

AI AND DATA INTEGRATION

Learning Value Predictors for the Speculative Execution of Information Gathering Plans

Greg Barish and Craig A. Knoblock

University of Southern California / Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
{barish, knoblock}@isi.edu

Abstract

Speculative execution of information gathering plans can dramatically reduce the effect of source I/O latencies on overall performance. However, the utility of speculation is closely tied to how accurately data values are predicted at runtime. Caching is one approach that can be used to issue future predictions, but it scales poorly with large data sources and is unable to make intelligent predictions given previously unseen input data, even when there is an obvious relationship between past input and the output it generated. In this paper, we describe a novel way to combine classification and transduction for a more efficient and accurate value prediction strategy, one capable of issuing predictions about previously unseen hints. We show how our approach results in significant speedups for plans that query multiple sources or sources that require multi-page navigation.

1 Introduction

The performance of Web information gathering plans can suffer because of I/O latencies associated with the remote sources queried by these plans. A single slow Web source can bottleneck an entire plan and lead to poor execution time. When a plan requires multiple queries (either to the same source or to multiple sources), performance can be even worse, where the overhead is a function of the slowest sequence of sources queried.

When multiple queries are required, speculative plan execution (Barish and Knoblock 2002b) can be used to dramatically reduce the impact of aggregate source latencies. The idea involves using data seen early in plan execution as a basis for issuing predictions about data likely to be needed during later parts of execution. This allows data dependency chains within the plan to be broken and parallelized, leading to significant speedups.

To maximize the utility of speculative execution, a good value prediction strategy is necessary. The basic problem involves being able to use some hint h as the basis for issuing a predicted value v . One approach involves caching: we can note that particular hint h_x corre-

sponds to a particular value v_y so that future receipt of h_x can lead to prediction of v_y . As a result, a plan that normally queries source S_1 with h_x and subsequently source S_2 with v_y can be parallelized so that both S_1 and S_2 are queried in parallel, the latter speculatively. Unfortunately, caching has two major drawbacks. First, it does not scale well when the domain of hints is large. A second drawback is the inability to deal with *novel* (previously unseen) hints, even when an obvious relationship exists between hint and predicted value.

In this paper, we present an alternative to caching that involves automatically learning predictors that combine classification and transduction in order to generate predictions from hints. Our approach succeeds where caching fails: the predictors learned usually consume less space than that demanded by caching and they are capable of making reasonable predictions when presented with novel hints, the latter leading to better speedups. Specifically, our work contributes the following:

- An algorithm that learns efficient transducers capable of a variety of string transformations.
- An algorithm that combines classification and transduction to learn value predictors

The rest of this paper is organized as follows. The next section briefly reviews information gathering and provides a motivating example for speculative execution. In Section 3, we describe how classification and transduction can be used to build efficient and intelligent predictors. Section 4 describes our learning algorithms that combine both techniques. Section 5 describes experimental results of using our approach. Finally, Section 6 details the related work.

2 Preliminaries

Web information gathering plans retrieve, combine, and manipulate data located in remote Web sources. Such plans consist of a partially-ordered graph of operators $O_1..O_n$ connected in a producer/consumer fashion. Each operator O_i consumes a set of inputs $a_1..a_p$, fetches data or performs a computation based on that input, and produces one or more outputs $b_1..b_q$. The types of operators used in information gathering plans vary, but most either retrieve or perform computations on data.

To better illustrate a Web information gathering plan, we consider the example plan *CarInfo*, shown in Figure 1. Given constraints on car type and pricing, *CarInfo* locates the make and model which has a median price closest to that specified and then retrieves the full review of that car from the Web site ConsumerGuide.com. The plan consists of four *Wrapper* operators that fetch and extract data from various parts of the remote source. Specifically, the plan involves: (a) querying CarsDirect.com for the car make and model having a median price closest to that specified, (b) querying ConsumerGuide for the resulting make and model, (c) retrieving the link to the summary page for that car (using the link provided in the search results), and (d) retrieving the full review using the link provided on the summary page.

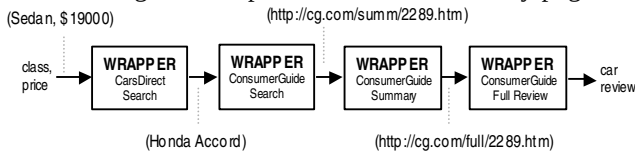


Figure 1: The CarInfo plan

For example, for the input (*Sedan*, \$19000), the car returned is (*Honda Accord*), the summary URL for this car is (<http://cg.com/summ/2289.htm>) and the full review URL (<http://cg.com/full/2289.htm>).¹ Once at the full review URL, the review text can be extracted.

Existing research on information agent executors (Barish and Knoblock 2002a) and network query engines (Hellerstein *et al.* 2000, Naughton *et al.* 2001, Ives *et al.* 2002) has suggested a number of techniques for the efficient execution of information gathering plans. Most of these systems work on the principle of streaming dataflow, which executes plan operators as fast as their data dependencies allow and supports the pipelining of data between operators. Despite the benefits of streaming dataflow, plan execution can still be slow if there are many binding pattern relationships between sources in the plan. For example, notice that since steps (b), (c), and (d) are dependent on the steps that precede them, the plan in Figure 1 must be executed sequentially and profits little from the benefit of streaming dataflow. Thus, if each source in Figure 1 has an average latency of 2s, then the average plan execution time is the summation of source latencies, or (4*2s =) 8s.

2.1 Speculative Plan Execution

Speculative execution is one technique that can be used to overcome the effects of aggregate latencies in information gathering plans containing multiple binding patterns.

As described in (Barish and Knoblock 2002b), a plan is transformed into one capable of speculative plan execution by the insertion of two additional operators – **Speculate** and **Confirm** – at various points the plan,

¹ For the sake of brevity, we have abbreviated the URLs associated with ConsumerGuide.com (although we retain the key aspects of its structure).

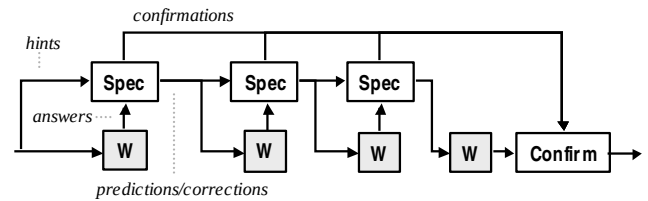


Figure 2: CarInfo transformed for speculative execution

based on an iterative analysis of the most expensive path to execute within that plan. For example, one possible result of transforming the plan in Figure 1 for speculative execution is shown in Figure 2.

As shown, the **Speculate** operator receives copies of data sent to operators executing earlier in the plan. Based on the hints it receives, **Speculate** can generate predicted values for later operators that can be transmitted immediately to those operators. Thus, the earlier and later parts of the plan can be parallelized. After the earlier operators finish executing, **Speculate** can assess whether or not its initial predictions were correct and forward the results onto a **Confirm** operator, which ensures that speculative data does not prematurely exit the plan or cause some other irreversible action to occur. Finally, notice that Figure 2 shows that speculation can be *cascading*: speculation about one operator can drive the speculation of another operator, leading to greater degrees of parallelism and thus arbitrary speedups.

As a result of the transformation shown in Figure 2, execution would then proceed as follows. Input data, such as (*Sedan*, \$19000), would result in the retrieval of the initial search results in parallel with the predicted make and model – which would drive the predictions of summary and full-review URLs so that all four retrievals (three speculative) were executed at once. If all predictions are correct, the resulting execution time can be reduced to only 2s plus the overhead to speculate, a maximum speedup of about 4. However, the **average** speedup depends on the average accuracy of prediction: *the greater this accuracy, the higher the average speedup*.

3 Value Prediction Strategies

Caching can be used to implement value prediction when speculatively executing plans such as *CarInfo*. Unfortunately, caching does not allow predictions to be issued for unseen hints. As a result, the average accuracy of prediction can be low when the domain of possible hints is large. Further, trying to achieve better accuracy under these conditions can require significant amounts of memory. In this section, we describe how a hybrid approach, consisting of classification and transduction, results in a more intelligent and space-efficient prediction strategy.

3.1 Classification

Classification involves extracting knowledge from a set of data (instances) that describes how the attributes of those instances are associated with a set of target classes. Given a set of instances, classification rules can be learned so that future instances can be classified cor-

rectly. Once learned, a classifier can also make reasonable predictions about new instances – a combination of attribute values that had not previously been seen. The ability for classification to accommodate new instances is intriguing for the speculative execution of information gathering plans because, unlike with caching, it is possible to make predictions about novel hints.

For example, consider the prediction of the make and model of a car in the *CarInfo* plan. It turns out that CarsDirect.com returns the same answer (*Honda Accord*) for “Sedan” as it does for other types (such as “All” and “Coupe”) in the same price range. The association of the same make and model to multiple criteria combinations occurs somewhat frequently on CarsDirect.com.

To see why classification is a more effective technique than caching for the prediction of the make and model, consider prediction of a car based on type and price:

Type	Price	Car
Sedan	18000	Saturn S Series
Sedan	19000	Honda Accord
Sedan	20000	VW Beetle
Coupe	18000	Saturn S Series
Coupe	19000	Honda Accord
Coupe	20000	VW Beetle

The data above is what a cache would contain. In contrast, a classifier like Id3 (Quinlan 1986) would induce the following decision tree:

```

pri <= 18000 : Saturn S Series (2.0)
pri > 18000 :
| pri <= 19000 : Honda Accord (2.0)
| pri > 19000 : VW Beetle (2.0)

```

When presented with an instance previously seen, such as (*Sedan*, 19000), both the cache and the classifier would result in the same prediction: (*Honda Accord*). However, when presented with a new instance, such as (*Coupe*, 18500), the cache would be unable to make a prediction. In contrast, the classifier would issue the correct prediction of (*Honda Accord*). Any errors made by classification would be caught automatically later in execution by the **Confirm** operator.

The decision tree above is also more space efficient than a cache for the same data. The cache requires storing 8 unique values. The decision tree above requires only storing 5 values (just those shown) plus the information required to describe tree structure and attribute value conditions (i.e., $pri \leq 18000$).

