multi-normal distributions require simple linear combinations of this integral.

In the end, $\mathrm{P}[V_T](t)$ may still be a function of time. In this event, we report $\mathrm{P}[V_T]$ as the maximum instantaneous probability of violation during the timeline unit. (Such an assumption works for systems where each random value is chosen once and not resampled.) To avoid checking all $t \in T$, we currently only check a constant number of critical times from T, including the endpoints.

The probability of constraint violation for each timeline unit is compared to the user-specified acceptable risk level, and any violations that are more likely than the risk tolerance are flagged as plan conflicts. A planning algorithm can use the tolerance to help decide whether and where to add, order, move, or remove an activity.

In our application, we use an iterative repair planner that chooses one over-risk-tolerance timeline unit at a time and attempts to reconcile the risk by moving, adding, or deleting resource contributors. For our purposes, the plan is scored according to how many remaining "too risky" timeline units remain, and the planner gradually hill-climbs toward plans with only tolerable risk levels.

2.1 Comparison Approximations

Approximation methods were implemented for comparison against the fully probabilistic system described above. Each fits within the same planning and heuristic framework, but maintains the net resource distributions differently.

Means Only: One very natural approximation method is to disregard all uncertainty and consider only the one value of maximum likelihood as representative of a distribution. For normal distributions, this is the mean. Because durations are also estimated by the mean, there are no multi-modal distributions, and resource values are tracked as a single value on each unit. The Means Only approximation is equivalent to an assumption that everything behaves as expected.

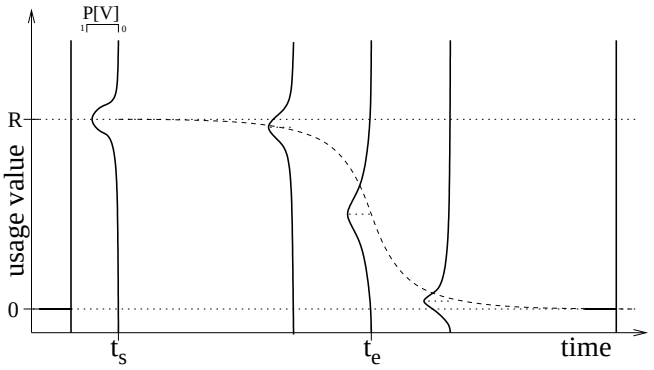


Figure 4: Single peak approximation for resource usage distribution in place of multi-modal distribution (figure 2).

Pessimistic: Similar to the Means Only approximation, the Pessimistic approximation only tracks one value from each distribution. Instead of choosing the value of maximum likelihood, however, it chooses the "worst case" value. For a normal distribution, our pessimistic system tracks only the value $\mu + 2\sigma$ (or $\mu - 2\sigma$), and considers that to be the actual resource reservation. The choice of which direction constitutes the worst case is inherently domain dependent and must be specified.

Single Peak: A possible limiting factor of the Fully Probabilistic system are the $\mathrm{O}(2^{|A|})$ peaks required when combining reservations of uncertain-duration activities A. The Single Peak approximation uses a single normal distribution in place of this set, as in figure 4 (compare with figure 2). This forfeits accurate representation in favor of much improved time complexity. The Single Peak approximation is optimistic in that it underestimates reservations.

Chebyshev Bound: The Chebyshev Bound approximation is similar to the Single Peak approximation in that both eliminate the multi-modal distributions that arise from uncertain duration. However, the Chebyshev Bound uses a more rigorous mathematical foundation for its approximation: for any random variable R, no matter the distribution, the probability of receiving a sample further than l from the distribution mean μ is given by the single-tailed version of Chebyshev's inequality:

$$\mathrm{P}[R - \mu \geq l] \leq \frac{\sigma^2}{\sigma^2 + l^2}$$

Because the Chebyshev Bound assumes so little about a distribution, it is necessarily pessimistic. Like the Single Peak, Chebyshev tracks only a single mean and standard deviation, and the sum of two approximated values is taken to have the worst case standard deviation of $\sigma_\Sigma = \sum_i {\sigma_{n_i}}^2$. We apply the one-sided Chebyshev inequality to the net mean and standard deviation, and report the resulting upper bound on violation probability as *the* violation probability.

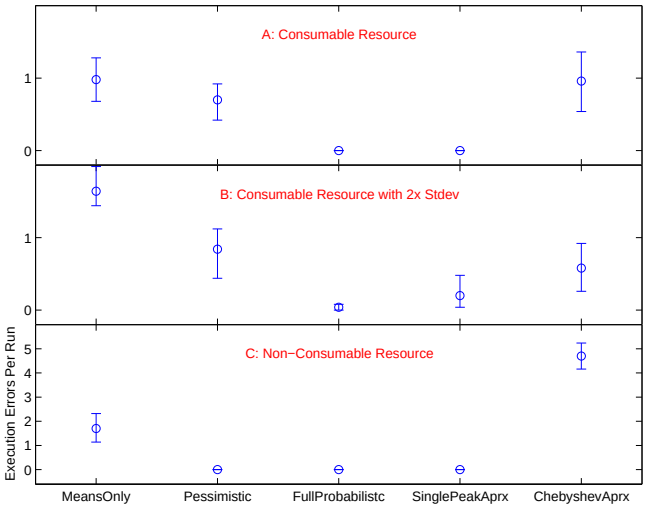


Figure 5: Execution error means for the three abstract domain variations. The 99% confidence of the mean is shown as an error bar.

3 Results

The Full Probabilistic system was evaluated against each of the comparison algorithms in two disparate planning domains. The first domain is an abstract testbed, and the second is a much more complex orbiting spacecraft domain.

For each domain, a random problem generator provided the initial schedule for the planner to repair. An iterative optimization planner was then run for a fixed number of iterations on the seed plan. The planner was augmented to use each of the full probabilistic and approximation algorithms, and an output plan was saved for each. The saved plans were then executed on a stochastic simulator that reported the number of resource constraint violations that occurred. Notably, no replanning was allowed as information became available during simulation. It would be possible to augment our experiments with more elaborate execution models (flexible time points, replanning, etc), but such was not investigated in the present work.

3.1 Abstract Domain

The abstract testbed domain has a single resource and a series of activities that may consume or replenish that resource. The model was run with both permanent and transient resource reservations, and with different levels of reservation uncertainty. A valid solution existed for every generated problem.

A comparison of the simulation error means for each approximation method is show in figure 5. As expected, the Means Only approximation stacked activities until the resource value was very close to its limit. This resulted in simulation errors when the simulated values exceeded the mean. The Pessimistic approximation only fared slightly better, likely due to its representation deficiency: a simulation error occurs when a resource exceeds its limit or falls below zero. After a sequence of several overestimated consumers, the Pessimistic approximation replenished those reservations with twice as many underestimated replenishers. This causes the resource to fall well below zero, and an error is reported. In real systems, resources may have one-sided constraints.

The Full Probabilistic system fared the best, consistently achieving nearly zero errors in each domain. It added an appropriate amount of both resource and schedule slack to accommodate the specified risk tolerance of 5%. The Single Bump approximation also performed well, only having difficulty when the resource uncertainty was doubled in (B). Notably, the Chebyshev approximation did not meet expectations: it turned out to be so very pessimistic in its distribution estimation that it failed to find good solutions, floundering with imagined conflicts.

The price of using the Fully Probabilistic system is of course computation time. For problems in which duration was not uncertain, the Fully Probabilistic system was about 10 times slower (unoptimized) than non probabilistic approaches. When duration was made uncertain, however, a vast difference appeared. Notably, the Single Peak approximation was almost 100 times faster than Full Probabilistic, on par with the non-probabilistic approaches. Systems where computational time is at a premium would likely fare well to adopt a simple Single Peak approximation and instead leave the Full Probabilistic approach for systems where execution errors are extremely high cost.

3.2 Orbiter Domain

The second domain is a more realistic mock up of an orbiting spacecraft model. The model is based on a synthesis of ideas from actual models for the EO-1 spacecraft [Chien *et al.*, 2003] and the proposed ASE spacecraft [Chien *et al.*, 2002]. In addition, we strove to model many of the issues presented in similar planning competition models [Long & Fox, 2002]. There are other planning systems that have treated similar spacecraft domains, e.g. [Globus *et al.*, 2002; Frank *et al.*, 2001].

The modeled hypothetical spacecraft is an earth-orbiting satellite equipped with a camera for imaging the planet. The craft must take actions only when sufficient power is available to its solar panels or by drawing on its battery. The craft must avoid overrunning its battery, memory, and disk space capacities. In addition, the processing power and antenna bandwidth are modeled as system resources. Finally, the external environment is modeled as providing limited availability windows for downlinks, imaging, and solar power.

The probe is tasked with acquiring images during target visibility windows, processing those images in RAM, recording them to disk, and later downlinking them to a ground station. The probe has to reason about 10 resources and has 10 different activities to complete its goals. Each activity makes reservations on multiple resource timelines. In this domain, the random problem generator does not guarantee that its problems will always have a completely valid solution (that is, the problems could be over-constrained since the planner is forbidden to shed goals).

As before, the Fully Probabilistic system achieves statistically significantly fewer simulation errors than either of the non-probabilistic systems, and generates plans on par with the Single Peak approximation's. The box plot in figure 6 conveniently shows a comparison of the error counts for each system. On a per-problem basis, the Full Probabilistic system had a mean 3.05 fewer simulation errors, with a

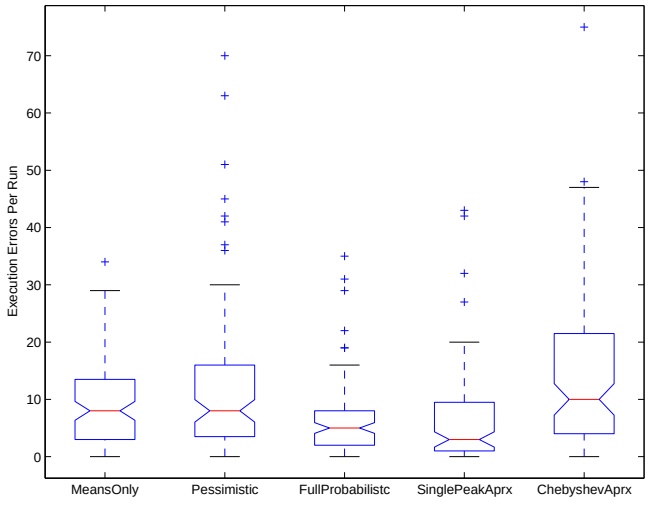


Figure 6: Execution error distribution for each reasoning system. The box plot shows the median as a horizontal line, a 95% confidence of the median as a notch, and the interquartile range as a box. The whiskers extend to encompass 1.5 more interquartile ranges, and outliers are plotted beyond that.

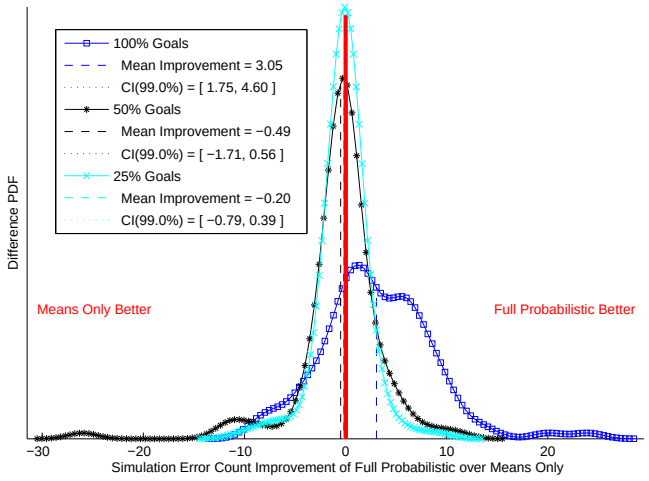


Figure 7: Execution error improvement distribution for problems of different goal densities. (The improvement is measured as the per-problem difference in errors.)

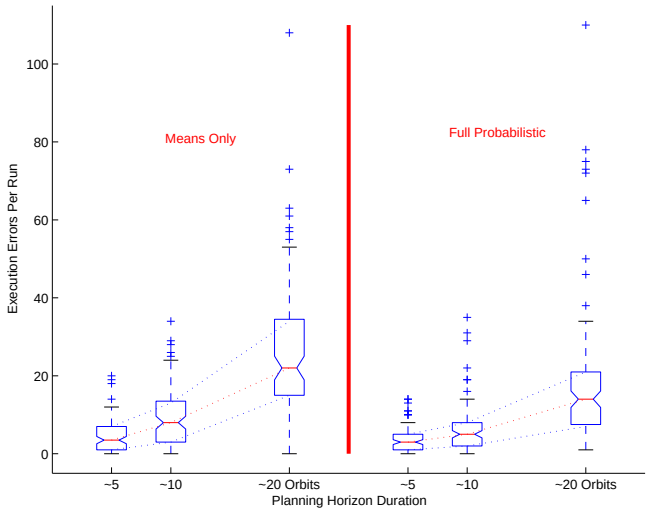


Figure 8: Execution Error Distribution for Problems of Different Sizes

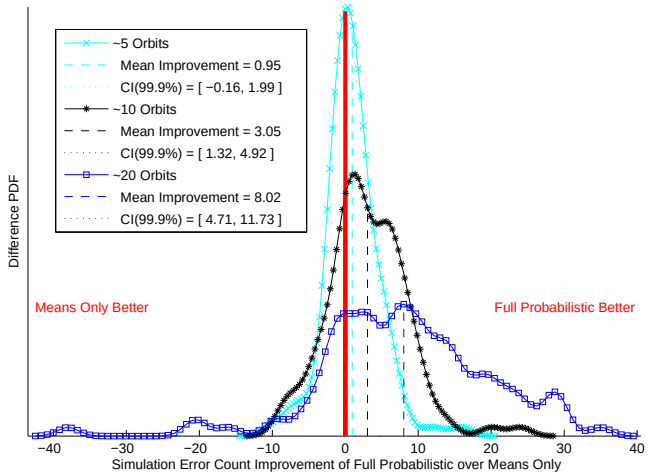


Figure 9: Execution Error Improvement Distribution for Problems of Different Sizes

99.9% confidence interval of [1.35, 4.90]. The Pessimistic approximation still suffers from the double resource bound problem noted for the abstract model, but still achieves performance comparable to the Means Only approach. The overly pessimistic Chebyshev system still fares worse than the Fully Probabilistic system, but is not statistically significantly worse than the either of the non-probabilistic systems. Notably, the Single Peak approximation achieves an error rate that is comparable to - perhaps even better than (confidence of 85%) - the Fully Probabilistic system. This is likely an artifact of our domain, in which activities seldom have tails that stack up into large multi-modal distributions.

Various parameters of the system were changed to evaluate the relative sensitivity of each approach. One such parameter is the user-specified risk tolerance. As expected, the payoff (in terms of reduced simulation errors) for using the Fully Probabilistic over a Means Only approach diminishes as the risk tolerance is increased. At a risk tolerance of 5%, they are distinct with 100% confidence, but even at 10% risk tolerance the statistical significance has dropped to 80%. At a risk tolerance of 50%, the Full Probabilistic system becomes mathematically equivalent to the Means Only system.

The difficulty of the problem also plays an important role in determining the Full Probabilistic system's dominance. As problem difficulty (measured as number of goals required) decreases, the Means Only approach gains on and eventually overtakes the Full Probabilistic approach in terms of simulation errors. Figure 7 shows the relevant confidence intervals.

Perhaps the most important change is that due to overall problem size. Figure 8 shows that both the probabilistic and non-probabilistic systems suffer a roughly exponential growth in simulation errors as a function of problem size. However, the slope of the Full Probabilistic system's function is significantly lower than that for Means Only. This indicates that the difference in simulation error counts will probably grow roughly exponentially was well. Figure 9 demonstrates

this fact more clearly by showing the per-problem improvement distribution. At large problem sizes, the Fully Probabilistic system vastly dominates the Means Only approach, while at small problem sizes, there is hardly any difference.

4 Conclusions

We have described an approach for directly dealing with plan uncertainty by collecting and merging the probability distributions from action duration and resource usage. The essential idea is that by maintaining such merged distributions, a planning system can ask specific questions about the risk of violating constraints at any time. Being able to ask such questions allows the planner to better balance its risk posture against its desire to achieve goals.

We have shown that augmenting a planner with such a probabilistic reasoning system allows for plans with execution-time quality superior to that which can be obtained without directly considering uncertainty. Though the underlying structure of the planner's decisions are not changed, the more robust risk assessment afforded by a probabilistic system allow the planner to focus its decisions on the most probable errors. As problem size increases or as resources become more saturated with subscriptions, such focus becomes more important to finding plans that perform well on execution.

The fully probabilistic system makes its gains using a $\mathrm{O}(2^n)$ algorithm, but we have also shown that a simple approximation technique that still tracks distributions can achieve comparable (and sometimes superior) results with only a $\mathrm{O}(n)$ algorithm.

The techniques we have demonstrated are applicable to most planning problems that satisfy a few constraints. First the resource and duration distributions of actions must be known. Second, the system must have a relatively high risk averseness for the probabilistic system to make a difference. In the current implementation, we have not handled many desirable planner capabilities such as direct temporal constraints or discrete state resources. We believe the techniques are still applicable for problems with such characteristics, albeit with some modification. Probabilistic reasoning is especially suited to problems of large size and high cost of failure.

References

[Bertsekas & Tsitsiklis, 1996] Bertsekas, D., and Tsitsiklis, J. 1996. *Neuro-dynamic Programming*. Athena, Belmont, MA.

[Blythe, 1999] Blythe, J. 1999. Decision-theoretic planning. *AI Magazine* 20(2):37–54.

[Boutilier, Dean, & Hanks, 1999] Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

[Boyan & Littman, 2000] Boyan, J., and Littman, M. 2000. Exact solutions to time-dependent MDPs. In *Advances in Neural Information Processing Systems*, 1026–1032.

[Bresina *et al.*, 2002] Bresina, J.; Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D.; and Washington, R. 2002. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

[Chien *et al.*, 2000] Chien, S.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 2000. Using iterative repair to improve the responsiveness of planning and scheduling. In *Proceedings of the International Conference on AI Planning and Scheduling*, 300–307.

[Chien *et al.*, 2002] Chien, S.; Sherwood, R.; Rabideau, G.; Castano, R.; Davies, A.; Burl, M.; Knight, R.; Stough, T.; Roden, J.; Zetocha, P.; Wainwright, R.; Gaasbeck, J. V.; Cappelaere, P.; and Oswald, D. 2002. The Techsat-21 autonomous space science agent. In *International Conference on Autonomous Agents*.

[Chien *et al.*, 2003] Chien, S.; Sherwood, R.; Tran, D.; Castano, R.; Cichy, B.; Davies, A.; Rabideau, G.; Tang, N.; Burl, M.; Mandl, D.; Frye, S.; Hengemihle, J.; Agostino, J.; Bote, R.; Trout, B.; Shulman, S.; Ungar, S.; Gaasbeck, J. V.; Boyer, D.; Griffin, M.; Burke, H.; Greeley, R.; Doggett, T.; Williams, K.; Baker, V.; and Dohm, J. 2003. Autonomous science on the EO-1 mission. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*.

[Dearden *et al.*, 2003] Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D.; and Washington, R. 2003. Incremental contingency planning. In *ICAPS Workshop on Uncertainty and Incomplete Information*.

[Frank *et al.*, 2001] Frank, J.; Jnsson, A.; Morris, R.; and Smith, D. 2001. Planning and scheduling for fleets of earth observing satellites. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space*.

[Globus *et al.*, 2002] Globus, A.; Crawford, J.; Lohn, J.; and Morris, R. 2002. Scheduling earth observing fleets using evolutionary algorithms: Problem description and approach. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

[Gordon, Salmond, & Smith, 1993] Gordon, N.; Salmond, D.; and Smith, A. 1993. Novel approach to nonliner/non gaussian bayesian state estimation.

[Koenig, 2001] Koenig, S. 2001. Agent-centered search. *AI Magazine* 22(4):109–131.

[Long & Fox, 2002] Long, D., and Fox, M. 2002. Domain modelling and the satellite observation domain. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

[Puterman, 1994] Puterman, M. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York.

[Smart & Kaelbling, 2000] Smart, W., and Kaelbling, L. 2000. Practical reinforcement learning in continuous spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning*.

Optimized Execution of Action Chains Using Learned Performance Models of Abstract Actions*

Freek Stulp and Michael Beetz
Intelligent Autonomous Systems Group, Technische Universität München
Boltzmannstrasse 3, D-85747 Munich, Germany
{stulp,beetz}@in.tum.de

Abstract

Many plan-based autonomous robot controllers generate chains of abstract actions in order to achieve complex, dynamically changing, and possibly interacting goals. The execution of these action chains often results in robot behavior that shows abrupt transitions between subsequent actions, causing suboptimal performance. The resulting motion patterns are so characteristic for robots that people imitating robotic behavior will do so by making abrupt movements between actions.

In this paper we propose a novel computation model for the execution of abstract action chains. In this computation model a robot first learns situation-specific performance models of abstract actions. It then uses these models to automatically specialize the abstract actions for their execution in a given action chain. This specialization results in refined chains that are optimized for performance. As a side effect this behavior optimization also appears to produce action chains with seamless transitions between actions.

1 Introduction

In recent years, a number of autonomous robots, including WITAS [Doherty *et al.*, 2000], Minerva [Thrun *et al.*, 1999], and Chip [Firby *et al.*, 1996], have shown impressive performance in long term demonstrations. These robots have in common that they generate, maintain, and execute chains of discrete actions to achieve their goals. The use of plans enables these robots to flexibly interleave complex and interacting tasks, exploit opportunities, and optimize their intended course of action.

To allow for plan-based control, the plan generation mechanisms are equipped with libraries of actions and causal models of these actions. These models include specifications of action effects and of the conditions under which the actions are executable. For good reasons, the action models are specified abstractly and disregard many aspects of the situations before, during, and after their execution. This abstractness has several big advantages. Programmers need to supply fewer actions because viewed at an abstract level the actions are applicable to a broader range of situations. Also the models themselves become more concise. This not only eases the job of the programmers but also the computational task of the automatic planning systems. At more abstract levels the search space of plans is substantially smaller and fewer interactions between actions need to be considered.

The advantages of abstraction, however, come at a cost. Because the planning system considers actions as black boxes with performance independent of the prior and subsequent steps, the planning system cannot tailor the actions to the contexts of their execution. This often yields suboptimal behavior with abrupt transitions between actions. The resulting motion patterns are so characteristic for robots that people trying to imitate robotic behavior will do so by making abrupt movements between actions. In contrast, one of the impressive capabilities of animals and humans is their capability to perform chains of actions in optimal ways and with seamless transitions between subsequent actions.

Let us illustrate these points using a simple scenario in autonomous robot soccer that is depicted in Figure 1. The starting situation is shown in the left sub-figure. The planner issues a three step plan: 1) go to the ball; 2) dribble the ball to shooting position; 3) shoot. If the robot naively executed the first action (as depicted in the center sub-figure), it might arrive at the ball with the goal at its back. This is an unfortunate position from which to start dribbling towards the goal. The problem is that in the abstract view of the planner, being at the ball is considered sufficient for dribbling the ball and the dynamical state of the robot arriving at the ball is considered to be irrelevant for the dribbling action.

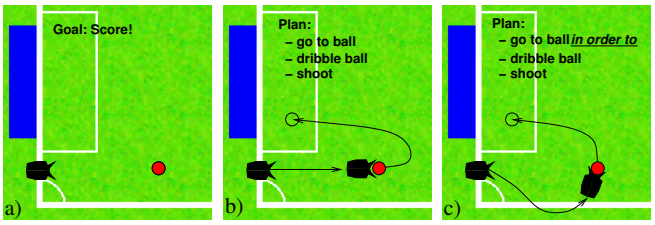


Figure 1: Alternative execution of the same plan

What we would like the robot to do instead is to go to the ball *in order* to dribble it towards the goal afterwards. The

*The work described in this paper was partially funded by the Deutsche Forschungsgemeinschaft in the SPP-1125.

robot should, as depicted in the Figure 1c, perform the first action sub-optimally in order to achieve a much better position for executing the second plan step. The behavior shown in Figure 1c exhibits seamless transitions between plan steps and has higher performance, achieving the ultimate goal in less time.

In this paper we propose a novel computational model for plan execution that enables the planner to keep its abstract action models and that optimizes action chains at execution time. The basic idea of our approach is to learn performance models of abstract actions off-line from observed experience. These performance models are rules that predict the situation- and parameterization-specific performance of abstract actions, e.g. the expected duration. Then at execution time, our system determines the set of parameters that are not set by the plan and therefore define the possible action executions. It then determines for each abstract action the parameterization such that the predicted performance of the action chain is optimal.

The technical contributions of this paper are threefold. First, we propose a novel computational model for the execution time optimization of action chains, presented in section 2. Second, we show how situation-specific performance models for abstract actions can be learned automatically (section 3). Third, we describe a mechanism for subgoal (post-condition) refinement for action chain optimization. We apply our implemented computational model to chains of navigation plans with different objectives and constraints and different task contexts (section 4). We show for typical action chains in robot soccer our computational model achieves substantial and statistically significant performance improvements for action chains generated by robot planners (section 5).

2 System overview

This section introduces the basic concepts upon which we base our computational model of action chain optimization. Using these concepts we define the computational task and sketch the key ideas for its solution.

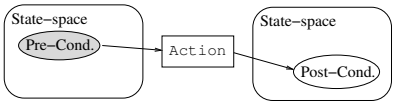
2.1 Conceptualization

Our conceptualization for the computational problem is based on the notion of actions, performance models of actions, teleo-operators, teleo-operator libraries, and chains of teleo-operators. In this section we will introduce these concepts.

Actions are control programs that produce streams of control signals, based on the current estimated state, thereby influencing the state of the world. One of the actions used here is `goToPose`, which navigates the robot from the current pose (at time t) $[x_t,y_t,\phi_t]$ to a future destination pose $[x_d,y_d,\phi_d]$ by setting the translational and rotational velocity of the robot:

`goToPoseAction`$(x_t,y_t,\phi_t,x_d,y_d,\phi_d) \rightarrow v_{tra},v_{rot}$

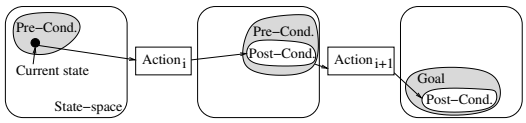
Teleo-operators (TOPs) consist of an action, as well as pre- and post-conditions [Nilsson, 1994]. The post-condition represents the intended effect of the TOP, or its goal. It specifies a region in the state space in which the goal is satisfied. The pre-condition region with respect to a temporally extended action is defined as the set of world states in which continuous execution of the action will eventually satisfy the post-condition. They are similar to Action Schemata or STRIPS operators in the sense that they are temporally extended actions that can be treated by the planner as if they were atomic actions.



The `goToPoseTOP` has the empty pre-condition, as it can be executed from any state in the state space. Its post-condition is $[x_t \approx x_d,y_t \approx y_d, \phi_t \approx \phi_d]$. Its action is `goToPoseAction`.

TOP libraries contain a set of TOPs that are frequently used within a given domain. In many domains, only a small number of control routines suffices to execute most tasks, if they are kept general and abstract, allowing them to be applicable in many situations. Our library contains the TOPs: `goToPoseTOP` and `dribbleBallTOP`.

A **TOP chain** for a given goal is a chain of TOPs such that the pre-condition of the first top is satisfied by the current situation, and the post-condition of each step satisfies the pre-condition of the subsequent TOP. The post-condition of the last TOP must satisfy the goal. It represents a valid plan to achieve the goal.



Performance models of actions map a specific situation onto a performance measure. In this paper the performance measure is time. Alternatives could be chance of success or accuracy. These models can be used to predict the performance outcome of an action if applied in a specific situation, by specifying the current state (satisfying the pre-conditions) and end state (satisfying the post-conditions).

`goToPoseAction.performance`$(x_t,y_t,\phi_t,x_d,y_d,\phi_d) \rightarrow t$

2.2 Computational task and solution idea

The on-line computational task is to optimize the overall performance of a TOP chain. The input consists of a TOP chain that has been generated by a planner, that uses a TOP library as a resource. The output is an intermediate refined subgoal that optimizes the chain, and is inserted in the chain. Executing the TOP chain is simply done by calling the action of each TOP. This flow is displayed in Figure 2.

To optimize action chains, the pre- and post-conditions of the TOPs in the TOP chains are analyzed to determine which variables in the subgoal may be freely tuned. These are the variables that specify future states of the robot, and are not constrained by the pre- and post-conditions of the respective TOP. For the optimization of these free variables, performance models of the actions are required. Off-line, these models are learned from experience for each action in the TOP library. They are used by the subgoal refinement system during execution time, but available as a resource to other systems as well.

One of the big advantages of our approach is that neither TOP library, nor the generation of TOP chains (the planner)

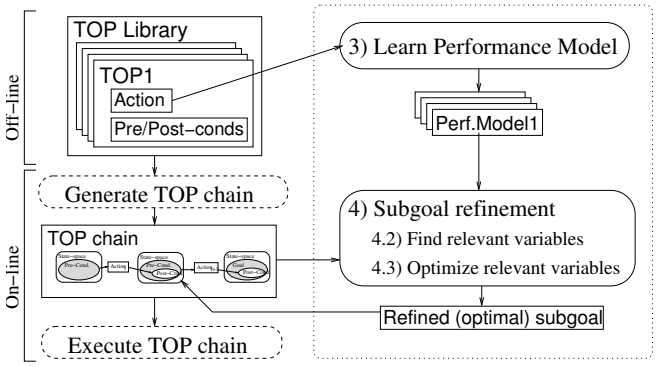


Figure 2: System Overview

nor the TOP chain executor need to be modified in any way to accommodate the action chain optimization system.

3 Learning performance models

The actual optimization of TOP chains, to be discussed in section 4, needs performance models of each action in the TOP library. For each action, the robot learns a function that maps situations to the cost of performing this action in the respective situation. In this paper, the performance measure is time, although our mechanisms applies to other cost functions without requiring any change. The robot will learn the performance function 1) from experience 2) using a transformed state space 3) by partitioning the state space 4) by approximating functions to the data in each of these partitions. We will first motivate *why*, and then explain *how* this has been implemented.

Let us consider the navigation action `goToPoseAction`. This navigation action is based on computing a Bezier curve, and trying to follow it as closely as possible. `dribbleBallAction` uses the same method, but restricts deceleration and rotational velocity, so as not to loose the ball. We abstract away from their implementation, as our methods consider the actions to be black boxes, whose performance we learn from observed experience.

The robot learns the performance function **from experience**. It executes the action under varying situations, observes the performance, and logs the experience examples. Since the method is based solely on observations, it is also possible to acquire models of actions whose internal workings are not accessible. The robot executed each action 1000 times, with random initial and destination poses. The robot recorded the direct variables and the time it took to reach the destination state at 10Hz, thereby gathering 75 000 examples of the format $[x_t,y_t,\phi_t,x_d,y_d,\phi_d,time]$ per action. These examples were gathered using our simulator, which uses learned dynamics models of the Pioneer I platform. It has proven to be accurate enough to port control routines from the simulator to the real robot without change. Using our Pioneer I robots, acquiring this amount of data would take approximately two hours of operation time.

The variables that were recorded do not necessarily correlate well with the performance. We therefore design a **transformed feature space** with less features, but the same potential for learning accurate performance models. In Figure 3 it is shown how exploiting transformational and rotational invariance reduces our original six-dimensional feature space into a three-dimensional one, with the same predictive power.

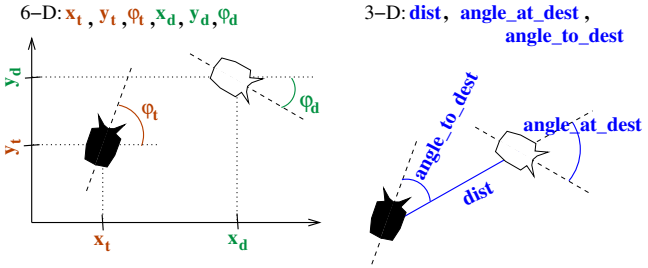


Figure 3: Transformation of the original state space into a lower-dimensional feature space.

Currently, we perform the transformation manually for each action. In our ongoing research we are investigating methods to automate the transformation. By explicitly representing and reasoning about the physical meaning of state variables, we research feature language generation methods.

The last step is to **approximate a function** to the transformed data. This is done using model trees. Model trees are functions that map continuous or nominal features to a continuous value. The function is learned from examples, by a piecewise partitioning of the feature space. A linear function is fitted to the data in each partition. Model trees are a generalization of decision trees, in which the nominal values at the leaf nodes are replaced by line segments. We use model trees because 1) they can be transformed into sets of rules that are suited for human inspection and interpretation 2) comparative research shows they are the best [Belker, 2004; Balac, 2002] 3) They tend to use only relevant variables. This means we can start off with many more features than are needed to predict performance, having the model tree function as an automatic feature selector. The tree was actually learned on an 11-dimensional feature space $[x,y,\phi,x_g,y_g,\phi_g,dx,dy,dist,angle\_to\_dest,angle\_at\_dest]$, the model tree algorithm automatically discovered that only $[dist,angle\_to\_dest,angle\_at\_dest]$ are necessary to accurately predict performance.

We have trained a model tree on the gathered data, yielding rules of which we will now present an example. In Figure 4, we depict an example situation in which $dist$ and $angle\_to\_dest$ are to 2.0m and 0° respectively. Given these values we could plot a performance function for varying values of $angle\_at\_dest$. These plots are also depicted in Figure 4, once in a Cartesian, once in a polar coordinate system. In the linear plot we can clearly see five different line segments. This means that the model tree has partitioned the feature space for $dist$=2 and $angle\_to\_dest$=0 into five areas, each with its own linear model. Below the two plots one of the learned model tree rules that applies to this situation is displayed. An arrow indicates its linear model in the plots.

The polar plot clearly shows the dependency of predicted execution time on the angle of approach for the example situation. Approaching the goal at 0 degrees is fastest, and would take a predicted 2.1s. Approaching the goal at 180

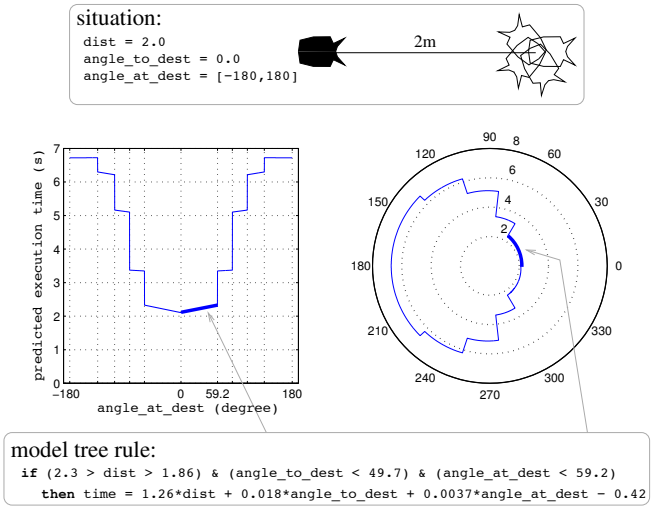


Figure 4: An example situation, two graphs of time prediction for this situation with varying $angle\_at\_dest$, and the model tree rule for one of the line segments.

degrees means the robot would have to navigate around the goal point, taking much longer (6.7s). p To evaluate the accuracy of the performance models, we again randomly generate 1000 new test situations. For the `goToBall` routine, the mean absolute error and root-mean-square error between predicted and actual execution time were 0.31s and 0.75s. For the `dribbleBall` routine these values were 0.29s and 0.73s. As we will see, these errors are accurate enough to optimize action chains.

4 Automatic subgoal refinement

As depicted in Figure 2, the automatic subgoal refinement system takes the performance models and a chain of teleoperators as an input, and returns a refined intermediate goal state that has been optimized with respect to the performance of the overall action chain. To do this we need to specify all the variables in the task, and recognize which of these variables influence the performance and are not fixed. These variables form a search space in which we will optimize the performance using the learned action models.

4.1 State variables

In the dynamic system model [Dean and Wellmann, 1991] the world changes through the interaction of two processes: the *controlling process*, in our case the low-level control programs implementing the action chains generated by the planner, and the *controlled process*, in our case the behavior of the robot. The evolution of the dynamic system is represented by a set of *state variables* that have changing values. The controlling process steers the controlled process by sending *control signals* to it. These control signals directly set some of the state variables and indirectly other ones. The affected state variables are called the *controllable* state variables. The robot for instance can set the translational and rotational velocity directly, causing the robot to move, thereby indirectly influencing future poses of the robot.

For the robot, a subset of the state variables is *observable* to its perceptive system, and they can be estimated using a state estimation module. For any controller there is a distinction between *direct* and *derived* observable state variables. All direct state variables for the navigation task are depicted in Figure 5. Direct state variables are directly provided by state estimation, whereas derived state variables are computed by combinations of direct variables. No extra information is contained in derived variables, but if chosen well, derived variables are better correlated to the control task.

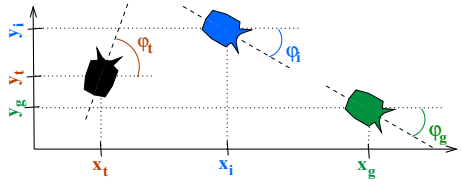


Figure 5: Direct state variables relevant to the navigation task

State variables are also used to specify goals internal to the controller. These variables are *bound*, conform to planning terminology. It is the controller's goal to have the bound internal variables (approximately) coincide with the external observable variables. The robot's goal to arrive at the intermediate position could be represented by the state variables $[x_i,y_i]$. By setting the velocities, the robot can influence its current position $[x_t,y_t]$ to achieve $[x_t \approx x_i,y_t \approx y_i]$.

4.2 Determining the search space

To optimize performance, only variables that actually influence performance should be tuned. In our implementation, this means only those variables that are used in the model tree to partition the state space at the nodes, or used in the linear functions at the leaves.

In both the learned model trees for the actions `goToPoseAction` and `dribbleBallAction`, the relevant variables are $dist$, $angle\_to\_dest$ and $angle\_at\_dest$. These are all derived variables, computed from the direct variables $[x_t,y_t,\phi_t,x_i,y_i,\phi_i]$ and $[x_i,y_i,\phi_i,x_g,y_g,\phi_g]$, for the first and second action respectively. So by changing these direct variables, we would change the indirect variables computed from them, which in effect would change the performance.

But may we change all these variables at will? Not x_t,y_t, or ϕ_t, as we cannot simply change the current state of the world. Also we may not alter bound variables that the robot has committed to, being $[x_i,y_i,x_g,y_g,\phi_g]$. Changing them would make the plan invalid.

This only leaves the free variable ϕ_i, the angle at which the intermediate goal is approached. This acknowledges our intuition from Figure 1 that changing this variable will not make the plan invalid, and that it will also influence the overall performance of the plan. We are left with a one-dimensional search space to optimize performance.

4.3 Optimization

To optimize the action chain, we will have to find those values for the free variables for which the overall performance of the action chain is the highest. The overall performance is

estimated by summing over the performance models of all actions that constitute the action chain. In Figure 6 the first two polar plots represent the performance of the two individual actions for different values of the only free variable, which is the angle of approach. The overall performance is computed by adding those two, and is depicted in the third polar plot.

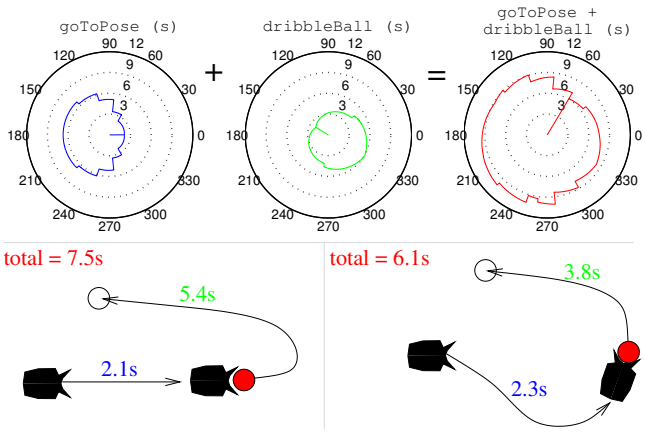


Figure 6: Selecting the optimal subgoal by finding the optimum of the summation of all action models in the chain.

The fastest time in the first polar plot is 2.1s, for angle of approach of 0.0 degrees. The direction is indicated from the center of the plot. However, the total time is 7.5s, because the second action takes 5.4s for this angle . These values can be read directly from the polar plots. However, this value is not the optimum overall performance. The minimum of the overall performance is 6.1s, as can be read from the third polar plot. Below the polar plots, the situation of Figure 1 is repeated, this time with the predicted performance for each action.

We expect that for higher-dimensional search spaces, exhaustive search may be infeasible. Therefore, other optimization techniques will have to be investigated.

5 Results

To determine the influence of subgoal refinement on the overall performance of the action chain, we generated 1000 situations with random robot, ball and final goal positions. The robot executed each navigation task twice, once with subgoal refinement, and once without. The results are summarized in Table 1. First of all, the overall increase in performance over the 1000 runs is 10%. We have split these cases into those in which the subgoal refinement yielded a higher, equal or lower performance in comparison to not using refinement. This shows that the performance improved in 533 cases, and in these cases causes a 21% improvement. In 369 cases, there was no improvement. This is to be expected, as there are many situations in which the three positions are already optimally aligned (e.g. in a straight line), and subgoal refinement will have no effect.

Unfortunately, applying our method causes a decrease of performance in 98 out of 1000 runs. To analyze in which cases subgoal refinement decreases performance, we labeled each of the above runs `Higher`, `Equal` or `Lower`. We then trained a decision tree to predict this nominal value. This tree yields four simple rules which predict the performance difference correctly in 86% of given cases. The rules declare that performance will stay equal if the three points are more or less aligned, and will only decrease if the final goal position is in the same area as which the robot is, but only if the robot's distance to the intermediate goal is smaller than 1.4m. Essentially, this last rule states that the robot using the Bezier-based `goToBallAction` has difficulty approaching the ball at awkward angles if it is close to it. In these cases, small variations in the initial position lead to large variations in execution time, and learning an accurate, general model of the action fails. The resulting inaccuracy in temporal prediction causes suboptimal optimization. Note that this is a shortcoming of the action itself, not the chain optimization methods. We will investigate if creating a specialized action for the cases in which Bezier based navigation is unsuccessful could solve these problems.

Before filtering	Total	Higher	Equal	Lower
# runs	1000	533	369	98
improvement	10%	21%	0%	-10%
After filtering	**Total**	**Higher**	**Equal**	**Lower**
# runs	1000	505	485	10
improvement	12%	23%	0%	-6%

Table 1: Results, before and after filtering for cases in which performance loss is predicted.

We then gathered another 1000 runs, as described above, but only applied subgoal refinement if the decision tree predicted applying it would yield a higher performance. Although increase in overall performance is not so dramatic (from 10% to 12%), the number of cases in which performance is worsened by applying subgoal refinement has decreased from 98 (10%) to 10 (1%). Apparently, the decision tree correctly filtered out cases in which applying subgoal refinement would decrease performance.

Summarizing: subgoal refinement with filtering yields a 23% increase in performance half of the time. Only once in a hundred times does it cause a small performance loss.

6 Related Work

Most similar to our work is the use of model trees to learn performance models to optimize Hierarchical Transition Network plans [Belker, 2004]. In this work, the models are used to select the next action in the chain, whereas we refine an existing action chain. Therefore, the planner can be selected independently of the optimization process.

Reinforcement Learning (RL) is another method that seeks to optimize performance, specified by a reward function. Recent attempts to combat the curse of dimensionality in RL have turned to principled ways of exploiting temporal abstraction. Several of these *Hierarchical Reinforcement Learning* methods, e.g. (Programmable) Hierarchical Abstract Machines, MAXQ, and Options, are described in the overview paper [Barto and Mahadevan, 2003]. All these approaches use the concept of actions (called 'machines', 'subtasks', or 'options' respectively). In our view, the benefits of our methods are that they acquire more informative per-

formance measures, facilitate the reuse of action models, and scale better to continuous and complex state spaces.

The performance measures we can learn (time, success rate, accuracy) are *informative* values, with a meaning in the physical world. Future research aims at developing meaningful composites of individual models. We will also investigate dynamic objective functions. In some cases, it is better to be fast at the cost of accuracy, and sometimes it is better to be accurate at the cost of speed. By weighting the performance measures time and accuracy accordingly in a composite measure, these preferences can be expressed at execution time. Since the (Q-)Value compiles all performance information in a single non-decomposable numeric value, it cannot be reasoned about in this fashion.

The methods we proposed *scale* better to continuous and complex state spaces. We are not aware of the application of Hierarchical Reinforcement Learning to (accurately simulated) continuous robotic domains.

In Hierarchical Reinforcement Learning, the performance models of actions (Q-Values) are learned in the calling context of the action. Optimization can therefore only be done in the context of the pre-specified hierarchy/program. In contrast, the success of action selection in complex robotic projects such as WITAS, Minerva and Chip depends on the on-line autonomous sequencing of actions through planning. Our methods learn abstract performance models of actions, independent of the context in which they are performed. This makes them *reusable*, and allows for integration in planning systems.

The only approach we know of that explicitly combines planning and RL is RL-TOPS (*Reinforcement Learning - Teleo Operators*) [Ryan and Pendrith, 1998]. Abrupt transitions arise here too, and the author recognizes that "cutting corners" between actions would improve performance, but does not present a solution.

Many behavior based approaches also achieve smooth motion by a weighted mixing of the control signals of various actions [Saffiotti *et al.*, 1995]. Since there are no discrete transitions between actions, they are also not visible in the execution. Since achieving optimal behavior is not an explicit goal, it is left to chance, not objective performance measures.

A very different technique for generating smooth transitions between skills has been developed for quadruped robots [Hoffmann and Düffert, 2004], also in the RoboCup domain. The periodic nature of robot gaits allows their meaningful representation in the frequency domain. Interpolating in this domain yields smooth transitions between walking skills. Since the actions we use are not periodic, these methods do unfortunately not apply.

7 Conclusion and Future Work

On-line optimization of action chains allows the use of planning with abstract actions, without losing performance. Optimizing the action chain is done by refining under-specified intermediate goals, which requires no change in the planner or plan execution mechanisms. To predict the optimal overall performance, performance models of each individual abstract action are learned off-line and from experience, using model trees. It is interesting to see that requiring optimal performance can implicitly yield smooth transitions in robotic and natural domains, even though smoothness in itself is not an explicit goal in either domain.

Applying subgoal refinement to the presented scenario yields good results. However, the computational model underlying the optimization is certainly not specific to this scenario, or to robot navigation. In principle, learning action models from experience using model trees is possible for any action whose relevant state variables can be observed and recorded. The notion of controllable, bound and free state variables are taken directly from the dynamic system model and planning approaches, and apply to any scenario that uses these paradigms. Our future research therefore aims at applying these methods in other domains, for instance robots with articulated arms and grippers, for which we also have a simulator available.

Currently, we are evaluating if subgoal refinement improves plan execution on real Pioneer I robots as much as it does in simulation. Previous research has shown that action models learned in simulation can be applied to real situations with good result [Buck *et al.*, 2002; Belker, 2004].

References

[Balac, 2002] N. Balac. *Learning Planner Knowledge in Complex, Continuous and Noisy Environments.* PhD thesis, Vanderbilt University, 2002.

[Barto and Mahadevan, 2003] A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event systems*, 2003.

[Belker, 2004] T. Belker. *Plan Projection, Execution, and Learning for Mobile Robot Control*. PhD thesis, Department of Applied Computer Science, Univ. of Bonn, 2004.

[Buck *et al.*, 2002] S. Buck, M. Beetz, and T. Schmitt. Reliable Multi Robot Coordination Using Minimal Communication and Neural Prediction. In *Selected Contributions of the Dagstuhl Seminar "Plan-based Control of Robotic Agents"*, 2002.

[Dean and Wellmann, 1991] T. Dean and M. Wellmann. *Planning and Control*. Morgan Kaufmann Publishers, 1991.

[Doherty *et al.*, 2000] P. Doherty, G. Granlund, K. Kuchcinski, E. Sandewall, K. Nordberg, E. Skarman, and J. Wiklund. The WITAS unmanned aerial vehicle project. In *Proceedings ECAI-00*, 2000.

[Firby *et al.*, 1996] R. Firby, P. Prokopowicz, M. Swain, R. Kahn, and D. Franklin. Programming CHIP for the IJCAI-95 robot competition. *AI Magazine*, 17(1):71–81, 1996.

[Hoffmann and Düffert, 2004] J. Hoffmann and U. Düffert. Frequency space representation and transitions of quadruped robot gaits. In *Proceedings of the 27th conference on Australasian computer science*, 2004.

[Nilsson, 1994] N.J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1994.

[Ryan and Pendrith, 1998] M. Ryan and M. Pendrith. RL-TOPs: an architecture for modularity and re-use in reinforcement learning. In *Proc. 15th International Conf. on Machine Learning*, 1998.

[Saffiotti *et al.*, 1995] A. Saffiotti, K. Konolige, and E.H. Ruspini. A multivalued-logic approach to integrating planning and control. *Artificial Intelligence*, 1995.

[Thrun *et al.*, 1999] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A tour-guide robot that learns. In *KI - Kunstliche Intelligenz*, pages 14–26, 1999.

UNCERTAINTY

Encoding formulas with partially constrained weights in a possibilistic-like many-sorted propositional logic

Salem Benferhat
CRIL-CNRS, Université d'Artois
rue Jean Souvraz
62307 Lens Cedex. France.
benferhat@cril.univ-artois.fr

Henri Prade
IRIT (CNRS-UPS)
118, Route de Narbonne
31062 Toulouse Cedex 04, France.
prade@irit.fr

Abstract

Possibilistic logic offers a convenient tool for handling uncertain or prioritized formulas and coping with inconsistency. Propositional logic formulas are thus associated with weights belonging to a linearly ordered scale. However, especially in case of multiple source information, only partial knowledge may be available about the relative ordering between weights of formulas. In order to cope with this problem, a two-sorted counterpart of possibilistic logic is introduced. Pieces of information are encoded as clauses where special literals refer to the weights. Constraints between weights translate into logical formulas of the corresponding sort and are gathered in a distinct auxiliary knowledge base. An inference relation, which is sound and complete with respect to preferential model semantics, enables us to draw plausible conclusions from the two knowledge bases. The inference process is characterized by using "forgetting variables" for handling the symbolic weights, and hence an inference process is obtained by means of a DNF compilation of the two knowledge bases.

1 Introduction

Information is often pervaded with uncertainty, and logics of different types have been developed for handling uncertain pieces of knowledge, some based on probability theory, others based on non-additive formalisms such as possibility theory. The lack of total certainty of a piece of information is then assessed by means of an evaluation that estimates the degree of certainty of the piece of information under the form of a precise value, or at least of a bound that constrains its value. For instance, a possibilistic logic expression of the form (p, a) encodes the constraint $N(p) \geq a$, where p is a proposition, $a \in (0, 1]$, and N is a necessity measure [Dubois *et al.*, 1994], with the intended meaning that p is certain at level at least a.

In this paper, only partial information is supposed to be available about the relative ordering between the weights of the formulas in the base. More precisely, propositional formulas are associated with symbolic weights, and a set of constraints on these weights is specified to express the relative importance of these weights. These weights may be compound symbolic expressions (e.g., as the result of formal computations) involving maximum and minimum operations.

For handling these symbolic weights, we propose an approach that uses a logical encoding of the weights. Indeed, one may think of expressing that a piece of information p is not totally certain, by stating that p is true *under some (unspecified) condition*, namely "things are not abnormal". Thus, the uncertain statement p can be written $p \vee A$, which can be read "p is true or it is abnormal (A)". Then note that if one simultaneously asserts $p \vee A$ and $\neg p \vee A'$, i.e. it is somewhat certain that p is true and it is also somewhat certain that p is false, $A \vee A'$ can be derived from the two pieces of information, which expresses that in a way or another we are in an abnormal situation. In this paper, since $p \vee A$ will be used as an encoding of the possibilistic formula (p, a), A may be a propositional formula if a is a symbolic expression. For instance, the case of multiple abnormality situations for p will be encoded by $p \vee A_1 \vee ... \vee A_n$.

This suggests a way of dealing with propositional formulas with partially constrained weights in a purely logical setting. In possibilistic logic, formulas with a weight strictly greater than the inconsistency level of the knowledge base are immune to inconsistency and can be safely used in deductive reasoning. In the proposed approach, the weights attached to the inferred formulas are handled as symbolic expressions, as well as the inconsistency level of the base. A procedure is described which enables us to determine when the available information is enough for knowing if a symbolic weight is greater than another or not.

The paper proposes a unified and general framework to represent and handle partially-constrained weighted formulas, using propositional logic. Its main contributions are:

- to encode available pieces of information Σ using a propositional knowledge base K_Σ. Each symbolic weight a is represented by a propositional expression A. Each compound expression is associated with a propositional formula, obtained by replacing maximum by a conjunction, and minimum by a disjunction. An uncertain formula (p, a) will be represented by a propositional logical formula $p \vee A$, where p and A are built using two different sets of variables: V and S. Intuitively, $p \vee A$ means that p is uncertain, and its uncertainty degree is

encoded by a logical formula A.

- to encode constraints C on symbolic weights with another propositional knowledge base K_C. The inequality $a \geq b$ will be represented using material implication. The fact that a is at least as large as b, will be encoded by a propositional formula $A \rightarrow B$ (i.e., $\neg A \vee B$). Maximum and minimum operations are also encoded here using respectively conjunctions and disjunctions.
- to define an inference relation in order to draw plausible conclusions from K_Σ and K_C. This inference is sound and complete with respect to a semantics based on preferential models. It extends the possibilistic logic (where the constraints on symbolic weights induce a total preordering).
- to characterize this inference process by "forgetting" variables of V, and by "forgetting" negative literals from S. The inference process basically comes down to infer from K_Σ and K_C the strongest positive formula (that does not contain negation symbol), denoted $Linc(K_\Sigma, K_C)$, that only contains variables from S. Intuitively, $Linc(K_\Sigma, K_C)$ represents the logical counterpart of the inconsistency degree of the knowledge base.
- to use recent results on compilations (e.g., [Darwiche, 2004]) to compute plausible inferences. More precisely, the knowledge base $K_\Sigma \cup K_C$ is first compiled into DNF. These formats allow then to have a linear computation of $Linc(K_\Sigma, K_C)$.

After a brief survey of possibilistic logic in Section 2, Section 3 states the problem of reasoning from possibilistic formulas with partially constrained uncertainty weights. Section 4 provides its purely logical counterpart as a two-sorted logic and defines a sound and complete inference process from the two knowledge bases encoding respectively the pieces of uncertain information, and the constraints on the uncertainty levels. The handling of symbolic weights in the inference process is then characterized in section 5 in terms of forgetting variables, and in terms of DNF (or d-DNNF compilation) of the knowledge bases. This section also briefly considers the case of totally ordered information corresponding to standard possibilistic knowledge bases.

2 Brief background on possibilistic logic

We start with a brief refresher on possibilistic logic (for more details see [Dubois *et al.*, 1994]). A possibilistic logic formula is a pair made of a classical logic formula and a weight a expressing certainty. The weight $a \in (0, 1]$ of a formula p is interpreted as the lower bound of a necessity measure N, i.e., the possibilistic logic expression (p, a) is understood as $N(p) \geq a$. Since $N(p \wedge q) = min(N(p), N(q))$ it is always possible to put a possibilistic formula (p, a) under the form of a conjunction of clauses (p_i, a) if $p \equiv \wedge_i\ p_i$. The basic inference rule in possibilistic logic put in clausal form is the resolution rule: $(\neg p \vee q, a); (p \vee r, b) \vdash (q \vee r, min(a, b))$. Classical resolution is retrieved when all the weights are equal to 1. Let $\Sigma = \{(p_i, a_i) : i = 1, n\}$ be a knowledge base. The level of inconsistency of Σ is defined as:

$Inc(\Sigma) = max\{a : \Sigma_a \vdash \bot)\}$ (by convention $max\emptyset = 0$).
where $\Sigma_a = \{p_i : (p_i, a_i) \in \Sigma,$ and $a_i \geq a\}$.

It can be shown that $Inc(\Sigma) = 0$ iff $\Sigma^* = \{p_i : (p_i, a_i) \in \Sigma\}$ is consistent in the usual sense.

Refutation can be easily extended to possibilistic logic. Proving (p, a) from Σ amounts to adding $(\neg p, 1)$, put in clausal form, to Σ, and using the above rules repeatedly until getting $\Sigma \cup \{(\neg p, 1)\} \vdash (\bot, a)$. Clearly, we are interested here in getting the empty clause $\bot$ with the greatest possible weight, i.e., $Inc(\Sigma \cup \{(\neg p, 1)\}) \geq a$. Indeed, the conclusion (p, a) is valid only if $a > Inc(\Sigma)$.

Semantic aspects of possibilistic logic, including soundness and completeness results with respect to the above syntactic inference, are presented in [Dubois *et al.*, 1994]. Semantically, a possibilistic knowledge base $\Sigma = \{(p_i, a_i) : i = 1, n\}$ is understood as a complete pre-ordering on the set of interpretations: $\omega >_\Sigma \omega'$ iff $max\{a_i : (p_i, a_i) \in \Sigma$ and $\omega' \not\models p_i\} > max\{a_j : (p_j, a_j) \in \Sigma$ and $\omega \not\models p_j\}$. Thus ω is all the less plausible as it falsifies formulas of higher degrees.

3 Possibilistic logic with symbolic weights

3.1 Representing beliefs

Let V be a set of propositional variables. Let L_V be a propositional language built from V using the propositional connectors $\wedge, \vee, \neg$. Let H be a set of symbolic weights or variables. Each symbol in H takes its value in the interval $[0, 1]$.

In the following, ordinary propositions are denoted by lower case letters $p, q, r, ...$, symbolic weights are denoted by lower case letters from the beginning of the alphabet $a, b, c....$

Let $\Sigma = \{(p_i, a_i) : i = 1, ..., n\}$ be a knowledge base with symbolic weights, where (p_i, a_i) expresses that p_i is a believed with a symbolic weight a_i. a_i can be a compound expression, namely a max/min expression. More precisely, max/min expressions are obtained only using the two following rules: i) $a \in H$ is a max/min expression, ii) if a and b are max/min expressions, then $max(a, b)$ and $min(a, b)$ are also max/min expressions.

Example 1 *Let p and r be two propositional symbols. Let a, b, c, d, e be five symbolic weights. In the following, we will use the following base:*

$\Sigma = \{(p, max(a, b)), (\neg p \vee r, min(c, d)), (\neg r, e))\}$,

to illustrate the main concepts of the paper. Σ reflects that p is asserted by two sources having reliability a and b respectively, while for $\neg p \vee r$ it is unsure if the information should be considered as having reliability equal to c or to d.

3.2 Representing constraints

Constraints bearing on symbolic weights are described by a set of inequalities. A simple form of constraints is: $a_i \geq b_i$, where a_i and b_i are elements of H. These constraints $a_i \geq b_i$ restrict the possible values that a_i and b_i can take in $[0, 1]$.

More generally, the set of constraints are inequalities between max/min expressions of the form:

$C = \{a_i \geq b_i : i = 1, .., s$ and
a_i and b_i are max/min expressions$\}$.

Note that any set of constraints C can be equivalently rewritten into a canonical form:

$C = \{max(a_{i1}, ..., a_{in}) \geq min(b_{i1}, ..., b_{im}) : i = 1, .., r, \text{ and } a_{ij} \in H, b_{jk} \in H\}$.

This follows from the facts that i) $a \geq max(b, c)$ is equivalent to $a \geq b$ and $a \geq c$, ii) $min(b, c) \geq a$ is equivalent to $b \geq a$ and $c \geq a$, and iii) max and min are distributive.

In the following, we assume that constraints in C are in this canonical form.

Given a set of constraints C, we are interested in checking whether a given equality $a \geq b$ (a and b are max/min expressions) follows from this set of constraints.

Definition 1 *Let a and b be two max/min expressions.*

- *An assignment I is a function that assigns to each symbolic weight a degree belonging to $(0, 1]$.*
- *An assignment I is a solution of C if it satisfies each constraint of C.*
- *$a \geq b$ follows from C if each solution of C is also a solution of $a \geq b$.*

The derivation of strict inequality can be defined recursively as follows. Let a and b be two elements of H, then $a > b$ follows from C iff $a \geq b$ follows from C and $b \geq a$ does not follow from C (i.e., there is no proof for $b \geq a$). Now, let a_1, ..., a_n, b_1, ..., b_m be symbolic weights in H. Then $max(a_1, ..., a_n) > min(b_1, ..., b_m)$ is derived from C if there exist a_i and b_j such that $a_i > b_j$. Lastly, the derivation from C of strict inequality between two general max/min expressions can be defined using the fact that this derivation is equivalent to a derivation of a set strict inequalities of the canonical form $max(a_1, ..., a_n) > min(b_1, ..., b_m)$, with a_i's and b_i's as symbolic weights.

Example 2 *With the base of the above example, we will also consider the following set of constraints :*

$C = \{a \geq e, c \geq a, d \geq e\}$.

Note that since $e \geq a$ does not belong to the reflexive and transitive closure of C, we may consider using a closed world assumption that the inequality is strict. The same holds for other inequalities.

3.3 Plausible inference and semantics

Given a knowledge base with symbolic weights, $Inc(\Sigma)$ is now a max/min expression of symbolic weights and the plausible inference of p amounts to establish from C that $Inc(\Sigma \cup \{(\neg p, 1)\}) > Inc(\Sigma)$.
Here the weight 1 continues to be the top certainty level, i.e., $\forall a \in H, 1 \geq a$.

Our semantics is based on preferential models. We use the principle of best-out ordering defined in [Benferhat *et al.*, 1993], to derive from Σ a partial pre-order on the set of interpetations, denoted by $>_\Sigma$, in agreement with possibilistic logic. Let ω, and ω' be two interpretations. Then:

$\omega >_\Sigma \omega'$ iff for each (p_j, b_j) such that $\omega \not\models p_j$, there exists (q_i, a_i) such that $\omega' \not\models q_i$ and $a_i > b_j$ follows from C.

Then, a conclusion q is said to be a plausible consequence of (Σ, C) if q is true in all preferred models (w.r.t. $>_\Sigma$).

An important result is the complete and soundness result, namely :

Proposition 1 *$Inc(\Sigma \cup \{(\neg p, 1)\}) > Inc(\Sigma)$ follows from C iff p is true in all preferred models w.r.t $>_\Sigma$.*

Example 3 *Let us consider the above example, where :*
$\Sigma = \{(p, max(a, b)), (\neg p \vee r, min(c, d)), (\neg r, e))\}$, *and*
$C = \{a \geq e, c \geq a, d \geq e\}$.

Given Σ and C we are interested to check if r follows from Σ and C. Let us compute $Inc(\Sigma)$. By definition, we have:

$$Inc(\Sigma) = min(max(a, b), min(c, d), e). \quad (1)$$

Let us simplify this expression, with the help of the constraints in C, we have:
$Inc(\Sigma) = min(max(a, b), min(c, d), e)$
$= max(e, min(b, e))$, *hence:*

$$Inc(\Sigma) = e. \quad (2)$$

We now need to compute $Inc(\Sigma \cup (\neg r, 1))$, we get:
$Inc(\Sigma \cup (\neg r, 1)) = max(Inc(\Sigma), min(max(a, b), min(c, d)))$
$= max(e, max(min(a, c, d), min(b, c, d)))$
$= max(e, min(a, d), min(b, c, d))$
$= max(min(a, d), min(b, c, d))$.

It can be checked from C that $Inc(\Sigma \cup \{(\neg r, 1)\}) > Inc(\Sigma)$, since for instance $min(a, d) > e$.

4 Propositional logic encoding of partially-constrained weighted formulas

In standard possibilistic logic (recalled in Section 2), the weights a_i's are assumed to be known. In this section, these weights are only partially known. Propositions are then associated with symbolic weights or variables. These symbolic weights are related by a set of constraints. The following subsections describe in detail the representation of uncertain beliefs and constraints, in a propositional logic setting.

4.1 Encoding constraints

This subsection presents the encoding of constraints on the set of symbolic weights using propositional logic. In the following we associate to each symbolic weight a of the knowledge base a propositional symbol denoted by the corresponding capital letter A. We denote by S the set of propositional symbols associated with H (with $S \cap V = \emptyset$). Let L_S be the propositional language built from S using the propositional connectors $\wedge, \vee, \neg$.

One possible way to check if $a \geq b$ follows from C is to use propositional logic, and encode C as a set of clauses.

Given C, its encoding in propositional logic is immediate. Namely, a constraint $a \geq b$ is translated into $\neg A \vee B$ in the agreement with the fact that a and b are lower bounds (of a necessity measure) and thus a refers to the set of numbers $[0, 1]$ and $[a, 1] \subseteq [b, 1]$ holds iff $a \geq b$. The translation of $a \geq b$ into $\neg A \vee B$ can be read as as "if the situation is at least very abnormal (A), it is at least abnormal (B)" (indeed, the greater a, the more certain p in (p, a), and the more exceptional a situation where p is false).

To refer to the maximum (max), we use the conjunction operator ($\wedge$), namely max(a,b) will be encoded by $A \wedge B$. Indeed, the tautology $\neg(A \wedge B) \vee A$ reflects $max(a,b) \geq a$. To refer to the minimum *(min)*, we use the disjunctive operator ($\vee$), namely *min(a,b)* will be encoded using $A \vee B$. A clause $\neg A_1 \vee ... \vee \neg A_m \vee B_1 \vee ... \vee B_n$ will hence encodes a constraint $max(a_1, ..., a_m) \geq min(b_1, ..., b_n)$. More formally,

Definition 2 *Let C be a set of constraints. The propositional logic base associated with C, denoted by K_C, is defined by :* $K_C = \{\neg A_1 \vee ... \vee \neg A_m \vee B_1 \vee ... \vee B_n : max(a_1, ..., a_m) \geq min(b_1, ..., b_n) \in C\}$.

Example 4 *A total order $C = \{a_1 \geq a_2, a_2 \geq a_3,$ $..., a_{n-1} \geq a_n\}$ is encoded by :* $K_C = \{\neg A_i \vee A_{i+1} : i = 1, .., n-1\}$.

The following proposition shows that inequalities induced from C can be obtained using our propositional encoding:

Proposition 2 *Let a, and b be two symbolic weights. Then : $a \geq b$ follows from C iff $K_C \models \neg A \vee B$.*

This proposition can be easily generalized for any inequality of the form $a \geq b$ where a and b are max/min expressions, using remarks of Section 3.2. Namely any derivation of inequalities between max/min expressions, can be redefined in terms of derivations between symbolic weights. For instance, it can be checked that $a > b$ follows from C iff $K_C \models \neg A \vee B$ holds but $K_C \models \neg B \vee A$ does not hold.

We call S-positive formulas the formulas built from S by only using the conjunction and disjunction operator. For instance, $A \vee B$ is an S-positive formula, while $\neg A \vee B$ is not an S-positive formula.

4.2 Encoding uncertain information

As suggested in the introduction, the idea is to manipulate symbolic weights as formulas. Thus, a possibilistic formula (p, a) is associated with the classical clause $p \vee A$ where A means something as "the situation is abnormal". Interestingly enough, this view agrees with the qualitative representation of uncertainty in terms of lower bounds of a necessity measure used in possibilistic logic. The following definition gives the propositional logic encoding of possibilistic knowledge base:

Definition 3 *Let $\Sigma = \{(p_i, a_i)\}$ be a possibilistic knowledge base. Let A_i be a S-positive formula associated with a_i (by replacing in a_i the minimum with the disjunction, and the maximum by the conjunction). Then the propositional base associated with Σ, denoted by K_Σ, is defined by :* $K_\Sigma = \{p_i \vee A_i : (p_i, a_i) \in \Sigma$ *and A_i is the S-positive formula associated with a_i}*

4.3 Characterizing plausible inferences

Until now, we have shown how to encode in propositional logic uncertain beliefs Σ and the constraints C on weights associated with these beliefs. This section defines the notion of plausible conclusions that can be drawn from (Σ, C). The set K_C is used at two stages: first it is used for simplifying the expression of the inconsistency degree, and then it is used to check, in the refutation, if the inconsistency degree of the augmented base increases.

Definition 4 *An S-prime formula of $K_\Sigma \cup K_C$ is an S-positive formula, denoted by $Linc(K_\Sigma)$, such that i) $K_\Sigma \cup K_C \models Linc(K_\Sigma)$, and ii) there is no ψ (not equivalent to $Linc(K_\Sigma)$) such that $K_\Sigma \cup K_C \models \psi$ and $\psi \models Linc(K_\Sigma)$.*

Up to logical equivalence, $Linc(K_\Sigma)$ is unique.

This definition allows to have a sound and complete inference relation with respect to the semantics given above. Indeed, the following proposition shows that $Linc(K_\Sigma)$ is the logical counterpart of $Inc(\Sigma)$.

Proposition 3 *Let K_Σ and K_C. Let $\delta(\Sigma)$ be a propositional formula obtained from $Inc(\Sigma)$ by replacing maximum by a conjunction, minimum by a disjunction, and the symbolic weights by their associated literals. Then $\delta(\Sigma)$ is logically equivalent to $Linc(K_\Sigma)$ given in Definition 4.*

5 Computing plausible inference using DNF formats

We propose a characterisation of plausible inference using the idea of forgetting variables (see for instance [Lang *et al.*, 2003; Darwiche and Marquis, 2004] for more details). Forgetting a variable p from K comes down to remove any reference of p in K.

Definition 5 *Let p be a propositional symbol of V. Then :* $ForgetVariable(K, p) = K_{p=\bot} \vee K_{p=\top}$

$K_{p=\bot}$ (resp. $K_{p=\top}$) is the knowledge base obtained from K by replacing p by false (resp. true). To forget a set of variables, we forget variable by variable, namely if A denotes a set of variables, then: $ForgetVariable(K, A) = ForgetVariable(ForgetVariable(K, p), A - \{p\})$.
It is also possible to only forget literals (atoms or negated atoms):

Definition 6 *Let l be a literal. Then :* $ForgetLiteral(K, l) = K_{l=\top} \vee (\neg l \wedge K)$

Some properties of ForgetVariable ([Darwiche and Marquis, 2004], [Lang *et al.*, 2003]), viewing a base as a conjunct of its formulas:

(1) $ForgetVariable(\phi \vee \psi, A) = Forgetvariable(\phi, A) \vee Fogetvariable(\psi, A)$.

(2) if ψ does not contain any varaible of A, then $ForgetVariable(\phi \wedge \psi, A) = \psi \wedge ForgetVariable(\phi, A)$

(3) if ϕ is a consistent conjunction of literals, then $ForgetVariable(\phi, A)$ consists in removing the variables in A from ϕ.

ForgetLiteral satisfies (1) and (2) which is enough for the purpose of the paper. The following shows that getting $Linc(K_\Sigma)$ is equivalent to first forget all formulas of the language, and then all negative literals of S

Proposition 4 *Let NegS be the set of negative literals in $K_\Sigma \cup K_C$. Let $F = ForgetVariable(K_\Sigma, V)$. Then $Linc(K_\Sigma)$ is equivalent to $ForgetLiteral(K_C \wedge F, NegS)$.*

This result is very important since it provides an efficient way to draw plausible conclusion from Σ and C. Indeed, forgetting a variable (resp. literal) can be achieved in a polynomial time if knowledge bases are in some formats like DNF or d-DNNF [Darwiche and Marquis, 2004]. The procedure for checking if a proposition p can be derived from Σ and C can be described as follows:

Step 1: transform Σ and C into K_Σ and K_C

Step 2: Put K_Σ into a DNF (or d-DNNF) form

Step 3.1: Forget Variables of V from K_Σ

Step 3.2: Forget Negated atoms of S from $K_C \cup K_\Sigma$. This gives $Linc(K_\Sigma)$.

Step 4: Put $K_\Sigma \cup \{\neg p\}$ into a DNF (or d-DNNF) form

Step 5.1: Forget Variables of V from $K_\Sigma \cup \{\neg p\}$

Step 5.2: Forget Negated atoms of S from $K_C \cup K_\Sigma \cup \{\neg p\}$. This gives $Linc(K_{\Sigma \cup \{\neg p\}})$.

Step 6: Use proposition 2 and results of Section 4.1 to check if p is a plausible consequence of Σ and C or not.

Forgetting a variable in a DNF amounts to forget it in each term, and forgetting it in a term amounts just to suppress the term. This clearly shows that this is polyomial in time. A similar procedure applies as well to d-DNNF format. This format known as Deterministic, Decomposable Negation Normal Form has been proposed recently [Darwiche, 2004] is a compact format, and has allowed the computation of generally intractable logical queries in time polynomial in the form size. An algorithm has been presented in [Darwiche, 2004] for compiling Conjunctive Normal Forms into d-DNNF directly. Our approach can clearly take advantage of this format as well.

Let us illustrate the above procedure with the following example:

Example 5 *Let us consider again Example 3, where we have:* $\Sigma = \{(p, max(a,b)), (\neg p \vee r, min(c,d)), (\neg r, e))\}$, *and* $C = \{a \geq e, c \geq a, d \geq e\}$. *Let* A, B, C, D, E *be the propositional symbols associated with symbolic weights* a, b, c, d, e.

Step 1 : Encoding Σ and C

The encoding of C in propositional logic gives :

$$K_C = \{\neg A \vee E, \neg C \vee A, \neg D \vee E\}$$

The encoding of Σ gives :

$K_\Sigma = \{p \vee (A \wedge B), \neg p \vee r \vee C \vee D, \neg r \vee E\}$.

Step 2 : Putting K_Σ into a DNF form

We first put K_Σ in a DNF form, which gives :

$(p \wedge C \wedge \neg r) \vee (p \wedge D \wedge \neg r) \vee (A \wedge B \wedge \neg p \wedge \neg r) \vee (A \wedge B \wedge C \wedge \neg r) \vee (A \wedge B \wedge D \wedge \neg r) \vee (p \wedge r \wedge E) \vee (p \wedge C \wedge E) \vee (p \wedge D \wedge E) \vee (A \wedge B \wedge \neg p \wedge E) \vee (A \wedge B \wedge r \wedge E) \vee (A \wedge B \wedge C \wedge E) \vee (A \wedge B \wedge D \wedge E)$

Step 3.1: Forgetting Variables of V

Now we forget the two variables p and r of V. By using properties (1)-(3); and after simplification, we get:

$ForgetVariable(K_\Sigma, V) \equiv C \vee D \vee E \vee (A \wedge B)$

Note that this is exactly the logical counterpart of $Inc(\Sigma)$ given by (1) in Example 3 (after remplacing, $\vee$ by minimum operation, $\wedge$ by maximum operation).

Step 3.2: Forgetting Negated atoms of S

Now, let us forget literals from $K_C \wedge ForgetVariable(K_\Sigma, V)$. *We get:*

$$Linc(K_\Sigma) = E.$$

Again this is exactly the logical counterpart of the expression (2) given in Example 3.

We are now interested to check if r is a plausible consequence of Σ and C.

Step 4 : Putting in DNF form $\Sigma \cup \{\neg r\}$

The DNF form associated with $\Sigma \cup \{\neg r\}$ is:

$(p \wedge C \wedge \neg r) \vee (p \wedge D \wedge \neg r) \vee (A \wedge B \wedge \neg p \wedge \neg r) \vee (A \wedge B \wedge C \wedge \neg r) \vee (A \wedge B \wedge D \wedge \neg r) \vee (p \wedge C \wedge E \wedge \neg r) \vee (p \wedge D \wedge E \wedge \neg r) \vee (A \wedge B \wedge \neg p \wedge E \wedge \neg r) \vee (A \wedge B \wedge C \wedge E \wedge \neg r) \vee (A \wedge B \wedge D \wedge E \wedge \neg r)$

Step 5.1. : Forgetting variables of V in $\Sigma \cup \{\neg r\}$

$ForgetVariable(\Sigma \cup \{\neg r\}, V) = (C \wedge \neg A) \vee (D \wedge \neg A) \vee (C \wedge E) \vee (D \wedge E) \vee (A \wedge B \wedge E)$

Step 5.2 : Forgetting Negative atoms

Forgetting negative literals of S gives :

$Linc(K_\Sigma \cup \{\neg r\}) = (A \wedge B \wedge E) \vee (C \wedge E \wedge A) \vee (D \wedge E)$

Step 6: Checking plausible inference

Using results of Section 4.1, it can be checked that r is a plausible consequence of Σ and C, as we have already seen in Example 3 that .

We finish this section by briefly discussing the case where weights associated with constraints are totally ordered. Let $\Sigma = \{(p_i, a_i) : i = 1, ..., n\}$. We assume without loss of generality that $a_1 \geq a_2 \geq ... \geq a_n$. The knowledge bases K_Σ and K_C are :

$K_\Sigma = \{p_i \vee A_i : (p_i, a_i) \in \Sigma\}$, and

$K_C = \{\neg A_i \vee A_{i+1} : i = 1, ..., n-1\}$.

Let D be the result of putting $K_\Sigma \cup K_C$ into a DNF form, and $Linc(K_\Sigma, K_C)$ be the result of forgetting variables of V and negative atoms from H. Then for totally ordered weights, it is possible to compute a compiled base (as in standard possibilistic logic), denoted by $Cons(K_\Sigma \cup K_C)$, as follows:

- If the inconsistency degree of Σ is a_j (namely $Inc(\Sigma) = a_j$), then $Cons(K_\Sigma \cup K_C)$ is logically equivalent to $ForgetVariable(\neg A_1 \wedge ... \wedge \neg A_{j-1} \wedge D, H)$
- If Σ is consistent, then $Linc(K_\Sigma, K_C)$ $Cons(K_\Sigma \cup K_C)$ is logically equivalent to $ForgetVariable(\neg A_1 \wedge ... \wedge \neg A_n \wedge D, H)$.

Note that $Cons(K_\Sigma \cup K_C)$ is under DNF form, and we have: p is a possibilistic consequence of Σ iff $Cons(K_\Sigma \cup K_C) \vdash p$.

Example 6 *Let* $\Sigma = \{(p, 0.8), (q, 0.5), (\neg p \vee \neg q, 0.4), (\neg p, 0.2)\}$. *Let A, B, C, E be propositional symbols associated respectively with the weights of the knowledge base (namely* $0.8, 0.5, 0.4, 0.2$*). The two propositional bases are:* $K_\Sigma = \{A \vee p, B \vee q, C \vee \neg p \vee \neg q, E \vee \neg p\}$, *and* $K_C = \{\neg A \vee B, \neg B \vee C, \neg C \vee E\}$.

The DNF associated with $K_\Sigma \cup K_C$ is:

$D = (p \wedge q \wedge C \wedge E \wedge \neg A) \vee (p \wedge B \wedge C \wedge E) \vee (A \wedge B \wedge C \wedge E)$

Let us forget variables of V, and negative atoms from H, we get :

$C \wedge E \vee B \wedge C \wedge E \vee A \wedge B \wedge C \wedge E$

which is equivalent to : $C \wedge E$ *Now let us add* $\neg A \wedge \neg B$ *to* D *we get:*

$D = p \wedge q \wedge C \wedge E \wedge \neg A \wedge \neg B$

Now forgetting variables of S gives : $Cons(K_\Sigma \cup K_C) = p \wedge q$*, from which it can be checked that* ψ *is a possibilistic consequence of* Σ *iff* $p \wedge q \vdash \psi$.