In short, classifiers such as decision trees can function as better, more space-efficient predictors. And in the worst case, where each instance corresponds to a unique class, a classifier simply emulates a cache.

3.2 Transduction

Transducers are finite state machines that transform input to output by using the former to iteratively proceed through a series of states that progressively produce the latter. One type of transducer is a *string-to-string sequential transducer*, defined by (Mohri 1997) as $T = (Q, i, F, \Sigma, \Delta, \delta, \sigma)$, where Q is the set of states, $i \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, Σ and Δ are

finite sets corresponding to input and output alphabets, δ is the state-transition function that maps $Q \times \Sigma$ to Q , and σ is the output function that maps $Q \times \Delta$ to Δ^* . We are interested in a particular type of sequential transducer called a p -subsequential transducer that allows at most p output symbols to be appended to the output (i.e., exist on the final state transition arc).

Value prediction by transduction makes sense for Web information gathering plans primarily because of how Web sources organize information and how Web requests (i.e., HTTP queries) are standardized. In the case of the former, Web sources often use predictable hierarchies to catalog information. For example, in the *CarInfo* example, the summary URL for the 1999 Honda Accord was <http://cg.com/summ/2289.htm> and the full review was <http://cg.com/full/2289.htm>. Notice that both URLs use the same piece of dynamic information (2289), but in different ways. By learning this transduction, we can then predict future full review URLs for corresponding summary URLs we have never previously seen. Transducers can also allow us to predict HTTP queries. For example, an HTTP GET query for the IBM stock chart is <http://finance.yahoo.com/q?s=ibm&d=c>. By exploiting the regularity of this URL structure, the system can predict the URL for the Cisco Systems (CSCO) chart.

In this paper, we define two new types of transducers that extend the traditional definition of p -subsequential transducers. The first is a high-level transducer, called a *value transducer* that describes how to construct the predicted value based on the regularity and transformations observed in a set of examples of past hints and values. Value transducers build the predicted value through substring-level operations {*Insert*, *Classify*, and *Transduce*}. **Insert** constructs the static parts of predicted values. **Classify** categorizes hint information into part of a predicted value. Finally, **Transduce** transforms hint information into part of a predicted value. **Transduce** uses a second type of special transducer, a *hint transducer*, in which the operations {*Accept*, *Copy*, *Replace*, *Upper*, *Lower*} all function on individual characters of the hint and perform the same transformation as their name implies, with respect to the predicted value.

To illustrate, consider the transducers shown in Figure 3, for predicting the full-review URL in the *CarInfo* example. Figure 3 shows the value transducer performs high-level operations – the insertion of substrings and the call to a lower-level transduction. The second transducer (in abbreviated form) uses the **Accept** and **Copy** opera-

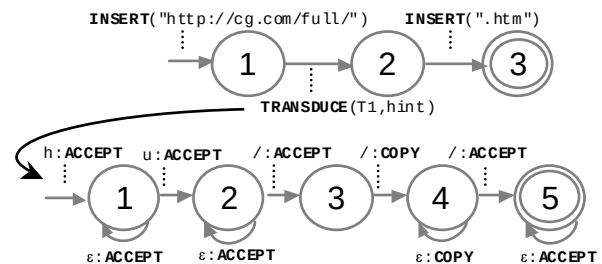


Figure 3: Value transducer for CarInfo

tions to transform the part of the hint value into its proper point in the predicted value. Thus, the first step builds the “*http://cg.com/full/*” part, the second step copies the “2289” part and the third step appends the “.htm” part.

In short, transducers lend themselves to value prediction because of the way information is stored by and queried from Web sources. They are a natural fit because URLs are strings that are often the result of simple transformations on earlier input. Transducers are also space efficient: once learned, they occupy a finite size and can be applied to an endless number of inputs. Overall, for Web content that cannot be queried directly (instead requiring an initial query and then further navigation), transducers serve as compact predictors that capitalize on the regularity of Web queries and source structure.

4 A Unifying Learning Algorithm

In this section, we present algorithms that induce value transducers (VTs), which combine classification and transduction, as predictors for the speculative execution of information gathering plans. The predictors learned represent not only a hybridization of classification and transduction, but also of caching, since classifiers default to caches when there is no feature of the key that is a particularly good indicator of the resulting value.

To learn a VT, the general approach consists of:

1. For each attribute of the answer tuple, identify an *SD Template* that distinguishes static from dynamic parts of the target string by analyzing the regularity between values of this attribute for all answers.
2. For each static part, add an **Insert** arc to the VT.
3. For each dynamic part, determine if transduction can be used; if so, add a **Transduce** arc to VT.
4. If no transducer can be found, classify the dynamic part based on the relevant attributes of the hint and add a **Classify** arc to the VT.

We implemented this in the algorithm LEARN-VALUE-TRANSDUCER, shown below. The algorithm takes a set of hints, a set of corresponding answers, and returns a VT that fits the data:

```

1  Function LEARN-VALUE-TRANSDUCER
   returns ValueTransducer
2  Input: the set of hints H, the set of answers A
3  VT  $\leftarrow \emptyset$ 
4  tmpl  $\leftarrow$  LEARN-SD-TEMPLATE (A);
5  ForEach element e in tmpl
6    If e is a static element
7      Add Insert (e.value) arc to VT
8    Else if e is a dynamic element
9      DA  $\leftarrow$  the set of dynamic strings in A for this tmpl element
10     HT  $\leftarrow$  LEARN-HINT-TRANSDUCER (H, DA)
11     If HT  $\neq \emptyset$ 
12       Add Transduce (HT) arc to VT
13     else
14       C  $\leftarrow$  LEARN-CLASSIFIER (H, DA)
15       Add Classify (C) arc to VT
16  Return VT
17 End /* LEARN-VALUE-TRANSDUCER */

```

In this algorithm, learning a classifier can be achieved by decision tree induction algorithms such as Id3 (Quinlan 1986). Learning the SD template and the hint transforming transducer, however, require unique algorithms.

4.1 Learning templates of string sets

Learning a VT requires first identifying a *template* for the target value that describes what parts are static and what parts are dynamic. Next, each static part is replaced with **Insert** operations and each dynamic part becomes a candidate for either transduction or classification.

To identify an SD template, we use an approach based on the *longest common subsequence* (LCS) between a set of values. First, an LCS identification algorithm similar to the one described by (Hirschberg 1975) is applied to the set of answer values. We then iterate through the LCS on each answer value to determine the set of possible static/dynamic templates that fit that answer. Only those templates common to all are kept – from this, one of the set is returned (though all are valid). The algorithm that implements this, LEARN-SD-TEMPLATE, is shown below.

```

1  Function LEARN-SD-TEMPLATE returns Template
2  Input: set of strings S
3  tmpl  $\leftarrow \emptyset$ 
4  lcs  $\leftarrow$  GET-LCS(S)
5  If seq  $\neq \emptyset$ 
6    tmplSet  $\leftarrow \emptyset$ 
7    ForEach string s in S
8      curTplSet  $\leftarrow$  EXTRACT-TEMPLATES (s, lcs)
9      tmplSet  $\leftarrow$  tmplSet  $\cap$  curTplSet
10   If tmplSet  $\neq \emptyset$ 
11     tmpl  $\leftarrow$  choose any member of tmplSet /* all are valid */
12  Return tmpl
13 End /* LEARN-SD-TEMPLATE */

```

4.2 Learning hint transducers

To learn a hint transducer (HT), we also make use of SD-template identification. However, instead of identifying a template that fits all answers, we identify templates that fit all *hints*. Based on this template, we then construct an HT that accepts the static parts of the hint string and performs the character-level transformation on the dynamic parts. A sketch of the algorithm that implements this, LEARN-HINT-TRANSDUCER, is shown below.

```

1  Function LEARN-HINT-TRANSDUCER returns HintTransducer
2  Input: the set of hints H, the set of resulting strings S
3  Use LCS to identify static parts between all H
4  ForEach H,S pair (h, s)
5    h'  $\leftarrow$  extraction of h replacing static chars with token ‘A’
6    A  $\leftarrow$  Align (h', s) based on lowercased string edit distance
7    Annotate A with character level operations
8  End
9  RE  $\leftarrow$  Build a reg expr that fits all annotations (using LCS)
10 If RE  $\neq \emptyset$ 
11   Return general predictive transducer based on RE
   that accepts static sequences of H where necessary
   and transduces dynamic sequences.
12 Else
13   Return  $\emptyset$ 
14 End /* LEARN-HINT-TRANSDUCER */

```

For example, suppose prior hints $\{\text{Dr. Joe Smith, Dr. Jane Thomas}\}$ corresponded to values $\{\text{joe_s, jane_t}\}$. The algorithm would first identify the static part of the hints and rewrite the hints using the Accept operation, i.e., $\{\text{AAAAJoe Smith, AAAAJane Thomas}\}$ where **A** refers to the operation **Accept**. It would then align each hint and value based on string edit distance and annotate with character operations required for transformation to the observed values, resulting in $\{\text{AAAAALCCRLDDDD, AAAALCCRLDDDD}\}$. Next, it would use the LCS to build the regular expression $\{\text{A*LC*RLD*}\}$ fitting these examples and, from this, the HT (partial form shown) in Figure 4:

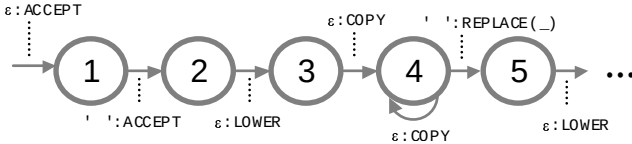


Figure 4: Partial form of the HT for the names example

5 Experimental Results

To evaluate our approach to value prediction, we compared it with caching on three sample plans that can benefit from speculative execution. These plans were chosen because all query multiple sources and/or multiple times within a source in order to retrieve information that is not possible to query directly. These plans normally require sequential execution; however, with speculative execution, significant speedup is possible.

These plans included *CarInfo* (which we have already described), *RepInfo*, and *PhoneInfo*. *RepInfo*, based on the plan described in (Barish and Knoblock 2002b), queries the site Vote-Smart.org for the issue positions of U.S. federal representatives for a particular nine-digit zip code. Its plan involves three queries: one for the list of representatives in the desired zip code, navigation to the profile page for each member, and navigation to their corresponding issue positions page. *PhoneInfo* is a similar plan that takes a U.S. phone number, does a reverse lookup of that number (on SuperPages.com) to find the state of origin and then queries the US Census (QuickFacts.census.gov) about demographics for that state. When querying the Census, additional navigation is required to get from the initial summary page about the state to the corresponding demographics details page.

All three were modified for speculative execution, with results similar to that shown in Figure 2. We then learned predictors for each, based on the data observed during execution. The learning was done offline, after each execution, so that future executions could benefit from the resulting predictors. The *CarInfo* predictors, as described, involved classification (make/model/year to car summary page) and transduction (summary to full review page). *RepInfo* required classification (nine-digit zip to political district page) and transduction (district to issue positions page). Finally, *PhoneInfo* required classi-

fication (phone number to state) and transduction (initial state facts page to detailed demographics page).

When learning the predictors, instances were drawn from typical distributions for that domain; for example, instances for *RepInfo* were drawn from a list of addresses of individuals that contributed to presidential campaigns (obtained from the FEC) – a distribution that closely approximates the U.S. geographic population distribution. Similarly, the phone numbers used in *PhoneInfo* came from a distribution of numbers for common last names.

We implemented speculative execution in Theseus, a streaming dataflow system for Web information gathering. Since our tests involved thousands of requests, we ran our plans using Theseus on local copies of the relevant data and simulated network latencies during retrieval. In doing so, we assumed each source had a latency of 2 seconds (note that the particular latency chosen does not matter – speedups will be the same).

Our results focus on three measurements. The first, shown in Table 1, show the average number of examples required in order to learn the correct transducer required by each of the plans. Notice, that *RepInfo* required more than *PhoneInfo* or *CarInfo* because there was a higher likelihood of the examples sharing some part of their dynamic data in common – and this was extracted by the LCS-based algorithm as a static element.

	CarInfo (full-review URL)	RepInfo (issue positions URL)	PhoneInfo (demographics URL)
Number of examples	6	11	7

Table 1: Learning the correct transducer

The second set of results, shown in Figure 5, focuses on the accuracy of classification, as measured over a 10-fold cross-validation sample of the data (where no data in the test fold was in any of the training folds). As expected, domains with larger sets of discrete target classes (such as *RepInfo*) required many more examples than those with smaller numbers of classes (like *PhoneInfo*).

The final set of results show how the learning we have described resulted in better average plan execution

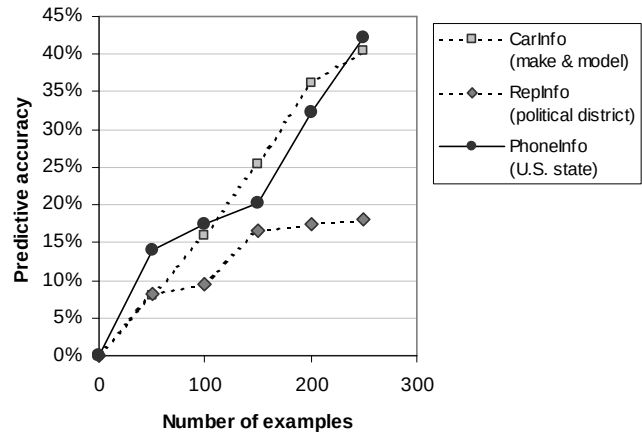


Figure 5: Accuracy of classification

speedups than did caching. Due to space constraints, we show results from only one of the plans – *CarInfo* – in Figure 6. This figure shows that, while the benefit of caching degrades significantly as the composition of future requests contains a greater number of unseen examples, the learning allows accurate predictions to be made – even for a mix containing entirely new requests. Furthermore, as the number of examples increases, speedup from learning increases and the accuracy of speculation (even on 100% unseen instances) gradually increases. Though omitted here, these same general trends hold for both *RepInfo* and *PhoneInfo* as well (although to different degrees).

6 Related Work

Learning to speculatively execute programs has been well-studied in computer science. Historically, however, computer architecture research has focused mostly on predicting *control* (e.g., branch prediction), not data. While hardware-level value prediction has recently become an active area of research, the type of data being predicted (e.g., memory locations) is different.

At a high-level, this work could be considered a form of speedup learning. In speedup learning, the goal is to improve problem solving performance through experience. Past research has focused on a number of areas, including learning sequences of operators (Fikes *et al.* 1972) and learning control knowledge to aid in choosing what operators to execute next (Minton 1988). Although the use of learning in this work is for the purpose of value prediction (i.e., how to execute), the goal of using learned knowledge to improve performance is the same.

There has not been any previous work on value prediction for information gathering systems. (Hull *et al.* 2000) proposed speculation in a decision flow framework, but one in which only control predictions were necessary. There has also been past work on information gathering with partial results (Hellerstein *et al.* 1997; Shanmugasundaram *et al.* 2000), but these systems do not predict data values and instead use approximate values from intermediate aggregate operators in order to obtain approximate final results.

Surprisingly, there has been little work on the learning of subsequential transducers. One existing algorithm is

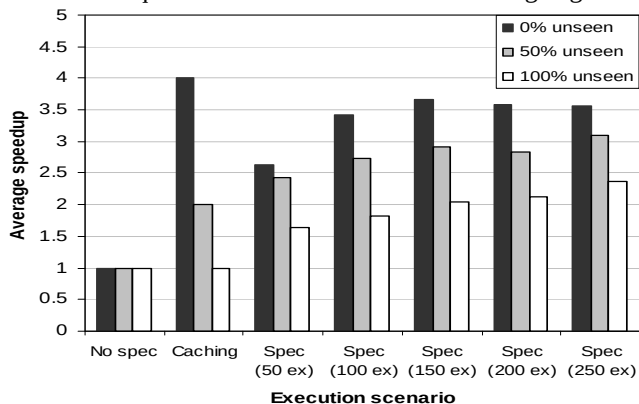


Figure 6: Learning vs. caching in *CarInfo*

OSTIA (Oncina *et al.* 1993), which is able to induce traditional subsequential transducers capable of automating translations of decimal to Roman numbers or English word spellings of numbers to their decimal equivalents. Our work differs from OSTIA mainly in that the transducers we learn capture *the general process* of a regular class of string transformations. After learning from only a few examples, our algorithm can achieve a high-degree of accuracy on such classes. In contrast, while OSTIA can learn more complex types of subsequential transducers, it can require a very large number of examples before it can learn the proper rule (Gildea and Jurafsky 1996).

The transducer learning algorithm suggested by (Hsu and Chang 1999) viewed transduction as a means for information extraction. Our use is similar in that one part of our approach involves extracting dynamic values from hints. However, the transducers we describe go beyond extraction – they transform the source string so that it can be integrated into a predicted value. In doing so, our transduction process also makes use of classification.

Finally, while our use of classification applies to predicting any type of data value in an information gathering plan, our typical use of transduction is for the prediction of URLs. Other approaches have explored point-based (Zukerman *et al.* 1999) or path-based (Su *et al.* 2000) methods of URL prediction, attempting to understand *request models* based on either time, the order of requests, or the associations between requests. However, unlike our approach, these techniques do not try to understand very general *patterns in request content* and thus cannot predict URLs that have not been previously requested.

7 Conclusions

Successful speculative execution of information gathering plans is fundamentally linked with the ability to make good predictions. In this paper, we have described how a hybrid approach based on two simple techniques – classification and transduction – can be combined and applied to the problem. The approach we describe represents a hybridization of not only classification and transduction, but also of caching, since classifiers effectively function as caches when no classification is possible.

Our experimental results show that learning such predictors can lead to significant speedups when gathering information that must be queried indirectly. We believe that a bright future exists for data value prediction at the information gathering level, primarily because of the potential speedup enabled by speculative execution and because of the availability of resources (i.e., memory) that exist at higher levels of execution, enabling more sophisticated machine learning techniques to be applied.

8 Acknowledgements

We wish to thank Steven Minton and Cenk Gizen for very helpful and inspiring discussions related to techniques for template induction.

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under contract/agreement numbers F30602-01-C-0197 and F30602-00-1-0504, in part by the Air Force Office of Scientific Research under grant numbers F49620-01-1-0053 and F49620-02-1-0270, in part by the United States Air Force under contract number F49620-02-C-0103, in part by gifts from the Intel and Microsoft Corporations. The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copy right 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