6 Related works

Using abnormality predicates in a logical setting is explicitly underlying several non-monotonic formalisms such as circumscription [McCarthy, 1980]. The idea is then to minimize abnormality and circumscribe it to a minimal number of individuals. The use of abnormality propositional literals is different here. Namely, no minimization process takes place (we are not dealing with exception-tolerant reasoning), and the symbolic weight attached to a formula p can itself be a compound formula reflecting the complex conditions under which it holds that p is true. What is proposed is more in the spirit of multiple source information, where the confidence of the information depends on the source, or on the topic, for instance.

Our representation framework for qualitative uncertainty does not require the knowledge of a complete pre-ordering of the different certainty levels attached to formulas. The uncertainty pervading a proposition may be either viewed as a precise notion, or as imprecisely stated by means of constraints that are opened to revision if new information becomes available. In both kinds of framework, one may only have a partial knowledge of the uncertainty. In the first case, uncertainty is handled under the form of absolute statements such as $g(p) = a$ (where g is an uncertainty measure and p is a proposition), or under the form of relative statements such as $p > q$ (where $>$ is a partial ordering encoding a plausibility relation expressing here that the plausibility of p is strictly greater than the one of q, as discussed in [Halpern, 1997]. Then, adding the piece of information that p entails q leads to inconsistency since if p entails q classically, it is expected that the plausibility of q is at least equal to the one of p. However, note that in possibilistic logic $N(p) \geq a$ and $N(q) \geq b$ together with the inequality $a \geq b$ is a weaker statement that $N(p) \geq N(q)$. Benferhat et al. [Benferhat *et al.*, 2004] propose a semantic approach for reasoning with partially ordered information in a possibilistic logic setting. The logical handling of formulas with unknown certainty weights together with the constraints relating these weights presented here turns to be much simpler and computationally more tractable.

7 Conclusion

The problem of reasoning with pieces of information having different confidence levels is raised by the handling of multiple source information. In case of partial information on the relative values of these levels, the problem becomes more difficult. An elegant method is proposed here for solving it, by rewriting the uncertain pieces of information in a two-sorted logic, and encoding the available information on the relative values of the certainty levels in a logical way. Putting the two bases in DNF format and using forgetting variables techniques enable us to compute the symbolic counterpart of the inconsistency level of a knowledge base in a linear way, and then to draw plausible inferences. Moreover, the representation technique that is used also provides a way for compiling a standard possibilistic knowledge base, a result analogous to the one obtained by Darwiche and Marquis [Darwiche and Marquis, 2004] for another type of weighted logic, the penalty logic, using d-DNNF format. This paper also briefly discusses the case of totally ordered weights, from which the number of extra variables (used to encode weights) and the number of extra binary clauses (used to encode the total ordering) necessary for the inference process is equal to the number of different weights used in base. The full development of a compilation of standard possibilistic knowledge bases in propositional logic is left for further research. Besides, another potential expected benefit of the approach, is a contribution to the solution of the drowning problem in possibilistic logic. Indeed, it can be checked that the symbolic inconsistency level of $K = \{p \vee A, \neg p \vee B, q \vee C\}$, is $Inc(K) = A \vee B$, where C does not appear, which should provide a way for finding out the "free" formulas in K that are not involved in any inconsistency conflict.

Acknowledgments: The authors would like to thank Pierre Marquis for his useful discussions.

References

[Benferhat *et al.*, 1993] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Procs. of the Thirteenth Intern. Joint Conf. on Art. Int. (IJCAI'93)*, pages 640–645, 1993.

[Benferhat *et al.*, 2004] S. Benferhat, S. Lagrue, and O. Papini. Reasoning with partially ordered information in a possibilistic framework. *Fuzzy Sets and Systems, Vol. 144, pp. 25-41*, 2004.

[Darwiche and Marquis, 2004] A. Darwiche and P. Marquis. Compiling propositional weighted bases. *Artif. Intell.*, 157(1-2):81–113, 2004.

[Darwiche, 2004] A. Darwiche. New advances in compiling cnf into decomposable negation normal form. In *Procs. of Europ. Conf. on Art. Intell. (ECAI2000)*, pages 328–332, 2004.

[Dubois *et al.*, 1994] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University Press, 1994.

[Halpern, 1997] J. Y. Halpern. Defining relative likelihood in partially-ordered preferential structures. *J. of AI Research*, 7:1–24, 1997.

[Lang *et al.*, 2003] J. Lang, P. Liberatore, and P. Marquis. Propositional independence - formula-variable independence and forgetting. *J. of Art. Intell. Research*, 18:391–443, 2003.

[McCarthy, 1980] J. McCarthy. Circumscription : A form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

Bounded Policy Iteration for Decentralized POMDPs

Daniel S. Bernstein
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
bern@cs.umass.edu

Eric A. Hansen
Dept. of CS and Engineering
Mississippi State University
Mississippi State, MS 39762
hansen@cse.msstate.edu

Shlomo Zilberstein
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
shlomo@cs.umass.edu

Abstract

We present a bounded policy iteration algorithm for infinite-horizon decentralized POMDPs. Policies are represented as joint stochastic finite-state controllers, which consist of a local controller for each agent. We also let a joint controller include a correlation device that allows the agents to correlate their behavior without exchanging information during execution, and show that this leads to improved performance. The algorithm uses a fixed amount of memory, and each iteration is guaranteed to produce a controller with value at least as high as the previous one for all possible initial state distributions. For the case of a single agent, the algorithm reduces to Poupart and Boutilier's bounded policy iteration for POMDPs.

1 Introduction

The Markov decision process (MDP) framework has proven to be useful for solving problems of sequential decision making under uncertainty. For some problems, an agent must base its decision on partial information about the system state. In this case, it is often better to use the more general partially observable Markov decision process (POMDP) framework. Though POMDPs are difficult to solve in the worst case, much progress has been made in the development of practical dynamic programming algorithms [Smallwood and Sondik, 1973; Cassandra *et al.*, 1997; Hansen, 1998; Poupart and Boutilier, 2003; Feng and Zilberstein, 2004].

Even more general are problems in which a team of decision makers, each with its own local observations, must act together. Domains in which these types of problems arise include networking, multi-robot coordination, e-commerce, and space exploration systems. To model such problems, we can use the decentralized partially observable Markov decision process (DEC-POMDP) framework. Though this model has been recognized for decades (see, e.g., [Witsenhausen, 1971]), there has been little work on efficient algorithms for it.

Recently, an exact dynamic programming algorithm was proposed for DEC-POMDPs [Hansen *et al.*, 2004]. Though the algorithm was presented in the context of finite-horizon problems, there are various ways to extend it to the infinite-horizon case. However, in both cases, it suffers from the fact that the memory requirements grow quickly with each iteration, and in practice it has only been used to solve very small problems. It is likely that any optimal algorithm would suffer this problem, as finite-horizon DEC-POMDPs have been shown to be NEXP-complete, even for just two agents [Bernstein *et al.*, 2002].

In this paper, we present a memory-bounded dynamic programming algorithm for infinite-horizon DEC-POMDPs. The algorithm uses a stochastic finite-state controller to represent the joint policy for the agents. A straightforward approach is to use a set of independent local controllers, one for each agent. We provide an example to illustrate that higher value can be obtained through the use of shared randomness. As such, we define a joint controller to be a set of local controllers along with a *correlation device*. The correlation device is a finite-state machine that sends a signal to all of the agents on each time step. Its behavior can be determined prior to execution time, and thus it does not require that the agents exchange information after receiving local observations.

Our algorithm generalizes *bounded policy iteration* for POMDPs [Poupart and Boutilier, 2003] to the multi-agent case. On each iteration, a node is chosen from one of the local controllers or the correlation device, and its parameters are updated through the solution of a linear program. The generalization has the same theoretical guarantees as in the POMDP case. Namely, an iteration is guaranteed to produce a new controller with value at least as high for every possible initial state distribution.

In our experiments, we applied our algorithm to idealized networking and robot navigation problems. Both problems are too large for exact dynamic programming, but could be handled by our approximation algorithm. We found that the addition of a correlation device gives rise to better solutions. In addition, larger controllers most often lead to better solutions.

A number of approximation algorithms have been developed previously for DEC-POMDPs [Peshkin *et al.*, 2000; Nair *et al.*, 2003; Emery-Montemerlo *et al.*, 2004]. However, the previous algorithms do not guarantee both bounded memory usage and monotonic value improvement for all initial state distributions. Furthermore, the use of correlated stochastic policies in the DEC-POMDP context is novel. The

importance of correlation has been recognized in the game theory community [Aumann, 1974], but there has been little work on algorithms for finding correlated policies.

2 Background

In this section, we present our formal framework for multi-agent decision making. A *decentralized partially-observable Markov decision process (DEC-POMDP)* is a tuple $\langle I, S, \{A_i\}, \{O_i\}, P, R\rangle$, where

- I is a finite set of agents indexed $1, \ldots, n$
- S is a finite set of states
- A_i is a finite set of actions available to agent i and $\vec{A} = \times_{i \in I} A_i$ is the set of joint actions, where $\vec{a} = \langle a_1, \ldots, a_n\rangle$ denotes a joint action
- O_i is a finite set of observations for agent i and $\vec{O} = \times_{i \in I} O_i$ is the set of joint observations, where $\vec{o} = \langle o_1, \ldots, o_n\rangle$ denotes a joint observation
- P is a set of Markovian state transition and observation probabilities, where $P(s', \vec{o}|s, \vec{a})$ denotes the probability that taking joint action $\vec{a}$ in state s results in a transition to state s' and joint observation $\vec{o}$
- $R : S \times \vec{A} \rightarrow \Re$ is a reward function

In this paper, we consider the case in which the process unfolds over an infinite sequence of stages. At each stage, all agents simultaneously select an action, and each receives the global reward and a local observation. The objective of the agents is to maximize the expected discounted sum of rewards received. We denote the discount factor γ and require that $0 \leq \gamma < 1$.

3 Finite-State Controllers

Our algorithm uses stochastic finite-state controllers to represent policies. In this section, we first define a type of controller in which the agents act independently. We then provide an example demonstrating the utility of correlation, and show how to extend the definition of a joint controller to allow for correlation among agents.

3.1 Local Finite-State Controllers

In a DEC-POMDP, each agent must select an action based on its history of local observations. Finite-state controllers provide a way to represent local policies using a finite amount of memory. The state of the controller is based on the observation sequence, and the agent's actions are based on the state of its controller. We allow for stochastic transitions and stochastic action selection, as this can help to make up for limited memory. This type of controller has been used previously in the single-agent context [Platzman, 1980; Meuleau *et al.*, 1999; Poupart and Boutilier, 2003].

Formally, we define a *local finite-state controller* for agent i to be a tuple $\langle Q_i, \psi_i, \eta_i\rangle$, where Q_i is a finite set of controller nodes, $\psi_i : Q_i \rightarrow \Delta A_i$ is an action selection function, and $\eta_i : Q_i \times A_i \times O_i \rightarrow \Delta Q_i$ is a transition function. The functions ψ_i and η_i parameterize the conditional distribution $P(a_i, q_i'|q_i, o_i)$.

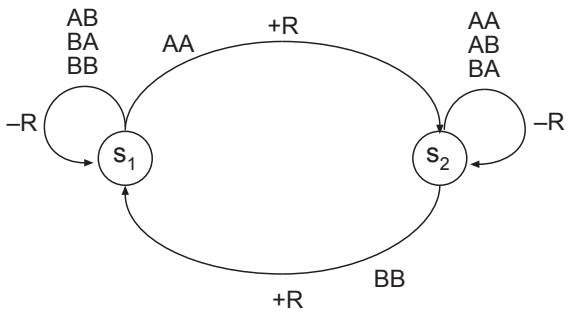


Figure 1: This figure shows a DEC-POMDP for which the optimal memoryless joint policy requires correlation.

Taken together, the agents' controllers determine the conditional distribution $P(\vec{a}, \vec{q}'|\vec{q}, \vec{o})$. This is denoted an *independent joint controller*. In the following subsection, we show that independence can be limiting.

3.2 The Utility of Correlation

The joint controllers described above do not allow the agents to correlate their behavior via a shared source of randomness. We will use a simple example to illustrate the utility of correlation in partially observable domains where agents have limited memory. This example generalizes the one given in [Singh *et al.*, 1994] to illustrate the utility of stochastic policies in single-agent partially observable settings.

Consider the DEC-POMDP shown in Figure 1. This problem has two states, two agents, and two actions per agent (A and B). The agents each have only one observation, and thus cannot distinguish between the two states. For this example, we will consider only memoryless policies.

Suppose that the agents can independently randomize their behavior using distributions $P(a_1)$ and $P(a_2)$, and consider the policy in which each agent chooses either A or B according to a uniform distribution. This yields an expected reward of $-\frac{R}{2}$ per time step, which results in an expected long-term reward of $\frac{-R}{2(1-\gamma)}$. It is straightforward to show that no independent policy yields higher reward than this one for all states.

Next, let us consider the larger class of policies in which the agents may act in a correlated fashion. In other words, we consider all joint distributions $P(a_1, a_2)$. Consider the policy that assigns probability $\frac{1}{2}$ to the pair AA and probability $\frac{1}{2}$ to the pair BB. This yields an average reward of 0 at each time step and thus an expected long-term reward of 0. The difference between the rewards obtained by the independent and correlated policies can be made arbitrarily large by increasing R.

3.3 Correlated Joint Controllers

In the previous subsection, we established that correlation can be useful in the face of limited memory. In this subsection, we extend our definition of a joint controller to allow for correlation among the agents. To do this, we introduce an additional finite-state machine, called a correlation device, that provides

Variables: $\epsilon, x(c, a_i), x(c, a_i, o_i, q_i')$
Objective: Maximize ϵ
Improvement constraints:

$$\forall s, q_{-i}, c \quad V(s, \vec{q}, c) + \epsilon \leq \sum_{\vec{a}} P(a_{-i}|c, q_{-i})[x(c, a_i)R(s, \vec{a}) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} x(c, a_i, o_i, q_i')P(q_{-i}'|c, q_{-i}, a_{-i}, o_{-i})P(s', \vec{o}|s, \vec{a})P(c'|c)V(s', \vec{q}', c')]$$

Probability constraints:

$$\forall c \quad \sum_{a_i} x(c, a_i) = 1, \quad \forall c, a_i, o_i \quad \sum_{q_i'} x(c, a_i, o_i, q_i') = x(c, a_i)$$

$$\forall c, a_i \quad x(c, a_i) \geq 0, \quad \forall c, a_i, o_i, q_i' \quad x(c, a_i, o_i, q_i') \geq 0$$

Table 1: The linear program used to find new parameters for agent i's node q_i. The variable $x(c, a_i)$ represents $P(a_i|q_i, c)$, and the variable $x(c, a_i, o_i, q_i')$ represents $P(a_i, q_i'|c, q_i, o_i)$.

extra signals to the agents at each time step. The device operates independently of the DEC-POMDP process, and thus does not provide the agents with information about the other agents' observations. In fact, the random numbers necessary for its operation could be determined prior to execution time.

Formally, a *correlation device* is a tuple $\langle C, \psi \rangle$, where C is a set of states and $\psi : C \to \Delta C$ is a state transition function. At each step, the device undergoes a transition, and each agent observes its state.

We must modify the definition of a local controller to take the state of the correlation device as input. Now, a local controller for agent i is a conditional distribution of the form $P(a_i, q_i'|c, q_i, o_i)$. The correlation device together with the local controllers form a joint conditional distribution $P(c', \vec{a}, \vec{q}'|c, \vec{q}, \vec{o})$. We will refer to this as a *correlated joint controller*. Note that a correlated joint controller with $|C| = 1$ is effectively an independent joint controller.

The value function for a correlated joint controller can be computed by solving the following system of linear equations, one for each $s \in S$, $\vec{q} \in \vec{Q}$, and $c \in C$:

$$V(s, \vec{q}, c) = \sum_{\vec{a}} P(\vec{a}|c, \vec{q})[R(s, \vec{a}) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} P(s', \vec{o}|s, \vec{a})P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) \cdot P(c'|c)V(s', \vec{q}', c')].$$

We sometimes refer to the value of the controller for an initial state distribution. For a distribution δ, this is defined as

$$V(\delta) = \max_{\vec{q}, c} \sum_{s} \delta(s)V(s, \vec{q}, c).$$

It is assumed that, given an initial state distribution, the controller is started in the joint node which maximizes value from that distribution.

4 Bounded Policy Iteration

We now describe our bounded policy iteration algorithm for improving correlated joint controllers. To improve a correlated joint controller, we can either change the correlation device or one of the local controllers. Both improvements can be done via a *bounded backup*, which involves solving a linear program. Following an improvement, the controller can be reevaluated through the solution of a set of linear equations. Below, we describe how a bounded backup works, and prove that it always produces a new controller with value at least as high for all initial state distributions.

4.1 Improving a Local Controller

We first describe how to improve a local controller. To do this, we choose an agent i, along with a node q_i. Then, we search for new parameters for the conditional distribution $P(a_i, q_i'|c, q_i, o_i)$.

The search for new parameters works as follows. We assume that the original controller will be used from the second step on, and try to replace the parameters for q_i with better ones for just the first step. In other words, we look for the best parameters satisfying the following inequality:

$$V(s, \vec{q}, c) \leq \sum_{\vec{a}} P(\vec{a}|c, \vec{q})[R(s, a) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o})P(s', \vec{o}|s, \vec{a}) \cdot P(c'|c)V(s', \vec{q}', c)]$$

for all $s \in S$, $q_{-i} \in Q_{-i}$, and $c \in C$. Note that the inequality is always satisfied by the original parameters. However, it is often possible to get an improvement.

Finding new parameters can be done using linear programming, as shown in Table 1. We note that this linear program is the same as that of Poupart and Boutilier [2003] for POMDPs, with the nodes of the other local controllers and correlation device considered part of the hidden state. Its size is polynomial in the sizes of the DEC-POMDP and the joint controller, but exponential in the number of agents.

4.2 Improving the Correlation Device

The procedure for improving the correlation device is very similar to the procedure for improving a local controller. We

Variables: $\epsilon, x(c')$
Objective: Maximize ϵ
Improvement constraints:

$$\forall s, \vec{q} \quad V(s, \vec{q}, c) + \epsilon \quad \leq \quad \sum_{\vec{a}} P(\vec{a}|c, \vec{q})[R(s, \vec{a}) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) P(s', \vec{o}|s, \vec{a}) x(c') V(s', \vec{q}', c')]$$

Probability constraints:

$$\forall c' \quad \sum_{c'} x(c') = 1, \quad \forall c' \quad x(c') \geq 0$$

Table 2: The linear program used to find new parameters for the correlation device node c. The variable $x(c')$ represents $P(c'|c)$.

first choose a device node c, and consider changing its parameters for just the first step. We look for the best parameters satisfying the following inequality:

$$V(s, \vec{q}, c) \quad \leq \quad \sum_{\vec{a}} P(\vec{a}|c, \vec{q})[R(s, a) + \gamma \sum_{s', \vec{o}, \vec{q}', c} P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) P(s', \vec{o}|s, \vec{a}) \cdot P(c'|c) V(s', \vec{q}', c')]$$

for all $s \in S$ and $\vec{q} \in \vec{Q}$.

As in the previous case, the search for parameters can be formulated as a linear program. This is shown in Table 2. This linear program is also polynomial in the sizes of the DEC-POMDP and joint controller, but exponential in the number of agents.

4.3 Monotonic Improvement

We have the following theorem, which says that performing either of the two updates cannot lead to a decrease in value for any initial state distribution.

Theorem 1 *Performing a bounded backup on a local controller or the correlation device produces a correlated joint controller with value at least as high for every initial state distribution.*

Proof. Consider the case in which some node q_i of agent i's local controller is changed. Let V_o be the value function for the original controller, and let V_n be the value function for the new controller. Recall that the new parameters for $P(a_i, q_i'|c, q_i, o_i)$ must satisfy the following inequality for all $s \in S$, $q_{-i} \in Q_{-i}$, and $c \in C$:

$$V_o(s, \vec{q}, c) \quad \leq \quad \sum_{\vec{a}} P(\vec{a}|c, \vec{q})[R(s, a) + \gamma \sum_{s', \vec{o}, \vec{q}', c'} P(\vec{q}'|c, \vec{q}, \vec{a}, \vec{o}) P(s', \vec{o}|s, \vec{a}) \cdot P(c'|c) V_o(s', \vec{q}', c)]$$

Notice that the formula on the right is the Bellman operator for the new controller, applied to the old value function. Denoting this operator T_n, the system of inequalities implies that $T_n V_o \geq V_o$. By monotonicity, we have that for all $k \geq 0$, $T_n^{k+1}(V_o) \geq T_n^k(V_o)$. Since $V_n = \lim_{k \to \infty} T_n^k(V_o)$, we have that $V_n \geq V_o$. Thus, the value of the new controller is higher than that of the original controller for all possible initial state distributions.

The argument for changing nodes of the correlation device is almost identical to the one given above. □

4.4 Local Optima

Although bounded backups give nondecreasing values for all initial state distributions, convergence to optimality is not guaranteed. There are a couple of factors contributing to this. First is the fact that only one local controller, or the correlation device, is improved at once. Thus, it is possible for the algorithm to get stuck in a suboptimal Nash equilibrium in which each of the controllers and the correlation device is optimal with the others held fixed. It is an open problem whether there is a linear program for updating more than one controller at a time.

Of course, a bounded backup does not find the *optimal* parameters for one controller with the others held fixed. Thus, a sequence of such updates may converge to a local optimum without even reaching a Nash equilibrium. For POMDPs, Poupart and Boutilier [2003] provide a characterization of these local optima, and a heuristic for escaping from them. This could be applied in our case, but it would not address the suboptimal Nash equilibrium problem.

5 Experiments

We implemented bounded policy iteration and tested it on two different problems, an idealized networking scenario and a problem of navigating on a grid. Below, we describe our experimental methodology, the specifics of the problems, and our results.

5.1 Experimental Setup

Although our algorithm guarantees nondecreasing value for all initial state distributions, we chose a specific distribution to focus on for each problem. Experiments with different distributions yielded qualitatively similar results.

We define a *trial run* of the algorithm as follows. At the start of a trial run, a size is chosen for each of the local controllers and the correlation device. The action selection and transition functions are initialized to be deterministic, with

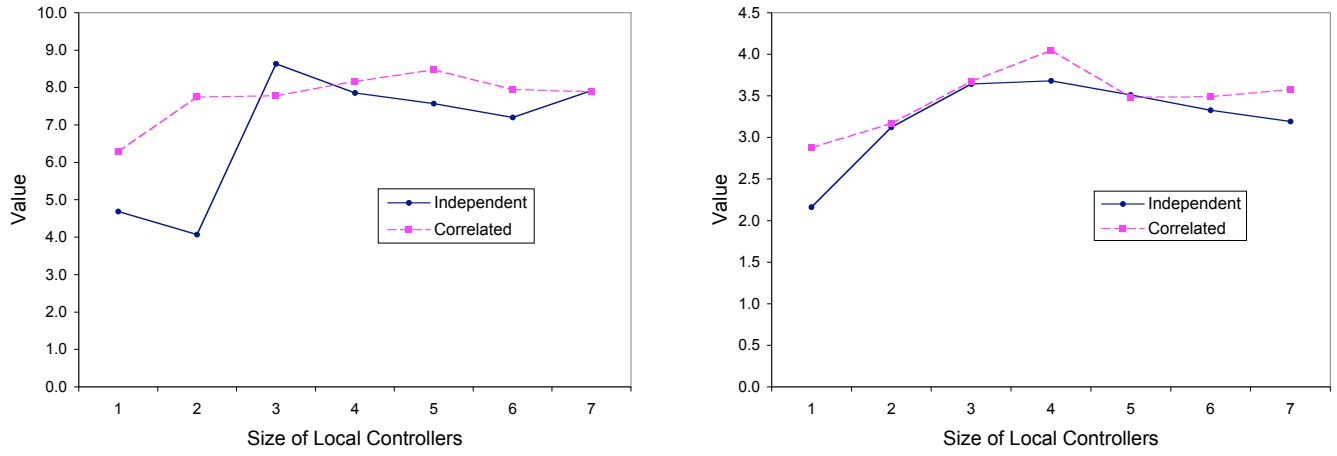


Figure 2: Average value per trial run plotted against the size of the local controllers, for (a) the multi-access broadcast channel problem, and (b) the robot navigation problem. The solid line represents independent controllers (a correlation device with one node), and the dotted line represents a joint controller including a two-node correlation device.

the outcomes drawn according to a uniform distribution. A *step* consists of choosing a node uniformly at random from the correlation device or one of the local controllers, and performing a bounded backup on that node. After 50 steps, the run is considered over. In practice, we found that values usually stabilized within 15 steps.

We varied the sizes of the local controllers from 1 to 7 (the agents' controllers were always the same sizes as each other), and we varied the size of the correlation device from 1 to 2. Thus, the number of joint nodes ranged from 1 to 98. Memory limitations prevented us from using larger controllers. For each combination of sizes, we performed 20 trial runs. We recorded the highest value obtained across all runs, as well as the average value over all runs.

5.2 Multi-Access Broadcast Channel

Our first domain is an idealized model of control of a multi-access broadcast channel [Ooi and Wornell, 1996]. In this problem, nodes need to broadcast messages to each other over a channel. Only one node may broadcast at a time, otherwise a collision occurs. The nodes share the common goal of maximizing the throughput of the channel.

At the start of each time step, each node decides whether or not to send a message. The nodes receive a reward of 1 when a message is successfully broadcast and a reward of 0 otherwise. At the end of the time step, each node observes its own buffer, and whether the previous step contained a collision, a successful broadcast, or nothing attempted.

The message buffer for each agent has space for only one message. If a node is unable to broadcast a message, the message remains in the buffer for the next time step. If a node i is able to send its message, the probability that its buffer will fill up on the next step is p_i. Our problem has two nodes, with $p_1 = 0.9$ and $p_2 = 0.1$. There are 4 states, 2 actions per agent, and 5 observations per agent. The discount factor is 0.9. The start state distribution is deterministic, with the buffer for agent 1 containing a message and the buffer for agent 2 being empty.

5.3 Meeting on a Grid

In this problem, we have two robots navigating on a two-by-two grid with no obstacles. Each robot can only sense whether there are walls to its left or right, and the goal is for the robots to spend as much time as possible on the same square. The actions are to move up, down, left, or right, or to stay on the same square. When a robot attempts to move to an open square, it only goes in the intended direction with probability 0.6, otherwise it either goes in another direction or stays in the same square. Any move into a wall results in staying in the same square. The robots do not interfere with each other and cannot sense each other.

This problem has 16 states, since each robot can be in any of 4 squares at any time. Each robot has 4 observations, since it has a bit for sensing a wall to its left or right. The total number of actions for each agent is 5. The reward is 1 when the agents share a square, and 0 otherwise, and the discount factor is 0.9. The initial state distribution is deterministic, placing both robots in the upper left corner of the grid.

5.4 Results

For each combination of controller sizes, we looked at the best solutions found across all trial runs. The values for these solutions were the same for all controller sizes except for the few smallest.

It was more instructive to compare average values over all trial runs. Figure 2 shows graphs of average values plotted against controller size. We found that, for the most part, the average value increases when we increase the size of the correlation device from one node to two nodes (essentially moving from independent to correlated).

For small controllers, the average value tends to increase with controller size. However, as the controllers get larger,

there is no clear trend. This behavior is somewhat intuitive, given the way the algorithm works. For new node parameters to be acceptable, they must not decrease the value for any combination of states, nodes for the other controllers, and nodes for the correlation device. This becomes more difficult as controllers get larger, and thus it is easier to get stuck in a local optimum.

Improving multiple controllers at once would help to alleviate the aforementioned problem. As mentioned earlier, we do not currently have a way to do this using linear programming, and it thus remains an interesting topic for future work.

6 Conclusion and Future Work

We have presented a bounded policy iteration algorithm for DEC-POMDPs. Besides the fact that it uses finite memory, the algorithm has a number of other appealing theoretical guarantees. First, by using correlated joint controllers, we can achieve higher value than with independent joint controllers of the same size. Second, assuming a constant number of agents, each iteration of the algorithm completes in polynomial time. Finally, monotonic value improvement is guaranteed for all states on each iteration.

Our empirical results are encouraging. By bounding the size of the controller, we are able to achieve a tradeoff between computational complexity and the quality of the approximation. Up to a point, increasing the sizes of the local controllers leads to higher values on average. After this point, average values tend to level off or decrease. Increasing the size of the correlation device leads to higher value, which is consistent with our theoretical results.

For future work, there are many more experiments that can be done with bounded policy iteration. For instance, in moving to a larger controller, we could use the previous controller as a starting point, rather than starting over with a random controller. Poupart and Boutilier's [2003] escape technique could be useful here. Also, rather than choosing nodes uniformly at random for updating, we could develop a principled way to order the nodes.

We are also looking into ways of extending the algorithm to handle problems with large numbers of agents. In many problems, each agent interacts with only a small subset of the other agents. This additional structure can be exploited to reduce the size of the problem representation, and it should be possible to extend our algorithm to take advantage of these local interactions.

Finally, it would be interesting to extend bounded policy iteration to the noncooperative setting, where each agent has a separate reward function. One approach is to require that a change in parameters does not lead to a decrease in value for *any* agent. Another approach is to consider just the value function for the agent whose node is being updated. This should move the joint controller towards a Nash equilibrium.

7 Acknowledgments

We thank Martin Allen and Özgür Şimşek for helpful discussions of this work. This work was supported in part by the National Science Foundation under grants IIS-0219606 and IIS-9984952, by NASA under cooperative agreement NCC 2-1311, and by the Air Force Office of Scientific Research under grant F49620-03-1-0090.

References

[Aumann, 1974] Robert J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1:67–96, 1974.

[Bernstein *et al.*, 2002] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[Cassandra *et al.*, 1997] Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of UAI*, pages 54–61, 1997.

[Emery-Montemerlo *et al.*, 2004] Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schnieder, and Sebastian Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of AAMAS*, 2004.

[Feng and Zilberstein, 2004] Zhengzhu Feng and Shlomo Zilberstein. Region-Based incremental pruning for POMDPs. In *Proceedings of UAI*, pages 146–153, 2004.

[Hansen *et al.*, 2004] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of AAAI*, pages 709–715, 2004.

[Hansen, 1998] Eric Hansen. Solving POMDPs by searching in policy space. In *Proceedings of UAI*, pages 211–219, 1998.

[Meuleau *et al.*, 1999] Nicolas Meuleau, Kee-Eung Kim, Leslie Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of UAI*, pages 417–426, 1999.

[Nair *et al.*, 2003] Ranjit Nair, David Pynadath, Makoto Yokoo, Milind Tambe, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of IJCAI*, 2003.

[Ooi and Wornell, 1996] James M. Ooi and Gregory W. Wornell. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proceedings of the 35th Conference on Decision and Control*, pages 293–298, 1996.

[Peshkin *et al.*, 2000] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. In *Proceedings of UAI*, pages 489–496, 2000.

[Platzman, 1980] Loren K. Platzman. A feasible computational approach to infinite-horizon partially-observed Markov decision processes. Technical report, Georgia Institute of Technology, 1980. Reprinted in *Working Notes of the 1998 AAAI Fall Symposium on Planning Using Partially Observable Markov Decision Processes*.

[Poupart and Boutilier, 2003] Pascal Poupart and Craig Boutilier. Bounded finite state controllers. In *Proceedings of NIPS*, 2003.

[Singh *et al.*, 1994] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Proceedings of ICML*, 1994.

[Smallwood and Sondik, 1973] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[Witsenhausen, 1971] Hans S. Witsenhausen. Separation of estimation and control for discrete time systems. *Proceedings of the IEEE*, 59(11):1557–1566, 1971.

A Decision-Theoretic Approach to Task Assistance for Persons with Dementia

Jennifer Boger
IBBME, Univ. Toronto
jen.boger@utoronto.ca

Pascal Poupart
School of Computer Science, Univ. Waterloo
ppoupart@cs.uwaterloo.ca

Jesse Hoey
Dept. Computer Science, Univ. Toronto
jhoey@cs.toronto.edu

Craig Boutilier
Dept. of Computer Science, Univ. Toronto
cebly@cs.toronto.edu

Geoff Fernie
Toronto Rehabilitation Institute
Fernie.Geoff@torontorehab.on.ca

Alex Mihailidis
Dept. of Occup. Therapy, Univ. Toronto
alex.mihailidis@utoronto.ca

Abstract

Cognitive assistive technologies that aid people with dementia (such as Alzheimer's disease) hold the promise to provide such people with an increased level of independence. However, to realize this promise, such systems must account for the specific needs and preferences of individuals. We argue that this form of customization requires a sequential, decision-theoretic model of interaction. We describe both fully and partially observable Markov decision process (POMDP) models of a handwashing task, and show that, despite the potential computational complexity, these can be effectively solved and produce policies that are evaluated as useful by professional caregivers.

1 Introduction

Dementia is a clinical syndrome characterized by the deterioration of a person's memory and cognitive function. It is estimated that there are nearly 18 million older adults with dementia worldwide, with this number expected to reach 35 million by 2050 [1]. Alzheimer's disease is the most common form of dementia and accounts for more than half of dementia diagnoses [3]. Home care offers tremendous advantages for such individuals (e.g., economically, quality of life) over placement in a long-term healthcare facility. However, significant barriers often prevent home care from being a viable option, including the physical and emotional burden placed on family members and the cost of maintaining professional care in the home.

A pressing need for people with advanced dementia arises from the difficulty they have completing *activities of daily living (ADLs)*. A caregiver generally must guide them, step by step, through such routine activities as handwashing, toileting, dressing, moving about, taking medication, etc. Cognitive assistive technologies that help guide *users* (i.e., care recipients) through ADLs can relieve some of the stress of home care on family members and care professionals. However, current monitoring and reminding devices cannot accurately track user behavior at a step-by-step level, nor adapt to specific users. Moreover, many devices require explicit feedback (e.g., button presses), which cannot reasonably be expected of people with moderate-to-severe dementia.

Our objective is to design and develop systems that actively monitor a user attempting a task and offer assistance in the form of task guidance (e.g., prompts or reminders) when it is most appropriate and in a form that will do the most good. Such systems should tailor their guidance decisions (w.r.t. both form and timing) to specific individuals and circumstances, based on what will work best for them. In this work, we extend the prototype COACH system [12] in which behavioral monitoring of a *handwashing task* is realized with a vision system. We extend COACH by modeling the guidance decision process in a decision-theoretic fashion, more precisely, as a *partially observable Markov decision process (POMDP)*. There are several important factors that should influence guidance decisions that strongly suggest the use of this model, including perceptual noise, the stochastic nature of user behavior, the need to trade off various objective criteria (e.g., task completion, caregiver burden, user frustration and independence), and the need to tailor guidance to specific individuals and circumstances.

Our key contributions are as follows. First, we develop a POMDP model for a specific ADL (handwashing). While the model itself is of value for handwashing, the general *form* of the model is appropriate for almost any ADL guidance or intervention module. Second, we show that these models exhibit considerable structure, allowing them to be specified easily, and the underlying decision problem to be solved approximately by recent state-of-the-art algorithms. In particular, we obtain an exact solution for a fully observable MDP simplification of the POMDP; and an approximate solution is obtained for the POMDP despite its size and complexity. Third, controlled evaluation of the MDP policy by professional caregivers demonstrates its significant value as an assistive device. While MDP performance is not as good as that of professional caregivers, it is more than adequate to serve as the basis for a supplemental device which can relieve the burden on caregivers or family members. Finally, in simulation, we show that the approximate POMDP policy outperforms the exact MDP policy. This suggests that accounting for partial observability and user customization via the POMDP approach offers considerable benefits. These benefits are currently being quantified in clinical trials which will be reported in a longer version of this paper.[1]

[1] Trials are not yet complete so cannot be reported here.

2 Cognitive Assistive Technologies & COACH

Assistive technology (AT) is increasingly used to offset the impact of impairments resulting from injury, disease, and the aging process and related disorders. Typically these technologies have focused on assisting users with mobility impairments. Recent growth in the number of people with cognitive disabilities, such as dementia, has resulted in considerable research being conducted into developing *cognitive assistive technologies (CATs)* to address the difficulties that this population faces. In broad terms, CATs attempt to compensate for existing impairments by using devices, tools, or techniques that either compensate for a person's impaired cognitive ability, or translate a problem into one that matches the user's strengths. To date, computerized CATs have been of limited scope, especially in the use of AI techniques. Prototypes of cognitive aids include handheld devices that remind a person to complete basic ADLs (e.g., taking medication [8]). However, most of these systems were not designed for older adults, particularly those with moderate-to-severe dementia, and would likely not be acceptable for this population.

Several intelligent systems that use AI and ubiquitous computing techniques are currently being developed for the older adult. These include the Aware Home Project [13], the Assisted Cognition Project [7] and the Nursebot Project [15]. These new projects are similar to the work described in this paper in the sense that they attempt to incorporate AI and a decision-theoretic approach to overcome the shortcomings of current approaches. In particular, the Autominder System [17], one aspect of the Nursebot Project, applies a POMDP in the development of the planning and scheduling aspect of the system [15]. These systems do not incorporate advanced prompting techniques and algorithms, but rather are being developed as scheduling and memory aids.

COACH, our first prototype of an intelligent environment for older adults with dementia, monitors progress and provides assistance during handwashing. It uses computer vision and simple AI techniques to learn to associate hand positions (2D coordinates) with specific handwashing steps (e.g., turning on the water, using the soap), and to adjust its parameters and cuing strategies [12]. COACH uses audio prompts for each of the possible plan steps, with each step associated with a *general*, a *moderate*, or a *specific* prompt. For example, a general prompt to turn on the water simply states "Please turn on the water," while a specific prompt uses the subject's name and elaborates on the the process ("... by placing your hands on the tap in front of you ..."). Clinical trials with 10 subjects with moderate-to-severe dementia showed that the number of handwashing steps that the subjects were able to complete without assistance from the caregiver increased overall by approximately 25% when the device was used [10].

Although this prototype and its algorithms showed some success, we have identified several remaining challenges. Limitations include the assumption of full observability (i.e., that the environment and user context are fully represented by the data provided), inability to tailor its prompting strategy to specific users, and the reliance on deterministic hand-coded rules to make decisions (w.r.t. when and how to provide assistance). The ability to plan appropriate courses of action automatically, deal with partial observability, and customize behavior to specific users has led to the development of the POMDP approach described here.

3 A POMDP Approach

The decision problem faced by COACH is fraught with uncertainty, both with respect to the observability of the environment, and the effects of system actions. Furthermore, our goal is to satisfy a number of different objective criteria that often conflict and cannot be achieved with certainty. For this reason, a POMDP provides the best formal framework for modeling the decision process. POMDPs allow one to model imprecise information about the current environment state, uncertainty in the effects of actions, and multiple, conflicting objectives. Optimal policies will proffer courses of action that balance the importance of specific objectives with their odds of success; furthermore, they account for the long-term impact of decisions, including the *value of information* inherent in an action, thereby allowing to the system to actively *learn* about certain environment or user characteristics.

Many domain characteristics associated with ADL guidance (handwashing being just one example) strongly suggest the use of a POMDP. These include:

> Sensing (e.g., vision in our system) is prone to noise, hence environment variables must be estimated probabilistically (e.g., the location of a user's hands and body must be estimated due to sensor noise or occlusion). As a result, the task step the user is attempting to engage must also be estimated.
>
> Successful completion of a task step given a specific form of guidance is generally stochastic (e.g., a user may react appropriately to an audio prompt only some percentage of the time).
>
> Conflicting objectives need to be traded off against one another and their odds of success (e.g., maximizing the odds of task completion, maximizing the number of successful task steps without caregiver aid, minimizing caregiver intervention, and minimizing user frustration).
>
> A given form of guidance may have different odds of success for specific individuals, hence requiring customization to specific individuals based on our current estimates (e.g., a specific user may tend to get frustrated with prompting that is too frequent).