- [Barish and Knoblock 2002a] Greg Barish and Craig A. Knoblock. An Efficient and Expressive Language for Information Gathering on the Web. *Proceedings of the Sixth International Conference on AI Planning and Scheduling (AIPS 2002) Workshop: Is There Life After Operator Sequencing?* Toulouse, France. 2002.
- [Barish and Knoblock 2002b] Greg Barish and Craig A. Knoblock. Speculative Execution for Information Gathering Plans. *Proceedings of the Sixth International Conference on AI Planning and Scheduling (AIPS 2002)*. Toulouse, France. 2002.
- [Fikes et al. 1972] Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence* 3(4): 251-288. 1972.
- [Gildea and Jurafsky 1996] Daniel Gildea and Daniel Jurafsky. Learning Bias and Phonological-Rule Induction. *Computational Linguistics* 22(4): 497-530. 1996.
- [Hellerstein et al. 1997] Joseph M. Hellerstein, Peter J. Haas and Helen J. Wang. Online Aggregation. *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1997)*. Tucson, AZ: 171-182. 1997.
- [Hellerstein et al. 2000] Joseph M. Hellerstein, Michael J. Franklin, Sirish Chandrasekaran, Amol Deshpande, Kris Hildrum, Sam Madden, Vijayshankar Raman and Mehul A. Shah. Adaptive Query Processing: Technology in Evolution. *IEEE Data Engineering Bulletin* 23(2):7-18. 2000.
- [Hirschberg 1975] Daniel S. Hirschberg. A Linear Space Algorithm for Computing Maximal Common Subsequences. *Communications of the ACM* 18(6): 341-343. 1975.
- [Hsu and Chang 1999] Chu-Nan Hsu and Chien-Chi Chang. Finite-State Transducers for Semi-Structured Text Mining. *Proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*. 1999.
- [Hull et al. 2000] Richard Hull, Francois Llirbat, Bharat Kumar, Gang Zhou, Guozhu Dong and Jianwen Su. Optimization Techniques for Data-Intensive Decision Flows. *Proceedings of the 16th International Conference on Data Engineering*. San Diego, CA: 281-292. 2000.
- [Ives et al. 2002] Zachary G. Ives, Alon Y. Halevy and Daniel S. Weld. An XML Query Engine for Network-Bound Data. *VLDB Journal* 11(4): 380-402. 2002.
- [Minton 1988] Steven Minton. *Learning search control knowledge*. Kluwer Academic Publishers. Boston, MA. 1988.
- [Mohri 1997] Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics* 23(2): 269-311. 1997.
- [Naughton et al. 2001] Jeffrey F. Naughton, David J. DeWitt, David Maier, Ashraf Aboulmaga, Jianjun Chen, Leonidas Galanis, Jaewoo Kang, Rajasekar Krishnamurthy, et al. The Niagara Internet Query System. *IEEE Data Engineering Bulletin* 24(2): 27-33. 2001.
- [Oncina et al. 1993] Jose Oncina, Pedro Garcia and Enrique Vidal. Learning Subsequential Transducers for Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(5): 448-458. 1993.
- [Shanmugasundaram et al. 2000] Jayavel Shanmugasundaram, Kristin Tufte, David J. DeWitt, Jeffrey F. Naughton and David Maier. Architecting a Network Query Engine for Producing Partial Results. *Proceedings of SIGMOD 3rd Intl Workshop on Web and Databases (WebDB 2000)*. Dallas, TX: 17-22. 2000.
- [Su et al. 2000] Zhong Su, Qiang Yang, Ye Lu and Hong-Jiang Zhang. WhatNext: A Prediction System for Web Request Using N-Gram Sequence Models. *Proceedings of the First International Conference on Web Information Systems Engineering (WISE-2000)*: 214-221. 2000.
- [Zuckerman et al. 1999] Ingrid Zukerman, David W. Albrecht and Ann E. Nicholson. Predicting User's Requests on the WWW. *Proceedings of the 7th International Conference on User Modeling*. 1999.

Logic Programs for Consistently Querying Data Integration Systems

Loreto Bravo

Pontificia Universidad Católica de Chile
Departamento de Ciencia de Computación
Santiago, Chile.
lbravo@ing.puc.cl

Leopoldo Bertossi

Carleton University
School of Computer Science
Ottawa, Canada
bertossi@scs.carleton.ca

Abstract

We solve the problem of obtaining answers to queries posed to a mediated integration system under the local-as-view paradigm that are consistent wrt to certain global integrity constraints. For this, the query program is combined with logic programming specifications under the stable model semantics of the class of minimal global instances, and of the class of their repairs.

1 Introduction

For several reasons a database may become inconsistent wrt certain integrity constraints (ICs). In such a situation, possibly most of the data is still consistent. In [Arenas *et al.*, 1999] consistent data in a single relational database is characterized as the data that is invariant under all minimal restorations of consistency, i.e. true in all repaired versions of the original instance (the *repairs*). In that paper and others [Arenas *et al.*, 2000; Greco *et al.*, 2001], some mechanisms have been developed for retrieving consistent answer when queries are posed to such an inconsistent database.

When independent data sources are integrated, inconsistencies wrt to global ICs are likely to occur, specially when the sources are virtually integrated by means of a mediator, because the data sources are kept completely independent. The mediator provides a global schema as an interface, and is responsible for generating query plans to answer global queries by retrieving data sets from the sources and combining them into a final answer set for the user.

“Local-as-view” (LAV) is a common paradigm for data integration that describes each data source as a set of views over the global schema. Another one, “global-as-view” (GAV), defines every global relation as a view of the set of relations in the sources ([Lenzerini, 2002] is a good survey). Query answering is harder in LAV [Abiteboul *et al.*, 1998]. On the other side, the LAV approach allows more flexibility when new sources are integrated into an existing system. However, the flexibility to add new sources, without having to consider the other sources in the system, makes inconsistencies wrt global ICs more likely.

Example 1 Consider the LAV based global integration system $\mathfrak{G}_1$ with a global relation $R(X, Y)$ and two source relations $v_1 = \{V_1(a, b), V_1(c, d)\}$ and $v_2 = \{V_2(a, c), V_2(d, e)\}$

that are described by the view definitions $V_1(X, Y) \leftarrow R(X, Y)$; $V_2(X, Y) \leftarrow R(X, Y)$. The global functional dependency (FD) $R: X \rightarrow Y$ is violated through the pair of tuples $\{(a, b), (a, c)\}$. $\square$

In a virtual integration system, the mediator should solve potential inconsistencies when the query plan is generated. In [Bertossi *et al.*, 2002], under the LAV approach, a methodology for generating query plans to compute answers to limited forms of queries that are consistent wrt an also restricted class of universal ICs was presented. The limitation comes from a first step where a query transformation for consistent query answering (CQA) is performed [Arenas *et al.*, 1999]. Next, query plans are generated for the transformed query. However, [Bertossi *et al.*, 2002] provides the right semantics for CQA in mediated integrated systems (see Section 2).

Example 2 (example 1 continued) If we pose to the global system the query $Q : Ans(X, Y) \leftarrow R(X, Y)$, we obtain the answers $\{Ans(a, b), Ans(c, d), Ans(a, c), Ans(d, e)\}$. However, only the tuples $Ans(c, d), Ans(d, e)$ should be returned as consistent answers wrt the FD $R: X \rightarrow Y$. $\square$

In this paper, under the LAV approach and assuming that sources are open (or incomplete) [Abiteboul *et al.*, 1998], we solve the problem of retrieving consistent answers to global queries. We consider arbitrary universal ICs and referential ICs, in consequence all the ICs that are used in database praxis [Abiteboul *et al.*, 1995]. View definitions are conjunctive queries, and global queries are expressed in Datalog and its extensions with negation. The methodology can be summarized as follows. First, in Section 3, the minimal legal global instances of a mediated system are specified by means of logic programs with a stable model, or answer sets, semantics. Next, in Section 4, the repairs of the minimal global instances are specified as the stable models of disjunctive logic programs. Those programs contain annotation constants, like those used to specify repairs of single relational databases for CQA [Barcelo *et al.*, 2002]. Finally, in Section 5, consistent answers to queries are obtained by running a query program in combination with the previous two specification programs.

2 Preliminaries

2.1 Global schemas and view definitions

A global schema $\mathcal{R}$ is modeled by a finite set of relations $\{R_1, R_2, \dots, R_n\}$ over a fixed domain $\mathcal{U}$. With these relation

symbols and the elements of $\mathcal{U}$ treated as constants, a first-order language $\mathcal{L}(\mathcal{R})$ can be defined. This language can be extended with new defined predicates and built-ins. In particular, we will extend the global schema with a *local schema* $\mathcal{S}$, i.e. a finite set of new view predicates $V_1, V_2, \dots$, that will be used to describe the relations in the local sources.

Each *view*, denoted by, say a new predicate V , is defined by means of conjunctive query [Abiteboul *et al.*, 1995], i.e. an $\mathcal{L}(\mathcal{R} \cup \mathcal{S})$ -formula of the form $\varphi_V: V(\bar{t}) \leftarrow \text{body}(\varphi_V)$, where $\bar{t}$ is a tuple containing variables and/or constants, and $\text{body}(\varphi_V)$ is a conjunction of $\mathcal{R}$ -atoms.

A *database instance* D over schema $\mathcal{R}$ can be considered as a first-order structure with domain $\mathcal{U}$, where the extensions of the relations R_i are finite. (The extensions of built-in predicates may be infinite, but fixed.) A global *integrity constraint* (IC) is an $\mathcal{L}(\mathcal{R})$ -sentence ψ . An instance D satisfies ψ , denoted $D \models \psi$, if ψ is true in D .

Given a database instance D over schema $\mathcal{R}$, and a view definition φ_V , $\varphi_V(D)$ denotes the extension of V obtained by applying the definition φ_V to D . If the view already has an extension v (corresponding to the contents of a data source), it is possible that v is incomplete and stores only some of the tuples in $\varphi_V(D)$; and we say the view extension v is *open* wrt D [Abiteboul *et al.*, 1998]. Most mechanisms for deriving query plans assume that sources are open, e.g. [Duschka *et al.*, 2000].

A *source* S is a pair $\langle \varphi, v \rangle$, where φ is the view definition, and v is an extension for φ . An *open global system* $\mathfrak{G}$ is a finite set of open sources. The global schema $\mathcal{R}$ consists of the relation names that do not have a definition in the global system. The underlying domain $\mathcal{U}$ for $\mathcal{R}$ is a proper superset of the “active” domain. The latter consists of all the constants appearing in the view extensions v_i of the sources, and in their definitions. A global system $\mathfrak{G}$ defines a set of legal global instances [Lenzerini, 2002].

Definition 1 Given an open global system $\mathfrak{G} = \{\langle \varphi_1, v_1 \rangle, \dots, \langle \varphi_n, v_n \rangle\}$, the set of legal global instances is $\text{Linst}(\mathfrak{G}) = \{D \text{ instance over } \mathcal{R} \mid v_i \subseteq \varphi_i(D), i = 1, \dots, n\}$. $\square$

Example 3 (example 2 continued) Let us denote by φ_1, φ_2 the view definitions of V_1, V_2 , resp. in $\mathfrak{G}_1$. $D = \{R(a, b), R(c, d), R(a, c), R(d, e)\}$ is a legal global instance, because $v_1 = \{V_1(a, b), V_1(c, d)\} \subseteq \varphi_1(D) = \{V_1(a, b), V_1(c, d), V_1(a, c), V_1(d, e)\}$ and $v_2 = \{V_2(a, c), V_2(d, e)\} \subseteq \varphi_2(D) = \{V_2(a, b), V_2(c, d), V_2(a, c), V_2(d, e)\}$. Supersets of D are also legal instances. $\square$

The semantics of query answers in mediated integration systems is given by the notion of *certain answer*. In this paper we will consider queries expressed in Datalog and its extensions with negation.