3.1 The POMDP Model

We now describe a specific POMDP model for the handwashing ADL that captures the desiderata described above. We also describe a fully observable counterpart of this model that illustrates how MDPs can be applied. The core MDP addresses stochasticity and multiple (long-term) objectives, while the POMDP extension captures partial observability, noise, and the ability to customize behavior to specific individuals by estimation of hidden user characteristics. We take pains to describe several variants of each model. This is because the efficacy study conducted using human caregivers was based on an earlier version of the MDP. We wish to evaluate our results using this baseline; but also describe

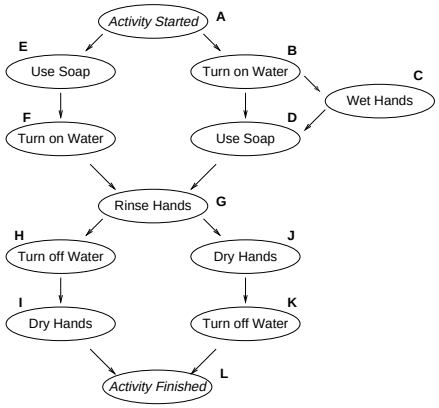


Figure 1: Plan graph for handwashing ADL.

improvements to the model which will be used in upcoming clinical trials.

A discrete-time POMDP consists of: a finite set of states; a finite set of actions; a stochastic transition model , with denoting the probability of moving from state to when action is taken; a finite observation set ; a stochastic observation model with denoting the probability of observation in state ; and a reward function assigning reward with state transition to induced by action . Given a specific POMDP, our goal is to find a *policy* that maximizes the expected discounted sum of rewards attained by the system. Since the system state is not known with certainty, a policy maps either *belief states* (i.e., distributions over) or action-observation *histories* into choices of actions. We refer to [9] for a detailed overview of POMDP concepts and algorithms.

State Variables

Our state space is characterized by four classes of variables: those that capture the state of the environment; those that summarize the ADL plan steps completed thus far; those summarizing system behavior; and those reflecting certain hidden aspects of the user's personality or mental state.

Environment variables represent the underlying physical state of the environment; these are HL (hand location: at tap, at soap, at towel, at sink, at water, away) and WO (water on: boolean).[2]

Activity status variables capture the ADL steps that the user has completed. Figure 1 shows the legitimate sequences of steps (any path from start to finish) that constitute successful handwashing. There are activity status variables. PS, whose domain is the nodes (A-K) of the plan graph (is shown only for convenience), denotes the most recent step the user has completed. MPS denotes the *maximum* plan step completed, since a user may regress in the plan graph. We use this to reward progress in the graph without rewarding duplication of steps (see reward description below); PSR denotes that the current step has been repeated. Finally, Prg indicates whether "progress" is being made within the current plan step; for example, hands moving to sink position prior to step is indicative of progress being made towards the completion of step despite the fact that the plan step hasn't yet been completed.

System behavior variables provide a summary of the system history relevant to prediction of user responses to a prompt. These are: NP or number of prompts issued for the current plan step (with domain 0–3+); NW or number of time steps waited since last prompt (0–4+); LP, the type of last prompt (see description of prompting actions below); PL, the specificity of the last prompt (see actions below); and Rgr, the number of times the user has regressed in the plan (0–3+), which is used as an (stochastic) indication of general responsiveness of the user.

Finally, *user variables* reflect aspects of the user's mental state that impact their response to prompts. Our current prototype uses only one variable, $Resp$ (general responsiveness), taking values low and $high$, to provide a very crude characterization of user type. However, more sophisticated user modeling can be incorporated into the POMDP, using transition and observation models of precisely the same form as those described below.[3]

Actions and Dynamics

The system has 20 actions, 18 of which comprise prompts for six different plan tasks (water on/off, use soap, wet hands, rinse hands, dry hands) at three levels of specificity (general, moderate, specific). General prompts gently cue the user, while specific prompts are designed to get the user's attention and provide a detailed description of the task. For example, a prompt to use the soap might be worded as "use the soap now" at a general specificity level, but might include references to the color or location of the soap at a specific level, such as "use the soap in the pink bottle on your left". Using the patient's name is another strategy that is effective for specific prompting. The wording of the prompts was chosen based on prior experience, and was fixed for the duration of the experiments we describe here.[4] The other two actions are the "null" action and "call caregiver." The latter action ends the process and is presumed to result in successful task completion. This is an important aspect of our system, since our aim is to develop assistive technologies that supplement rather than replace human caregivers. Our goal is to relieve stress and burden on caregivers by increasing the odds of successful "independent" task completion and reducing the need for constant (human) monitoring. However, in the efficacy study presented here, we remove this action temporarily to ensure the integrity of our evaluations (as described below). In the ongoing clinical trials, it plays a central role.

Transition probabilities describe the stochastic state changes induced by each action. These are specified by *dynamic Bayesian networks* (DBNs) over the state variables,

[2]Other relevant aspects of the environment (e.g., hands wet) can be inferred from plan step as discussed below.

[3]Other user variables might characterize auditory and visual ability, overall dementia level, general level of independence, tendency to frustration, etc. We have solved instances of the POMDP with additional user variables, but do not report these here, since they do not correspond to the MDP used in our efficacy study.

[4]Our current work includes clinical trials to assess the effectiveness of different wordings and prompting strategies. Visual prompts, played through a one-way mirror are being explored as well.

with the conditional probability tables (CPTs) represented by algebraic decision diagrams (ADDs) [6]. The considerable structure in the domain leads to a very compact specification of the dynamics. The precise parameters of the model were produced using a handcrafted prior reflecting our experience with patients in earlier clinical trials using the hand-coded prompting system and our interactions with caregivers.[5]

Space prevents a complete description of the dynamics, but we mention some of the key intuitions underlying the model. The probability of the user taking a specific "user action" such as turning on the water—corresponding to a state change in an environment variable (i.e., WO becoming true)—tends to be higher at appropriate plan steps, but the probability depends on the precise prompting history. For example, the longer the system waits for a response (modeled by NW), the less likely it is that the user will complete the current plan step independently (i.e., without prompting). Furthermore, the more "unsuccessful" prompting has been for the current step (modeled by PL and NP), the less likely it is that the patient will complete the step at all. Such "response" probabilities vary formulaicly with $Resp$ and Rgr. Higher values of Rgr are indicative of lower likelihood of eventual success, while success is more probable when $Resp$ is $high$ than when it is low.[6]

Activity status variables are updated deterministically as a function of environment variables (note that environment variables are themselves only partially observable) in the obvious way. PS may move back and forth through the plan graph (since the user can regress, e.g., by re-wetting their hands), but MPS records the maximal step in the plan reached at any point, and never regresses. System variables are updated deterministically as well (e.g., number of prompts NP is updated with each prompt for the current step, but reset to zero when the plan step changes). Finally, the user variable $Resp$ is static, and does not change value over time (though our *estimates* of its value does).[7]

Rewards

Rewards are associated with various state transitions and costs with specific actions. A large reward is given for full task completion (+300) and smaller rewards (+3) are associated with the (first) achievement of each plan step to encourage/reward even partial success (MPS is used to ensure rewards are not repeated). Action costs are also incorporated to ensure that prompting only occurs when needed. Each prompt is given a small negative reward, with the more time-consuming specific prompts penalized slightly more than simpler general prompts (the costs are -4, -5 and -7). The cost of calling the caregiver should be set so that this action only occurs if the predicted odds of completion are too low, or predicted costs of completion too high.

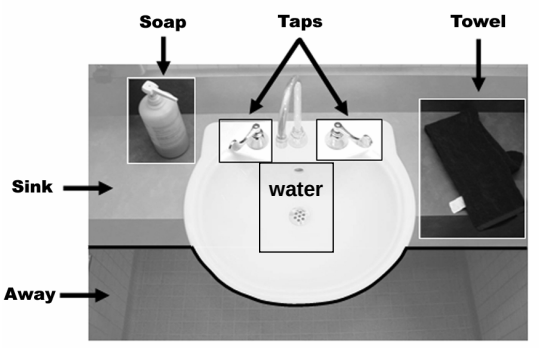


Figure 3: Possible hand locations.

The design of the reward function is based on brief interactions with caregivers, earlier clinical trials with patients using hand-coded COACH, and evaluation of numerous versions of MDP (rather than POMDP) policies. While the rewards seem reasonable, fine-tuning of the reward function is an ongoing research project (see Section 5).

Observations

The handwashing task has two observables: the *reported flow of water* (on/off), and the *reported location* of the user's hand(s). The POMDP is designed to be integrated with a computer vision system that estimates hand position (and water flow indirectly) based on skin color [11]. The vision system reports estimated hand position (-coordinates) which is translated into one of six possible regions (see Fig. 3).

We assume noise in the observation function that detects hand position (i.e., the correct hand position is detected with probability , while each incorrect position is detected with probability . Similarly water flow is detected correctly with probability . These probabilities are estimated based on our prior experience, but a detailed empirical study is needed to assess these accurately.

3.2 A Fully Observable Model

The POMDP model can be made fully observable with two simple changes. First, if we remove the hidden variable $Resp$, a key source of partial observability disappears, though we sacrifice the ability to customize behavior to specific individuals.[8] Second, if we assume the tracking system computes hand position and water flow without error, the system becomes fully observable. The assumption of perfect observability is not unreasonable if we use a switch sensor on the tap and additional cameras to disambiguate obscured views and reduce the inherent noise in image processing.

3.3 Computational Results

To study the feasibility of the model, we solved both the MDP and POMDP formulations of the handwashing ADL. The cost of calling the caregiver in both trials was set very high ( ), to ensure that this action was never taken,

[5] Some versions of our model (see Section 5) include updates of this prior using data produced by coding 21 hours of video capturing 600 trials of human caregivers interacting with patients.

[6] The influence of $Resp$ on user response is estimated based on the authors' previous clinical experience. Planned clinical evaluation of the POMDP policy will be used to assess this more precisely.

[7] This is true for the time scales we currently consider, but certain aspects of a user's state, such as level of dementia, general responsiveness, etc., can change (e.g., deteriorate) over longer periods.

[8] Customization based on other fully observable attributes, such as hearing and visual ability, is still possible in an MDP.

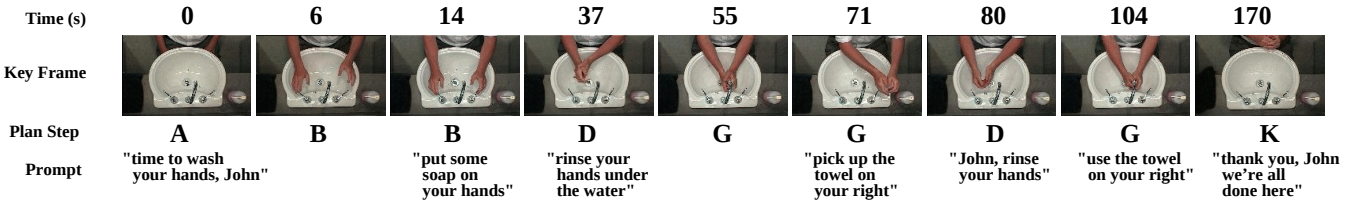


Figure 2: Example sequence during a trial in which prompts were selected by the MDP policy. Prompts were read by a human caregiver. The plan steps are those from Figure 1. The user turns the water on and off independently (at 6 and 160 seconds, respectively), but must be prompted for all other steps. The user regresses at 71 seconds by putting more soap on his hands, which the system recognizes and thus gives a second prompt to rinse hands.

as required by the evaluation procedure we describe in Section 4. The version of the fully observable MDP we solved has 12 variables (25,090,560 states) and 20 actions, rendering explicit state-based approaches impossible. However, due to the DBN structure, we were able to solve this problem exactly using the SPUDD algorithm [6]. An optimal policy was produced in 64 minutes, and was represented using an ADD with internal nodes and leaves. The optimal value function was an ADD having internal nodes and leaves. We discuss evaluation of this policy below.

The POMDP model has one additional variable, hence a state space of size 50,181,120, the same action space, and 12 observations. It is far beyond the reach of any exact solution techniques or model-based approximation methods. As a result, we developed a new approximation algorithm which exploits the structure in the system dynamics and rewards (as represented by our ADDs). *Perseus-ADD* consists of the Perseus algorithm [18], a randomized point-based version of value iteration, reconstructed to take advantage of ADD structure [16]. The algorithm results in a policy of 36 state-value functions (-vectors), computed in 42.7 hours.

The relative quality of the MDP and POMDP policies demonstrates the importance of accounting for noise and user characteristics. While the POMDP policy is approximate, we evaluated it in simulation from the starting state, using 500 trials of 60 steps, with $Resp$ uniformly distributed in the initial belief state, and the specific value of $Resp$ drawn randomly for each trial. The average value obtained by the POMDP policy is 100.1 over 60 steps. The value of the optimal policy for the fully observable MDP (ignoring $Resp$) is 134.2 (which provides an upper bound on the value of the unknown optimal POMDP solution). However, this MDP value cannot be realized in the partially observable environment. So we implemented the MDP policy in this environment by computing the most likely state at each stage of the simulation and applying the corresponding MDP policy choice. In simulation, this attains an average value of 95.6 over 60 steps. We see that the MDP model provides a reasonable approximation for this POMDP despite its limitations. However, the full POMDP policy (despite only being solved approximately) outperforms the MDP policy by a significant margin. The value of POMDP modeling is apparent when one considers that the computational overhead in solving the policy is borne offline—once computed, the optimal policy is applied in real-time without any significant computation. All that is required online is simple belief-state updating, the critical component in allowing the policy to choose different prompting strategies for different users as beliefs about these users evolve.

It is also interesting to note the customization of the POMDP policy based on its estimate of the *Resp* variable. When a patient tends to be slow to respond to prompts, it is better to wait several time steps before repeating the prompt. For instance, when a patient has reached plan step *G* and was prompted to dry her hands two time steps ago, the POMDP policy repeats the prompt when the probability of $Resp$ $high$ is sufficiently high, but waits when it is low. Similarly, when plan step *J* is reached and the patient was prompted to turn off the water two time steps ago, the prompt is repeated only when responsiveness is believed to be high. In contrast, the MDP policy doesn't have the ability to estimate the level of responsiveness since it is not directly observable. As a result, in both cases it repeats the prompt at the risk of annoying a slower, less responsive patient.

4 Caregiver Evaluation

While the evaluation of the MDP and POMDP policies in simulation is useful, the true value of the system can ultimately only be gauged in clinical trials. Clinical trials with Alzheimer's patients are currently underway to test our results. These trials are based on improved versions of the models (see Section 5). Results of these trials will be reported in a longer version of this paper.

Before undertaking clinical trials, an *efficacy study* of the MDP policy was undertaken to confirm its plausibility before applying it to dementia patients. In this study, an actor with considerable experience with dementia patients simulated the behavior of a user in the handwashing ADL during 33 trials, each of which was videotaped. This simulated patient was guided using either the prompting strategy given by the MDP policy, or by a professional caregiver with over 30 years experience. During the caregiver trials, the caregiver acted naturally, prompting using her own strategy. During the MDP trials, the same professional caregiver read (verbatim) the prompts provided by the MDP policy, in order to prevent verbal distinction between caregiver and MDP trials. The *HL* and *WO* variables (describing the location of the actor's hands and whether the water was on) were manually annotated by a researcher during the trials, thus fulfilling the perfect observability assumption of the MDP (note that using the vision system would not have allowed accurate implementation of the MDP policy). The videos were then viewed by 30 professional caregivers (different from the prompter), who evaluated the performance of the prompting in each trial. These

evaluators were unaware that the prompts in some trials were selected by a computer (this is why the *call caregiver* action could not be used).

Fig. 2 depicts snapshots of one of the MDP-guided scenarios used in the efficacy study. In this case, the handwashing subject is able to complete the steps of turning on the water (step) and turning the water off (step) independently. The subject ignores the prompt given to him at to dry his hands and regresses in the activity to step by applying soap instead. The planning system copes with this by identifying the regression, and prompting him to rinse his hands a second time. The majority of the time the prompting screen remained blank, as portrayed at and , allowing the subject time to attempt each step on his own.

Six of the 33 scenarios (three MDP and three human) were chosen for evaluation by 30 professional caregivers.[9] Each evaluator rated the strategy employed in each of the six scenarios, using a five-point Likert scale, on five criteria: *Identification* (the prompt(s) given appropriately identified the next task); *Detail* (level of prompt detail appropriate); *Time* (an appropriate amount of time to attempt task provided before being prompted); *Repetitions* (number of prompt repetitions was appropriate); and *Overall Effectiveness* (patient was guided effectively). Free-form comments were also provided.[10] Quantitative results (see Fig. 4) clearly indicate (results are statistically significant) that the professional caregiver outperforms the MDP policy in all evaluated aspects. Indeed, we had no expectation that the MDP policy would perform as well. Of course, our goals are much more modest; we intend the system to supplement human caregivers, not replace them. Furthermore, the standard of an experienced professional caregiver sets the bar quite high.

Despite this, we were quite encouraged by the performance of the MDP. The average rating of the MDP was higher than that of the human professional in 28 of 150 (18.7%) evaluator-aspect pairs. Furthermore, qualitative comments indicate that the evaluators viewed the MDP policy as adequate (and we believe the evaluators to be rather critical given the average rating of the human caregiver); and none suspected the prompting to be computerized. These facts suggest the MDP prototype can serve as the basis for a supplemental assistive device. Finally, many of the critical comments made of the MDP policy reflected issues that could easily be remedied (and do not reflect a weakness in the MDP approach per se). For instance, many evaluators felt that turning the water on before asking a patient to use soap, or turning the water off before drying hands, provides a natural cue as to what the next activity step is (different transition priors or further data for training the transition model could readily reflect this). Suggestions were also made regarding the language construction of prompts, the use of positive feedback and a friendly voice, and checking water temperature before allowing the patient to immerse his/her hands. None of these require modification of the current MDP model.

[9] Selection criteria were based on scenario length (no more than 3 minutes) and the actor's head being not visible in scene

[10] A detailed technical report on experimental methodology and results is available elsewhere [2].

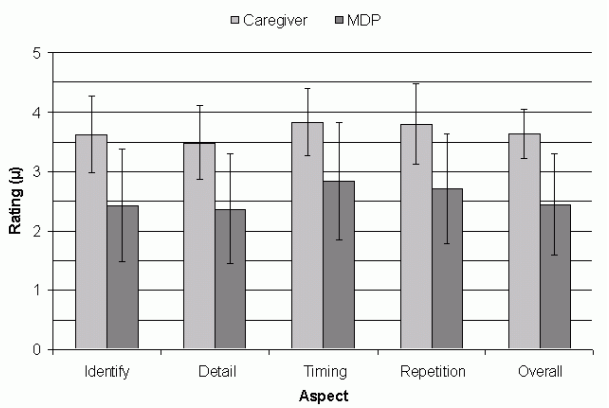


Figure 4: Caregiver ratings of MDP effectiveness for 5 criteria.

Addressing certain evaluator criticisms would require some modifications of the model (or fine-tuning of parameters). These include: using prompts that are tailored to the abilities of the user; giving the patient enough time for step completion; and incorporating visual cues (something that we have developed and tested independently). Finally, in some situations, the patient asked the caregiver a question, such as "where's the soap?". With no dialog model or speech sensors, the MDP cannot react to this (which evaluators viewed negatively). However, POMDP models of spoken dialog [19] could be used for this purpose in the future.

5 Further Model Validation & Improvement

The efficacy study supports the value of the POMDP approach. Although the study was based on a handcrafted MDP model, the policy evaluated by caregivers showed promise as a valuable means of relieving caregiver burden. Since the POMDP policy outperforms this MDP policy in simulation, we expect the POMDP model to offer additional value in a realistic settings. Ultimately, the value of the POMDP approach must be verified in a clinical setting. Clinical trials are currently underway in which both the MDP and POMDP approaches are being compared to human caregivers in guiding Alzheimer's patients through handwashing. The POMDP policy being evaluated is, in fact, somewhat different than the one reported here; we focus on the direct comparison of the POMDP described above with the MDP since it is a direct extension of the MDP (and thus differences in performance are largely due only to the power of the model to handle noise and user characteristics). The model used in the current clinical trials is significantly enhanced in several ways: a more realistic reward/cost function; a more realistic model the influence of level of responsiveness and response delays on step completion; model parameters estimated from video data of human caregivers; and a simplified and improved plan graph reflecting suggestions of the MDP policy-evaluators.

Our experience to date suggests a number of important avenues for model improvement. A difficult task is constructing reasonable reward models; while transition models can be constructed from data, rewards cannot. One future aim is to validate the reward model indirectly by having caregivers critique the policies directly. While caregivers often have dif-

ficulty quantifying the utility of, say, partial versus complete task success, or the cost of prompting, they often "recognize a good policy when they see it." Using precise caregiver policy critique, we will re-engineer the reward function so that the resulting optimal policy is consistent with the suggestions (reminiscent of revealed preference in economics or *inverse reinforcement learning* [14]).

Another important direction is learning model structure and user behaviors from sequence data, rather than imposing our own structure on tasks. Preliminary work along these lines is reported in [5]. Finally, generalizing the model to more directly account for continuous nature of both time, states (e.g., hand position) and observations in this domain is of critical importance for more accurate modeling and useful prompting. Preliminary results dealing with continuous observations in a simplified version of our domain, using a new algorithm for solving continuous observation POMDPs, are reported in [4].

6 Concluding Remarks

We have proposed a decision-theoretic view of ADL assistance for people with dementia, arguing that POMDPs provide an ideal model for such problems. While we developed this model for a specific ADL, the modeling principles embodied in our approach are broadly applicable. Despite the size and complexity of the model, we have demonstrated that the MDP and POMDP solution algorithms can successfully be used to solve these problems, thus producing (approximately) sequentially optimal policies for ADL prompting.

Considerable future work is required to apply the system in practice. Clinical trials are currently in progress to help evaluate the effectiveness of the model and to help improve it. Of particular interest are ways to to improve the (admittedly simple) user model. Sessions with caregivers are also planned to help refine our reward function. We are currently exploring many of the interesting technical issues associated with revising MDP and POMDP models (especially reward functions) based on policy critique. We are also working toward applying POMDP models to other ADLs, such as toileting, and general living patterns. Modeling these different tasks in a consistent, decision-theoretic fashion will eventually lead to hierarchies of POMDP CAT systems throughout a home or care facility, all working towards the same goal. The ability of these POMDPs to monitor *user variables*, such as $Resp$, means that changes in a patient's state of health can be assessed automatically over different time scales.

Acknowledgements

The authors thank the Bonnie Fernie for help with the experiments, and the caregivers who participated in the evaluations. The authors gratefully acknowledge the support of the Natural Sciences and Engineering Research Council (NSERC), Intel Corp., the American Alzheimer Association, and the Institute for Robotics and Intelligent Systems (IRIS).

References

[1] J. Bates, J. Boote, and C. Beverly. Psychosocial interventions for people with a dementing illness: A systematic review. *Journal of Advanced Nursing*, 45(6):644–658, 2004.

[2] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, and A. Mihailidis. A planning system based on Markov decision processes to guide people with dementia through activities of daily living. Manuscript, 2005.

[3] D. Geldmacher and P. Whitehouse Jr. Differential diagnosis of Alzheimer's disease. *Neurology*, 48(5-Suppl.6):S2–S9, 1997.

[4] J. Hoey and P. Poupart. Solving POMDPs with Continuous or Large Discrete Observation Spaces. In *IJCAI-05*, Edinburgh, 2005. to appear.

[5] J. Hoey, P. Poupart, C. Boutilier, and A. Mihailidis. Semi-supervised learning of patient-caregiver interactions using partially observable Markov decision processes. Working paper, 2005.

[6] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: Stochastic planning using decision diagrams. In *UAI*, pages 279–288, Stockholm, 1999.

[7] H. Kautz, L. Arnstein, G. Borriello, O. Etzioni, and D. Fox. An overview of the assisted cognition project. In *AAAI-2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care*, Edmonton, 2002.

[8] N. L. Kirsch. Computer-assisted interactive task guidance: Facilitating the performance of a simulated vocational task. *Journal of Head Trauma Rehabilitation*, 7(3):17–25, 1992.

[9] W. S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

[10] A. Mihailidis, J. C. Barbanel, and G. R. Fernie. The efficacy of an intelligent cognitive orthosis to facilitate handwashing by persons with moderate-to-severe dementia. *Neuropsychological Rehabilitation*, 14(1/2):135–171, 2003.

[11] A. Mihailidis, B. Carmichael, and J. Boger. The use of computer vision in an intelligent environment to support aging-in-place, safety, and independence in the home. *IEEE Trans. on Information Technology in Biomedicine*, 8(3):1–11, 2004.

[12] A. Mihailidis, G. R. Fernie, and J. C. Barbanel. The use of artificial intelligence in the design of an intelligent cognitive orthosis for people with dementia. *Assistive Technology*, 13:23–39, 2001.

[13] E. Mynatt, I. Essa, and W. Rogers. Increasing the opportunities for aging in place. In *Proceedings of ACM Universal Usability Conference*, Arlington, VA., 2000.

[14] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670, Stanford, CA, 2000.

[15] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3–4), 2003.

[16] P. Poupart. *Exploiting structure to efficiently solve large scale partially observable Markov decision processes.* Ph.D. thesis, Dept. of Computer Science, University of Toronto, 2005.

[17] M. E. Pollack. Planning technology for intelligent cognitive orthotics. In *AIPS*, pages 322–331, Toulouse, 2002.

[18] N. Vlassis and M. T. J. Spaan. A fast point-based algorithm for POMDPs. In *Proc. Belgian-Dutch Conf. on Machine Learning*, Brussels, 2004.

[19] B. Zhang, Q. Cai, J. Mao, and B. Guo. Planning and acting under uncertainty: A new model for spoken dialogue system. In *UAI*, pages 572–579, Seattle, WA, 2001.

Sensitivity Analysis in Markov Networks

Hei Chan and Adnan Darwiche
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
{hei,darwiche}@cs.ucla.edu

Abstract

This paper explores the topic of sensitivity analysis in Markov networks, by tackling questions similar to those arising in the context of Bayesian networks: the tuning of parameters to satisfy query constraints, and the bounding of query changes when perturbing network parameters. Even though the distribution induced by a Markov network corresponds to ratios of multi-linear functions, whereas the distribution induced by a Bayesian network corresponds to multi-linear functions, the results we obtain for Markov networks are as effective computationally as those obtained for Bayesian networks. This similarity is due to the fact that conditional probabilities have the same functional form in both Bayesian and Markov networks, which turns out to be the more influential factor. The major difference we found, however, is in how changes in parameter values should be quantified, as such parameters are interpreted differently in Bayesian networks and Markov networks.

1 Introduction

In the domain of uncertainty reasoning, graphical models are used to embed and represent probabilistic dependencies between random variables. There are two major types of graphical models: Bayesian networks [Pearl, 1988; Jensen, 2001] and Markov networks [Pearl, 1988]. In both models, there are two components, a qualitative part and a quantitative part. In the qualitative part, a graph is used to represent the interactions between variables, such that variables with direct interactions are connected by edges, and information of independence relationships between variables can be inferred from the network structure. In the quantitative part, a set of parameters are specified to quantify the strengths of influences between related variables. The joint probability distribution can then be induced from the two components.

Bayesian networks and Markov networks differ in both the qualitative and quantitative parts of the models. For the qualitative part, directed edges are used in a Bayesian network to represent influences from one variable to another, and no cycles are allowed in the graph, while undirected edges are used in a Markov network to represent symmetrical probabilistic dependencies between variables.

For the quantitative part, the parameters in a Bayesian network are specified in conditional probability tables (CPTs) of the variables, where each CPT consists of the conditional probability distributions of a variable given instantiations of its parents, which are the variables that directly influence this variable. Because these parameters are conditional probabilities, they are more intuitive, but have to obey certain properties. For example, the probabilities in each conditional distribution must be normalized and sum to 1. On the other hand, the parameters in a Markov network are given in potentials, defined over instantiations of cliques, which are maximal sets of variables such that every pair of variables in a clique are connected by an edge. These parameters do not need to be normalized and can be changed more freely.

The topic of sensitivity analysis is concerned with understanding and characterizing the relationship between the local parameters quantifying a network and the global queries computed based on the network. Sensitivity analysis has been investigated quite comprehensively in Bayesian networks [Laskey, 1995; Castillo *et al.*, 1997; Jensen, 1999; Kjærulff and van der Gaag, 2000; Chan and Darwiche, 2002; 2004; 2005]. Earlier work on the subject has observed that the distribution induced by a Bayesian network corresponds to multi-linear functions, providing effective computational methods for computing $\partial Pr(\mathbf{e})/\partial\theta_{x|\mathbf{u}}$: the partial derivative of a joint probability with respect to a network parameter. These earlier results were recently pursued further, providing effective methods for solving the following problems:

- *What is the necessary parameter change we need to apply such that a given query constraint is satisfied?* For example, we may want some query value $Pr(z \mid \mathbf{e})$ to be at least p by changing network parameters. Efficient solutions were given to this problem for computing minimum changes for single parameters [Chan and Darwiche, 2002], and multiple parameters in a single CPT [Chan and Darwiche, 2004].
- *What is the bound on the change in some query value if we apply an arbitrary parameter change?* Here, we are interested in finding a general bound for any query and any parameter of any network. It has been shown that the log-odds change in any conditional query in a

Bayesian network is bounded by the log-odds change in any single parameter [Chan and Darwiche, 2002].

- *What is the bound on the difference between some query value induced by two networks that have the same structure but different parameter values?* The solution to this problem is based on a newly proposed distance measure [Chan and Darwiche, 2005], which allows one to bound the difference between the query values under two distributions. Based on this distance measure, and given two Bayesian networks that differ by only a single CPT, the global distance measure between the distributions induced by the networks can be computed from the local distance measure between the CPTs, thereby allowing one to provide a bound on the difference between the query values. Moreover, if we are given multiple CPTs that do not intersect with each other, the global distance measure can still be computed as the sum of the local distance measures between the individual CPTs.

In this paper, we address these key questions of sensitivity analysis but with respect to Markov networks. The main question of interest is the extent to which these promising results hold in Markov networks as well. There is indeed a key difference between Bayesian networks and Markov networks which appears to suggest a lack of parallels in this case: whereas the joint distribution induced by a Bayesian network corresponds to multi-linear functions, the joint distribution induced by a Markov network corresponds to ratios of multi-linear functions. As it turns out, however, the conditional probability function has the same functional form in both Bayesian and Markov networks, as ratios of multi-linear functions. This similarity turns out to be the key factor here, allowing us to derive similarly effective results for Markov networks. This is greatly beneficial because we can answer the previous three questions in the context of Markov networks as well, with the same computational complexity. For example, we can go through each parameter in a Markov network, compute the minimum single parameter changes necessary to enforce a query constraint, and find the one that perturbs the network the least. Alternatively, we can change all parameters in a single clique table, and find the change that minimizes network perturbation. Afterwards, we can compute a bound on any query change using the distance measure incurred by the parameter change.

Our results, however, point to a main semantical difference between Bayesian networks and Markov networks, relating to how one should quantify and measure parameter changes. That is, how should one quantify a parameter change from .3 to .4? In a Bayesian network, parameters are interpreted as conditional probabilities, and the measure which quantifies the change is the *relative odds change* in the parameter value. This means query values are much more sensitive to changes in extreme probability values, whether close to 0 or 1. On the other hand, in a Markov network, parameters are interpreted as compatibility ratios, and the measure which quantifies the change is the *relative change* in the parameter value. This difference stems from how the parameters in the two models are interpreted and will be explained in more depth later.

This paper is structured as follows. Section 2 is dedicated to the definition and parameterization of Markov networks. Section 3 presents a procedure which tunes parameters in a Markov network in order to satisfy a query constraint. Section 4 introduces a distance measure that can be used to bound query change between distributions induced by two Markov networks. Section 5 concludes the paper.

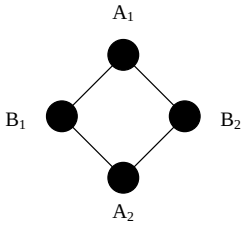


Figure 1: A sample Markov network structure in terms of an undirected graph G.

2 Markov networks

A Markov network $M = (G, \Theta)$ consists of two parts: the network structure in terms of an undirected graph G, and the parameterization Θ. Each node in the undirected graph G represents a random variable, and two variables are connected by an edge if there exists a direct interaction between them. The absence of an edge between two variables means any potential interaction between them is indirect and conditional upon other variables.

As an example, consider four individuals, A_1, A_2, B_1, B_2, and we wish to represent the possible transmission of a contagious disease among them. A network structure which consists of these variables is shown in Figure 1. Each pair of variables $\{A_i, B_j\}$ is connected by an edge, meaning these pairs of individuals are in direct contact of each other. However, A_1 and A_2 are not connected with an edge, meaning they do not interact directly with each other, although diseases can still be transmitted between them indirectly through B_1 or B_2.

To quantify the edges in a Markov network, we first define *cliques* in the network structure. A clique $\mathbf{C}$ is a maximal set of variables where each pair of variables in the set $\mathbf{C}$ is connected by an edge. In the network structure in Figure 1, there are four cliques: $\{A_1, B_1\}$, $\{A_1, B_2\}$, $\{A_2, B_1\}$, $\{A_2, B_2\}$.

For each clique $\mathbf{C}$, we define a table $\Theta_{\mathbf{C}}$ that assigns a nonnegative number $\theta_{\mathbf{c}}$ to each instantiation $\mathbf{c}$ of variables $\mathbf{C}$, which measures the relative degree of compatibility associated with each instantiation $\mathbf{c}$. The numbers given indicate the level of compatibility between the values of the variables in the clique, where direct interactions exist between every pair of variables. For example, in our example network, we define the same clique table Θ_{A_i,B_j} shown in Table 1 for each clique $\{A_i, B_j\}$. For example, it can be viewed that both A_i and B_j not having the disease is four times more compatible than both A_i and B_j having the disease, since $\theta_{\bar{a_i},\bar{b_j}} = 4\theta_{a_i,b_j}$.[1]

[1]Parameters may be estimated on an ad hoc basis [Geman and Geman, 1984] or by statistical models. It can be difficult to assign meanings and numbers to Markov network parameters intuitively [Pearl, 1988, pages 107–108].

A_i	B_j	Θ_{A_i,B_j}
a_i	b_j	1
a_i	$\bar{b}_j$	2
$\bar{a}_i$	b_j	3
$\bar{a}_i$	$\bar{b}_j$	4

Table 1: The clique table of $\{A_i, B_j\}$ of the Markov network whose structure is shown in Figure 1.

2.1 Joint and conditional probabilities induced by Markov networks

We now proceed to find the distribution induced by a Markov network. If $\mathbf{X}$ is the set of all variables in the Markov network $M = (G, \Theta)$, the joint potential ψ over $\mathbf{X}$ induced by M is defined as:

$$\psi(\mathbf{x}) \stackrel{def}{=} \prod_{\mathbf{c} \sim \mathbf{x}} \theta_{\mathbf{c}}. \quad (1)$$

That is, $\psi(\mathbf{x})$ is the product of parameters $\theta_{\mathbf{c}}$ in each clique $\mathbf{C}$, such that instantiation $\mathbf{c}$ is consistent with $\mathbf{x}$. The joint distribution Pr^M induced by Markov network M is then defined as its normalized joint potential:

$$Pr^M(\mathbf{x}) \stackrel{def}{=} \frac{\psi(\mathbf{x})}{\sum_{\mathbf{x}} \psi(\mathbf{x})} = K\psi(\mathbf{x}),$$

where $K = 1/\sum_{\mathbf{x}} \psi(\mathbf{x})$ is the normalizing constant. From this equation, we can easily verify that the specific parameter values in a clique are not important, but their ratios are. This is a major departure from Bayesian networks, where specific parameter values in CPTs are important.

Given the network structure in Figure 1 with the parameterization in Table 1, we can compute the joint probability of any full instantiation of variables induced by our example network. For example, the joint probability of $a_1, \bar{a}_2, b_1, \bar{b}_2$ is equal to $K\theta_{a_1,b_1}\theta_{a_1,\bar{b}_2}\theta_{\bar{a}_2,b_1}\theta_{\bar{a}_2,\bar{b}_2} = K \cdot 1 \cdot 2 \cdot 3 \cdot 4$, where K is the normalizing constant.

For an instantiation $\mathbf{e}$ of variables $\mathbf{E} \subseteq \mathbf{X}$, we can express the joint probability $Pr^M(\mathbf{e})$ as:

$$Pr^M(\mathbf{e}) = \sum_{\mathbf{x} \sim \mathbf{e}} Pr^M(\mathbf{x}) = \sum_{\mathbf{x} \sim \mathbf{e}} K\psi(\mathbf{x}) = K\phi(\mathbf{e}),$$

where the function $\phi(\mathbf{e})$ is defined as the sum of the potentials of $\mathbf{x}$ which are consistent with $\mathbf{e}$:

$$\phi(\mathbf{e}) \stackrel{def}{=} \sum_{\mathbf{x} \sim \mathbf{e}} \psi(\mathbf{x}) = \sum_{\mathbf{x} \sim \mathbf{e}} \prod_{\mathbf{c} \sim \mathbf{x}} \theta_{\mathbf{c}}, \quad (2)$$

and any conditional probability $Pr^M(z \mid \mathbf{e})$ as:

$$Pr^M(z \mid \mathbf{e}) = \frac{Pr^M(z, \mathbf{e})}{Pr^M(\mathbf{e})} = \frac{K\phi(z, \mathbf{e})}{K\phi(\mathbf{e})} = \frac{\phi(z, \mathbf{e})}{\phi(\mathbf{e})}. \quad (3)$$

From Equation 2, we can see that both $\phi(z, \mathbf{e})$ and $\phi(\mathbf{e})$ are sums of products of the network parameters. Therefore, the conditional probability $Pr^M(z \mid \mathbf{e})$ induced by a Markov network can be expressed as the ratio of two multi-linear functions of the network parameters.

As a comparison, the parameterization Θ of a directed acyclic graph N for a Bayesian network consists of a set of CPTs, one for each variable X, which are conditional probabilities in the form of $\theta_{x|\mathbf{u}} = Pr(x \mid \mathbf{u})$, where $\mathbf{U}$ are the parents of X. The joint distribution Pr^B induced by Bayesian network $B = (N, \Theta)$ is:

$$Pr^B(\mathbf{x}) \stackrel{def}{=} \prod_{x, \mathbf{u} \sim \mathbf{x}} \theta_{x|\mathbf{u}}.$$

The joint probability $Pr^B(\mathbf{e})$ can be expressed as:

$$Pr^B(\mathbf{e}) = \sum_{\mathbf{x} \sim \mathbf{e}} \prod_{x, \mathbf{u} \sim \mathbf{x}} \theta_{x|\mathbf{u}},$$

and any conditional probability $Pr^B(z \mid \mathbf{e})$ as:

$$Pr^B(z \mid \mathbf{e}) = \frac{Pr^B(z, \mathbf{e})}{Pr^B(\mathbf{e})} = \frac{\sum_{\mathbf{x} \sim z, \mathbf{e}} \prod_{x, \mathbf{u} \sim \mathbf{x}} \theta_{x|\mathbf{u}}}{\sum_{\mathbf{x} \sim \mathbf{e}} \prod_{x, \mathbf{u} \sim \mathbf{x}} \theta_{x|\mathbf{u}}}. \quad (4)$$

We can easily see the similarities between the expressions of $Pr^M(z \mid \mathbf{e})$ and $Pr^B(z \mid \mathbf{e})$ in Equations 3 and 4 lie in the fact that both can be expressed as ratios of two multi-linear functions of the network parameters. This is different for joint probabilties, where in a Bayesian network they are multi-linear functions of the network parameters, while in a Markov network they are ratios of multi-linear functions of the network parameters, since the normalizing constant can be expressed as $K = 1/\sum_{\mathbf{x}} \psi(\mathbf{x}) = 1/\phi(\text{true})$. This similarity in terms of conditional probabilities will be the basis of our results in the following section.

3 Tuning of parameters in Markov networks

We now answer the following question in the context of Markov networks: what is the necessary change we can apply to certain parameter(s) such that a query constraint is satisfied? For example, if Pr is the probability distribution induced by a Markov network, we may want to satisfy the query constraint $Pr(z \mid \mathbf{e}) \geq k$, for some $k \in [0, 1]$.[2]

Notice that given all the parameters $\theta_{\mathbf{c}}$ in the clique $\mathbf{C}$, $\phi(\mathbf{e})$ can be expressed as a multi-linear function of these parameters, as shown in Equation 2. Since $\phi(\mathbf{e})$ is linear in each parameter $\theta_{\mathbf{c}}$, and no two parameters in the same clique are multiplied together, if we apply a change of $\Delta\theta_{\mathbf{c}}$ to each parameter $\theta_{\mathbf{c}}$ in the clique $\mathbf{C}$, the change in $\phi(\mathbf{e})$ is:

$$\Delta\phi(\mathbf{e}) = \sum_{\mathbf{c}} \frac{\partial\phi(\mathbf{e})}{\partial\theta_{\mathbf{c}}} \Delta\theta_{\mathbf{c}}.$$

To ensure the query constraint $Pr(z \mid \mathbf{e}) \geq k$, from Equation 3, this is equivalent to ensuring the inequality $\phi(z, \mathbf{e}) \geq k \cdot \phi(\mathbf{e})$. If the current ϕ values of $z, \mathbf{e}$ and $\mathbf{e}$ are $\varphi(z, \mathbf{e})$ and $\varphi(\mathbf{e})$ respectively, this inequality becomes:

$$\varphi(z, \mathbf{e}) + \Delta\phi(z, \mathbf{e}) \geq k(\varphi(\mathbf{e}) + \Delta\phi(\mathbf{e})),$$

or:

$$\varphi(z, \mathbf{e}) + \sum_{\mathbf{c}} \frac{\partial\phi(z, \mathbf{e})}{\partial\theta_{\mathbf{c}}} \Delta\theta_{\mathbf{c}} \geq k \left(\varphi(\mathbf{e}) + \sum_{\mathbf{c}} \frac{\partial\phi(\mathbf{e})}{\partial\theta_{\mathbf{c}}} \Delta\theta_{\mathbf{c}} \right).$$

Rearranging the terms, we get the following theorem and corollary.

[2] We can easily expand our results in this section into other forms of constraints, such as $Pr(z \mid \mathbf{e}) \leq k$, $Pr(z_1 \mid \mathbf{e}) - Pr(z_2 \mid \mathbf{e}) \geq k$, or $Pr(z_1 \mid \mathbf{e})/Pr(z_2 \mid \mathbf{e}) \geq k$.

Theorem 1 *To ensure the probability distribution Pr induced by a Markov network satisfies the query constraint $Pr(z \mid \mathbf{e}) \geq k$, we must change each parameter $\theta_{\mathbf{c}}$ in the clique $\mathbf{C}$ by $\Delta\theta_{\mathbf{c}}$ such that:*

$$\sum_{\mathbf{c}} \alpha(\theta_{\mathbf{c}})\Delta\theta_{\mathbf{c}} \geq -(\varphi(z,\mathbf{e}) - k \cdot \varphi(\mathbf{e})), \tag{5}$$

where $\varphi(z,\mathbf{e})$ and $\varphi(\mathbf{e})$ are the current ϕ values of $z, \mathbf{e}$ and $\mathbf{e}$ respectively, and:

$$\alpha(\theta_{\mathbf{c}}) = \frac{\partial\phi(z,\mathbf{e})}{\partial\theta_{\mathbf{c}}} - k\frac{\partial\phi(\mathbf{e})}{\partial\theta_{\mathbf{c}}}. \tag{6}$$

Corollary 1 *If instead of changing all parameters in the clique $\mathbf{C}$, we are only allowed to change a single parameter $\theta_{\mathbf{c}}$ by $\Delta\theta_{\mathbf{c}}$, the solution of Theorem 1 becomes:*

$$\alpha(\theta_{\mathbf{c}})\Delta\theta_{\mathbf{c}} \geq -(\varphi(z,\mathbf{e}) - k \cdot \varphi(\mathbf{e})), \tag{7}$$

which returns a solution interval of possible $\Delta\theta_{\mathbf{c}}$.

Therefore, to solve for $\Delta\theta_{\mathbf{c}}$ in Inequality 7 for all parameters in the Markov network, we need to compute the original values $\varphi(z,\mathbf{e})$ and $\varphi(\mathbf{e})$, which should already be known when computing the original probability of $z \mid \mathbf{e}$, and the partial derivatives $\partial\phi(z,\mathbf{e})/\partial\theta_{\mathbf{c}}$ and $\partial\phi(\mathbf{e})/\partial\theta_{\mathbf{c}}$, to find $\alpha(\theta_{\mathbf{c}})$ in Equation 6 for all parameters $\theta_{\mathbf{c}}$. To do this, we can use a procedure in time complexity $O(n\exp(w))$ where n is the number of variables and w is the treewidth of the Markov network, similar to the one proposed to compute partial derivatives in a Bayesian network [Darwiche, 2003]. This can greatly help users debug their network when they are faced with query results that do not match their expectations.

We now use our example network to illustrate this procedure. The probability distribution Pr induced by the current parameterization gives us the conditional query value $Pr(\bar{a_2} \mid a_1) = .789$. Assume that we would like to change a single parameter in the clique $\{A_2, B_1\}$ to ensure the constraint $Pr(\bar{a_2} \mid a_1) \geq .9$. We need to compute the α values for each parameter in the clique using Equation 6, and then use Inequality 7 to solve for the minimal $\Delta\theta_{\mathbf{c}}$. The solutions are:

$$\begin{aligned}
\Delta\theta_{a_2,b_1} &\leq -2.93;\\
\Delta\theta_{a_2,\bar{b_1}} &\leq -1.467;\\
\Delta\theta_{\bar{a_2},b_1} &\geq 12;\\
\Delta\theta_{\bar{a_2},\bar{b_1}} &\geq 8.8.
\end{aligned}$$

However, notice that since the parameter values have to be non-negative, the solution for $\Delta\theta_{a_2,b_1}$ is impossible to achieve. Therefore, no possible single parameter change on θ_{a_2,b_1} is possible to ensure our query constraint. However, we can decrease the parameter $\theta_{a_2,\bar{b_1}}$ from 2 to .533 to ensure our query constraint.

If we are interested in changing all the parameters in the clique $\{A_2, B_1\}$ to satisfy our query constraint, we need to find a solution that satisfies Inequality 5. As a consequence, we are now faced with multiple solutions of changing the parameters, and we want to commit to one which disturbs the network the least. We will discuss this in the next section using a distance measure which measures network perturbation.

4 Bounding belief change between Markov networks using a distance measure

We now answer the following question in the context of Markov networks: what is the bound on the difference between some query value induced by two networks that have the same structure but different parameter values? We will answer it by using a previously proposed distance measure.

Let Pr and Pr' be two probability distributions over the same set of variables $\mathbf{X}$. We use a distance measure $D(Pr, Pr')$ defined as follows [Chan and Darwiche, 2005]:

$$D(Pr, Pr') \stackrel{def}{=} \ln \max_{\mathbf{x}} \frac{Pr'(\mathbf{x})}{Pr(\mathbf{x})} - \ln \min_{\mathbf{x}} \frac{Pr'(\mathbf{x})}{Pr(\mathbf{x})},$$

where we will define $0/0 \stackrel{def}{=} 1$ and $\infty/\infty \stackrel{def}{=} 1$.[3]

The significance of this distance measure is that if the distance measure is $D(Pr, Pr') = d$, the change in the probability of any conditional query $z \mid \mathbf{e}$ is bounded by:

$$e^{-d} \leq \frac{O'(z \mid \mathbf{e})}{O(z \mid \mathbf{e})} \leq e^{d},$$

where $O'(z \mid \mathbf{e})$ and $O(z \mid \mathbf{e})$ are the odds of $z \mid \mathbf{e}$ under distributions Pr' and Pr respectively.[4] Given $p = Pr'(z \mid \mathbf{e})$, this result can also be expressed in terms of probabilities:

$$\frac{p \cdot e^{-d}}{p(e^{-d} - 1) + 1} \leq Pr'(z \mid \mathbf{e}) \leq \frac{p \cdot e^{d}}{p(e^{d} - 1) + 1}. \tag{8}$$

The advantage of this distance measure is that it can be computed using local information for Bayesian networks. Given distributions Pr^B and $Pr^{B'}$ induced by two Bayesian networks B and B' that differ by only the CPT of a single variable X, if $Pr(\mathbf{u}) > 0$ for every $\mathbf{u}$ of parents $\mathbf{U}$, the distance measure between Pr^B and $Pr^{B'}$ can be computed as:

$$\begin{aligned}
D(Pr^B, Pr^{B'}) &= D(\Theta_{X|\mathbf{U}}, \Theta'_{X|\mathbf{U}})\\
&= \ln \max_{x,\mathbf{u}} \frac{\theta'_{x|\mathbf{u}}}{\theta_{x|\mathbf{u}}} - \ln \min_{x,\mathbf{u}} \frac{\theta'_{x|\mathbf{u}}}{\theta_{x|\mathbf{u}}}.
\end{aligned} \tag{9}$$

This is useful because we can compute the bound on any query change using Inequality 8 by computing the local distance between the CPTs $\Theta_{X|\mathbf{U}}$ and $\Theta'_{X|\mathbf{U}}$ in B and B'. For example, if X is binary, and we only change a single parameter $\theta_{x|\mathbf{u}}$ while applying a complementary change on $\theta_{\bar{x}|\mathbf{u}}$, the bound on the change in query $z \mid \mathbf{e}$ is given by:

$$\frac{O(\theta_{x|\mathbf{u}})}{O(\theta'_{x|\mathbf{u}})} \leq \frac{O^{B'}(z \mid \mathbf{e})}{O^{B}(z \mid \mathbf{e})} \leq \frac{O(\theta'_{x|\mathbf{u}})}{O(\theta_{x|\mathbf{u}})}, \tag{10}$$

where $O(\theta'_{x|\mathbf{u}})$ and $O(\theta_{x|\mathbf{u}})$ are the odds of parameters $\theta'_{x|\mathbf{u}}$ and $\theta_{x|\mathbf{u}}$ respectively. Intuitively this means the relative change in the query odds is bounded by the relative change in the parameter odds.

[3] This measure is a distance measure because it satisfies positiveness, symmetry and the triangle inequality.

[4] The odds of $z \mid \mathbf{e}$ under distribution Pr is defined as: $O(z \mid \mathbf{e}) \stackrel{def}{=} Pr(z \mid \mathbf{e})/(1 - Pr(z \mid \mathbf{e}))$.

We can get a similar result for Markov networks, where the distance measure between distributions induced by two Markov networks that differ by only a single clique table can be computed by the distance measure between the tables.

Theorem 2 *Given distributions Pr^M and $Pr^{M'}$ induced by two Markov networks M and M' which differ by only the parameters in a single clique $\mathbf{C}$, such that the clique tables are $\Theta_{\mathbf{C}}$ and $\Theta'_{\mathbf{C}}$ respectively, the distance measure between Pr^M and $Pr^{M'}$ is given by:*

$$\begin{aligned} D(Pr^M, Pr^{M'}) &= D(\Theta_{\mathbf{C}}, \Theta'_{\mathbf{C}}) \\ &= \ln \max_{\mathbf{c}} \frac{\theta'_{\mathbf{c}}}{\theta_{\mathbf{c}}} - \ln \min_{\mathbf{c}} \frac{\theta'_{\mathbf{c}}}{\theta_{\mathbf{c}}}, \quad (11) \end{aligned}$$

if $\partial\psi(\mathbf{x})/\partial\theta_{\mathbf{c}} \neq 0$ for all $\mathbf{c} \sim \mathbf{x}$.[5]

Proof Given instantiation $\mathbf{c}$ such that $\mathbf{c} \sim \mathbf{x}$, the joint potential ψ of $\mathbf{x}$ is linear in the parameter $\theta_{\mathbf{c}}$, and the ratio of $\psi'(\mathbf{x})$ and $\psi(\mathbf{x})$ induced by M' and M respectively is:

$$\frac{\psi'(\mathbf{x})}{\psi(\mathbf{x})} = \frac{\theta'_{\mathbf{c}}}{\theta_{\mathbf{c}}},$$

if $\partial\psi(\mathbf{x})/\partial\theta_{\mathbf{c}} \neq 0$. We have:

$$\frac{Pr^{M'}(\mathbf{x})}{Pr^M(\mathbf{x})} = \frac{K'\psi'(\mathbf{x})}{K\psi(\mathbf{x})} = \frac{K'\theta'_{\mathbf{c}}}{K\theta_{\mathbf{c}}}.$$

Note that since the parameters have changed, the normalizing constants K and K' for networks M and M' respectively are different. Therefore, the distance measure between Pr^M and $Pr^{M'}$ is given by:

$$\begin{aligned} D(Pr^M, Pr^{M'}) &= \ln \max_{\mathbf{x}} \frac{Pr^{M'}(\mathbf{x})}{Pr^M(\mathbf{x})} - \ln \min_{\mathbf{x}} \frac{Pr^{M'}(\mathbf{x})}{Pr^M(\mathbf{x})} \\ &= \ln \max_{\mathbf{c}} \frac{K'\theta'_{\mathbf{c}}}{K\theta_{\mathbf{c}}} - \ln \min_{\mathbf{c}} \frac{K'\theta'_{\mathbf{c}}}{K\theta_{\mathbf{c}}} \\ &= \ln \max_{\mathbf{c}} \frac{\theta'_{\mathbf{c}}}{\theta_{\mathbf{c}}} - \ln \min_{\mathbf{c}} \frac{\theta'_{\mathbf{c}}}{\theta_{\mathbf{c}}} \\ &= D(\Theta_{\mathbf{C}}, \Theta'_{\mathbf{C}}), \end{aligned}$$

if $\partial\psi(\mathbf{x})/\partial\theta_{\mathbf{c}} \neq 0$ for all $\mathbf{c} \sim \mathbf{x}$.□

Therefore, the global distance measure between the distributions induced is equal to the local distance measure between the individual clique tables. This is useful for computing the bound on query change after changing the parameters in a clique table. For example, what is the bound on the change in some query value if we apply an arbitrary change on the single parameter $\theta_{\mathbf{c}}$? The distance measure in this case is:

$$D(Pr^M, Pr^{M'}) = \left| \ln \frac{\theta'_{\mathbf{c}}}{\theta_{\mathbf{c}}} \right|,$$

[5]This condition is satisfied when the network parameters are all strictly positive. However, this is a sufficient condition, not a necessary condition. The necessary condition is there exists some $\mathbf{x}$ such that $\partial\psi(\mathbf{x})/\partial\theta_{\mathbf{c}} \neq 0$ for all $\mathbf{c}$. This means that changing any parameter $\theta_{\mathbf{c}}$ will have some impact on the joint potential ψ.

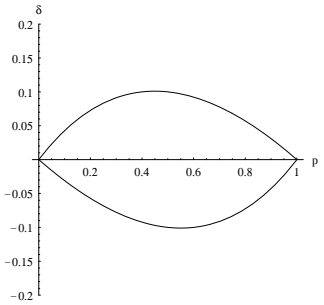


Figure 2: The amount of parameter change δ that would guarantee the query $Pr(z \mid \mathbf{e}) = .75$ to stay within the interval $[.667, .818]$, as a function of the current Bayesian network parameter value p.

and the change in query $z \mid \mathbf{e}$ is bounded by:

$$\frac{\theta_{\mathbf{c}}}{\theta'_{\mathbf{c}}} \leq \frac{O^{M'}(z \mid \mathbf{e})}{O^M(z \mid \mathbf{e})} \leq \frac{\theta'_{\mathbf{c}}}{\theta_{\mathbf{c}}}. \quad (12)$$

This means for Markov networks, *the relative change in query odds is bounded by the relative change in the parameter itself, not the relative change in the parameter odds* as for Bayesian networks. This is an important distinction between Markov networks and Bayesian networks.

As an example, suppose we have a network and we want to ensure the robustness of the query $Pr(z \mid \mathbf{e})$ after we apply a parameter change. Assume that we define robustness as the relative change in any query odds to be less than 1.5. For example, if currently $Pr(z \mid \mathbf{e}) = .75$, the new query value must stay in the interval $[.667, .818]$ after the parameter change. We may ask, how much change can we apply to a parameter if we want to ensure robustness? *The answers for Bayesian networks and Markov networks are different due to our previous results*, as we will show next.

For a Bayesian network, the amount of permissible change is determined by Inequality 10, and is plotted in Figure 2. We can see that the amount of permissible change is small when the parameters have extreme values close to 0 or 1, because the relative odds change is large when even a very small absolute change is applied.

On the other hand, for a Markov network, the amount of permissible change is determined by Inequality 12, and is plotted in Figure 3. We can see that the amount of permissible change is proportional to the parameter values, because relative change is used instead of relative odds change.

Therefore, for a Bayesian network, the sensitivity of the network with respect to a parameter is largest for extreme values close to 0 or 1, and becomes smaller as the parameter approaches .5, while for a Markov network, the sensitivity of the network with respect to a parameter is proportional to its value, and increases as it grows larger.

The distance measure is useful in many aspects of sensitivity analysis. For example, given the possible single parameter changes in our example in the previous section, we can choose the one which perturbs the network the least according to the distance measure. In this case, the most preferred

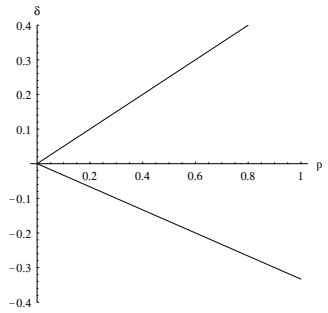


Figure 3: The amount of parameter change δ that would guarantee the query $Pr(z \mid \mathbf{e}) = .75$ to stay within the interval $[.667, .818]$, as a function of the current Markov network parameter value p.

single parameter change is to decrease the parameter $\theta_{a_2,\bar{b_1}}$ from 2 to .533, incurring a distance measure of 1.322.

Moreover, we can also use the distance measure to find the optimal solution for changing all parameters in a clique table, which is the solution that satisfies Inequality 5 and minimizes the distance measure. As the distance measure $D(\Theta_{\mathbf{C}}, \Theta'_{\mathbf{C}})$ is computed using the maximum relative change in the parameters, the relative changes in all parameters must be the same for the optimal solution, because to obtain another solution that satisfies the constraint, we must increase the relative change in one parameter while decreasing the relative change in another, thereby incurring a larger distance measure.

For example, in our example from the previous section, to ensure our query constraint, we would like to decrease the parameters θ_{a_2,b_1} and $\theta_{a_2,\bar{b_1}}$ and increase the parameters $\theta_{\bar{a_2},b_1}$ and $\theta_{\bar{a_2},\bar{b_1}}$, such that the relative changes in all parameters are the same. However, since only the ratios between the parameters are important, we can keep the first two parameters constant and only increase the last two parameters. The optimal solution of Inequality 5 is:

$$\begin{aligned}
\Delta\theta_{a_2,b_1} &= 0; \\
\Delta\theta_{a_2,\bar{b_1}} &= 0; \\
\Delta\theta_{\bar{a_2},b_1} &= 6.257; \\
\Delta\theta_{\bar{a_2},\bar{b_1}} &= 8.676.
\end{aligned}$$

This parameter change incurs a distance measure of .350. It involves all parameters in the clique table and incurs a smaller distance measure than any of the single parameter changes computed earlier.

Finally, if we change the parameters in different clique tables which do not share any variables, the distance measure can be computed as the sum of the local distance measures between the clique tables. This result is also similar to the case of Bayesian networks [Chan and Darwiche, 2004].

5 Conclusion

In this paper, we expanded some of the main results in the topic of sensitivity analysis from Bayesian networks to Markov networks. We were able to find many parallels between the results in both models even given the differences in how their parameters are interpreted. Our results allow for the effective debugging of Markov networks by changing parameters in a single potential for ensuring query constraints. We also showed how to compute a bound on any query change between two Markov networks that have the same structure, but differ in the parameters of a single potential, and to choose parameter changes which minimize network disturbance. Finally, we identified a key semantical difference between Bayesian networks and Markov networks, relating to how parameter changes should be quantified.

References

[Castillo *et al.*, 1997] Enrique Castillo, José Manuel Gutiérrez, and Ali S. Hadi. Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A (Systems and Humans)*, 27:412–423, 1997.

[Chan and Darwiche, 2002] Hei Chan and Adnan Darwiche. When do numbers really matter? *Journal of Artificial Intelligence Research*, 17:265–287, 2002.

[Chan and Darwiche, 2004] Hei Chan and Adnan Darwiche. Sensitivity analysis in Bayesian networks: From single to multiple parameters. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 67–75, Arlington, Virginia, 2004. AUAI Press.

[Chan and Darwiche, 2005] Hei Chan and Adnan Darwiche. A distance measure for bounding probabilistic belief change. *International Journal of Approximate Reasoning*, 38:149–174, 2005.

[Darwiche, 2003] Adnan Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50:280–305, 2003.

[Geman and Geman, 1984] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[Jensen, 1999] Finn V. Jensen. Gradient descent training of Bayesian networks. In *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 190–200, Berlin, Germany, 1999. Springer-Verlag.

[Jensen, 2001] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.

[Kjærulff and van der Gaag, 2000] Uffe Kjærulff and Linda C. van der Gaag. Making sensitivity analysis computationally efficient. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 317–325, San Francisco, California, 2000. Morgan Kaufmann Publishers.

[Laskey, 1995] Kathryn B. Laskey. Sensitivity analysis for probability assessments in Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 25:901–909, 1995.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Francisco, California, 1988.

Compiling Bayesian Networks with Local Structure

Mark Chavira and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095-1596
{chavira,darwiche}@cs.ucla.edu

Abstract

Recent work on compiling Bayesian networks has reduced the problem to that of factoring CNF encodings of these networks, providing an expressive framework for exploiting local structure. For networks that have local structure, large CPTs, yet no excessive determinism, the quality of the CNF encodings and the amount of local structure they capture can have a significant effect on both the offline compile time and online inference time. We examine the encoding of such Bayesian networks in this paper and report on new findings that allow us to significantly scale this compilation approach. In particular, we obtain order–of–magnitude improvements in compile time, compile some networks successfully for the first time, and obtain orders–of–magnitude improvements in online inference for some networks with local structure, as compared to baseline jointree inference, which does not exploit local structure.

1 Introduction

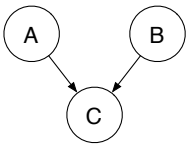
It was shown recently that compiling Bayesian networks corresponds to factoring multi–linear functions (MLFs) [Darwiche, 2003]. In particular, each Bayesian network can be characterized by an MLF of exponential size, whose evaluation and differentiation solves the exact inference problem. Moreover, the MLF can be factored into an arithmetic circuit (AC) whose size is not necessarily exponential, allowing one to use ACs as a compiled representation of Bayesian networks. Interestingly, [Park and Darwiche, 2003] has shown that building a jointree [Jensen *et al.*, 1990; Shenoy and Shafer, 1986] for a Bayesian network corresponds in a precise sense to a process of factoring the MLF into an AC, which is embedded by the jointree structure.

These findings created new possibilities for performing exact inference, as they provided a new computational framework based on factoring MLFs. In fact, a specific MLF factoring method was proposed in [Darwiche, 2002], which can exploit the structure inherent in network parameters. According to this approach, one encodes the MLF using a propositional theory in Conjunctive Normal Form (CNF), factors the CNF, and then immediately extracts the AC from the CNF factorization. The benefit of this logical approach is twofold. First, it allows one to encode local structure in the form of determinism and context specific independence (CSI) [Boutilier *et al.*, 1996]. For example, using this approach, it became practical to compile some Bayesian networks with binary variables and excessive determinism (induced by relational models) and having treewidths in excess of 200 by compiling the AC in minutes and evaluating them in seconds [Chavira *et al.*, 2004]. The second advantage of this approach is its ability to accommodate different representations of conditional probability tables (CPTs) (decision trees, rules, noisy–or, etc.), without the need for algorithmic change. In this paper, we consider only tabular representations.

The critical computational step in the above approach is clearly that of factoring/compiling the CNF, which is done using an exhaustive and refined version of the DPLL algorithm [Davis *et al.*, 1962; Darwiche, 2004]. The key observation underlying this paper is that the efficiency of the factoring step—both factoring time and size of factorization—can be significantly improved through careful CNF encodings which capture as much local structure as possible, and by passing additional information about these CNFs to the factoring algorithm. This becomes especially true when handling networks that have local structure, large CPTs, yet no excessive determinism. We do indeed propose a particular CNF encoding which appears quite effective on networks with such properties. We also identify a key semantic property of the resulting CNFs, which we exploit in the CNF factoring algorithm. By incorporating these findings, we show dramatic improvements in the both offline compile time and online inference. In the offline compilation phase, we get an order–of–magnitude improvement in some cases and an ability to compile some networks for the first time. In the online phase, we observe orders–of–magnitude improvements on some well known benchmarks, such as *Pathfinder*, *Munin1*, and *Water*, over online inference by the baseline jointree algorithm.

2 Factoring Multi–linear Functions

Our investigation is based on three technical observations [Darwiche, 2003]: that every Bayesian network can be interpreted as an exponentially–sized MLF whose evaluation and differentiation solves the exact inference problem; that such an MLF can be factored into an AC whose size may not be ex-



row	A	B	C	$\Pr(c \mid a, b)$
1	a_1	b_1	c_1	$\theta_{c_1\|a_1b_1} = 0$
2	a_1	b_1	c_2	$\theta_{c_2\|a_1b_1} = 0.5$
3	a_1	b_1	c_3	$\theta_{c_3\|a_1b_2} = 0.5$
4	a_1	b_2	c_1	$\theta_{c_1\|a_1b_2} = 0.2$
5	a_1	b_2	c_2	$\theta_{c_2\|a_1b_1} = 0.3$
6	a_1	b_2	c_3	$\theta_{c_3\|a_1b_1} = 0.5$
7	a_2	b_1	c_1	$\theta_{c_1\|a_2b_2} = 0$
8	a_2	b_1	c_2	$\theta_{c_2\|a_2b_2} = 0$
9	a_2	b_1	c_3	$\theta_{c_3\|a_2b_1} = 1$
10	a_2	b_2	c_1	$\theta_{c_1\|a_2b_1} = 0.2$
11	a_2	b_2	c_2	$\theta_{c_2\|a_2b_2} = 0.3$
12	a_2	b_2	c_3	$\theta_{c_3\|a_2b_2} = 0.5$

Figure 1: A small Bayesian network with one of its CPTs, showing local structure in the form of determinism and CSI.

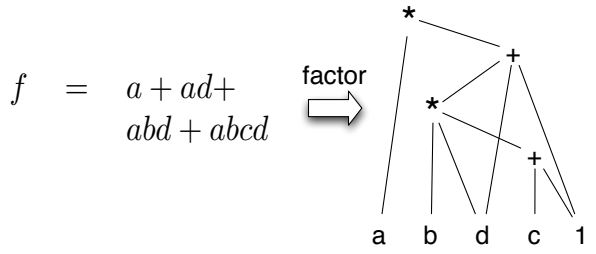


Figure 2: An MLF factored into an AC.

ponential; and that the MLF factoring process can be reduced to factoring a CNF encoding of the MLF.

The MLF for a network contains two types of variables. For each value x of each network variable X, there is an *indicator variable* λ_x. For each network parameter $\Pr(x|\mathbf{u})$, there is a *parameter variable* $\theta_{x|\mathbf{u}}$. The MLF contains a term for each instantiation of the network variables, and the term is the product of all indicators and parameters that are consistent with the instantiation. For the network in Figure 1, variables A and B have two values, and variable C has three values. The MLF corresponding to this network is as follows:

$$\lambda_{a_1}\lambda_{b_1}\lambda_{c_1}\theta_{a_1}\theta_{b_1}\theta_{c_1|a_1,b_1} + \lambda_{a_1}\lambda_{b_1}\lambda_{c_2}\theta_{a_1}\theta_{b_1}\theta_{c_2|a_1,b_1} + \ldots$$
$$\lambda_{a_2}\lambda_{b_2}\lambda_{c_2}\theta_{a_2}\theta_{b_2}\theta_{c_2|a_2,b_2} + \lambda_{a_2}\lambda_{b_2}\lambda_{c_3}\theta_{a_2}\theta_{b_2}\theta_{c_3|a_2,b_2}$$

To compute the probability of evidence $\mathbf{e}$, we evaluate the MLF after setting indicators that contradict $\mathbf{e}$ to 0 and other indicators to 1. For example, to compute $\Pr(a_2, b_1)$, we set indicators λ_{a_1} and λ_{b_2} to 0, set other indicators to 1, and evaluate the reduced MLF: $\Pr(a_2, b_1) =$

$$\theta_{a_2}\theta_{b_1|a_2}\theta_{c_1|a_2,b_1} + \theta_{a_2}\theta_{b_1|a_2}\theta_{c_2|a_2,b_1} + \theta_{a_2}\theta_{b_1|a_2}\theta_{c_3|a_2,b_1}$$

As is obvious from the above example, the MLF has an exponential size. Yet the MLF can be factored into an AC whose size may not be exponential, leading one to formulate the exact inference problem as a problem of factoring MLFs into ACs. An AC is a DAG with internal nodes labeled with multiplications/additions and leaves labeled with variables and constants; see Figure 2.[1]

[1]Representations such as ADDs and their variations are more restricted representations of MLFs (they can be unfolded into ACs).

One way to factor an MLF into an AC is by encoding the MLF into CNF and then factoring the CNF. To illustrate the encoding scheme, consider again the MLF f in Figure 2 over real–valued variables a, b, c, d. The basic idea is to specify this MLF using a propositional theory that has exactly four models, one for each term in f. Specifically, the propositional theory $\Delta_f = V_a \wedge (V_b \Rightarrow V_d) \wedge (V_c \Rightarrow V_b)$ over Boolean variables V_a, V_b, V_c, V_d has exactly four models and encodes f as follows:

Model	V_a	V_b	V_c	V_d	*encoded term*
σ_1	true	false	false	false	a
σ_2	true	false	false	true	ad
σ_3	true	true	false	true	abd
σ_4	true	true	true	true	$abcd$

That is, model σ encodes term t since $\sigma(V_j) = true$ precisely when term t contains the real–valued variable j.

By factoring/compiling the CNF Δ_f as discussed in [Darwiche, 2004], one can immediately extract an AC representation of the MLF f in time and space proportional to the factored CNF [Darwiche, 2002]. We will discuss this process later, but we first provide more detail on the encoding step. We start here with the baseline encoding [Darwiche, 2002], which we refer to as PREV.

The CNF has one Boolean variable V_λ for each indicator variable λ, and one Boolean variable V_θ for each parameter variable θ. For brevity though, we will abuse notation and simply write λ and θ instead of V_λ and V_θ. CNF clauses fall into three sets. First, for each network variable X with domain $x_1, x_2, \ldots, x_n$, we have:

$$\underline{\text{Indicator clauses}}: \quad \lambda_{x_1} \vee \lambda_{x_2} \vee \ldots \vee \lambda_{x_n}$$
$$\neg\lambda_{x_i} \vee \neg\lambda_{x_j}, \text{ for } i < j$$

For example, variable C in Figure 1 generates:

$$\lambda_{c_1} \vee \lambda_{c_2} \vee \lambda_{c_3}, \ \neg\lambda_{c_1} \vee \neg\lambda_{c_2}, \ \neg\lambda_{c_1} \vee \neg\lambda_{c_3}, \ \neg\lambda_{c_2} \vee \neg\lambda_{c_3}$$

These clauses ensure that exactly one indicator variable for C appears in each term of the MLF. The remaining sets of clauses correspond to network parameters. In particular, for each parameter $\theta_{x_n|x_1,x_2,\ldots,x_{n-1}}$, we have:

$$\underline{\text{IP clause}}: \lambda_{x_1} \wedge \lambda_{x_2} \wedge \ldots \wedge \lambda_{x_n} \Rightarrow \theta_{x_n|x_1,x_2,\ldots,x_{n-1}}$$
$$\underline{\text{PI clauses}}: \theta_{x_n|x_1,x_2,\ldots,x_{n-1}} \Rightarrow \lambda_{x_i}, \text{ for each } i$$

For example, parameter $\theta_{c_1|a_1,b_1}$ in Figure 1 generates:

$$\lambda_{a_1} \wedge \lambda_{b_1} \wedge \lambda_{c_1} \Rightarrow \theta_{c_1|a_1,b_1}$$
$$\theta_{c_1|a_1,b_1} \Rightarrow \lambda_{a_1}, \ \ \theta_{c_1|a_1,b_1} \Rightarrow \lambda_{b_1}, \ \ \theta_{c_1|a_1,b_1} \Rightarrow \lambda_{c_1} \quad (1)$$

The models of this CNF are in one–to–one correspondence with the terms of the MLF. In particular, each model of the CNF will correspond to a unique network variable instantiation, and will set to true only those indicator and parameter variables which are compatible with that instantiation.

3 Encoding Techniques

The PREV encoding as discussed does not encode information about parameter values (local structure). However, it is quite

Table 1: —: jointree ran out of memory.

Network	Max Cluster	AC Inference Time (s)	Improvement Over JT
bm-5-3	23	0.0068	4,028
stud-3-2	25	0.0052	1,181
mm-4-8-3	26	0.0516	1,114
mm-3-8-5	54	0.6835	—
bm-22-3	104	4.7000	—
stud-6-24	233	13.0000	—

easy to encode information about determinism within this encoding. Consider Figure 1 and the parameter $\theta_{c_1|a_1,b_1} = 0$, which generates the four clauses in (1). These clauses ensure that the parameter $\theta_{c_1|a_1,b_1}$ appears in an MLF term iff that term contains the indicators λ_{a_1}, λ_{b_1} and λ_{c_1}. However, given that this parameter is known to be 0, all terms that contain this parameter must vanish. Therefore, we can suppress the generation of a Boolean variable for this parameter, and then replace the above clauses by a single clause: $\neg\lambda_{a_1} \vee \neg\lambda_{b_1} \vee \neg\lambda_{c_1}$. This clause has the effect of eliminating all CNF models which correspond to vanishing terms, those containing the parameter $\theta_{c_1|a_1,b_1}$.

Armed with determinism, the PREV encoding can produce impressive results when applied to networks with only binary variables, that contain small CPTs, and that contain large numbers of 0 parameters. For example, [Chavira *et al.*, 2004] reports on networks with such properties, generated from relational models, and Table 1 reviews some of these results. As is clear from the table, one gets exponential improvements over the standard jointree method, which does not take advantage of network determinism.[2] Similar results have also been reported in [Darwiche, 2002] with respect to Bayesian networks corresponding to digital circuits.

For Bayesian networks with local structure, large CPTs, yet no excessive determinism, the encoding of determinism alone may not be so effective. Table 2 lists a set of benchmark networks, some having variables with large cardinalities, others having very large CPTs, and where the amount of determinism is not necessarily excessive. Table 3 provides statistics on the CNFs generated for some of these networks, according to the PREV encoding, while also encoding determinism as discussed above. These CNFs are quite large, but the striking property they have is the large percentage (up to 99% in some cases) of Boolean variables that represent parameters versus those representing indicators. Some of these CNFs proved challenging to factor, some taking too long and others running out of memory. There are two key observations, however, that allowed us to handle these networks successfully, leading to significant improvements in both offline compile time and online inference time. We explain each of these in some detail next, but after providing an overview of the CNF factoring/compilation algorithm of [Darwiche, 2004].

[2]The technique of "zero compression" can be employed to exploit determinism in jointrees, but it requires inference on the full jointree first, which is prohibitive in this case.

Table 3: CNFs generated by PREV (determinism encoded).

Network	Vars	Parm Vars	Clauses	Literals
pathfinder	55229	54781	300576	821814
water	6630	6514	49367	152686
mildew	38540	37924	683552	1958952
munin1	9551	8556	49363	129358
munin4	48864	43216	247582	641839
diabetes	113527	108845	814412	2196008

3.1 How the CNF factoring/compilation process works

We provide in this section a sketch of the CNF factoring process. We note, however, that this section may be skipped on a first reading of the paper as it is not strictly needed for the following sections.

Consider again the CNF $\Delta_f = V_a \wedge (V_b \Rightarrow V_d) \wedge (V_c \Rightarrow V_b)$ from the previous section, and the MLF $f = a + ad + abd + abcd$ that it encodes. We now briefly describe the CNF factoring process which allows us to produce the AC shown in Figure 2. First, the output of the factoring process is shown on the left of Figure 3: It is a logical form known as negation normal form (NNF) which satisfies decomposability (conjunctions do not share variables), determinism (disjuncts must be logically incompatible), and smoothness (disjuncts must mention the same sets of variables). Such a factorization is generated using an exhaustive version of the DPLL procedure [Davis *et al.*, 1962]. In particular, the algorithm will pick a variable x in the CNF, will factor $\Delta|x$ and $\Delta|\neg x$ separately, and then combine the results into $x \wedge factor(\Delta|x) \vee \neg x \wedge factor(\Delta|\neg x)$. To improve performance, the algorithm keeps a cache that stores CNFs that have been factored and their factorizations and checks this cache before trying to factor a CNF Γ. Finally, before picking a variable x to split Γ on, the algorithm checks the CNF to see if it can be broken into disconnected components, say α_1 and α_2. In that case, the algorithm factors the components separately and combines their results into $factor(\alpha_1) \wedge factor(\alpha_2)$. The factoring algorithm we use [Darwiche, 2004], utilizes a decomposition tree (dtree) to manage this decomposition process. In particular, a dtree for a CNF is a binary tree whose leaves correspond to the CNF clauses. Moreover, each node in the dtree is associated with a set of variables whose instantiation is guaranteed to decompose the CNF into two independent components.

The above procedure generates NNFs that are decomposable and deterministic. Smoothness can be established easily by a postprocessing step. Given an NNF that satisfies the required properties, we can extract an AC by simply replacing conjunctions with multiplications, disjunctions with additions, and negative literals with the constant 1. Positive literals are replaced by the real–valued variables they encode. This decoding process is shown in Figure 3; see [Darwiche, 2002] for more details.

This factoring algorithm is indeed a logical version of the recursive conditioning (RC) algorithm [Darwiche, 2001b].[3]

[3]Similar algorithms have been used recently to solve probabilis-

Table 2: The networks with which we experimented.