Definition 2 [Abiteboul *et al.*, 1998] Given an open global system $\mathfrak{G}$ and a query $Q(\bar{X})$ to the system, a tuple $\bar{t}$ is a *certain answer* to Q in $\mathfrak{G}$ if for every global instance $D \in \text{Linst}(\mathfrak{G})$, it holds $D \models Q[\bar{t}]$. We denote with $\text{Certain}_{\mathfrak{G}}(Q)$ the set of certain answers to Q in $\mathfrak{G}$. $\square$

The inverse-rules algorithm [Duschka *et al.*, 2000] for generating query plans under the LAV approach assumes that sources are open and each source relation V is defined as

a conjunctive view over the global schema: $V(\bar{X}) \leftarrow P_1(\bar{X}_1), \dots, P_n(\bar{X}_n)$, with $\bar{X} \subseteq \bigcup_i \bar{X}_i$.

Then, for $j = 1, \dots, n$, $P_j(\bar{X}'_j) \leftarrow V(\bar{X})$ is an inverse rule for P_j . The tuple $\bar{X}_j$ is transformed to obtain the tuple $\bar{X}'_j$ as follows: if $X \in \bar{X}_j$ is a constant or is a variable appearing in $\bar{X}$, then X is unchanged in $\bar{X}'_j$. Otherwise, X is a variable X_i that does not appear in $\bar{X}$, and it is replaced by the term $f_i(\bar{X})$, where f_i is a fresh Skolem function. We denote the set of inverse rules of the collection $\mathcal{V}$ of source descriptions in $\mathfrak{G}$ by $\mathcal{V}^{-1}$.

Example 4 Consider the mediated data integration system $\mathfrak{G}_3$ with global schema $\mathcal{R} = \{P, R\}$. The set $\mathcal{V}$ of local view definitions consists of $V_1(X, Z) \leftarrow P(X, Y), R(Y, Z)$, and $V_2(X, Y) \leftarrow P(X, Y)$. The set $\mathcal{V}^{-1}$ consists of the rules $P(X, f(X, Z)) \leftarrow V_1(X, Z)$; $R(f(X, Z), Z) \leftarrow V_1(X, Z)$; and $P(X, Y) \leftarrow V_2(X, Y)$. $\square$

The inverse rules are then used to answer queries expressed in terms of the global relations, that now have definitions in terms of the sources. The answers obtained with the inverse rule algorithm are maximally contained in the queries [Duschka *et al.*, 2000], and coincide with the certain answers [Abiteboul *et al.*, 1998].

2.2 Global systems and consistency

We assume that we have a set of global integrity constraints IC that is consistent as a set of logical sentences, and *generic*, in the sense that it does not entail any ground database literal. ICs used in database praxis are always generic.

Definition 3 [Bertossi *et al.*, 2002] (a) Given a global system, $\mathfrak{G}$, an instance D is *minimal* if $D \in \text{Linst}(\mathfrak{G})$ and is *minimal wrt set inclusion*, i.e. there is no other instance in $\text{Linst}(\mathfrak{G})$ that is a proper subset of D (as a set of atoms). We denote by $\text{Mininst}(\mathfrak{G})$ the set of minimal legal global instances of $\mathfrak{G}$ wrt set inclusion. (b) A global system $\mathfrak{G}$ is *consistent wrt IC*, if for all $D \in \text{Mininst}(\mathfrak{G})$, $D \models IC$. $\square$

Example 5 (example 4 continued) Assume that $\mathfrak{G}_3$ has the source contents $v_1 = \{V_1(a, b)\}$, $v_2 = \{V_2(a, c)\}$, resp. and that $\mathcal{U} = \{a, b, c, u\}$. Then, the elements of $\text{Mininst}(\mathfrak{G}_3)$ are of the form $D_z = \{P(a, z), R(z, b), P(a, c)\}$ for some $z \in \mathcal{U}$. The global FD $P(X, Y): X \rightarrow Y$ is violated exactly in those minimal legal instances D_z for which $z \neq c$. Thus, $\mathfrak{G}_3$ is inconsistent. $\square$

Definition 4 The ground tuple $\bar{a}$ is a *minimal answer* to a query Q posed to $\mathfrak{G}$ if for every $D \in \text{Mininst}(\mathfrak{G})$, $\bar{a} \in Q(D)$, where $Q(D)$ is the answer set for Q in D . The set of minimal answers is denoted by $\text{Minimal}_{\mathfrak{G}}(Q)$. $\square$

Clearly $\text{Certain}_{\mathfrak{G}}(Q) \subseteq \text{Minimal}_{\mathfrak{G}}(Q)$. For monotone queries [Abiteboul *et al.*, 1995], the two notions coincide [Bertossi *et al.*, 2002]. Nevertheless, in Example 5 the query $\text{Ans}(X, Y) \leftarrow \neg P(X, Y)$ has (b, a) as a minimal answer, but not as a certain answer, because there are legal instances that contain $P(b, a)$. Since consistency was defined wrt minimal global instances, the notion of minimal answer is particularly relevant.

Given a database instance D , we denote by $\Sigma(D)$ the set of ground formulas $\{P(\bar{a}) \mid P \in \mathcal{R} \text{ and } D \models P(\bar{a})\}$.

Definition 5 [Arenas et al., 1999] (a) Let D, D' be database instances over the same schema and domain. The distance, $\Delta(D, D')$, between D and D' is the symmetric difference $\Delta(D, D') = (\Sigma(D) \setminus \Sigma(D')) \cup (\Sigma(D') \setminus \Sigma(D))$. (b) For database instances D, D', D'' , we define $D' \leq_D D''$ if $\Delta(D, D') \subseteq \Delta(D, D'')$. $\square$

Definition 6 (based on [Arenas et al., 1999]) Let $\mathfrak{G}$ be a global system and IC a set of global ICs. A repair of $\mathfrak{G}$ wrt IC is a global database instance D' , such that $D' \models IC$ and D' is $\leq_D$ -minimal for some $D \in \text{Mininst}(\mathfrak{G})$. $\square$

Thus, a repair of a global system is a global database instance that minimally differs from a minimal legal global database instance. If $\mathfrak{G}$ is already consistent, then the repairs are the elements of $\text{Mininst}(\mathfrak{G})$. In Definition 6 we are not requiring that a repair respects the property of the sources of being open, i.e. that the extension of each view in the repair contains the corresponding view extension in the source. Thus, it may be the case that a repair – still a global instance – does not belong to $\text{Linst}(\mathfrak{G})$. If we do not allow this, a global system might not be repairable. Repairs are used as an auxiliary concept to define the notion of consistent answer.

Example 6 (example 1 continued) The only element in $\text{Mininst}(\mathfrak{G}_1)$ is $D_0 = \{R(a, b), R(c, d), R(a, c), R(d, e)\}$, that does not satisfy IC . Then, $\mathfrak{G}_1$ is inconsistent. The repairs are the global instances that minimally differ from D_0 and satisfy the FD, namely $D_0^1 = \{R(a, b), R(c, d), R(d, e)\}$ and $D_0^2 = \{R(a, c), R(c, d), R(d, e)\}$. Notice that they do not belong to $\text{Linst}(\mathfrak{G}_1)$. $\square$

Definition 7 [Bertossi et al., 2002] (a) Given a global system $\mathfrak{G}$, a set of global integrity constraints IC , and a global first-order query $Q(\bar{X})$, we say that a (ground) tuple $\bar{t}$ is a consistent answer to Q wrt IC iff for every repair D of $\mathfrak{G}$, $D \models Q[\bar{t}]$. (b) We denote by $\text{Consis}_{\mathfrak{G}}(Q)$ the set of consistent answers to Q in $\mathfrak{G}$. $\square$

Example 7 (example 6 continued) For the query $Q_1(X): \exists Y R(X, Y)$, a is a consistent answer. $Q_2(X, Y): R(X, Y)$ has $(c, d), (d, e)$ as consistent answers. $\square$

If $\mathfrak{G}$ is consistent wrt IC , then $\text{Consis}_{\mathfrak{G}}(Q) = \text{Minimal}_{\mathfrak{G}}(Q)$. Furthermore, if the ICs are generic, then for any $\mathfrak{G}$ it holds $\text{Consis}_{\mathfrak{G}}(Q) \subseteq \text{Minimal}_{\mathfrak{G}}(Q)$ [Bertossi et al., 2002]. Notice also that the notion of consistent answer can be applied to queries written in Datalog or its extensions with built-ins and stratified negation.

3 Specification of Legal Instances

The specification of the class $\text{Mininst}(\mathfrak{G})$ for system $\mathfrak{G}$ is given using normal logic programs, whose rules are inspired by the inverse-rules algorithm. They use auxiliary predicates instead of function symbols, but their functionality is enforced using the choice predicate [Giannotti et al., 1991].

Definition 8 Given an open global system $\mathfrak{G}$, the program, $\Pi(\mathfrak{G})$, contains the following clauses:

1. Fact $\text{dom}(a)$ for every constant $a \in \mathcal{U}$; and the fact $V(\bar{a})$ whenever $V(\bar{a}) \in v_i$ for some source extension v_i in $\mathfrak{G}$.
2. For every view (source) predicate V in the system with

description $V(\bar{X}) \leftarrow P_1(\bar{X}_1), \dots, P_n(\bar{X}_n)$, the rules $P_j(\bar{X}_j) \leftarrow V(\bar{X}), \bigwedge_{X_i \in (\bar{X}_j \setminus \bar{X})} F_i(\bar{X}, X_i), j = 1, \dots, n$.
 3. For every predicate $F_i(\bar{X}, X_i)$ introduced in 2., the rule $F_i(\bar{X}, X_i) \leftarrow V(\bar{X}), \text{dom}(X_i), \text{choice}((\bar{X}), (X_i))$. $\square$

The predicate $F_i(\bar{X}, X_i)$ is replacing the Skolem function $f_i(\bar{X}) = X_i$ introduced in Section 2.1, and, through the choice predicate, it assigns values in the domain to the variables in the head of the rule that are not in $\bar{X}$. There is a new Skolem predicate for each pair formed by a description rule as in item 2. above and a different existentially quantified variable in it. The predicate $\text{choice}((\bar{X}), (X_i))$ ensures that for every (tuple of) value(s) for $\bar{X}$, only one (tuple of) value(s) for X_i is non deterministically chosen between the constants of the active domain.

Example 8 (examples 4 and 5 continued) Program $\Pi(\mathfrak{G}_3)$ contains the following rules:

1. $\text{dom}(a), \text{dom}(b), \text{dom}(c), \text{dom}(u), V_1(a, b), V_2(a, c)$.
2. $P(X, Z) \leftarrow V_1(X, Y), F_1(X, Y, Z),$
 $R(Z, Y) \leftarrow V_1(X, Y), F_1(X, Y, Z),$
 $P(X, Y) \leftarrow V_2(X, Y).$
3. $F_1(X, Y, Z) \leftarrow V_1(X, Y), \text{dom}(Z),$
 $\text{choice}((X, Y), (Z)). \square$

For every program Π with the choice operator, there is its stable version, $SV(\Pi)$, whose stable models correspond to the so-called choice models of Π [Giannotti et al., 1991]. The program $SV(\Pi)$ is obtained as follows: (a) Each choice rule $r: H \leftarrow B, \text{choice}((\bar{X}), (Y))$ in Π is replaced by the rule $H \leftarrow B, \text{chosen}_r(\bar{X}, Y)$. (b) For each rule as in (a), the following rules are added

$$\begin{aligned} \text{chosen}_r(\bar{X}, Y) &\leftarrow B, \text{not } \text{diffChoice}_r(\bar{X}, Y). \\ \text{diffChoice}_r(\bar{X}, Y) &\leftarrow \text{chosen}_r(\bar{X}, Y'), Y \neq Y'. \end{aligned}$$

The rules defined in (b) ensure that, for every tuple $\bar{X}$ where B is satisfied, the predicate $\text{chosen}_r(\bar{X}, Y)$ satisfies the functional dependency $\bar{X} \rightarrow Y$. In (a) the choice operator is replaced by the new predicate chosen_r that forces the expected functional dependency.

Example 9 (example 8 continued) Program $SV(\Pi(\mathfrak{G}_3))$ contains the following rules:

1. $\text{dom}(a), \text{dom}(b), \text{dom}(c), \text{dom}(u), V_1(a, b), V_2(a, c)$.
2. $P(X, Z) \leftarrow V_1(X, Y), F_1(X, Y, Z),$
 $R(Z, Y) \leftarrow V_1(X, Y), F_1(X, Y, Z),$
 $P(X, Y) \leftarrow V_2(X, Y).$
3. $F_1(X, Y, Z) \leftarrow V_1(X, Y), \text{dom}(Z), \text{chosen}_1(X, Y, Z).$
4. $\text{chosen}_1(X, Y, Z) \leftarrow V_1(X, Y), \text{dom}(Z),$
 $\text{not } \text{diffChoice}_1(X, Y, Z),$
 $\text{diffChoice}_1(X, Y, Z) \leftarrow \text{chosen}_1(X, Y, Z'),$
 $\text{dom}(Z), Z' \neq Z.$

Its stable models are:

$$\begin{aligned} \mathcal{M}_1 &= \{\text{dom}(a), \text{dom}(b), \text{dom}(c), \text{dom}(u), V_1(a, b), \\ &\quad V_2(a, c), P(a, c), \text{diffChoice}_1(a, b, a), \\ &\quad \text{chosen}_1(a, b, b), \text{diffChoice}_1(a, b, c), \\ &\quad \text{diffChoice}_1(a, b, u), F_1(a, b, b), R(b, b), P(a, b)\} \\ \mathcal{M}_2 &= \{\text{dom}(a), \text{dom}(b), \text{dom}(c), \text{dom}(u), V_1(a, b), \\ &\quad V_2(a, c), P(a, c), \text{chosen}_1(a, b, a), \\ &\quad \text{diffChoice}_1(a, b, b), \text{diffChoice}_1(a, b, c), \\ &\quad \text{diffChoice}_1(a, b, u), F_1(a, b, a), R(a, b), P(a, a)\} \end{aligned}$$

$$\begin{aligned}\mathcal{M}_3 &= \{dom(a), dom(b), dom(c), dom(u), V_1(a, b), \\ &\quad V_2(a, c), P(a, c), diffChoice_1(a, b, a), \\ &\quad diffChoice_1(a, b, b), chosen_1(a, b, c), \\ &\quad diffChoice_1(a, b, u), F_1(a, b, c), R(c, b)\} \\ \mathcal{M}_4 &= \{domd(a), domd(b), domd(c), domd(u), V_1(a, b), \\ &\quad V_2(a, c), P(a, c), diffChoice_1(a, b, a), \\ &\quad diffChoice_1(a, b, b), diffChoice_1(a, b, c), \\ &\quad chosen_1(a, b, u), F_1(a, b, u), R(u, b), P(a, u)\}. \quad \square\end{aligned}$$

Definition 9 The instance associated to a stable model $\mathcal{M}$ of $\Pi(\mathfrak{G})$ is $D_{\mathcal{M}} = \{P(\bar{a}) \mid P \in \mathcal{R} \text{ and } P(\bar{a}) \in \mathcal{M}\}$. $\square$

Example 10 (example 9 continued) $D_{\mathcal{M}_1}, D_{\mathcal{M}_2}, D_{\mathcal{M}_3}, D_{\mathcal{M}_4}$ are the elements of $Mininst(\mathfrak{G}_3)$, namely $\{P(a, b), R(b, b), P(a, c)\}, \{P(a, a), R(a, b), P(a, c)\}, \{P(a, c), R(c, b)\}, \{P(a, u), R(u, b), P(a, c)\}$, respectively. $\square$

Theorem 1 If $\mathcal{M}$ is a stable model of program $SV(\Pi(\mathfrak{G}))$, then $D_{\mathcal{M}} \in Mininst(\mathfrak{G})$. Furthermore, the minimal legal instances obtained in this way are all the minimal legal instances of $\mathfrak{G}$. $\square$

The program $\Pi(\mathfrak{G})$ (or its stable version) can be used to compute $Minimal_{\mathfrak{G}}(Q)$, where Q is a query expressed as a, say Datalog^{not} program $\Pi(Q)$. This can be done by running the combined program under the skeptical or cautious stable model semantics, that sanctions as true wrt the program what is true in all choice (resp. stable) models. If the query Q is monotone, e.g. a Datalog query, then in the same way $Certain_{\mathfrak{G}}(Q)$ can be computed.

4 Specification of Repairs of a Global System

In [Barcelo *et al.*, 2002] repairs of single relational databases are specified using disjunctive logic programs with stable model semantics. The approach works for arbitrary universal and referential ICs in the sense that the repairs of the database correspond to the stable models of the program. We briefly explain these programs, because they will be used to specify repairs of integration systems.

First, the database predicates in the program are expanded with an extra argument to be filled with one of a set of new annotation constants. An atom in (outside) the original database is annotated with $\mathbf{t_d}$ ($\mathbf{f_d}$). Annotations $\mathbf{t_a}$ and $\mathbf{f_a}$ are considered *advisory* values, to solve conflicts between the database and the ICs. If an atom gets the derived annotation $\mathbf{f_a}$, it means an advise to make it false, i.e. to delete it from the database. Similarly, an atom that gets the annotation $\mathbf{t_a}$ must be inserted into the database.

Example 11 Consider the integrity constraint $\forall x(P(x) \rightarrow R(x))$, and the inconsistent database instance $r = \{P(a)\}$. The logic program should have the effect of repairing the database. Single, local repair steps are obtained by deriving the annotations $\mathbf{t_a}$ or $\mathbf{f_a}$. This is done when each IC is considered in isolation, but there may be interacting ICs, and the repair process may take several steps and should stabilize at some point. In order to achieve this, we use annotations $\mathbf{t^*}$, $\mathbf{f^*}$. The latter, for example, groups together the annotations $\mathbf{f_d}$ and $\mathbf{f_a}$ for the same atom (rules 1. and 4. below). These derived annotations are used to give a feedback to the bodies of the rules that produce the local, single repair steps, so that

a propagation of changes is triggered (rule 2. below). The annotations $\mathbf{t^{**}}$ and $\mathbf{f^{**}}$ are just used to read off the literals that are inside (resp. outside) a repair. This is achieved by means of rules 6. below, that are used to interpret the models as database repairs. The following is the program:

1. $P(x, \mathbf{f^*}) \leftarrow P(x, \mathbf{f_a}). \quad P(x, \mathbf{t^*}) \leftarrow P(x, \mathbf{t_a}).$
 $P(x, \mathbf{t^*}) \leftarrow P(x, \mathbf{t_d}). \quad (\text{similarly for } R)$
2. $P(x, \mathbf{f_a}) \vee R(x, \mathbf{t_a}) \leftarrow P(x, \mathbf{t^*}), R(x, \mathbf{f^*}).$
3. $P(a, \mathbf{t_d}) \leftarrow.$
4. $P(x, \mathbf{f^*}) \leftarrow \text{not } P(x, \mathbf{t_d}). \quad R(x, \mathbf{f^*}) \leftarrow \text{not } R(x, \mathbf{t_d}).$
5. $\leftarrow P(\bar{x}, \mathbf{t_a}), P(\bar{x}, \mathbf{f_a}). \quad \leftarrow R(\bar{x}, \mathbf{t_a}), R(\bar{x}, \mathbf{f_a}).$
6. $P(x, \mathbf{t^{**}}) \leftarrow P(x, \mathbf{t_a}). \quad P(x, \mathbf{f^{**}}) \leftarrow P(x, \mathbf{f_a}).$
 $P(x, \mathbf{t^{**}}) \leftarrow P(x, \mathbf{t_d}), \text{not } P(x, \mathbf{f_a}).$
 $P(x, \mathbf{f^{**}}) \leftarrow \text{not } P(x, \mathbf{t_d}), \text{not } P(x, \mathbf{t_a}). \quad (\text{similarly for } R)$

Only rules 2. depend on the ICs. They say how to repair them when violations are found. Rules 3. contain the database atoms. Rules 4. capture the CWA. Rules 5. are denial program constraints to discard models that contain an atom annotated with both $\mathbf{t_a}$ and $\mathbf{f_a}$. The program has two stable models: $\{P(a, \mathbf{t_d}), P(a, \mathbf{t^*}), R(a, \mathbf{f^*}), R(a, \mathbf{t_a}), \underline{P(a, \mathbf{t^{**}})}, R(a, \mathbf{t^*}), \underline{R(a, \mathbf{t^{**}})}\}$, and $\{P(a, \mathbf{t_d}), P(a, \mathbf{t^*}), \underline{P(a, \mathbf{f^*})}, R(a, \mathbf{f^*}), \underline{P(a, \mathbf{f^{**}})}, \underline{R(a, \mathbf{f^{**}})}, P(a, \mathbf{f_a})\}$, the first one saying (look at underlined atoms) that $R(a)$ is inserted into the database; the second one, that $P(a)$ is deleted. $\square$

The next definition combines into one program the specification of the minimal legal instances and their repairs.