Network	Max Clust	Vars	Card	Ave Card	Total Parms	Max CPT Parms	Ave CPT Parms	%Det	%DP
alarm	7.2	37	2-4	2.8	752	108	20	0.9	24.6
bm	20.0	1005	2-2	2.0	6972	8	7	99.6	100.0
diabetes	17.2	413	3-21	11.3	461069	7056	1116	78.2	17.6
hailfinder	11.7	56	2-11	4.0	3741	1188	67	15.7	26.9
mildew	21.4	35	3-100	17.6	547158	280000	15633	93.2	25.1
mm	23.0	1220	2-2	2.0	8326	8	7	98.7	75.0
munin1	26.8	189	1-21	5.3	19466	600	103	66.5	61.2
munin2	18.6	1003	2-21	5.4	83920	600	84	63.3	69.5
munin3	17.8	1044	1-21	5.4	85855	600	82	63.1	71.3
munin4	21.4	1041	1-21	5.4	98183	600	94	64.5	65.3
pathfinder	15.0	109	2-63	4.1	97851	8064	898	56.1	5.1
pigs	17.4	441	3-3	3.0	8427	27	19	56.2	23.9
students	22.0	376	2-2	2.0	2616	8	7	90.7	79.3
tcc4f	10.0	105	2-2	2.0	3236	512	31	0.4	35.6
water	19.9	32	3-4	3.6	13484	3072	421	54.0	57.0

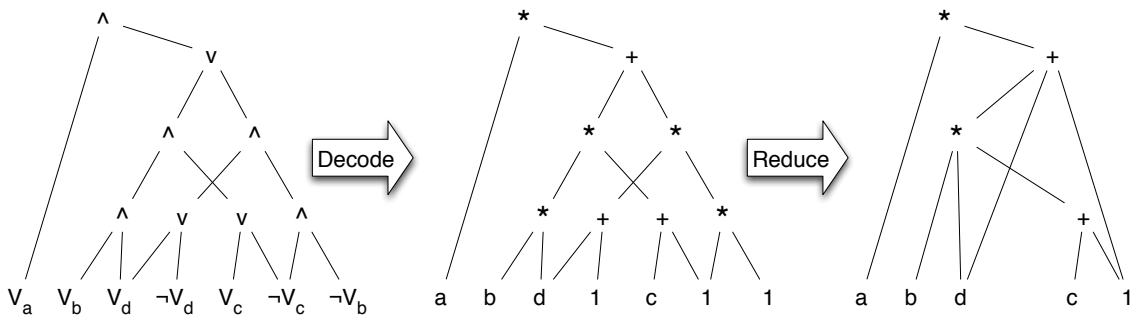


Figure 3: An NNF (left), its encoded AC (middle) and a simplification of the AC (right).

One can in principle use RC to compile a Bayesian network directly into an AC (bypassing the CNF encoding), but the approach we use allows us to capitalize on the state of the art in logical reasoning for handling determinism and CSI, even though it may incur more overhead in the factoring process. In particular, our CNF factoring algorithm uses unit resolution to propagate logical constraints; conflict directed backtracking to recover more efficiently when conditioning on variable settings that lead to contradictions; in addition to clause learning as a means for avoiding such contradictions early on in future conditioning. It also provides a more flexible framework for exploiting CSI through the use of two techniques. First, it can detect non–structural decompositions, that is, subproblems that become independent due to conditioning on specific variable values—such independence cannot be detected based only on structural considerations (network topology). This is done by removing clauses that become subsumed due to conditioning, therefore, disconnecting subsets of the CNF under specific variable values. Second, it uses a non–structural caching scheme which allows one to prove the equivalence of subproblems under specific variable values (therefore, avoiding multiple factorings of the same subproblem). Again, these equivalences cannot be proven if we were to only use structural considerations. Finally, the reduction to CNF allows one to more naturally accommodate other types of CPT structures (e.g., decision trees and rules) without the need for algorithmic change. The reduction to CNF factoring and the corresponding techniques lead to quite a bit of overhead in some cases, but this is justified since it will only be done once when compiling the network into an AC. If the application of these techniques lead to smaller ACs, the compile time can then be amortized over all online queries. We will illustrate this benefit more concretely in the experimental results section.

tic inference using #SAT [Bacchus *et al.*, 2003].

3.2 Encoding parameter equality / CSI

The CPT depicted in Figure 1 has 12 parameters, yet only 5 of these are distinct. Some of the equalities among parameters imply context–specific independence; others do not. For example, the equality between parameters in rows $4-6$ with those in rows $10-12$ imply that C is independent of A given $B = b_2$. However, the equality between parameters in rows 2 and 3 do not imply a CSI.

From a purely encoding viewpoint, one would clearly want to exploit parameter equality, at least to reduce the number of Boolean variables one must generate. Table 2 shows the extent to which parameter equality can help in this regard. In particular, the table reports as %DP the percentage of distinct parameters among non–extreme parameters. That is, the per-

centage of parameters that would remain if, for each CPT, we collapsed equal, non–extreme parameters into a single parameter. The dramatic example here is *pathfinder:* about half of its parameters are extreme, and among the other half, only about 5% are distinct within their CPTs. In addition to generating smaller CNFs, encoding parameter equality allows the compiler to run with less overhead, and to *generate smaller ACs since parameter equality provides more opportunities for factoring,* which immediately translates to gains in online inference.

A key observation here is that no two parameters in the same CPT can ever appear in the same MLF term, as they correspond to incompatible network instantiations. This observation suggests that we can use the same Boolean variable to represent multiple parameters, assuming that such parameters have equal values and appear in the same CPT. However, the idea will not work when applied to PREV. Consider again the CPT in Figure 1. If we use the same variable θ to represent parameters $\theta_{c_2|a_1,b_1}$ and $\theta_{c_3|a_1,b_2}$, which are both equal to $.5$, we would get the following two PI clauses in the CNF: $\theta \Rightarrow \lambda_{b_1}$ and $\theta \Rightarrow \lambda_{b_2}$, which is inconsistent with other clauses. More generally, PI clauses assert that a parameter implies the corresponding family instantiation. Therefore, if we simply use the same Boolean variable to represent equal parameters, we would be implying inconsistent family instantiations.[4]

The solution we adopt is to drop PI clauses from the encoding! Note that dropping PI clauses introduces additional (unintended) models into the CNF, allowing MLF terms which contain multiple parameters from the same CPT. These unintended models/terms, however, can be easily filtered during the decoding process given the following.

Theorem 1 *Consider a Bayesian network with n variables. Let Δ be a CNF encoding which includes the indicator, IP and PI clause sets, and let Γ be the CNF encoding which includes the indicator and IP clause sets only. Then Δ's models have cardinality $2n$ and are a subset of Γ's models. Moreover, if σ is a model of Γ but not Δ, then σ's cardinality is $> 2n$.*

Therefore, unintended models have a higher cardinality than original models (which all have the same cardinality). As it turns out, if Γ is an NNF which satisfies decomposability, determinism and smoothness, one can in linear time obtain another NNF Γ' whose models are exactly the minimum cardinality models of Γ and which satisfies the required properties [Darwiche, 2001a]. Therefore, we can safely drop PI clauses as long we minimize the resulting NNF before we decode the AC.

By including only indicator and IP clauses, we can now safely represent all equal parameters within the same CPT by a single Boolean variable in the CNF encoding. For the *Pathfinder* network for example, this drops the number of Boolean variables needed to represent non–extreme parameters from $42,946$ to $2,186$, a 95% reduction! Similar reductions are obtained for many other networks; see Table 2. As we show later, not only does this technique improve the compilation time, but can lead to significantly smaller ACs.

[4] A restricted case of encoding parameter equality was discussed in [Darwiche, 2002].

3.3 A more informed factoring algorithm

The CNF factoring algorithm employs two key techniques as discussed earlier. The first is variable splitting, which can be thought of as doing case analysis. The second is caching, so that one can avoid factoring the same CNF subset multiple times. Which variables the algorithm ends up splitting on can very much affect its running time, and the size of factorizations it generates. Moreover, the complexity of the caching scheme is proportional to the number of variables appearing in the cached CNF subset, as the state of such variables are used to generate keys that uniquely define CNFs. The following observations state interesting properties of our CNF encodings, which if passed to the factoring algorithm can significantly improve both the splitting and caching processes.

First, if two clauses share a parameter variable, then they must also share indicators over the same network variable. This property, and the presence of indicator clauses, allow the CNF factoring algorithm to restrict its splitting to indicator variables, which would be sufficient to decompose the problem into independent components (hence, no splitting/case analysis is needed on parameter variables). Second, given the structure of indicator and IP clauses, the state of indicator variables are sufficient to characterize the state of parameter variables. This property allows us to only involve indicator variables when generating CNF keys during the caching process. Both of the above optimizations can be exploited by simply identifying parameter variables to the factoring algorithm.

Another technique that we have used in some of the experiments involves the construction of a decomposition tree (dtree) for the given Bayesian network, and then converting it into a dtree for its CNF encoding. A dtree for a Bayesian network is simply a binary tree whose leaves correspond to the network CPTs [Darwiche, 2001b]. A dtree for a CNF is also a binary tree, but its leaves correspond to the CNF clauses. Since each clause in the CNF encoding is generated by a CPT, we can convert a network dtree into a CNF dtree by simply unfolding the dtree node corresponding to a CPT into a subtree whose leaves correspond to the clauses generated by that CPT. The main point of this technique is to more efficiently generate dtrees for very large CNF encodings that are generated by Bayesian networks with a small number of CPTs (this happens when the network contains very large CPTs).

3.4 Other Optimizations

Our CNF encodings utilize some additional enhancements, two of which are described next. First, we define a new type of clause, called an *eclause*, which has the same syntax as a regular clause but stronger semantics: it asserts that *exactly* one of it literals is true. We use eclauses for representing indicator clauses, therefore reducing the size of CNFs considerably in networks having multi–valued variables. Moreover, we outfit the DPLL procedure used in factoring the CNF to work directly with eclauses, without having to unfold them into regular clauses. For another optimization example, the indicators and parameters corresponding to the same state of a root variable are logically equivalent, making it possible to delete the parameter variables and the corresponding IP and PI clauses, which establish the equivalence.

4 Experimental Results

Experiments ran on a 1.6GHz Pentium M with 2GB of RAM, using the networks in Table 2. Two important columns in Table 2 are %Det, which is the percent of parameters that are equal to 0 or to 1 and %DP, which is the percent of non–extreme parameters remaining after collapsing equal parameters within the same CPT. These two values give an idea of the amounts of local structure in the form of determinism and possibly CSI. bm-5-3, mm-3-8-3, and stud-3-2 are networks on which PREV was already shown to perform well by only encoding determinism [Chavira *et al.*, 2004]. These networks contain only binary variables, are highly deterministic, and have small CPTs (no more than two parents per node). In contrast, the other networks contain variables with higher cardinalities and lesser degrees of determinism and, sometimes, very large CPTs. These networks came from various sources: http://www2.sis.pitt.edu/∼genie; Hughes Research Labs; and http://www.cs.huji.ac.il/labs/compbio/Repository.

The experiments serve to demonstrate three points. First, the new CNF encoding and the additional information we pass to the CNF factoring algorithm lead to significant improvement, both in the factoring time and the size of resulting factorizations. Table 4 (1st three columns) illustrates the improvement in factoring time, showing order of magnitude improvements in some cases and allowing us to factor some networks for the first time under the given memory constraints.

The second point illustrated by our experiments concerns the quality of factorizations (ACs) we obtain, compared to the ones embedded in jointrees. Recall that every jointree embeds an AC [Park and Darwiche, 2003], whose size depends only on the jointree size and, hence, is not dependent on local structure. Our experiments are therefore set to illustrate the extent to which local structure can help the factorization process. Table 4 illustrates significant improvements in AC size, as we obtained one to two orders of magnitude improvement on networks such as *Water*, *Pathfinder*, *Munin1*, and *Munin4*. For a more direct measure of improvement in online inference,[5] we generated sixteen random sets of evidence, and for each evidence set, we computed the marginal of each network variable using jointree propagation and then using AC evaluation/differentiation.[6] Table 4 shows the obtained improvements in online inference, which are similar to improvements in AC size.

Note that since the AC is compiled independently of evidence, the improvements apply for computing all marginals regardless of evidence. This is especially useful for tasks that apply a massive number of queries to Bayesian networks, including parameter estimation algorithms (e.g., EM) given known local structure in the form of determinism and CSI, and MAP algorithms based on branch&bound search.

[Poole and Zhang, 2003] present another approach for exploiting local structure, by providing a refinement on variable elimination (VE). Their approach does not involve a compilation step, and their results are sensitive to the given queries, so a direct comparison between the two approaches is not too meaningful. Yet, we mention here that on a re-parameterized version of Water (to introduce local structure), [Poole and Zhang, 2003] show a factor of 4 improvement over standard VE (total time over all queries; the speedup was more or less depending on the query). Water is the only network from Table 4 that [Poole and Zhang, 2003] report on. Note that the exploitation of local structure can incur overhead that may not be justifiable unless the savings due to local structure are significant enough (not all of the cases in Table 4 lead to significant savings). This appears to be less an of an issue in our approach, since the overhead is pushed into the compilation step. However, in [Poole and Zhang, 2003] which does not involve a compilation step, this overhead is incurred in every query which may or may not lead to overall savings depending on the query—being query specific, however, may sometimes pay off quite significantly, since work performed can sometimes be simplified given specific evidence and specific query variables.

Table 5: Effect of local structure on AC size (edge count).

Local structure	Pathfinder	Water	Munin4
Det/Equal Parms	42,810	134,140	5,762,690
Det only	130,380	138,501	9,997,267
Equal Parms only	200,787	11,111,104	17,612,036
No local structure	784,330	15,305,634	—
Jointree	981,178	13,777,166	116,136,985

Our final point regards the effect of local structure on the quality of factorizations. Consider Table 5, which shows AC sizes under different encodings of local structure. First, it is obvious that encoding local structure is responsible for the significant improvements on these networks. Second, in *Water*, determinism appears to be the main responsible factor. However, in *Pathfinder* and *Munin4*, parameter equality alone (without determinism) is sufficient to bring about most of the reported improvements, even though determinism alone can have a similar effect too. Note that there is some overlap between determinism and parameter equality since by encoding determinism (0 parameters), one is effectively collapsing all 0 parameters (applying implicit parameter equality). The results for *Pathfinder* and *Munin4* are therefore not surprising, suggesting possibly that parameter equality is more fundamental than determinism for these networks.

5 Conclusion

We considered the problem of compiling Bayesian networks into ACs. Our aim was to efficiently compile networks having local structure, yet large CPTs and no excessive determinism. We proposed a new encoding scheme that facilitates the representation of local structure in the form of parameter equality, and identified some of its properties that improve compile time. Our results demonstrate significant improvements in both offline/compile and online/inference time, leading to orders of magnitude improvement in online inference.

[5] Online inference time is affected by the number of AC nodes too (not reported), and by the structure of ACs which affects locality of reference (the ACs embedded in jointrees have better locality).

[6] AC differentiation provides more information that just marginals [Darwiche, 2003].

Table 4: Comparing the new and prev encodings with the jointree baseline.

Network	Offline Compile Time (s)			AC Edge Count			Online Inference Time (s)		
	PREV	NEW	Improv.	Jointree	NEW	Improv.	Jointree	NEW	Improv.
alarm	0.93	0.52	1.8	4,804	2,686	1.8	0.07	0.01	6.4
bm-5-3	2.51	1.11	2.3	86,532,336	18,693	4,629.1	165.15	0.09	1,814.8
diabetes	—	2,269.05	—	40,673,307	13,585,023	3.0	37.09	16.27	2.3
hailfinder	2.26	0.86	2.6	36,342	15,687	2.3	0.13	0.06	2.2
mildew	—	7,483.80	—	16,171,408	2,123,309	7.6	13.39	3.35	4.0
mm-3-8-3	7.44	1.87	4.0	98,044,208	263,835	371.6	248.15	0.21	1,181.6
munin1	—	1,534.97	—	1,047,211,866	30,620,744	34.2	1,321.68	44.91	29.4
munin2	3,248.42	225.46	14.4	24,281,678	4,791,974	5.1	28.32	6.59	4.3
munin3	1,553.43	151.72	10.2	14,333,412	2,436,598	5.9	18.69	3.65	5.1
munin4	2,440.30	677.92	3.6	116,136,985	5,762,690	20.2	137.94	7.70	17.9
pathfinder	226.37	20.36	11.1	981,178	43,064	22.8	1.68	0.07	23.7
pigs	110.10	17.84	6.2	2,347,299	1,302,215	1.8	3.45	1.60	2.2
students-3-2	1.53	0.82	1.9	37,799,472	27,292	1,385.0	55.98	0.07	799.7
tcc4f.obfuscated	4.11	1.15	3.6	29,064	22,284	1.3	0.14	0.05	2.8
water	34.82	4.83	7.2	13,777,166	134,142	102.7	22.68	0.21	107.5

Acknowledgments

We thank the reviewers for commenting on an earlier version of this paper. This work has been partially supported by NSF grant IIS-9988543 and MURI grant N00014-00-1-0617.

References

[Bacchus *et al.*, 2003] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *FOCS*, pages 340–351, 2003.

[Boutilier *et al.*, 1996] Craig Boutilier, Nir Friedman, Moisés Goldszmidt, and Daphne Koller. Context–specific independence in bayesian networks. In *UAI'96*, pages 115–123, 1996.

[Chavira *et al.*, 2004] M. Chavira, A. Darwiche, and M. Yaeger. Compiling relational bayesian networks for exact inference. In PGM-2004, pages 49–56, 2004.

[Davis *et al.*, 1962] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Journal of the ACM*, (5)7:394–397, 1962.

[Darwiche, 2001a] Adnan Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4):608–647, 2001.

[Darwiche, 2001b] Adnan Darwiche. Recursive conditioning. *Artificial Intelligence*, 126(1-2):5–41, 2001.

[Darwiche, 2002] Adnan Darwiche. A logical approach to factoring belief networks. In *Proceedings of KR*, pages 409–420, 2002.

[Darwiche, 2003] Adnan Darwiche. A differential approach to inference in bayesian networks. *JACM*, 50(3):280–305, 2003.

[Darwiche, 2004] Adnan Darwiche. New advances in compiling CNF to decomposable negational normal form. In *ECAI'2004*.

[Jensen *et al.*, 1990] F. V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, 4:269–282, 1990.

[Park and Darwiche, 2003] James Park and Adnan Darwiche. A differential semantics for jointree algorithms. *NIPS 15*, volume 1, pages 299–307. MIT Press, 2003.

[Poole and Zhang, 2003] D. Poole and N.L. Zhang. Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence*, 18:263–313, 2003.

[Shenoy and Shafer, 1986] Prakash P. Shenoy and Glenn Shafer. Propagating belief functions with local computations. *IEEE Expert*, 1(3):43–52, 1986.

The Inferential Complexity of Bayesian and Credal Networks

Cassio Polpo de Campos[1,2] **Fabio Gagliardi Cozman**[1]
[1] Universidade de São Paulo - Escola Politécnica
[2] Pontifícia Universidade Católica de São Paulo
cassio@pucsp.br, fgcozman@usp.br

Abstract

This paper presents new results on the complexity of graph-theoretical models that represent probabilities (Bayesian networks) and that represent interval and set valued probabilities (credal networks). We define a new class of networks with bounded width, and introduce a new decision problem for Bayesian networks, the maximin a posteriori. We present new links between the Bayesian and credal networks, and present new results both for Bayesian networks (most probable explanation with observations, maximin a posteriori) and for credal networks (bounds on probabilities a posteriori, most probable explanation with and without observations, maximum a posteriori).

1 Introduction

This paper builds a picture of inferential complexity in graph-theoretical models of uncertainty that goes significantly beyond existing results. We focus on Bayesian and credal networks — the former is a purely probabilistic model, while the latter admits interval and set valued probabilities and generalizes several theories of uncertainty. There already is quite a solid understanding about the inferential complexity of Bayesian networks; we add to this picture a new class of networks that strengthens existing results, and a new type of problem, maximin a posteriori (MmAP), that can be of interest in game-theoretic settings. Our main contributions are related to credal networks, as little is known at this point concerning their inferential complexity: we present new results concerning computation of probability bounds, most probable explanations (MPE) and maximum a posteriori (MAP). We show that, rather surprisingly, the MPE problem without observations in credal networks of bounded width is polynomial, while the MPE problem with observations and the MAP problem are Σ_2^p-complete for credal networks.

We also strengthen the connection between Bayesian and credal networks by showing links between the computation of probability bounds, the MAP and the MmAP problems. Our results suggest that moving from point to interval/set valued probabilities takes us "one step up" in terms of complexity classes.

Section 2 presents a few definitions and, most importantly, tries to justify our interest in credal networks. Section 3 describes the problems we are interested in. Section 4 contains our new results; in particular, Table 1 offers a summary of our contributions. Section 5 concludes the paper and suggests future research.

2 Bayesian and Credal Networks

A Bayesian network (or `BN`) represents a *single* joint probability density over a collection of random variables. We assume throughout that variables are categorical; variables are uppercase and their assignments are lowercase.

Definition A Bayesian network is a pair $(G, \mathbb{P})$, where: $G = (V_G, E_G)$ is a directed acyclic graph, with V_G a collection of vertices associated to random variables $\mathbb{X}$ (a node per variable), and E_G a collection of arcs; $\mathbb{P}$ is a collection of conditional probability densities $p(X_i|\mathrm{pa}(X_i))$ where $\mathrm{pa}(X_i)$ denotes the parents of X_i in the graph ($\mathrm{pa}(X_i)$ may be empty), respecting the relations of E_G.

In a `BN` every variable is independent of its nondescendants nonparents given its parents (Markov condition). This structure induces a joint probability density by the expression $p(X_1, \ldots, X_n) = \prod_i p(X_i|\mathrm{pa}(X_i))$. Given a `BN`, E denotes the set of observed variables in the network (the *evidence*); e denotes the observed value of E.

We consider the class of networks with `bounded induced-width` (or `BIW`); that is, networks where the subjacent graph (obtained removing arc's directions) of G has maximum degree and induced-width bounded by $\log(f(s))$. We define $f(s)$ as a polynomial function in the size s of input (this size is evaluated over all information needed to specify the problem). We also consider the class of `polytrees` (or `PT`); that is, a `BIW` network where the subjacent graph of G has no cycles. Otherwise, a network is said `multiply-connected`. Note that our definition of `BIW` networks is not, as usually done, based on *fixed* induced-width; rather, we allow the induced-width to vary with the network size.

Credal networks generalize Bayesian networks by allowing each variable to be associated with *sets* of joint probability measures rather than single probability measures [Cano *et al.*, 1993; Cozman, 2000a]. Such graph-theoretical models can be

viewed as Bayesian networks with relaxed numerical parameters; they can be used to study robustness of probabilistic models, to investigate the behavior of groups of experts, or to represent incomplete or vague knowledge about probabilities. Sets of probability measures are sufficiently powerful to represent belief functions, possibility measures, qualitative probabilities, and probabilistic logic statements, thus offering a general language that can convey many models of interest in artificial intelligence [Walley, 1996].

A set of probability distributions is called a *credal set* [Levi, 1980]. A credal set for X is denoted by $K(X)$. A conditional credal set is a set of conditional distributions, obtained applying Bayes rule to each distribution in a credal set of joint distributions [Walley, 1991]. Given a credal set $K(X)$, the *lower probability* and the *upper probability* of event A are defined respectively as $\underline{P}(A) = \min_{P(X)\in K(X)} P(A)$ and $\overline{P}(A) = \max_{P(X)\in K(X)} P(A)$. We assume that, given a credal set, finding a lower/upper probability is a polynomial operation.

Definition A credal network is a pair $(G, \mathbb{K})$, where: $G = (V_G, E_G)$ is a directed acyclic graph, with V_G a collection of vertices associated to random variables $\mathbb{X}$ (a node per variable), and E_G a collection of arcs; $\mathbb{K}$ is a collection of conditional credal sets $K(X_i|\mathrm{pa}(X_i))$, respecting the relations implied by E_G.

We assume a Markov condition in credal networks: every variable is independent of its nondescendants nonparents given its parents. In this paper we adopt the concept of *strong independence*; that is, the joint credal set represented by a credal network is a set where each vertex factorizes as a Bayesian network [Cozman, 2000b]. Thus we can really view a credal network as a set of Bayesian networks, all with identical graphs. Credal networks can also be classified as polytree, bounded induced-width or multiply-connected.

3 Reasoning

In this section we present decision versions of the inferences we focus in this paper.

Definition `BN-Pr`: given a Bayesian network $(G, \mathbb{P})$, evidence $E = e$ with $E \subseteq \mathbb{X}$, a query variable $Q \in \mathbb{X} \setminus E$ and its category q, and a rational number r, is $P(q|e) > r$?

Note the following distinction between MPE problems with or without evidence; the reason for this will be indicated later:

Definition `BN-MPE`: given a Bayesian network $(G, \mathbb{P})$, evidence $E = e$ with $E \subseteq \mathbb{X}$ and a rational number r, is there an instantiation x for $\mathbb{X} \setminus E$ such that $P(x, e) > r$?

Definition `BN-MPEe`: given a Bayesian network $(G, \mathbb{P})$, evidence $E = e$ with $E \subseteq \mathbb{X}$ and a rational number r, is there an instantiation x for $\mathbb{X} \setminus E$ such that $P(x|e) > r$?

Definition `BN-MAP`: given a Bayesian network $(G, \mathbb{P})$, evidence $E = e$ with $E \subseteq \mathbb{X}$, a set $Q \subseteq \mathbb{X} \setminus E$ and a rational number r, is there an instantiation q for Q such that $P(q|e) > r$?

We introduce the *maximin a posteriori* problem, which may be of interest in applications involving game-theoretic behavior with maximizers and minimizers [Kakade *et al.*, 2001]:

Definition `BN-MmAP`: given a Bayesian network $(G, \mathbb{P})$, some evidence $E = e$ with $E \subseteq \mathbb{X}$, the sets $A \subseteq \mathbb{X} \setminus E$ and $B \subseteq \mathbb{X} \setminus E$, with $A \cap B = \emptyset$ and a rational number r, is there an instantiation a for the A variables such that $\min_b P(a, b|e) > r$?

The corresponding problems in `CN` are now defined.

Definition `CN-Pr`: given a credal network $(G, \mathbb{K})$, evidence $E = e$ with $E \subseteq \mathbb{X}$, a query variable $Q \in \mathbb{X} \setminus E$ and its category q and a rational number r, is $\overline{P}(q|e) > r$?

In this definition we use upper queries, but lower queries may be of interest too. We can show that both queries lead to identical complexity results:

Lemma 1 *Evaluating marginal lower probabilities in* `CN` *is as hard as evaluating marginal upper probabilities.*

Proof Suppose we have a `CN-Pr` with marginal query $Q = q$. The calculation of $\underline{P}(q|e)$ can be done by inserting a binary child Q' to Q, where $P(q'|Q) = 1$ if $Q \neq q$ and 0 otherwise. Now $\overline{P}(q'|e) = \max \sum_{Q\neq q} P(Q|e) = 1 - \underline{P}(q|e)$. □

We now define *maximin* versions of MPE and MAP in credal networks (we could alternatively define maximax versions for these problems, with possibly different complexities).

Definition `CN-MAP`: given a credal network $(G, \mathbb{K})$, some evidence $E = e$ with $E \subseteq \mathbb{X}$, a set $Q \subseteq \mathbb{X} \setminus E$ and a rational number r, is there an instantiation q for the Q variables such that $\underline{P}(q|e) > r$?

`CN-MPEe` is obtained when `CN-MAP` has $Q = \mathbb{X} \setminus E$. `CN-MPE` is `CN-MPEe` without evidence, that is, $Q = \mathbb{X}$. We use abbreviations to refer to these problems (e.g. `PT-CN-Pr` is the belief updating problem in a polytree credal network).

4 Complexity results

Table 1 summarizes relevant complexity results.[1] We start by explaining the origin of the results in this table (numbering matches the numbers in the table):

1. [Dechter, 1996] describes algorithms with exponential time complexity on the induced-width of an elimination order; [Eyal, 2001] shows how to obtain a constant-factor approximation to optimal order in polynomial time for networks with bounded degree and bounded induced-width by $\log(f(s))$. Note that the result here is sightly different from the one in [Dechter, 1996]; we refer to the actual induced width of the graph, not the induced width of an ordering.
2. [Roth, 1996] shows complexity of functional version, [Litmman *et al.*, 2001] takes the decision version.
3. [Shimony, 1994] shows (by reduction from the vertex cover problem) that `BN-MPE` is NP-Complete, while Theorem 2 shows that `BN-MPEe` is PP-Complete.

[1] We assume that the reader is familiar with notions of complexity theory; for an introduction see [Papadimitriou, 1994]. Polynomial time means polynomial time in the size of input. A *reduction* means a polynomial time reduction; when a problem *is solvable by* another, there is a *reduction* from the former to the latter.

Problem	*Polytree*	*Bounded induced-width*	*Multiply-connected*
BN−Pr	Polynomial (1)	Polynomial (1)	PP-Complete (2)
BN−MPE	Polynomial (1)	Polynomial (1)	NP-Complete (3)
BN−MPEe	Polynomial (1)	Polynomial (1)	PP-Complete (3)
BN−MAP	NP-Complete (4)	NP-Complete (4)	NP^{PP}-Complete (5)
BN−MmAP	Σ_2^P-Complete (6)	Σ_2^P-Complete (6)	NP^{PP}-Hard (7)
CN−Pr	NP-Complete (8)	NP-Complete (8)	NP^{PP}-Complete (9)
CN−MPE	Polynomial (10)	Polynomial (10)	NP-Complete (11)
CN−MPEe	Σ_2^P-Complete (12)	Σ_2^P-Complete (12)	Σ_2^P-Hard and PP-Hard (13)
CN−MAP	Σ_2^P-Complete (12)	Σ_2^P-Complete (12)	NP^{PP}-Hard (7)

Table 1: Complexity results; numbers in parenthesis indicate the item that discusses the result.

4. [Park, 2002] reduces MAXSAT problem to BN−MAP in polytrees, a result that can be extended to BIW networks as both problems belong to NP (given the polynomial nature of BN−Pr in BIW networks).
5. [Park and Darwiche, 2004] by reduction from E−MAJSAT.
6. Theorem 8.
7. The complexity of BN−MAP implies it.
8. Theorem 3.
9. [Cozman *et al.*, 2004] by a reduction from E−MAJSAT.
10. Theorem 5.
11. Theorem 6.
12. Theorem 7.
13. BN−MPE and PT−CN−MPEe ensure it.

In the context of Bayesian networks, the difference between MPE and MPEe may seem academic, because any most probable explanation can be found with BN−MPE. However the same is not true for credal networks, where one cannot find a most probable explanation with evidence by simply running a version of MPE — and note that CN−MPE and CN−MPEe do display non-trivial differences. In fact, these differences were our motivation for differentiating MPE from MPEe. The following theorem clarifies the difference between these problems for Bayesian networks.

Theorem 2 BN−MPEe *is PP-Complete.*

Proof Pertinence is obtained from the fact that, after making a PP query to find $P(e)$ (this query is made once), we can decide whether a given instantiation x has $P(x|e) > r$ in linear time, by multiplying the probabilities. To show hardness, we reduce the decision problem #3SAT($\geq 2^{n/2}$) to it, which is PP-Complete [Bailey *et al.*, 2001] and can be stated as: *Given a set of boolean variables $X = \{X_1, \ldots, X_n\}$ and a 3CNF formula $\phi(X)$ with clauses $\{C_1, \ldots, C_m\}$, is $\phi(X)$ satisfied by at least $2^{n/2}$ of the instantiations of X?*

We construct a BN with binary nodes $X_1, \ldots, X_n$ (x_i and $\overline{x_i}$ are the categories) and $C_1, \ldots, C_m$ (c_i and $\overline{c_i}$), where X_i has no parents and uniform prior and C_i has three parents (the variables contained in the clause) with probabilities respecting the truth table for the clause. Furthermore, we insert a dummy binary node Y appearing non-negated in every clause (there will be 2^n instantiations with $\{Y = y\}$ satisfying ϕ; this ensures that the formula is satisfiable). Now we solve the BN−MPEe problem with queries $X_1, \ldots, X_n, Y$ and evidence $C_i = c_i$ for all $1 \leq i \leq m$ (indicating that all C_i are **true**). Then $P(X, Y|c_1, \ldots, c_m)$ is equal to

$$
\begin{aligned}
&= \frac{P(c_1, \ldots, c_m|X_1, \ldots, X_n, Y)\, P(X_1, \ldots, X_n, Y)}{P(c_1, \ldots, c_m)} \\
&= \frac{\frac{1}{2^{n+1}} P(c_1, \ldots, c_m|X_1, \ldots, X_n, Y)}{\sum_{X'_1, \ldots, X'_n, Y'} \left[P(c_1, \ldots, c_m|X'_1, \ldots, X'_n, Y') \frac{1}{2^{n+1}}\right]} \\
&= \frac{1}{\#sats} \text{ if } X_1, \ldots, X_n, Y \text{ satisfies } \phi \text{ and } 0 \text{ otherwise,}
\end{aligned}
$$

where $\#sats$ is the total number of satisfying instantiations. So, $\max P(X, Y|c_1, \ldots, c_m) \leq \frac{1}{2^n + 2^{n/2}}$ implies that formula $\phi(X)$ is satisfied by at least $2^{n/2}$ of all X instantiations. Note that if $\max P(X, Y|c_1, \ldots, c_m) = \alpha$ and $\alpha > \frac{1}{2^n + 2^{n/2}}$, then $P(X, Y|c_1, \ldots, c_m) = \alpha$ for all satisfying instantiations, which implies that there are less than $2^{n/2}$ instantiations of X satisfying $\phi(X)$. □

The hardness of PT−CN−Pr was stated by [da Rocha and Cozman, 2002], but the proof there was flawed, and the central argument used zero probabilities and vertex-based description in an essential way. The following proof corrects these difficulties.

Theorem 3 PT−CN−Pr *and* BIW−CN−Pr *are NP-Complete.*

Proof Pertinence of BIW−CN−Pr (which ensures pertinence of PT−CN−Pr) is immediate, as choosing a vertex of each credal set we have a BIW−BN−Pr problem to solve, which is polynomial. To show hardness we reduce the MAX−3−SAT problem to PT−CN−Pr. It can be formulated as follows: *Given a set of boolean variables $\{X_1, \ldots, X_n\}$, a 3CNF formula with clauses $\{C_1, \ldots, C_m\}$ and an integer $0 \leq k \leq m$, is there an assignment for the variables that satisfies at least k clauses?* Initially we remove all clauses that have both x_i and $\overline{x_i}$ and decrement k for each elimination (because those clauses are already satisfied). For each variable X_i we construct two nodes, namely X_i and S_i. The former is binary, has no parents and represents the state of X_i; the probabilities $P(X_i = x_i)$ and $P(X_i = \overline{x_i})$ are in $[\varepsilon, 1 - \varepsilon]$ ($0 < \varepsilon < \frac{1}{m+1}$ is a small constant). The latter may assume $m + 1$ categories

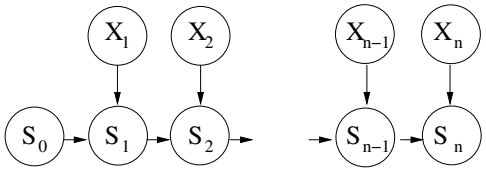


Figure 1: Polytree used in the network of Theorem 3.

(from 0 to m), has S_{i-1} and X_i as parents and is defined by

$$\begin{aligned} P(S_i = c|S_{i-1} = c, x_i) &= 0 \text{ if } x_i \in C_c, \text{ or 1 otherwise} \\ P(S_i = c|S_{i-1} = c, \overline{x_i}) &= 0 \text{ if } \overline{x_i} \in C_c, \text{ or 1 otherwise} \\ P(S_i = c|S_{i-1} \neq c, X_i) &= 0 \text{ for } X_i \in \{x_i, \overline{x_i}\}, \end{aligned}$$

for $c \neq 0$. When $S_i = 0$, we have:

$$\begin{aligned} P(S_i = 0|S_{i-1} = c, X_i) &= 1 - P(S_i = c|S_{i-1} = c, X_i) \\ P(S_i = 0|S_{i-1} = 0, X_i) &= 1 \text{ for } X_i \in \{x_i, \overline{x_i}\}. \end{aligned}$$

The rules above guarantee coherency in probabilities; note that we include a dummy node S_0 with $P(S_0 = c) = \frac{1}{m+1}$ for all c. Now, consider $P(S_n = c)$ for $c \neq 0$. Note that $P(S_n = c) = P(S_0 = c) \prod_i P(S_i = c|S_{i-1} = c)$ because, for $c \neq 0$, every time $\{S_i = c\}$ and $\{S_{i-1} \neq c\}$ appear together we are led to zero. Let $A_{c,i}$ be defined as follows:

$$A_{c,i} = \sum_{X_i \in \{x_i, \overline{x_i}\}} P(S_i = c|S_{i-1} = c, X_i)\, P(X_i)\,.$$

We get $P(S_n = c) = \frac{1}{m+1} \prod_{i \in \{1,\ldots,n\}} A_{c,i}$. Note that $A_{c,i}$ may assume three values: $A_{c,i} = 1$ if X_i does not influence C_c; $A_{c,i} = \varepsilon$ if X_i satisfies C_c and $A_{c,i} = 1 - \varepsilon$ if X_i does not satisfy C_c. We may conclude that if $P(S_n = c) \leq \alpha = (1-\varepsilon)^2\varepsilon$, then some X_i satisfied C_c. Furthermore, we know that if C_c was not satisfied, then $P(S_n = c) = \beta = (1-\varepsilon)^3$. Note that $\alpha < \beta$. To find out how many clauses were not satisfied, we have to sum over all categories of S_n, obtaining $P(S_n = 0) = 1 - \sum_{c \in \{1,\ldots,m\}} P(S_n = c)$ and thus $\overline{P}(S_n = 0)$ minimizes this sum. We define

$$r_h = [(m+1)(1 - \overline{P}(S_n = 0)) - h\beta]/(m-h).$$

and then calculate $r_0, r_1, \ldots$ until $r_h \leq \alpha$ or $h = m-1$. We know that $1 - \overline{P}(S_n = 0)$ is the minimum sum of all $P(S_n = c)$, for $c \neq 0$. This sum is composed by two types of terms: those which are equal to β and those which are less than or equal to α. So what we are verifying with r_h is whether there are more than h terms of the sum that are equal to β or not. The last thing should be noted is that $\varepsilon < \frac{1}{m+1}$ ensures that if just one term of the sum equals to β, then the sum is greater than $m\alpha$, that is, if the sum is composed by $m-1$ terms equal to ε^3 and just one equal to β, it must sum greater than $m\alpha$, because one clause was not satisfied. $\varepsilon < \frac{1}{m+1}$ ensures that everywhere it is necessary (for any h). Thus the inference $\overline{P}(S_n = 0)$ solves MAX-3-SAT problem. If $r_h > \alpha$ for all $0 \leq h \leq m-1$, then no clause was satisfied. Otherwise h counts how many clauses were not satisfied. □

Because of the reduction from MAX-3-SAT, we can state that there is no polynomial time approximation scheme for PT-CN-Pr (or BIW-CN-Pr) unless P=NP.

Remark The proof still holds if we substitute all ε by zero; the proof becomes simpler but depends on events of zero probability, which may be inconvenient, as pointed out by [Zaffalon, 2003]. Note also that the proof can be rewritten using inequalities instead of vertices, because all credal sets are in binary nodes (pertinence and hardness still hold).

It is known that CN-Pr is solvable by BN-MAP, by conducting a *CCM* transform in a credal network [Cozman, 2000a]. The following lemma presents the reverse connection between inferences in Bayesian and credal networks.

Lemma 4 BN-MAP *is solvable by* CN-Pr *with joint queries without changing the topology of the network among the three types defined.*

Proof Suppose $X_1, \ldots, X_n$ are the MAP variables. Add a binary child X_i' to each X_i with $P(X_i'|X_i) \in [0,1]$ and the constraint $\sum_{X_i} P(x_i'|X_i) = 1$. Now we have

$$\max_{X_1,\ldots,X_n} P(X_1, \ldots, X_n|e) = \overline{P}(x_1', \ldots, x_n'|e)$$

After evaluating $\overline{P}(x_1', \ldots, x_n'|e)$, we just have to look at each X_i' node and set X_i according to which of the $P(x_i'|X_i)$ is equal to one (exactly one will be). □

It should be noted that joint queries are not more difficult than single marginal queries: we have that CN-Pr with joint queries is still NP$^{\text{PP}}$-Complete and BIW-CN-Pr with joint queries is still NP-Complete.