Definition 10 The repair program, $\Pi(\mathfrak{G}, IC)$, of $\mathfrak{G}$ wrt IC contains the following clauses:

1. Facts as in 1. in Definition 8.
2. Each of the rules 2. in Definition 8 is replaced by
 $P_j(\bar{X}_j, \mathbf{t_d}) \leftarrow V(\bar{X}), \bigwedge_{X_i \in (\bar{X}_j \setminus \bar{X})} F_i(\bar{X}, X_i).$
3. Exactly the same rules as in 3. in Definition 8.
4. For every predicate $P \in \mathcal{R}$, the clauses
 $P(\bar{X}, \mathbf{t^*}) \leftarrow P(\bar{X}, \mathbf{t_d}), dom(\bar{X}).^1$
 $P(\bar{X}, \mathbf{t^*}) \leftarrow P(\bar{X}, \mathbf{t_a}), dom(\bar{X}).$
 $P(\bar{X}, \mathbf{f^*}) \leftarrow P(\bar{X}, \mathbf{f_a}), dom(\bar{X}).$
 $P(\bar{X}, \mathbf{f^*}) \leftarrow dom(\bar{X}), \text{not } P(\bar{X}, \mathbf{t_d}).$
5. For every first-order global universal IC of the form
 $\bar{\forall}(Q_1(\bar{y}_1) \vee \dots \vee Q_n(\bar{Y}_n) \leftarrow P_1(\bar{x}_1) \wedge \dots \wedge P_m(\bar{x}_m) \wedge \varphi),$
where $P_i, Q_j \in \mathcal{R}$, and φ is a conjunction of built-in atoms, the clause

$$\bigvee_{i=1}^n P_i(\bar{X}_i, \mathbf{f_a}) \bigvee_{j=1}^m Q_j(\bar{Y}_j, \mathbf{t_a}) \leftarrow \bigwedge_{i=1}^n P_i(\bar{X}_i, \mathbf{t^*}), \bigwedge_{j=1}^m Q_j(\bar{Y}_j, \mathbf{f^*}), dom(\bar{X}), \varphi;$$
- where $\bar{X}$ is the tuple of all variables appearing in database atoms in the rule.
6. For every referential IC of the form $\forall \bar{x}(P(\bar{x}) \rightarrow \exists y Q(\bar{x}', y))$, with $\bar{x}' \subseteq \bar{x}$, the clauses
 $P(\bar{X}, \mathbf{f_a}) \vee Q(\bar{X}', \text{null}, \mathbf{t_a}) \leftarrow P(\bar{X}, \mathbf{t^*}), \text{not } aux(\bar{X}'),$
 $\text{not } Q(\bar{X}', \text{null}, \mathbf{t_d}), dom(\bar{X}).$
 $aux(\bar{X}') \leftarrow Q(\bar{X}', Y, \mathbf{t_d}), \text{not } Q(\bar{X}', Y, \mathbf{f_a}), dom(\bar{X}', Y).$
 $aux(\bar{X}') \leftarrow Q(\bar{X}', Y, \mathbf{t_a}), dom(\bar{X}', Y).$
7. For every predicate $P \in \mathcal{R}$, the interpretation clauses:
 $P(\bar{a}, \mathbf{f^{**}}) \leftarrow P(\bar{a}, \mathbf{f_a}).$
 $P(\bar{a}, \mathbf{f^{**}}) \leftarrow \text{not } P(\bar{a}, \mathbf{t_d}), \text{not } P(\bar{a}, \mathbf{t_a}).$

¹If $\bar{X} = (X_1, \dots, X_n)$, we abbreviate $dom(X_1) \wedge \dots \wedge dom(X_n)$ with $dom(\bar{X})$.

$$\begin{aligned} P(\bar{a}, \mathbf{t}^{**}) &\leftarrow P(\bar{a}, \mathbf{t}_a). \\ P(\bar{a}, \mathbf{t}^{**}) &\leftarrow P(\bar{a}, \mathbf{t}_a), \text{ not } P(\bar{a}, \mathbf{f}_a). \end{aligned} \quad \square$$

Rules 6. repair referential ICs by deletion of tuples or insertion of null values that are not propagated through other ICs [Barcelo *et al.*, 2003]. For this purpose, we consider that the new constant $null \notin \mathcal{U}$, in particular $dom(null)$ is not a fact. The choice models of program $\Pi(\mathfrak{G}, IC)$ that do not contain a pair of literals of the form $\{P(\bar{a}, \mathbf{t}_a), P(\bar{a}, \mathbf{f}_a)\}$ are called *coherent models*.

Definition 11 *The instance associated to a choice model $\mathcal{M}$ of $\Pi(\mathfrak{G}, IC)$ is $D_{\mathcal{M}} = \{P(\bar{a}) \mid P(\bar{a}, \mathbf{t}^{**}) \in \mathcal{M}\}$.* $\square$

Theorem 2 *If $\mathcal{M}$ is a coherent choice model $\Pi(\mathfrak{G}, IC)$, then $D_{\mathcal{M}}$ is a repair of $\mathfrak{G}$ wrt IC . Furthermore, the repairs obtained in this way are all the repairs of $\mathfrak{G}$.* $\square$

5 Consistent Answers

Now, we can obtain the answers to queries posed to a system $\mathfrak{G}$ that are consistent wrt to IC . We do the following:

1. We start with a query Q that is expressed as a stratified Datalog program, $\Pi(Q)$, whose extensional predicates are elements of the global schema $\mathcal{R}$. Each positive occurrence of those predicates, say $P(\bar{t})$, is replaced by $P(\bar{t}, \mathbf{t}^{**})$; and each negative occurrence, say $\text{not } P(\bar{t})$, by $P(\bar{t}, \mathbf{f}^{**})$. This program has a query predicate Ans that collects the answers to Q . In particular, first order queries can be expressed as stratified Datalog programs [Abiteboul *et al.*, 1995].
2. Program $\Pi(Q)$ is appended to the program $SV(\Pi(\mathfrak{G}, IC))$, the stable version of the repair program.
3. The consistent answers to Q are the ground Ans atoms in the intersection of all stable models of $\Pi(Q) \cup SV(\Pi(\mathfrak{G}, IC))$.

Example 12 (example 9 cont.) *Consider the global symmetry integrity constraint $sim : \forall x \forall y (R(x, y) \rightarrow R(y, x))$ on $\mathfrak{G}_3$. We want the consistent answers to the query $Q : P(x, y)$. First, the query is written as the query program clause $Ans(X, Y) \leftarrow P(X, Y, \mathbf{t}^{**})$. This query program, $\Pi(Q)$, is run with $SV(\Pi(\mathfrak{G}_3, sim))$, which on its turn has five stable models with the following associated repairs: (a) $D_{\mathcal{M}_1^r} = \{P(a, b), R(b, b), P(a, c)\}$, the repair of the already consistent minimal instances $D_{\mathcal{M}_1}$ in Example 10; (b) $D_{\mathcal{M}_2^r} = \{P(a, a), P(a, c)\}$ and $D_{\mathcal{M}_3^r} = \{R(a, b), R(b, a), P(a, a), P(a, c)\}$, the repairs of the inconsistent instance $D_{\mathcal{M}_2}$; (c) $D_{\mathcal{M}_4^r} = \{P(a, c)\}$ and $D_{\mathcal{M}_5^r} = \{R(c, b), R(b, c), P(a, c)\}$, the repairs of instance $D_{\mathcal{M}_3}$; and (d) $D_{\mathcal{M}_6^r} = \{P(a, u), P(a, c)\}$ and $D_{\mathcal{M}_7^r} = \{R(u, b), R(b, u), P(a, u), P(a, c)\}$, the repairs of $D_{\mathcal{M}_4}$.*

The corresponding stable models of $\Pi(Q) \cup SV(\Pi(\mathfrak{G}_3, sim))$ are: (a) $\bar{\mathcal{M}}_1^r = \mathcal{M}_1^r \cup \{Ans(a, b), Ans(a, c)\}$; (b) $\bar{\mathcal{M}}_2^r = \mathcal{M}_2^r \cup \{Ans(a, a), Ans(a, c)\}$; $\bar{\mathcal{M}}_3^r = \mathcal{M}_3^r \cup \{Ans(a, a), Ans(a, c)\}$; (c) $\bar{\mathcal{M}}_4^r = \mathcal{M}_4^r \cup \{Ans(a, c)\}$; $\bar{\mathcal{M}}_5^r = \mathcal{M}_5^r \cup \{Ans(a, c)\}$; (d) $\bar{\mathcal{M}}_6^r = \mathcal{M}_6^r \cup \{Ans(a, u), Ans(a, c)\}$; $\bar{\mathcal{M}}_7^r = \mathcal{M}_7^r \cup \{Ans(a, u), Ans(a, c)\}$. $Ans(a, c)$ is the only query atom in all stable models, then the tuple (a, c) is the only consistent answer to the query. $\square$

If $\mathfrak{G}$ is consistent, then the consistent answers to Q computed with this method coincide with the minimal answers to Q , and

then to the certain answers if Q is monotone. If we are interested in just the minimal answers to Q , without considering consistency issues, then they can be computed as above, but using $\Pi(\mathfrak{G})$ introduced in Definition 8 instead of $\Pi(\mathfrak{G}, IC)$.