Theorem 5 PT-CN-MPE *and* BIW-CN-MPE *are solvable in polynomial time.*

Proof It is enough to realize that the inner min of the BIW-CN-MPE query $\max_x \min_{P \in K(\mathbb{X})} \prod_i P(x_i|\text{pa}(x_i))$ factorizes, as the network is locally specified (note that x_i's are consistent with the observation e). The optimization becomes $\max_x \prod_i \underline{P}(x_i|\text{pa}(x_i))$, which is equivalent to BIW-BN-MPE. □

Theorem 6 CN-MPE *is NP-Complete.*

Proof Hardness is immediate, because BN-MPE is NP-Complete and can be trivially transformed to a CN-MPE (we just have to use credal networks composed by single probability densities). Pertinence is reached because, given an assignment x to the variables, the value of $\underline{P}(x)$ is given by $\prod_i \underline{P}(x_i|\text{pa}(x_i))$. This holds because each credal set $K(x_i|\text{pa}(x_i))$ is locally specified. □

Theorem 7 PT-CN-MPEe, BIW-CN-MPEe, PT-CN-MAP *and* BIW-CN-MAP *are all Σ_2^P-Complete.*

Proof Pertinence of them is immediate as BIW-CN-MAP belongs to Σ_2^{P} (given the MAP variables, we get a BIW-CN-Pr to solve). To see hardness we reduce to PT-CN-MPEe a version of QSAT_2 that is Σ_2^{P}-Complete: *Given a set of variables $X_1, \ldots, X_n$, an integer $0 < k \leq n$ and a boolean 3CNF formula $\phi(X)$ over these variables, is it true that, for all instantiations to the first k variables, there is an instantiation of the remaining $n-k$ that satisfies $\phi(X)$?*

Initially we construct a network similar to that of Theorem 3. The variables $X_1, \ldots, X_k$ are defined the same way as there. The variables $X_{k+1}, \ldots, X_n$ become ternary,

assuming the categories $x_i, \overline{x_i}, o_i$. Their probabilities are: $P(X_i = o_i) = \varepsilon$ and $P(X_i = x_i)$, $P(X_i = \overline{x_i})$ belong to $[0, 1-\varepsilon]$, where $0 < \varepsilon < \frac{1}{m+2}$ is a small constant. The probabilities of S_i given its parents are the same as there, except when $i > k$. In these cases we have $P(S_i = c|S_{i-1} = c, o_i) = 0$, for $c \neq 0$ (the case when $c = 0$ remains the same, that is, equals to 1).

Furthermore, we add a dummy node Q with S_n as parent and $P(q|S_n = c) = 0$ for $c \neq 0$ and 1 otherwise. We will solve the PT-CN-MPEe problem $\max \underline{P}(X, S|q)$, where $S = \{S_0, \ldots, S_n\}$ and $X = \{X_1, \ldots, X_n\}$. Let X^- be $\{X_1, \ldots, X_k\}$ and X^+ be $\{X_{k+1}, \ldots, X_n\}$; then

$$\underline{P}(X^-, X^+, S|q) = \frac{P(q|S)\,P(S|X)\,P(X^-)\,P(X^+)}{P(q)}.$$

First note that the given q forces $S_n = 0$ to get a non-zero probability. Furthermore, for all instantiations x^-, x^+, s of the variables, there is another instantiation with $s' = \{\forall i S_i = 0\}$ that has its probability greater than the former, that is, $\underline{P}(x^-, x^+, s'|q) > \underline{P}(x^-, x^+, s|q)$. This holds because the S_i nodes are not credal and the conditional probabilities $P(S_i = 0|S_{i-1} = 0)$ are equal to 1, for all i. Thus we know that the solution of the MPEe problem will be attained in an instantiation where all S_i are set to 0. Besides that, we have that choosing the category o_i for all X^+ variables lead us to greater probabilities than if we choose any other. That is,

$$\max_{X^-, X^+, S} \underline{P}(X^-, X^+, S|q) = \max_{X^-} \underline{P}(X^-, o, s|q)\,,$$

where o denotes $\{X_i = o_i \text{ for } i \in \{k+1, \ldots, n\}\}$ and s denotes $\{\forall i\ S_i = 0\}$. If we choose a category different from o_i whenever possible, the maximum probability would not reach the same value (in fact it will be zero, because $P(X_i \neq o_i) \in [0, 1-\varepsilon]$, and thus it may assume value zero).

$$\underline{P}(X^-, o, s|q) = \frac{\frac{1}{m+1}\varepsilon^{n-k}P(X^-)}{1 - \sum_{c\neq 0} P(S_n = c)}.$$

When finding $\underline{P}(X^-, o, s|q)$, the numerator $P(X^-)$ will become ε^k, because any solution that does not make $P(X^-) = \varepsilon^k$ will not be a minimum for $\underline{P}(X^-, o, s|q)$ (remember that $\varepsilon < \frac{1}{m+2}$). This holds because just one X^- variable using the extreme point $(1-\varepsilon)$ instead of ε is enough to make $\underline{P}(X^-, o, s|q)$ too large:

$$\frac{\frac{1}{m+1}\varepsilon^{n-k}\varepsilon^{k-1}(1-\varepsilon)}{1-R} > \frac{\frac{1}{m+1}\varepsilon^{n-k}\varepsilon^{k}}{1-R'}$$

where R and R' are any possible values for the sum $\sum_{c\neq 0} P(S_n = c)$ that appears in the denominator (note that these sums are restricted in $[0, \frac{m}{m+1}]$ by the probabilities of the network).

Summarizing, all S variables in the solution of MPEe are set to zero, all X^+ variables are set to o, and the instantiation chosen for X^- makes $P(X^-) = \varepsilon^k$, which implies that if x_i belongs to the instantiation of X^-, then $P(x_i) = \varepsilon$ and $P(\overline{x_i}) = 1 - \varepsilon$ (the opposite case is analogous). This means that the vertices of the credal sets of the X^- nodes are completely fixed by the instantiation chosen. Thus the only credal sets that can float in the denominator are the X^+ variables (S_i variables are not credal and X^- are already fixed as indicated). So, processing the MPEe we have

$$\underline{P}(X^-, o, s|q) = \frac{\frac{1}{m+1}\varepsilon^{n}}{1 - \min\left(\sum_{c\neq 0} P(S_n = c)\right)},$$

where $\min \sum_{c\neq 0} P(S_n = c)$ is evaluated over all possible vertices of the X^+ credal sets (we know that $P(X_i = o_i)$ is set to ε, but the probabilities $P(X_i = x_i)$ and $P(X_i = \overline{x_i})$ may vary between 0 and $1-\varepsilon$).

The formula $\phi(X)$ will be satisfied by X^- and X^+ if the sum $\sum_{c\neq 0} P(S_n = c)$ is less than $\delta_1 = \frac{m}{m+1}(1-\varepsilon)^2\varepsilon(1-\varepsilon)^{n-k}$ (this is the maximum value that a satisfied $\phi(X)$ may assume). All unsatisfied formulas lead to greater values. In fact the smallest value that a unsatisfied formula implies in the sum is greater than $\delta_2 = \frac{m}{m+1}(1-\varepsilon)^{n-k+3}$. Note that $\delta_2 > \delta_1$.

If we query the PT-CN-MPEe problem $\max \underline{P}(X, S|q)$ with $r = \frac{1}{m+1}\frac{\varepsilon^n}{1-\delta_1}$ and get a negative answer, then for all instantiations to X^-, there is an instantiation to X^+ that satisfy $\phi(X)$ (that is, the sum is bounded by δ_1). If the answer is positive, then there does exist an instantiation to X^- where no instantiation to X^+ can make $\phi(X)$ satisfied. □

Theorem 8 PT-BN-MmAP *and* BIW-BN-MmAP *are* Σ_2^P*-Complete.*

Proof Pertinence of BIW-BN-MmAP (which ensures pertinence of PT-BN-MmAP) is trivial. Given a instantiation for the MAP variables, we need to solve a minimization over the Y variables, which is NP-Complete (see it as a BIW-CN-Pr using Lemma 4). Hardness of PT-BN-MmAP (which ensures hardness of BIW-BN-MmAP) is reached by a reduction from another version of $\mathtt{QSAT}_2$ that is Σ_2^{p}-Complete: *Given a set of variables $X_1, \ldots, X_n$, an integer $0 < k \leq n$ and a boolean 3DNF formula $\phi(X)$ over these variables, is there an instantiation to the first k variables such that, for all instantiations of the remaining $n-k$, $\phi(X)$ is satisfied?*

We construct again a network following the ideas of Theorem 3. It includes a binary node to each X_i, without parents and with uniform prior. There are n nodes S_i with parents S_{i-1} and X_i. They have $m+1$ categories and are defined as follows (for $c \in \{1, \ldots, m\}$ and $i \in \{1, \ldots, n\}$):

$$P(S_i = c|S_{i-1} = c, x_i) = \begin{cases} 1 & \text{if } x_i \in \text{clause } c \\ 0 & \text{if } \overline{x_i} \in \text{clause } c \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

$$P(S_i = c|S_{i-1} \neq c, x_i) = 0.$$

The conditional probabilities of S_i given $\overline{x_i}$ are defined analogously. The probabilities of $P(S_i = 0|S_{i-1}, X_i)$ ensure that they sum exactly 1, as done in Theorem 3. Furthermore, $P(S_0)$ has uniform prior and we insert an additional binary node Q with S_n as parent, having $P(q|S_n = c) = 1$ if $c \neq 0$ and 0 otherwise.

So, we have that $P(X|q) = \frac{\frac{1}{2^n}P(q|X)}{P(q)}$ is equal to

$$= \frac{\frac{1}{(m+1)2^n}\sum_{c\neq 0}\prod_i A_{c,i}}{\frac{m}{(m+1)}\left(\frac{1}{2}\right)^n} = \frac{\sum_{c\neq 0}\prod_i A_{c,i}}{m}$$

where $A_{c,i} = \sum_{X_i} P(S_i = c|S_{i-1} = c, X_i)\, P(X_i)$ for $i \in \{1, \ldots, n\}$. $A_{c,i}$ assumes value zero only when the variable it represents denies the clause c. The numerator of $P(X|q)$ sums how many clauses are satisfied by the instantiation of X variables. Let $X^- = \{X_1, \ldots, X_k\}$ and $X^+ = \{X_{k+1}, \ldots, X_n\}$. We have $\min_{X^+} P(X^-, X^+|q) = 0$ if, given the instantiation for the X^- variables, there is an assignment to X^+ variables that can deny $\phi(X)$. This way, questioning if PT-BN-MmAP problem with MAP variables X^- and evidence q (which maximizes $\min_{X^+} P(X|q)$) has non-zero answer is enough to solve the $\mathtt{QSAT}_2$ problem. Given this X^- instantiation, all X^+ will satisfy $\phi(X)$. □

5 Conclusion

We can summarize the contributions of this paper as follows. Concerning Bayesian networks, we have first introduced a more general definition of bounded induced-width networks (demonstrating that many problems where induced-width actually grows with the network remain polynomial), and we have shown the difference between the BN-MPE and the BN-MPEe problems. More importantly, we have introduced the MmAP problem and presented its complexity. A possible improvement of our results would be to present a completeness result for multiply-connected networks.

Our most significant results pertain to credal networks, with direct implications to models that handle interval and set probabilities, belief functions, possibility measures, qualitative probabilities, and families of probabilistic logic. We have clarified the so far unexplored complexity of CN-MPE, CN-MPEe, and CN-MAP. The polynomial character of CN-MPE in some cases is rather surprising.

There are several interesting problems still to be explored. For example, binary networks (Bayesian and credal) could display lower complexity than their non-binary counterparts in problems such as PT-BN-MAP, BIW-BN-MAP and BIW-CN-Pr. These problems belong to NP and are clearly related, but are there polynomial time algorithms to solve them? We conjecture there are not, even as we note that there is a polynomial algorithm to solve CN-Pr in binary polytrees [Fagiuoli and Zaffalon, 1998].

Acknowledgments

We thank Aritanan Gruber for indicating relevant complexity results, Carlos Guestrin for pointing out a flaw in [da Rocha and Cozman, 2002], Thomas Lukasiewicz for initial thoughts on the complexity of credal networks, and a reviewer for suggesting a short proof for Theorem 5. This research received support from HP Brazil R&D; the second author was partially supported by CNPq, Brazil.

References

[Bailey *et al.*, 2001] D. D. Bailey, V. Dalmau, and P. G. Kolaitis. Phase transitions of PP-complete satisfiability problems. In *Int. Joint Conf. on Artificial Intelligence*, pp. 183–192, Seattle, Washington, 2001.

[Cano *et al.*, 1993] J. Cano, M. Delgado, and S. Moral. An axiomatic framework for propagating uncertainty in directed acyclic networks. *Int. Journal of Approximate Reasoning*, 8:253–280, 1993.

[Cozman *et al.*, 2004] F. G. Cozman, C. P. de Campos, J. S. Ide, and J. C. F. da Rocha. Propositional and relational Bayesian networks associated with imprecise and qualitative probabilistic assessments. In *XX Conf. on Uncertainty in Artificial Intelligence*, pp. 104–111, Banff, 2004. AUAI Press.

[Cozman, 2000a] F. G. Cozman. Credal networks. *Artificial Intelligence*, 120:199–233, 2000.

[Cozman, 2000b] F. G. Cozman. Separation properties of sets of probabilities. In *XVI Conf. on Uncertainty in Artificial Intelligence*, pp. 107–115, San Francisco, 2000. Morgan Kaufmann.

[da Rocha and Cozman, 2002] J. C. F. da Rocha and F. G. Cozman. Inference with separately specified sets of probabilities in credal networks. In *XVIII Conf. on Uncertainty in Artificial Intelligence*, pp. 430–437, San Francisco, 2002. Morgan Kaufmann.

[Dechter, 1996] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *XII Conf. on Uncertainty in Artificial Intelligence*, pp. 211–219, San Francisco, 1996. Morgan Kaufmann.

[Eyal, 2001] A. Eyal. *Dividing and Conquering Logic*. PhD thesis, Stanford University, 2001.

[Kakade *et al.*, 2001] S. Kakade, and L. Ortiz. Graphical economics. *COLT*, 2004.

[Fagiuoli and Zaffalon, 1998] E. Fagiuoli and M. Zaffalon. 2U: An exact interval propagation algorithm for polytrees with binary variables. *Artificial Intelligence*, 106(1):77–107, 1998.

[Levi, 1980] I. Levi. *The Enterprise of Knowledge*. MIT Press, Cambridge, Massachusetts, 1980.

[Litmman *et al.*, 2001] M. Litmman, S. M. Majercik, and T. Pitassi. Stochastic boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296, 2001.

[Papadimitriou, 1994] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, San Diego, 1994.

[Park and Darwiche, 2004] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

[Park, 2002] J. D. Park. MAP complexity results and approximation methods. In *XVIII Conf. in Uncertainty in Artificial Intelligence*, University of Alberta, Edmonton, Alberta, Canada, 2002. Morgan Kaufmann.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[Roth, 1996] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

[Shimony, 1994] S. E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410, 1994.

[Walley, 1991] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

[Walley, 1996] P. Walley. Measures of uncertainty in expert systems. *Artificial Intelligence*, 83:1–58, 1996.

[Zaffalon, 2003] M. Zaffalon and E. Fagiuoli. Tree-based credal networks for classification. *Reliable Computing*, 9(6):487–509, 2003.

Lifted First-Order Probabilistic Inference

Rodrigo de Salvo Braz and **Eyal Amir** and **Dan Roth**
University of Illinois at Urbana-Champaign
Department of Computer Science
201 N Goodwin Ave, Urbana, IL 61801-2302
braz@uiuc.edu, {eyal,danr}@cs.uiuc.edu

Abstract

Most probabilistic inference algorithms are specified and processed on a propositional level. In the last decade, many proposals for algorithms accepting first-order specifications have been presented, but in the inference stage they still operate on a mostly propositional representation level. [Poole, 2003] presented a method to perform inference directly on the first-order level, but this method is limited to special cases. In this paper we present the first exact inference algorithm that operates directly on a first-order level, and that can be applied to any first-order model (specified in a language that generalizes undirected graphical models). Our experiments show superior performance in comparison with propositional exact inference.

1 Introduction

Probabilistic inference algorithms are widely employed in artificial intelligence. However, most of them do not accept first-order specifications of models, which can abstract over classes of objects, requiring instead propositional ones which are longer and redundant because they must be specified variable by variable.

In the last decade, many proposals for algorithms accepting first-order specifications have been presented [Ngo and Haddawy, 1995; Ng and Subrahmanian, 1992; Jaeger, 1997; Kersting and De Raedt, 2000; Friedman *et al.*, 1999; Pfeffer *et al.*, 1999; Poole, 1993; Anderson *et al.*, 2002; Richardson and Domingos, 2004; Laskey, 2005], most of which based on the theoretic framework of [Halpern, 1990]. However, these solutions perform inference at a mostly propositional level, that is, dealing with the random variables instantiated from the first-order, parameterized variables in the first-order specification (usually only the relevant random variables will be present in the propositional version). In domains with a large number of objects this may be both costly and essentially unnecessary. For example, a medical application can be about a large population of people infected with a certain disease and have a model of the probability of death of a person (*any* person) with that disease. To answer the query "what is the probability that *someone* will die of this disease?", an algorithm that depends on propositionalization would have to instantiate a random variable per patient. However this is not necessary since one can reason about individuals on a general level, provided one knows the population size, in order to answer that query in a much shorter time.

In such a scenario it would be possible to devise a way of using the available model to answer the query without considering each individual. However, this would require a manual devising of a process *specific* to the model or query in question. What is missing to date is an algorithm that can receive a *general* first-order model and *automatically* answer queries like these without computional waste.

A first step in this direction was given by [Poole, 2003], which proposes a generalized version of the variable elimination algorithm ([Zhang and Poole, 1994]) that is *lifted*, that is, deals with groups of random variables at a first-order level. The algorithm receives a specification in which *parameterized* random variables stand for all of their instantiations and then eliminates them in a way that is equivalent to, but much cheaper than, eliminating all their instantiations at once.

The algorithm in [Poole, 2003], however, works only for a very particular type of model because its only elimination operation is what we call *inversion elimination*, which requires special conditions explained later. These special conditions may sometimes be met by carefully chosen elimination orderings, but in certain cases no such ordering exists and inversion elimination cannot be applied at all steps. This introduces the need for another type of elimination, *counting elimination*, which can always be applied but costs more. By putting these two operations together we present the first algorithm capable of dealing with any first-order probabilistic model that operates directly on first-order representations, without resorting to a propositional level. We also show that the algorithm is correct and provide experimental results comparing it to a propositional algorithm.

2 Motivation

A probabilistic model over a set of random variables is defined by a set of dependencies, each of them on a subset of those variables. In a propositional model, each dependency explicitly refers to the variables it affects. For example, consider a Markov network involving a potential function, or *fac-*

tor, such as:

$$\phi(epidemic, sick) = \begin{cases} 0.7, & \text{if } epidemic \wedge sick, \\ 0.3, & \text{if } epidemic \wedge \neg sick, \\ 0.5, & \text{otherwise.} \end{cases}$$

In fact, this type of potential function will be common enough in this paper that we define "if α then β p" to mean

$$\phi(Var(\alpha, \beta)) = \begin{cases} p, & \text{if } \alpha \wedge \beta, \\ 1-p, & \text{if } \alpha \wedge \neg\beta, \\ 0.5, & \text{otherwise.} \end{cases}$$

where α and β are boolean formulas on binary random variables and $Var(\alpha, \beta)$ is the set of these random variables. If p is omitted, it is assumed to be 1. "β p" is the same as "if $\top$ then β p" and "if α then β p else q" stands for both "if α then β p" and "if $\neg\alpha$ then β q". So we can write a model in the more readable fashion:[1]

if $epidemic$ then $sick$ 0.7 if $sick$ then $death$ 0.4

In most practical problems, the same factor holds on many different sets of variables, requiring propositional models to repeat that factor several times, modulo the specific variables involved each time. In our example, if we wish to keep track of whether each member of a population is sick (with distinct variables $sick(john), sick(mary), \ldots$), we would have to write

if $epidemic$ then $sick(john)$ 0.7
if $sick(john)$ then $death(john)$ 0.4
if $epidemic$ then $sick(mary)$ 0.7
if $sick(mary)$ then $death(mary)$ 0.4 ...

This renders the model specification unnecessarily complex and redundant. Moreover, an inference algorithm will consider each of those factors separately, even though there is some structure across them that should be exploited.

Currently, the most common way of dealing with these situations is to keep the original model and use it for each separate object (in this case, each person) as needed. This however does not help when different instances of the factor need to be used at the same time, as it would be the case to answer the query $P(sick(john) \wedge sick(mary))$, for example. In these situations, procedures specific to a given model have to be manually tailored, in what is a time consuming solution.

A natural way around this problem is to specify recurring factors in a parameterized way. This would allow us to express the same as above in the more succint way

if $epidemic$ then $sick(Person)$ 0.7
if $sick(Person)$ then $death(Person)$ 0.4

where $Person$ (and, in our notation, words starting with a capital) is a typed *logical variable* assuming any value from the set of people involved in the problem. The semantics of this representation is simply that it should be equivalent to the propositional model formed by all possible instantiations of its logical variables. Following [Poole, 2003], we call these parameterized factors *parfactors*.

This semantics immediately provides an inference algorithm for such a representation, namely the one in which we apply any regular propositional inference algorithm to the propositionalized model, but this would be overkill. For example, in order to solve the query $P(death(john)|sick(john))$ it is only necessary to consider the instantiations for $Person = john$, ignoring other values. One could also consider general queries such as $P(sick(Person)|death(Person))$ that do not require any instantiations at all in order to be solved. An extreme example of the benefit of directly using the first-order representation is given by adding the parfactor "if $death(Person)$ then $someDeath$" to the model and considering the query $P(someDeath|epidemics)$. The tree width of the propositionalized graphical model is the population size, while the query can in fact be answered in time *independent* from the population size (a similar example is shown in fig. 4). It is therefore desirable to have an algorithm performing *lifted inference*, that is, inference directly on the first-order level, which instantiates parfactors only when necessary.

The languages we mentioned before allow parameterized specifications of probabilistic models. However, no corresponding first-order inference algorithm has been provided; inference is still performed by generating some propositionalized form of the model and using regular propositional inference algorithms on them (although some systems, like SPOOK in [Pfeffer *et al.*, 1999], benefit from the first-order structure in some ways). In this paper we present an algorithm which performs exact lifted inference on first-order models.

It is also useful to allow deterministic constraints on the logical variables of parfactors. For example,

$$\text{if } sick(Person1) \wedge roommate(Person1, Person2) \text{ then } sick(Person2)\ 0.8, Person1 \neq Person2 \quad (1)$$

$$diabetes(Person)\ 0.01, Person \neq john \wedge Person \neq mary$$

The constraint $Person1 \neq Person2$ in the first factor states that only its instantiations satisfying this condition will be considered. In the second factor, the potential of 0.01 for the random variable $diabetes(Person)$ is assigned only for instantiations in which $Person$ is distinct from $john$ and $mary$.

3 Language, notation and semantics

Our language and semantics are essentially the same as those in [Poole, 2003], that is, those of a Markov network specified in a first-order language that allows parameterized random variables,[2] and are also similar to Markov logic networks [Richardson and Domingos, 2004]. A *parfactor* is a triad (ϕ, A, C) representing the applications of a potential function ϕ on all instantiations of a tuple of logical *atoms*

[1] The reason we define the model with this "conditional" potential function rather than with usual conditional probabilities is that we concern ourselves with Markov networks (undirected models) only in this paper.

[2] Poole discusses some aspects of directed models, however.

A according to assignments to the logical variables in these atoms that satisfy a constraint formula C. At this point, we restrict ourselves to constraints with a finite number of solutions so as to have finite models only (this prevents us from using function symbols – more on this in section 6). For example, (1) is represented by the parfactor (ϕ, A, C), where ϕ is the appropriate potential function, A is $\{sick(Person1), roommate(Person2), sick(Person2)\}$ and C is $Person1 \neq Person2$.

Note that we are in no way committed to the "if ... then" construction used, which is simply a notation for a specific type of potential function. Any potential function is allowed, and random variables can be multivalued rather than binary only.

Just as with regular undirected graphical models, here the joint probability distribution is determined by the product of all potential functions given an assignment to all random variables (which are instantiations of atoms in parfactors, and therefore *ground* atoms). The only difference is that in a first-order model this product involves all instantiations of all parfactors. Given a set of parfactors G, the joint distribution defined by it is

$$P(RV(G)) \propto \prod_{g \in G} \prod_{\theta \in \Theta_g} \phi_g(A_g\theta) \qquad (2)$$

where $RV(G)$ is the set of all random variables (ground atoms) involved in all instantiations of all its parfactors, Θ_g is the set of all assignments, or substitutions, to the logical variables of g that satisfy its constraint (the *solutions* to the constraint), ϕ_g is the potential function in g, A_g is the tuple of atoms in g and $A_g\theta$ is the instantiation of this tuple given an assignment θ to logical variables.

Further notation include, for a parfactor g, C_g for the constraint in g and, for a set of parfactors G, A_G for the atoms in G, C_G for the total constraint $\bigwedge_{g \in G} C_g$ and Θ_G for the set of solutions of C_G. For any object α, $LV(\alpha)$ and $RV(\alpha)$ are the sets of logical and random variables in α, respectively. Finally, all sets of parfactors are implicitly assumed to be *standardized apart*, that is, logical variables are renamed if necessary so that no logical variable is used in more than one parfactor in the set.

4 Inference

The inference problem is, given a set of random variables (ground atoms) Q representing a query, to calculate the marginal probability of Q given a model G (queries involving a condition can be issued by adding parfactors representing this condition to G). This is

$$P(Q) \propto \sum_{RV(G) \setminus Q} \phi(G)$$

where $\sum_{RV(G)\setminus Q}$ is a summation over all assignments to random variables not in Q and $\phi(G)$ is a shorthand notation for the right-hand side of equation (2).

Calculating this summation by brute force is intractable, but one can use independencies in the model to do it more efficiently. In propositional models, one way of doing this is the variable elimination (VE, [Zhang and Poole, 1994]) algorithm which calculates the total marginal by dividing it into smaller partial marginalizations, each on a single variable. The main contribution of this paper is a first-order version of VE, *FOVE*, which is shown in Figure 1 and works in a similar way by eliminating one (but maybe more) atoms and their respective constraints at each step. The advantage of *FOVE* is that, by eliminating an atom with its associated constraints, we are effectively eliminating all of its groundings in a *lifted* way, with a cost that is sometimes independent of the number of groundings.

PROCEDURE *FOVE*(G, Q)
G a set of parfactors, Q a set of random variables (the query).
1. If $RV(G) = Q$, return G.
2. $G \leftarrow$ *SHATTER*(G, Q) (figure 2).
3. $E \leftarrow$ *FIND-ELIMINABLE*(G, Q).
4. $G_E \leftarrow \{g \in G : RV(g)$ and $RV(E)$ intersect$\}$.
5. $G_{\bar{E}} \leftarrow G \setminus G_E$.
6. $g' \leftarrow$ *ELIMINATE*(G_E, E).
7. $G' \leftarrow \{g'\} \cup G_{\bar{E}}$.
8. Return *FOVE*(G', Q).

PROCEDURE *FIND-ELIMINABLE*(G, Q)
G a set of parfactors, $Q \subset RV(G)$, G shattered against Q.
1. Choose e from $A_G \setminus Q$.
2. $G_e \leftarrow \{g \in G : RV(g)$ and $RV(e)$ intersect$\}$.
3. If $LV(e) = LV(G_e)$ ($\{e\}$ is inversion-eliminable) return $\{e\}$.
4. Return *FIND-COUNT-ELIMINABLE*$(G, Q, \{e\})$.

PROCEDURE *ELIMINATE*(G, E)
G a set of parfactors, $E \subset RV(G)$.
1. $A' \leftarrow A_G \setminus E$.
2. $g \leftarrow (\prod_{g \in G} \phi_g^{|\Theta_G|/|\Theta_g|}, A', C_G)$ (fusion, section 4.4).
3. If $LV(E) = LV(g)$ (E is inversion-eliminable) return parfactor $(\sum_e \phi_g(A'\theta, e), A', C_g)$.
4. Return $(\sum_{N_1} \cdots \sum_{N_u} N_1! \ldots N_n! \prod_i^k \phi_g(v_i, A')^{|V_i|}, A', \top)$ (as detailed in section 4.3).

PROCEDURE *FIND-COUNT-ELIMINABLE*(G, Q, E)
G a set of parfactors, $Q \subseteq RV(G)$, $E \subseteq A_G \setminus Q$.
1. If $A_{G_E} \setminus E$ is ground (E is counting-eliminable) return E.
2. Choose a non-ground atom $e \in A_G \setminus E$.
3. Return *FIND-COUNT-ELIMINABLE*$(G, Q, E \cup \{e\})$.

Figure 1: First-order variable elimination algorithm.

4.1 *FOVE* correctness

We now show that *FOVE* is correct. The algorithm works in the following way: suppose we want to eliminate the atoms in a set E at a given step of it. Then we can write

$$\begin{aligned} P(Q) &\propto \sum_{RV(G)\setminus Q} \phi(G) \\ &= \sum_{RV(G)\setminus Q\setminus RV(E)} \sum_{RV(E)} \phi(G_E)\phi(G_{\bar{E}}) \\ &= \sum_{RV(G)\setminus Q\setminus RV(E)} \phi(G_{\bar{E}}) \sum_{RV(E)} \phi(G_E) \end{aligned}$$

where $RV(E)$ is the set of random variables resulting from all instantiations of E in G, G_E is the subset of parfactors in G depending on $RV(E)$, and $G_{\bar{E}}$ is $G \setminus G_E$.

If we can represent $\sum_{RV(E)} \phi(G_E)$ as the potential of a single parfactor g', (defined such that $\phi(g') = \prod_{\theta \in \Theta_g} \phi_g(A_g\theta)$), we can reduce the original marginal $\sum_{RV(G)\setminus Q} \phi(G)$ to a marginal on a model $G' = G_{\bar{E}} \cup \{g'\}$ which involves fewer random variables:

$$\begin{aligned} P(Q) \propto \sum_{RV(G)\setminus Q} \phi(G) &= \sum_{RV(G)\setminus Q\setminus RV(E)} \phi(G_{\bar{E}}) \sum_{RV(E)} \phi(G_E) \\ &= \sum_{RV(G)\setminus Q\setminus RV(E)} \phi(G_{\bar{E}})\phi(g') \\ &= \sum_{RV(G)\setminus Q\setminus RV(E)} \phi(G_{\bar{E}} \cup \{g'\}) = \sum_{RV(G')\setminus Q} \phi(G') \end{aligned}$$

There are two ways, described below, of calculating a parfactor g' such that $\phi(g') = \sum_{RV(E)} \phi(G_E)$: (1) *inversion elimination* or (2) *counting elimination*. (1) is the preferable one because it does not depend on the number of objects in the domain or, in other words, the size of $RV(E)$. However, this method requires certain conditions on E (explained later) that may be impossible to satisfy for any E in the atoms of G. (2) is less favorable as it depends on the size of $RV(E)$ (it is still better than propositionalization, though), but it is always possible to find an E on which it can be applied.

In the two next subsections, we assume that G_E has been replaced by an equivalent parfactor g. This operation is called *fusion* and is explained in section 4.4. We are thus left with the problem of expressing $\sum_{RV(E)} \phi(g)$ as a parfactor.

Finally, we assume that the parfactors and query have been *shattered*, as explained in section 4.5. The main property of shattered parfactors and query is that any two atoms in them have groundings which are either identical or completely disjoint. Why this matters will be explained as inversion and counting elimination are explained.

4.2 Inversion elimination

Inversion elimination assumes that E is a unary set $\{e\}$ such that $LV(e) = LV(g)$, where $LV(\alpha)$ is the set of logical variables in α. Let $\theta_1 \dots \theta_n$ be an enumeration of Θ_g. Then

$$\begin{aligned} \sum_{RV(e)} \phi(g) &= \sum_{RV(e)} \prod_{\theta \in \Theta_g} \phi_g(A_g\theta) \\ &= \sum_{e\theta_1} \cdots \sum_{e\theta_n} \phi_g(A_g\theta_1) \dots \phi_g(A_g\theta_n) \\ &= \sum_{e\theta_1} \phi_g(A_g\theta_1) \cdots \sum_{e\theta_n} \phi_g(A_g\theta_n) \end{aligned}$$

(because of shattering)

$$\begin{aligned} &= \Big(\sum_{e\theta_1} \phi_g(A_g\theta_1)\Big) \dots \Big(\sum_{e\theta_n} \phi_g(A_g\theta_n)\Big) \\ &= \prod_{\theta \in \Theta_g} \sum_{e\theta} \phi_g(A_g\theta) = \prod_{\theta \in \Theta_g} \sum_{e\theta} \phi_g(A'\theta, e\theta) \\ &= \prod_{\theta \in \Theta_g} \sum_{e} \phi_g(A'\theta, e) \quad \text{(by renaming)} \\ &= \prod_{\theta \in \Theta_g} \phi'(A'\theta) = \phi(g') \end{aligned}$$

for g' a new parfactor (ϕ', A', C_g) where A' is a tuple of the atoms distinct from e in A_g, $\phi'(A'\theta) = \sum_{e\theta} \phi_g(A_g\theta)$, and C_g is the constraint formula of g.

Note that the initial sum of products becomes a product of sums, hence the name *inversion elimination*. Also note that the sum providing ϕ' is over the assignments on the *parameterized* random variables. Therefore this elimination method does not depend on the number of groundings, but on the number of assignments to the parameterized random variable, which is much smaller.

The condition $LV(e) = LV(g)$ is essential for this method to work because it guarantees that the random variables being summed out have a one-to-one correspondence to the instantiations of g. This is a condition not taken into account by [Poole, 2003], whose method eliminates all random variables not in the query by inversion elimination, one by one. However, the proof above should make it clear that this is not always possible. A numerical contradiction can be found by trying to answer the query r for $p(X) \wedge q(Y) \wedge r$ 0.8, with type of X being $\{a\}$ and type of Y being $\{b, c\}$, since neither $p(X)$ or $p(Y)$ is suitable for inversion elimination. The correct answer is ≈ 0.78, but eliminating $p(X)$ and then $q(Y)$ by inversion produces ≈ 0.75.

4.3 Counting elimination

When we cannot find an atom e in G satisfying the conditions for inversion elimination, we can resort to *counting elimination*, which is based on counting arguments.

Counting elimination can be done on a set of atoms E such that the remaining atoms in g (and consequently G_E) are all ground. We can always find such an E in G, since the set of non-ground atoms in G is such a set. We however try to find the smallest such E since the cost of the method depends on the size of $RV(E)$. Note that counting elimination is only justified for $|E| > 1$ since $|E| = 1$ implies that E is inversion eliminable.

Once we have a proper E, let A' be the remaining atoms in g. Then, because A' is ground,

$$\begin{aligned} \sum_{RV(E)} \phi(g) &= \sum_{RV(E)} \prod_{\theta \in \Theta_g} \phi_g(E\theta, A'\theta) \\ &= \sum_{RV(E)} \prod_{\theta \in \Theta_g} \phi_g(E\theta, A') \end{aligned}$$

The last term above defines a potential function ϕ' on A'. The result obtained from counting elimination is a new parfactor $g' = (\phi', A', \top)$.

In order to calculate this term, we present a counting argument. Given an assignment on $RV(E)$,

$$\prod_{\theta \in \Theta_g} \phi_g(E\theta, A') = \prod_i^k \phi_g(v_i, A')^{|V_i|}$$

by grouping all applications of ϕ_g with the same v_i, where $v_1, \ldots, v_k$ are the different assignments to $E\theta$ and V_i is the set of different $E\theta$'s assigned v_i.

Now assume for a moment that E is in fact just one atom e. This means that the v_i's are the possible values for instances of e. Note that $\prod_i^k \phi_g(v_i)^{|V_i|}$ is a function of v_i and $|V_i|$, but not of V_i. In other words, it only matters how *many* instances of e are assigned v_i by a given assignment on $RV(e)$, but not *which* of them. Different assignments will induce different vectors $(V_1, \ldots, V_k)$, but if they induce the same vector $(|V_1|, \ldots, |V_n|)$ (denoted $\vec{N}$), that product will be the same. Also, given a vector $\vec{N}$, the number of assignments inducing it is $\vec{N}!$, the multinomial coefficient of $\vec{N}$ [3] (in the particular case where e is a binary variable, this becomes $\binom{|RV(e)|}{\vec{N}_0} = \binom{|RV(e)|}{\vec{N}_1}$). We can therefore group assignments according to $\vec{N}$ and write

$$\sum_{RV(E)} \phi(g) = \sum_{\vec{N}} \vec{N}! \prod_i^k \phi_g(v_i, A')^{\vec{N}_i}$$

Let us now consider the case where E contains multiple atoms. Let $E_1, \ldots, E_n$ be an enumeration of E and $R_1, \ldots, R_n$ be their respective groundings. Let us also assume that for any two atoms E_m, E_j in E, their groundings R_m, R_j are identical or disjoint. This condition is satisfied by shattered sets of parfactors, as discussed in 4.5. Moreover, we also assume that any two atoms in E are either identical or not unifiable at all, according to C_g. This is also granted for shattered sets of parfactors. For now, we also demand that for any two atoms E_m, E_j in E, $LV(E_m) \cap LV(E_j) = \emptyset$, leaving the case where this is false for later.

Let $S_1, \ldots, S_u$ be an enumeration of $\{R_j : 1 \leq j \leq n\}$, that is, a sequence of unique R_j's. We can consider each assignment as a composition of assignments on each S_i and write

$$\sum_{RV(E)} \phi(g) = \sum_{S_1} \cdots \sum_{S_u} \prod_i^k \phi_g(v_i, A')^{|V_i|}$$

As before, $|V_i|$ is the number of $E\theta$'s assigned v_i, but v_i is a *tuple* assignment to $E\theta$. $|V_i|$ can be calculated by choosing, for each component $v_{i,j}$, how many random variables in R_j can be assigned $v_{i,j}$ (this choice can be made in this fashion because atoms in E do not share logical variables). This is simply $|R_j|$, the number of random variables in R_j, unless some other component $v_{i,m}$, with $m < j$, $E_m \neq E_j$ and $R_m = R_j$, has already committed a random variable in R_j for itself. For this reason, it is important to know from the beginning whether $E_m\theta$ is either the same random variable as $E_j\theta$ or not, otherwise we would not know whether to have one less option from R_j (for the cases where $E_m\theta \neq E_j\theta$) or not to have to make a choice at all (in those cases where $E_m\theta = E_j\theta$ and the choice has already been made for $E_m\theta$ and therefore for $E_j\theta$). This information is available since E_m and E_j must be either identical or not unifiable, as stated above. From this reasoning,

$$\sum_{RV(E)} \phi(g) = \sum_{\vec{N}_1} \cdots \sum_{\vec{N}_u} \vec{N}_1! \ldots \vec{N}_n! \prod_i^k \phi_g(v_i, A')^{|V_i|}$$

with

$$|V_i| = \prod_j (|\vec{N}_{s(j),i}| - |\{m : m < j, E_m \neq E_j, R_m = R_j\}|),$$

where $s(j)$ is such that $S_{s(j)} = R_j$, $\vec{N}_{s(j)}$ is the vector corresponding to the counting of assigments on $S_{s(j)} = R_j$, with $\vec{N}_{s(j),i}$ being the number of random variables in R_j assigned $v_{i,j}$.

Finally, we consider the case where the condition that for any two atoms E_m, E_j in E, $LV(E_m) \cap LV(E_j) = \emptyset$ is not satisfied. We can reduce this case to the previous one by multiplying away the logical variables violating the condition. To multiply a logical variable vector $\bar{Z}$ away from a parfactor h, we calculate a new parfactor $h' = (\phi', A', C')$ where A' is the same as A_h but for the removal of the logical variables in $\bar{Z}$, $C' = C_{|LV(h) \setminus \bar{Z}}$ and, for any $\theta \in \Theta_{h'}$, $\phi'(A'\theta) = \prod_{\theta' \in C_{|\bar{Z}}} \phi_h(A_h\theta'\theta)$, where $C_{|\bar{W}}$, the *restriction* of C to a vector of logical variables $\bar{W}$, is defined as the constraint $\exists \bar{V} C$ for $\bar{V} = LV(C) \setminus \bar{W}$. This is simply the formula describing the solutions of C restricted to variables in $\bar{W}$ (for equational formulas without function symbols this can be simplified to an equational formula without quantifiers).

Multiplying away is an expensive operation that depends directly on the domain size. We are currently working on more sophisticated counting arguments that minimize its use.

4.4 Fusion

We now explain how a set of parfactors G can be replaced by a single, equivalent parfactor $fs(G) = (\phi', A_G, C_G)$, with $\phi'(A_G\theta) = \prod_{g \in G} \phi_g(A_g\theta)^{|\Theta_g|/|\Theta_G|}$ for any $\theta \in \Theta_G$.

$$\begin{aligned}
\phi(G) &= \prod_{g \in G} \prod_{\theta \in \Theta_g} \phi_g(A_g\theta) = \prod_{g \in G} \prod_{\theta \in \Theta_G} \phi_g(A_g\theta)^{|\Theta_g|/|\Theta_G|} \\
&= \prod_{\theta \in \Theta_G} \prod_{g \in G} \phi_g(A_g\theta)^{|\Theta_g|/|\Theta_G|} \\
&= \prod_{\theta \in \Theta_G} \phi'(A_G\theta) = \phi(fs(G))
\end{aligned}$$

The crucial step is the one in which we replace each original set of constraint solutions Θ_g by the global constraint solution set Θ_G. When this happens, each original instantiation of a parfactor is now instantiated $|\Theta_G|/|\Theta_g|$ many times more than before, but the power $|\Theta_g|/|\Theta_G|$ preserves the original potential value.

[3] Defined as $\vec{N}! = (\vec{N}_1, \ldots, \vec{N}_n)! = \frac{(\vec{N}_1 + \cdots + \vec{N}_n)!}{\vec{N}_1! \ldots \vec{N}_n!}$

4.5 Shattering

The elimination of atoms requires certain conditions guaranteed by the fact that the set of parfactors having been shattered against the query. This is based on discussion in [Poole, 2003].

A set of parfactors is *shattered* if, for every pair of atoms (p, q) in G, two conditions hold: (1), their groundings $RV(p)$ and $RV(q)$ are either identical or disjoint, and (2), in a condition needed by counting elimination, every pair of atoms in each parfactor must be either identical or never be instantiated to the same random variable by a single logical variable assignment. We call pairs of atoms satisfying these two conditions *proper pairs*. A set of parfactors is *shattered against* a set of ground atoms Q if the same conditions hold when the atoms in Q are included.

For example, parfactors $(\phi_1, p(X, a), \top)$ and $(\phi_2, p(b, Y), Y \neq d)$ are not shattered because $RV(p(X, a))$ and $RV(p(b, Y))$ overlap but are not identical, violating (1). In another example, parfactor $(\phi, (p(X), p(Y)), \top)$ is not shattered because, even though $RV(p(X)) = RV(p(Y))$, $p(X)$ and $p(Y)$ are instantiated to the same random variable by some logical variable assignments (those in which $X = Y$), violating (2).

The algorithm in figure 2 shatters a set of parfactors against a query. It works by repeatedly identifying pairs of improper pairs and *breaking* parfactors into equivalent sets of parfactors whose sets of instantiations are the same as the original ones, but inducing proper pairs. This is done by, through unification, determining the conditions for the groundings of improper pairs to coincide or not, and breaking the parfactors along these conditions. After this, unified atoms are rewritten so that they will be identical.

For example, if we have parfactor $(\phi_2, p(b, Y), Y \neq d)$ and query $p(b, c)$, $p(b, Y)$ and $p(b, c)$ are an improper pair. Their most general unifier (MGU) is $Y = c$, so we can break the parfactor into $(\phi_2, p(b, Y), Y \neq d \wedge Y = c)$ and $(\phi_2, p(b, Y), Y \neq d \wedge Y \neq c)$ which can be rewritten as $(\phi_2, p(b, Y), Y = c) = (\phi_2, p(b, c), \top)$ and $(\phi_2, p(b, Y), Y \neq d \wedge Y \neq c)$. In another example, parfactor $(\phi, (p(X), p(Y)), X \neq a)$ contains improper pair $p(X), p(Y)$. Their unification yields $X = Y$, so the parfactor is broken into $(\phi, (p(X), p(Y)), X \neq a \wedge X = Y)$ and $(\phi, (p(X), p(Y)), X \neq a \wedge X \neq Y)$, the first one being rewritten as $(\phi, (p(X), p(X)), X \neq a)$.

5 Empirical results

We use the implementation available at `http://l2r.cs.uiuc.edu/~cogcomp` to compare average run times between lifted and propositional inference (which produce the exact same results) for two different models while increasing the number of objects in the domain. The first one, (I) in figure 3, answers the query $P(death)$ from $\{epidemic\ 0.55,\ \text{if } epidemic \text{ then } sick(X)\ 0.7 \text{ else } 0.01, \text{if } sick(X) \text{ then } death\ 0.55\}$ and uses inversion elimination only. Figure 4 shows that this model can have a very large tree width when propositionalized but can be treated as a linear graph by lifted inference. The second one, (II) in figure 3, answers query $P(r)$ from $p(X) \wedge p(Y) \wedge r\ 0.51, X \neq Y$ and uses counting elimination only. In both cases propositional inference starts taking very long before any noticeable variation in lifted inference run times.

PROCEDURE *SHATTER*(G,Q)
G a set of parfactors, Q a set of atoms.
1. If there exist improper atom pair p, q in $A_G \cup Q$
   (a) For each $r \in \{p, q\}$
   If r comes from parfactor g
   i. $g' \leftarrow$ *NORMALIZE*$(\phi_g, A_g, C_g \wedge MGU(p, q))$.
   ii. $g'' \leftarrow$ *NORMALIZE*$(\phi_g, A_g, C_g \wedge \neg MGU(p, q))$.
   iii. $G \leftarrow (G \setminus \{g\}) \cup \{g', g''\}$.
   (b) Return *SHATTER*(G, Q).
2. Return G.

PROCEDURE *NORMALIZE*(g)
g a parfactor.
1. If there exists a pair of atoms p, q in A_g unified in C_g replace q by p in g.
2. Return *NORMALIZE*(g).

Figure 2: Shattering algorithm.

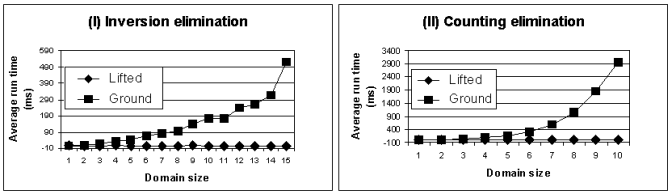


Figure 3: (I) Average run time for answering query $P(death)$ from $\{epidemic\ 0.55,\ \text{if } epidemic \text{ then } sick(X)\ 0.7 \text{ else } 0.01, \text{if } sick(X) \text{ then } death\ 0.55\}$, which requires inversion elimination only. (II) Average run time for answering query $P(r)$ from $p(X) \wedge p(Y) \wedge r\ 0.51, X \neq Y$, which requires counting elimination.

6 Discussion

We presented and showed the correctness of a lifted first-order probabilistic inference algorithm, the first one to our knowledge that covers all cases in its intended language. This allows expressive representations whose inference is made much cheaper by abstracting away from specific instances of random variables and dealing instead with whole classes of them at once. We believe this type of algorithm will be essential to the development of large and expressive probabilistic systems, especially when the particular model is not known in advance and a general and automatic approach is necessary.

We presented two ways of eliminating variables: inversion and counting elimination. Counting elimination is potentially much more expensive than inversion elimination, but we expect its occurrence in practical problems to be low; in any case, we believe that much better counting arguments exist. Investigating them is an interesting line for further research.

Many other interesting directions remain to be taken. A very natural extension would be to allow non-ground queries that produce not only probabilities but also bindings for logical variables. Also, the algorithm can be adapted, in a way similar to [Pfeffer and Koller, 2000], in order to work

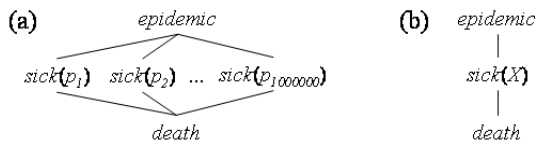


Figure 4: Computing an answer to query $P(death)$ from the epidemic model and a million people is expensive for the propositional grounded model (a) as it has a large tree width, but cheap for the lifted model (b) since it is a linear graph.

with infinite models, allowing for richer constraint languages where constraints may have infinite solutions, as for example those with function symbols. This is also related to Constraint Logic Programming [Van Hentenryck, 1989]. Given the complexity of the language, approximation schemes will be very important for practical applications; counting elimination seems a particularly good place to start given its relatively high cost but also regularity. Techniques from theorem proving will be particularly useful when models with a large number of parfactors are necessary and one has to apply them wisely. A complexity study is also necessary for, among other things, guiding the choice of efficient elimination orderings.

Much of the gain in performance from a lifted algorithm comes from the presence of a large number of indistinguishable objects in the domain, that is, objects about which we have exactly the same knowledge. It has been argued ([Chavira *et al.*, 2004]) that this does not occur often in practical applications. However, the current work simply provides a base for extensions with other important benefits. In an approximate inference setting, for example, the notion of indistinguishable objects is replaced by that of objects about which there is approximately the same knowledge (according to the current approximation factor), a much more practical situation. For non-ground queries, a problem of great practical interest, lifted inference is much more suitable, since the answers to such queries may be lifted themselves.

Acknowledgments

We would like to thank Vasin Punyakanok, Dav Zimak and the anonymous reviewers for their helpful comments. This research was supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program, by NSF grant ITR-IIS-0085980 and DAF Air Force Research Laboratory grant FA8750-04-2-0222 (DARPA REAL program).

References

[Anderson *et al.*, 2002] C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov models and their application to adaptive web navigation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, 2002.

[Chavira *et al.*, 2004] M. Chavira, A. Darwiche, and M. Yaeger. Compiling relational Bayesian networks for exact inference. In *Proceedings of the Second European Workshop on Probabilistic Graphical Models*, pages 49–56, 2004.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309, 1999.

[Halpern, 1990] J. Y. Halpern. An analysis of first-order logics of probability. In *Proceedings of IJCAI-89, 11th International Joint Conference on Artificial Intelligence*, pages 1375–1381, Detroit, US, 1990.

[Jaeger, 1997] M. Jaeger. Relational Bayesian networks. In Morgan Kaufmann, editor, *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 266–273, 1997.

[Kersting and De Raedt, 2000] K. Kersting and L. De Raedt. Bayesian logic programs. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 138–155, 2000.

[Laskey, 2005] K. B. Laskey. First-order Bayesian logic. Technical report, George Mason University Department of Systems Engineering and Operations Research, 2005.

[Ng and Subrahmanian, 1992] R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[Ngo and Haddawy, 1995] L. Ngo and P. Haddawy. Probabilistic logic programming and Bayesian networks. In *Asian Computing Science Conference*, pages 286–300, 1995.

[Pfeffer and Koller, 2000] A. Pfeffer and D. Koller. Semantics and inference for recursive probability models. In *AAAI/IAAI*, pages 538–544, 2000.

[Pfeffer *et al.*, 1999] A. Pfeffer, D. Koller, B. Milch, and Takusagawa. K. T. SPOOK: A system for probabilistic object oriented knowledge representation. In *Proceedings of the 14th Annual Conference on Uncertainty in AI (UAI-99)*, pages 541–550, 1999.

[Poole, 1993] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[Poole, 2003] D. Poole. First-order probabilistic inference. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 985–991, 2003.

[Richardson and Domingos, 2004] M. Richardson and P. Domingos. Markov logic networks. Technical report, Department of Computer Science, University of Washington, 2004.

[Van Hentenryck, 1989] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming*. Logic Programming Series, The MIT Press, Cambridge, MA, 1989.

[Zhang and Poole, 1994] N. L. Zhang and D. Poole. A simple approach to Bayesian network computations. In *Proceedings of the Tenth Biennial Canadian Artificial Intelligence Conference*, 1994.

Stationary Deterministic Policies for Constrained MDPs with Multiple Rewards, Costs, and Discount Factors

Dmitri Dolgov and Edmund Durfee
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109
{ddolgov, durfee}@umich.edu

Abstract

We consider the problem of policy optimization for a resource-limited agent with multiple time-dependent objectives, represented as an MDP with multiple discount factors in the objective function and constraints. We show that limiting search to stationary deterministic policies, coupled with a novel problem reduction to mixed integer programming, yields an algorithm for finding such policies that is computationally feasible, where no such algorithm has heretofore been identified. In the simpler case where the constrained MDP has a single discount factor, our technique provides a new way for finding an optimal deterministic policy, where previous methods could only find randomized policies. We analyze the properties of our approach and describe implementation results.

1 Introduction

Markov decision processes [Bellman, 1957] provide a simple and elegant framework for constructing optimal policies for agents in stochastic environments. The classical MDP formulations usually maximize a measure of the aggregate reward received by the agent. For instance, in widely-used discounted MDPs, the objective is to maximize the expected value of a sum of exponentially discounted scalar rewards received by the agent. Such MDPs have a number of very nice properties: they are subject to the principle of local optimality, according to which the optimal action for a state is independent of the choice of actions for other states, and the optimal policies for such MDPs are stationary, deterministic, and do not depend on the initial state of the system. These properties translate into very efficient dynamic-programming algorithms for constructing optimal policies for such MDPs (e.g., [Puterman, 1994]), and policies that are easy to implement in standard agent architectures.

This material is based upon work supported by Honeywell International, and by the DARPA/IPTO COORDINATORs program and the Air Force Research Laboratory under Contract No. FA8750–05–C–0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

However, there are numerous domains where the classical MDP model proves inadequate, because it can be very difficult to fold all the relevant feedback from the environment (i.e., rewards the agent receives and costs it incurs) into a single scalar reward function. In particular, the agent's actions, in addition to producing rewards, might also incur costs that might be measured very differently from the rewards, making it hard or impossible to express both on the same scale. For example, a natural problem for a delivery agent is to maximize aggregate reward for making deliveries, subject to constraints on the total time spent en route. Problems naturally modeled as constrained MDPs also often arise in other domains: for example, in telecommunication applications (e.g., [Lazar, 1983]), where it is desirable to maximize throughput subject to delay constraints.

Another situation where the classical MDP model is not expressive enough is where an agent receives multiple reward streams and incurs multiple costs, each with a different discount factor. For example, the delivery agent could face a rush-hour situation where the rewards for making deliveries decrease as a function of time (same delivery action produces lower reward if it is executed at a later time), while the traffic conditions improve with time (same delivery action can be executed faster at a later time). If the rewards decrease and traffic conditions improve on different time scales, the problem can be naturally modeled with two discount factors, allowing the agent to evaluate the tradeoffs between early and late delivery. Problems with multiple discount factors also frequently arise in other domains: for example, an agent can be involved in several financial ventures with different risk levels and time scales, where a model with multiple discount factors would allow the decision maker to quantitatively weigh the tradeoffs between shorter- and longer-term investments. Feinberg and Shwartz [1999] describe more examples and provide further justification for constrained models with several discount factors.

The price we have to pay for extending the classical model by introducing constraints and several discount factors is that stationary deterministic policies are no longer guaranteed to be optimal [Feinberg and Shwartz, 1994; 1995]. Searching for an optimal policy in the larger class of non-stationary randomized policies can dramatically increase problem complexity; in fact, the complexity of finding optimal policies for this broad class of constrained MDPs with multiple costs, re-

wards, and discount factors is not known, and no solution algorithms exist (aside from some very special cases [Feinberg and Shwartz, 1996]). Furthermore, even if they could be found, these non-stationary randomized policies might not be reliably executable by basic agent architectures. For example, [Paruchuri *et al.*, 2004] described how executing randomized policies in multiagent systems can be problematic.

In this paper, therefore, we focus on finding optimal stationary deterministic policies for MDPs with multiple rewards, costs, and discount factors. This problem has been studied before and has been proven to be NP-complete by Feinberg [2000], who formulated it as a non-linear and non-convex mathematical program. Unfortunately, aside from intractable techniques of general non-convex optimization, these problems have heretofore not been practically solvable.

Our contribution in this paper is to present an approach to solving this problem that reduces it to a mixed-integer linear program – a formulation that, while still NP-complete, has available a wide variety of very efficient solution algorithms and tools that make it practical to often find optimal stationary deterministic policies. As we will show, moreover, our approach can also be fruitfully employed for the subclass of MDPs that have multiple costs, but only a single reward function and discount factor. For these problems, linear programming can, in polynomial time, find optimal stationary randomized policies [Kallenberg, 1983; Heyman and Sobel, 1984], but the problem of finding optimal stationary deterministic policies is NP-complete [Feinberg, 2000], with no implementable solution algorithms existing previously (aside from the general non-linear optimization techniques). We show that our integer-programming-based approach finds optimal stationary deterministic policies, which can then be compared empirically to optimal randomized policies.

In the remainder of this paper, we first (in Section 2) establish a baseline by briefly reviewing techniques for solving unconstrained MDPs. In Section 3, we move on to constrained MDPs, and present our approach to solving constrained MDPs with a single reward, multiple costs, and one discount factor for the rewards and costs. We next expand this to the case with multiple rewards and costs, and several discount factors, in Section 4. Section 5 provides some empirical evaluations and observations, and Section 6 discusses our results and some thoughts about applying the same techniques to other flavors of constrained MDPs.

2 Background: Unconstrained MDPs

An unconstrained, stationary, discrete-time, fully-observable MDP can be defined as a 4-tuple $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$, where $\mathcal{S} = \{i\}$ is a finite set of states the agent can be in; $\mathcal{A} = \{a\}$ is a finite set of actions the agent can execute; $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the stochastic ($\sum_j p_{iaj} = 1$) transition function (p_{iaj} is the probability the agent goes to state j if it executes action a in state i); $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the bounded reward function (the agent gets a reward of r_{ia} for executing action a in state i).

A solution to an MDP is a policy (a procedure for selecting an action in every state) that maximizes some measure of aggregate reward. In this paper we will focus on MDPs with the total expected discounted reward optimization criterion, but our results can be extended to other optimization criteria (as discussed in Section 6). A policy is said to be *Markovian* (or *history-independent*) if the choice of action does not depend on the history of states and actions encountered in the past, but only on the current state and time. If, in addition to that, the policy does not depend on time, it is called *stationary* (by definition, a stationary policy is always Markovian). A *deterministic* policy always prescribes the execution of the same action in a state, while a *randomized* policy chooses actions according to a probability distribution.

A stationary randomized policy π can be described as a mapping of states to probability distributions over actions: $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, where π_{ia} defines the probability that the agent will execute action a when it encounters state i. A deterministic policy can be viewed as a degenerate case of a randomized policy for which there is only one action for each state that has a nonzero probability of being executed.

A policy π and the initial conditions $\alpha : \mathcal{S} \mapsto [0, 1]$ that specify the probability distribution over the state space at time 0 (the agent starts in state i with probability α_i) together determine the evolution of the system and the total expected discounted reward the agent will receive:

$$U_\gamma(\pi, \alpha) = \sum_{t=0}^{\infty} \sum_{i,a} \gamma^t \varphi_i(t) \pi_{ia} r_{ia}, \tag{1}$$

where $\varphi_i(t)$ refers to the probability of being in state i at time t, and $\gamma \in [0, 1)$ is the discount factor.

It is well-known (e.g., [Puterman, 1994]) that, for an unconstrained MDP with the total expected discounted reward optimization criterion, there always exists an optimal policy π^* that is stationary, deterministic, and *uniformly-optimal*, where the latter term means that the policy is optimal for all initial probability distributions over the starting state (i.e., $U_\gamma(\pi^*, \alpha) \geq U_\gamma(\pi, \alpha)\ \forall \pi, \alpha$).

There are several standard ways of solving such MDPs (e.g., [Puterman, 1994]); some use dynamic programming (value or policy iteration), others, which are much better suited for constrained problems, reduce MDPs to linear programs (LPs). A discounted MDP can be formulated as the following LP [D'Epenoux, 1963; Kallenberg, 1983] (this maximization LP is the dual to the more-commonly used minimization LP in the value function coordinates):

$$\max \sum_{i,a} r_{ia} x_{ia} \left| \begin{array}{l} \sum_a x_{ja} - \gamma \sum_{i,a} x_{ia} p_{iaj} = \alpha_j, \\ x_{ia} \geq 0. \end{array} \right. \tag{2}$$

The set of optimization variables x_{ia} is often called the *occupation measure* of a policy, where x_{ia} can be interpreted as the total expected discounted number of times action a is executed in state i. Then, $\sum_a x_{ia}$ gives the total expected discounted *flow* through state i, and the constraints in the above LP can be interpreted as the conservation of flow through each of the states. An optimal policy can be computed from a solution to the above LP as:

$$\pi_{ia} = x_{ia} / \sum_a x_{ia}. \tag{3}$$

Although this appears to lead to randomized policies, in the absence of external constraints, and if we use strictly positive

initial conditions ($\alpha_i > 0$), a basic feasible solution to this LP always maps to a deterministic policy that is uniformly-optimal [Puterman, 1994; Kallenberg, 1983]. This LP (2) for the unconstrained MDP serves as the basis for solving constrained MDPs that we discuss next.

3 Constrained MDPs

Suppose that the agent, besides getting rewards for executing actions, also incurs costs: $c^k : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R},\ k \in [1, K]$, where c^k_{ia} is the cost of type k incurred for executing action a in state i (e.g., actions might take time and consume energy, in which case we would say that there are two types of costs). Then, a natural problem to pose is to maximize the expected discounted reward subject to some upper bounds on the total expected discounted costs. Let us label the total expected discounted cost of type $k \in [1, K]$ as:

$$C^k_\gamma(\pi, \alpha) = \sum_{t=0}^{\infty} \sum_{i,a} \gamma^t \varphi_i(t) \pi_{ia} c^k_{ia}. \tag{4}$$

Then, we can abstractly write the optimization problem with cost constraints as

$$\max_\pi U_\gamma(\pi, \alpha) \,\big|\, C^k_\gamma(\pi, \alpha) \le \widehat{c}^k, \tag{5}$$

where $\widehat{c}^k$ is the upper bound on the cost of type k. If this problem is feasible, then there always exists an optimal stationary policy, and it can be computed as a solution to the following LP [Kallenberg, 1983; Heyman and Sobel, 1984]:

$$\max \sum_{i,a} r_{ia} x_{ia} \left| \begin{array}{l} \sum_a x_{ja} - \gamma \sum_{i,a} x_{ia} p_{iaj} = \alpha_j, \\ \sum_{i,a} c^k_{ia} x_{ia} \le \widehat{c}^k, \\ x_{ia} \ge 0. \end{array} \right. \tag{6}$$

Therefore, constrained MDPs of this type can be solved in polynomial time, i.e., adding constraints with the same discount factor does not increase the complexity of the MDP. However, due to the addition of constraints, the problem (5), in general, will not have uniformly-optimal policies. Furthermore, the LP (6) will yield randomized policies, which (as argued in Section 1) are often more difficult to implement than deterministic ones.

Thus, it can be desirable to compute optimal solutions to (5) from the class of stationary deterministic policies. This, however, is a much harder problem: Feinberg [2000] studied this problem, showed that it is NP-complete (using a reduction similar to [Filar and Krass, 1994]), and reduced it to a mathematical program by augmenting (6) with the following constraint, ensuring that only one x_{ia} per state is nonzero:

$$|x_{ia} - x_{ia'}| = x_{ia} + x_{ia'} \tag{7}$$

However, the resulting program (6,7) is neither linear nor convex, and thus presents significant computational challenges.

We show how (5) can be reduced to a mixed integer linear program (MILP) that is equivalent to (6,7). This is beneficial because MILPs are well-studied (e.g., [Wolsey, 1998]), and there exist efficient implemented algorithms for solving them. Our reduction uses techniques similar to the ones employed in [Dolgov and Durfee, 2004b], where an MDP with limited non-consumable resources is reduced to a MILP. The following proposition provides the basis for our reduction.

Proposition 1 *Consider an MDP $\langle \mathcal{S}, \mathcal{A}, p, r, \alpha \rangle$, a policy π, its corresponding occupation measure x (given α), a constant $X \ge x_{ia}\ \forall i \in \mathcal{S}, a \in \mathcal{A}$, and a set of binary variables $\Delta_{ia} = \{0, 1\},\ \forall i \in \mathcal{S}, a \in \mathcal{A}$.*

If x and Δ satisfy the following conditions

$$\sum_a \Delta_{ia} \le 1, \quad \forall i \in \mathcal{S}, \tag{8}$$

$$x_{ia}/X \le \Delta_{ia}, \quad \forall i \in \mathcal{S}, a \in \mathcal{A} \tag{9}$$

then, for all states i that, under π and α, have a nonzero probability of being visited ($\sum_a x_{ia} > 0$), π is deterministic, and the following holds:

$$\Delta_{ia} = 1 \Leftrightarrow x_{ia} > 0 \tag{10}$$

Proof: Consider a state i^* that, under policy π and initial distribution α, has a nonzero probability of being visited, i.e., $\sum_a x_{i^*a} > 0$. Then, since the occupation measure is non-negative, there must be at least one action in this state that has a non-zero occupation measure:

$$\exists a^* \in \mathcal{A} \quad \text{s.t.} \quad x_{i^*a^*} > 0.$$

Then, (9) forces $\Delta_{i^*a^*} = 1$, which, due to (8), forces zero values for all other Δ's for state i^*:

$$\Delta_{i^*a} = 0 \quad \forall a \ne a^*.$$

Given (9), this, in turn, means that the occupation measure for all other actions has to be zero:

$$x_{i^*a} = 0 \quad \forall a \ne a^*,$$

which, per (3), translates into the fact that the policy π is deterministic and $\Delta_{ia} = 1 \Leftrightarrow x_{ia} > 0$. ■

Proposition 1 immediately leads to the following MILP whose solution yields optimal stationary deterministic policies for (5):

$$\max \sum_i \sum_a x_{ia} r_{ia} \left| \begin{array}{l} \sum_a x_{ja} - \gamma \sum_{i,a} x_{ia} p_{iaj} = \alpha_j, \\ \sum_{i,a} c^k_{ia} x_{ia} \le \widehat{c}^k, \\ \sum_a \Delta_{ia} \le 1, \\ x_{ia}/X \le \Delta_{ia}, \\ x_{ia} \ge 0, \quad \Delta_{ia} \in \{0, 1\}, \end{array} \right. \tag{11}$$

where X can be computed in polynomial time by, for example, solving the LP (2) with the objective function replaced by $\max \sum_{i,a} x_{ia}$ and setting X to its maximum value.

The above reduction to an MILP is most valuable for domains where it is difficult to implement a randomized stationary policy because of an agent's architectural limitations. It is also of interest for domains where such limitations are not present, as it can be used for evaluating the quality vs. implementation-difficulty tradeoffs between randomized and deterministic policies during the agent-design phase.

4 Constrained MDPs with Multiple Discounts

We now turn our attention to the more general case of MDPs with multiple streams of rewards and costs, each with its own

discount factor $\gamma_n,\ n \in [1, N]$. The total expected reward is a weighted sum of the N discounted reward streams:

$$U(\pi, \alpha) = \sum_n \beta_n U_{\gamma_n}(\pi, \alpha) = \sum_n \beta_n \sum_{t=0}^{\infty} \sum_{i,a} \gamma_n^t \varphi_i(t) \pi_{ia} r_{ia}^n$$

where β_n is the weight of the n^{th} reward stream that is defined by the reward function $r^n : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. Similarly, each of the K total expected costs is a weighted sum of N cost streams:

$$C^k(\pi, \alpha) = \sum_n \beta_{kn} C^k_{\gamma_n}(\pi, \alpha) = \sum_{n,i,a} \beta_{kn} \sum_{t=0}^{\infty} \gamma_n^t \varphi_i(t) \pi_{ia} c_{ia}^{kn}$$

where β_{kn} is the weight of the n^{th} discounted stream of cost of type k, defined by the cost function $c^{kn} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$.

Notice that in this MDP with multiple discount factors, we have N reward functions and KN cost functions (unlike the constrained MDP from the previous section that had 1 and K reward and cost functions, respectively).

Our goal is to maximize the total weighted discounted reward, subject to constraints on weighted discounted costs:

$$\max_{\pi} U(\pi, \alpha) \,\Big|\, C^k(\pi, \alpha) \leq \widehat{c}^k, \quad \forall k \in [1, K]. \tag{12}$$

Feinberg and Shwartz [1994; 1995] developed a general theory of constrained MDPs with multiple discount factors and demonstrated that, in general, optimal policies are neither deterministic nor stationary. However, except for some special cases [Feinberg and Shwartz, 1996], there are no implementable algorithms for finding optimal policies for such problems. Because of this, and given the complexity of implementing non-stationary randomized policies (even if we could find them), it is worthwhile to consider the problem of constructing optimal stationary deterministic policies for such MDPs. Feinberg [2000] showed that finding optimal policies that belong to the class of stationary (randomized or deterministic) policies is an NP-complete task. He also formulated the problem of finding optimal stationary policies as the following mathematical program (again, based on (eq. 6)):

$$\max \sum_n \beta_n \sum_{i,a} r_{ia}^n x_{ia}^n \left| \begin{array}{l} \sum_a x_{ja}^n - \gamma_n \sum_{i,a} x_{ia}^n p_{iaj} = \alpha_j, \\ \sum_n \beta_{kn} \sum_{i,a} c_{ia}^{kn} x_{ia}^n \leq \widehat{c}^k, \\ x_{ia}^n / \sum_a x_{ia}^n = x_{ia}^{n+1} / \sum_a x_{ia}^{n+1}, \\ x_{ia}^n \geq 0, \end{array} \right. \tag{13}$$

This program has an occupation measure x^n for each discount factor $\gamma_n,\ n \in [1, N]$ and expresses the total reward and total costs as weighted linear functions of these occupation measures. The first set of constraints contains the conservation of flow constraints for each of the N occupation measures, and the third set of constraints ensures that all occupation measures map to the same policy (recall (3)).

As in the previous section, we can limit the search to deterministic policies by imposing the following additional constraint on the occupation measures in (13) [Feinberg, 2000]:

$$\left| \sum_n \left(x_{ia}^n - x_{ia'}^n \right) \right| = \sum_n \left(x_{ia}^n + x_{ia'}^n \right) \tag{14}$$

Because of the synchronization of the different occupation measures and the constraint that forces deterministic policies, this program (13,14) is non-linear and non-convex, and is thus very difficult to solve.

For finding optimal stationary deterministic policies, we present a reduction of the program (12) to a linear integer program that is equivalent to (13,14). Just like in the previous section, this reduction to an MILP allows us to exploit a wide array of efficient solution techniques and tools. Our reduction is based on the following proposition.

Proposition 2 *Consider an MDP $\langle \mathcal{S}, \mathcal{A}, p, r, \alpha \rangle$ with several discount factors $\gamma_n,\ n \in [1, N]$, a set of policies $\pi^n,\ n \in [1, N]$ with their corresponding occupation measures x^n (policy π^n and discount factor γ_n define x^n), a constant $X \geq x_{ia}^n\ \forall n \in [1, N], i \in \mathcal{S}, a \in \mathcal{A}$, and a set of binary variables $\Delta_{ia} = \{0, 1\}$.*

If x^n and Δ satisfy the following conditions

$$\sum_a \Delta_{ia} \leq 1, \quad \forall i \in \mathcal{S}, \tag{15}$$

$$x_{ia}^n / X \leq \Delta_{ia}, \quad \forall n \in [1, N], i \in \mathcal{S}, a \in \mathcal{A}, \tag{16}$$

then, the sets of reachable states $\mathcal{I}^n = \{i : \sum_a x_{ia}^n > 0\}$ defined by all occupation measures are the same, i.e., $\mathcal{I}^n = \mathcal{I}^{n'},\ \forall n, n' \in [1, N]$. Furthermore, all π^n are deterministic on $\mathcal{I}^n$, and $\pi_{ia}^n = \pi_{ia}^{n'}\ \forall n, n' \in [1, N]$.

Proof: Consider an initial state i^* (i.e., $\alpha_{i^*} > 0$). Following the argument of Proposition 1, the policy for that state is deterministic:

$$\exists a^* :\ x_{i^*a^*}^n > 0, \Delta_{i^*a^*} = 1;\ x_{i^*a}^n = 0, \Delta_{i^*a} = 0\ \forall a \neq a^*$$

This implies that all N occupation measures x^n must prescribe the execution of the same deterministic action a^* for state i^*, because all x_{ia}^n are tied to the same Δ_{ia} via (16).

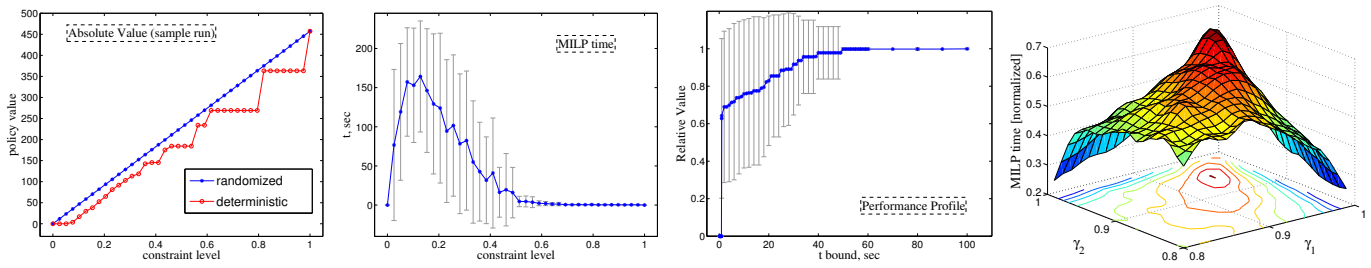
Therefore, all occupation measures x^n correspond to the same deterministic policy on the initial states $\mathcal{I}_0 = \{i : \alpha_i > 0\}$. We can then expand this statement by induction to all reachable states. Indeed, clearly the set of states $\mathcal{I}_1$ that are reachable from $\mathcal{I}_0$ in one step will be the same for all x^n. Then, by the same argument as above, all x^n map to the same deterministic policy on $\mathcal{I}_1$, and so forth. ■

It immediately follows from Proposition 2 that the problem of finding optimal stationary deterministic policies for an MDP with weighted discounted rewards and constraints (12) can be formulated as the following MILP:

$$\max \sum_n \beta_n \sum_{i,a} r_{ia}^n x_{ia}^n \left| \begin{array}{l} \sum_a x_{ja}^n - \gamma_n \sum_{i,a} x_{ia}^n p_{iaj} = \alpha_j, \\ \sum_n \beta_{kn} \sum_{i,a} c_{ia}^{kn} x_{ia}^n \leq \widehat{c}^k, \\ x_{ia}^n / X \leq \Delta_{ia}, \\ \sum_a \Delta_{ia} \leq 1, \\ x_{ia}^n \geq 0, \Delta_{ia} \in \{0, 1\}, \end{array} \right. \tag{17}$$

where $X \geq \max x_{ia}^n$ is a constant, as in Proposition 2.

Although this MILP produces policies that are only optimal in the class of stationary deterministic ones, at present



(a) (b) (c) (d)

Figure 1: Value of deterministic and randomized policies (a); solution time (b) and profile (c); MDP with two discounts (d).

time there are (to the best of our knowledge) no practical algorithms for finding optimal solutions from any larger policy class for constrained MDPs with multiple discount factors.

5 Experimental Observations

We have implemented the MILP algorithm for finding optimal stationary deterministic policies for constrained MDPs and empirically evaluated it on a class of test problems. In the following discussion, we focus on the constrained MDPs from Section 3, because these problems are better studied, and the existing algorithms for finding optimal randomized policies can serve as benchmarks, whereas there are no alternative algorithms for finding policies that are optimal for general constrained MDPs with multiple discount factors.

In our empirical analysis, we tried to answer the following questions: 1) how well do deterministic policies perform, compared to optimal randomized ones, and 2) what is the average-case complexity of the resulting MILPs. The answers to these questions are obviously domain-dependent, so the following discussion should not be viewed as a comprehensive characterization of the behavior of our algorithms on constrained MDPs. However, we believe that our experiments provide some interesting observations about such problems.

We experimented with a large set of randomly-generated problems and with a more meaningful manually-constructed domain, which we randomly perturbed in several ways. The big picture resulting from the experiments on the randomly-generated domains was very similar to the one from the manually-constructed example, providing a certain measure of comfort about the stability and validity of our observations. We report the results for the manually-constructed domain.

For our test domain, we used a simplistic model of an autonomous delivery agent, as mentioned in the introduction (based on the multiagent example from [Dolgov and Durfee, 2004b]). In the domain, an agent is operating in a grid world with delivery sites placed randomly throughout the grid. The agent moves around the grid (incurring small negative rewards for every move) and receives positive rewards for making deliveries. The agent's movement is nondeterministic, and the agent has some probability of getting stuck in randomly-placed dangerous locations. The agent also incurs a scalar cost (e.g., time) per move, and the objective is to maximize the total expected discounted reward subject to an upper bound on the total expected discounted cost.

The results of our experiments are summarized in Figure 1. Figure 1a shows the values of randomized and deterministic policies as functions of the constraint level ($[0, 1]$), where 0 means that only policies that incur zero cost are feasible (strictest possible constraint), whereas 1 means that the upper bound on cost equals the cost of the optimal unconstrained policy (agent is not constrained at all). The first observation, as illustrated in Figure 1a, is that the value of stationary deterministic policies for constrained problems is reasonably close to optimal. We can also observe that the value of deterministic policies changes in a very discrete manner (i.e., it jumps up at certain constraint levels), whereas the value of randomized policies changes continuously. This is, of course, only natural, given that the space of randomized policies is continuous, and randomized policies can gradually increase the probability of taking "better" actions as cost constraints are relaxed. On the other hand, the space of deterministic policies is discrete, and their quality jumps when the constraints are relaxed to permit the agent to switch to a better action. While the number and the size of these jumps in the value function depends on the dynamics of the MDP, the high-level picture was the same in all of our experiments.

Figure 1b shows the running time of the MILP solver as a function of the constraint level (here and in Figure 1c the plots contain values averaged over 100 runs, with the error bars showing the standard deviation). The data indicates that our MILPs (11) have an easy-hard-easy complexity profile, although without a sharp phase transition from hard to easy, i.e., the problems very quickly become hard, and then gradually get easier as cost constraints are relaxed.

This complexity profile gives rise to the question regarding the source of the difficulty for solving MILPs in the "hard" region: is it difficult to find good feasible solutions, or is it time-consuming to prove their optimality? Figure 1c suggests that the latter is the case, which can be considered as the more fortunate outcome, since algorithms with such performance profiles can be successfully used in an anytime manner. The figure contains a plot of the quality of the best solution found as a function of the time bound imposed on the MILP solver[1] for problems in the hardest constraint region (constraint level value of 0.13). As the graph shows, very good policies are usually produced rather quickly.

Let us conclude with a somewhat intriguing observation about the MILP solution time for constrained MDPs with multiple discount factors (Section 4). We generated and solved a large number of random MDPs with two discount

[1]CPLEX 8.1 on a P4 performed the role of the MILP solver.