6 Conclusions

We have presented the most general approach so far to specifying, by means of disjunctive logic programs with a stable model semantics, the database repairs of a mediated integration system with open sources under the LAV approach. Then, consistent answers to queries posed to such a system are computed by running a query program together with the specification of database repairs under the skeptical or cautious stable model semantics. The specification of the repairs is achieved by first specifying the class of minimal global legal instances of the integration system. To the best of our knowledge, this is also the first specification, under the LAV paradigm, of such global instances in a logic programming formalism. This specification is inspired by the inverse rules algorithms, where auxiliary functions are replaced by auxiliary predicates that are forced to be functional by means of the non deterministic choice operator.

The methodology works for conjunctive view definitions, but can be extended to disjunctive views using the corresponding extension of the inverse rules algorithm [Duschka, 1997]. Wrt the ICs and queries it can handle, the approach works for arbitrary universal and referential integrity constraints and queries expressed as Datalog^{not} programs. In consequence, the current approach to consistent query answering (CQA) subsumes and extends the methodologies presented in [Bertossi *et al.*, 2002] for integration systems, and the one in [Barcelo *et al.*, 2002] for stand alone relational databases.

For reasons of space, we just mention a few optimizations of the specification programs and their execution. The materialization of the CWA present in $\Pi(\mathfrak{G}, IC)$ can be avoided by program transformation. Classes of common ICs can be identified for which $SV(\Pi(\mathfrak{G}, IC))$ becomes head-cycle-free, and in consequence, can be transformed into a non disjunctive program [Ben-Eliyahu *et al.*, 1994; Barcelo *et al.*, 2003]. The program for CQA can be split [Lifschitz *et al.*, 1994] into the program that specifies minimal legal instances, the program that specifies their repairs and the query program. The first is non stratified, but its models can be computed bottom-up as fixpoints of an iterative operator [Giannotti *et al.*, 2001]; and the second one is locally stratified [Przymusiński, 1991]. Finally, if the query program is stratified, e.g. if the original query is first-order, then the consistent answers can be eventually computed by a bottom-up evaluation mechanism.

For CQA we have successfully experimented with *DLV* [Eiter *et al.*, 2000]. The current implementations of the disjunctive stable models semantics would be much more effective in database applications if it were possible to evaluate open queries in a form that is guided by the query rather than based on, first, massive grounding of the whole program and, second, considering what can be found in every (completely constructed) stable model of the program.

Wrt related papers, query answering in mediated integration systems *under the assumption* that certain global ICs hold has been treated in [Gryz, 1999; Duschka *et al.*, 2000; Grant *et al.*, 2002; Cali *et al.*, 2002]. However, in CQA, we do not assume that global ICs hold. Logic programming specifications of repairs of single relational databases have been presented in [Arenas *et al.*, 2000; Greco *et al.*, 2001; Barcelo *et al.*, 2002].

[Lembo *et al.*, 2002] considers integration systems under the GAV approach that do not satisfy global key dependencies. There, legal instances are allowed to be more flexible, allowing their computed views to accommodate the satisfaction of the ICs. In this sense, the notion of repair is implicit; and the legal instances are the repairs we have considered here. View definitions are expressed as Datalog queries; and the queries to the global system are conjunctive. The “repairs” of the global system are specified by normal programs under stable model semantics.

In [Bertossi *et al.*, 2002], CQA in possibly inconsistent integration systems under the LAV approach is considered. There, the notion of repair of a minimal legal instance is introduced. The algorithm for CQA is based on a query transformation mechanism [Arenas *et al.*, 1999] applied to first-order queries. The resulting query may contain negation, and is run on top of an extension of the inverse algorithm to the case of stratified Datalog^{not} queries. This approach is limited by the restrictions of the query transformation methodology.

Acknowledgments

Work funded by DIPUC, MECESUP, CONICYT, Carleton University Start-Up Grant 9364-01, NSERC Grant 250279-02. L. Bertossi is Faculty Fellow of the IBM Center for Advanced Studies, Toronto Lab. We are grateful to Pablo Barceló, Alvaro Cortés, Ariel Fuxman, Nicola Leone, and Alberto Mendelzon for useful conversations.

References

- [Abiteboul *et al.*, 1995] Abiteboul, S.; Hull, R. and Vianu, V. *Foundations of Databases*. Addison-Wesley, 1995.
- [Abiteboul *et al.*, 1998] Abiteboul, A. and Duschka, O. Complexity of Answering Queries Using Materialized Views. In *Proc. ACM PODS*, 1998, pp. 254–263.
- [Arenas *et al.*, 1999] Arenas, M., Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. In *Proc. ACM PODS*, 1999, pp. 68–79.
- [Arenas *et al.*, 2000] Arenas, M.; Bertossi, L. and Chomicki, J. Specifying and Querying Database Repairs Using Logic Programs with Exceptions. In *Flexible Query Answering Systems. Recent Developments*. H. Larsen *et al.* (eds.), Springer, 2000, pp. 27–41.
- [Barcelo *et al.*, 2002] Barcelo, P. and Bertossi, L. Repairing Databases with Annotated Predicate Logic. In *Proc. International Workshop on Non-Monotonic Reasoning*. S. Benferhat and E. Giunchiglia (eds.), 2002, pp. 160–170.
- [Barcelo *et al.*, 2003] Barcelo, P., Bertossi, L. and Bravo, L. Characterizing and Computing Semantically Correct Answers from Databases with Annotated Logic and Answer Sets. In ‘Semantics of Databases’, Springer LNCS 2582, 2003, pp. 1–27.
- [Ben-Eliyahu *et al.*, 1994] Ben-Eliyahu, R. and Dechter, R. Propositional Semantics for Disjunctive Logic Programs. *Annals of Mathematics in Artificial Intelligence*, 1994, 12:53–87.
- [Bertossi *et al.*, 2002] Bertossi, L., Chomicki, J., Cortes, A. and Gutierrez, C. Consistent Answers from Integrated Data Sources. In *Flexible Query Answering Systems*, Springer LNAI 2522, 2002, pp. 71–85.
- [Cali *et al.*, 2002] Cali, A.; Calvanese, D.; De Giacomo, G. and Lenzerini, M. Data integration Under Integrity Constraints. In *Proc. CAISE*, Springer LNCS 2348, 2002, pp. 262–279.
- [Duschka, 1997] Duschka, O. *Query Planning and Optimization in Information Integration*. PhD Thesis, Stanford University, December 1997.
- [Duschka *et al.*, 2000] Duschka, O., Genesereth, M. and Levy, A. Recursive Query Plans for Data Integration. *J. Logic Programming*, 2000, 43(1):49–73.
- [Eiter *et al.*, 2000] Eiter, T., Faber, W.; Leone, N. and Pfeifer, G. Declarative Problem-Solving in DLV. In *Logic-Based Artificial Intelligence*. J. Minker (ed.), Kluwer, 2000, pp. 79–103.
- [Giannotti *et al.*, 1991] Giannotti, F.; Pedreschi, D.; Sacca, D. and Zaniolo, C. Non-Determinism in Deductive Databases. In *Proc. DOOD*, Springer LNCS 566, 1991, pp. 129–146.
- [Giannotti *et al.*, 2001] Giannotti, F.; Pedreschi, D. and Zaniolo, C. Semantics and Expressive Power of Nondeterministic Constructs in Deductive Databases. *J. Computer and System Sciences*, 62(1):15–42, 2001.
- [Grant *et al.*, 2002] Grant, J. and Minker, M. A Logic-based Approach to Data Integration. *Theory and Practice of Logic Programming*, 2002, 2(3):323–368.
- [Greco *et al.*, 2001] Greco, G., Greco, S. and Zumpano, E. A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases. In *Proc. ICLP*, Springer LNCS 2237, 2001, pp. 348–364.
- [Gryz, 1999] Gryz, J. Query Rewriting Using Views in the Presence of Functional and Inclusion Dependencies. *Information Systems*, 1999, 24(7):597–612.
- [Lembo *et al.*, 2002] Lembo, D.; Lenzerini, M. and Rosati, R. Source Inconsistency and Incompleteness in Data Integration. In *Proc. KRDB*, 2002.
- [Lenzerini, 2002] Lenzerini, M. Data Integration: A Theoretical Perspective. In *Proc. ACM PODS*, 2002, pp. 233–246.
- [Lifschitz *et al.*, 1994] Lifschitz, V. and Turner, H. Splitting a Logic Program. In *Proc. ICLP*. The MIT Press, 1994, pp. 23–37.
- [Przymusiński, 1991] Przymusiński, T. Stable Semantics for Disjunctive Programs. *New Generation Computing*, 9(3/4):401–424, 1991.

Query rewriting and answering under constraints in data integration systems

Andrea Calì Domenico Lembo Riccardo Rosati

Dipartimento di Informatica e Sistemistica

Università di Roma “La Sapienza”

Via Salaria 113, I-00198 Roma, Italy

{cali, lembo, rosati}@dis.uniroma1.it

Abstract

In this paper we address the problem of query answering and rewriting in global-as-view data integration systems, when key and inclusion dependencies are expressed on the global integration schema. In the case of *sound* views, we provide sound and complete rewriting techniques for a maximal class of constraints for which decidability holds. Then, we introduce a semantics which is able to cope with violations of constraints, and present a sound and complete rewriting technique for the same decidable class of constraints. Finally, we consider the decision problem of query answering and give decidability and complexity results.

1 Introduction

The task of a data integration system is to combine data residing at different sources, providing the user with a unified view of them, called *global schema*. User queries are formulated over the global schema, and the system suitably queries the sources, providing an answer to the user, who is not obliged to have any information about the sources. The problem of data integration is a crucial issue in many application domains, e.g., re-engineering legacy systems, data warehousing, data mining, data exchange.

A central aspect of query processing is the specification of the relationship between the global schema and the sources; such a specification is given in the form of a so-called *mapping*. There are basically two approaches for specifying the mapping. The first approach, called *global-as-view* (GAV), requires that a view over the sources is associated with every element of the global schema. Conversely, the second approach, called *local-as-view* (LAV), requires the sources to be defined as views over the global schema [Lenzerini, 2002; Duschka and Levy, 1997].

The global schema is a representation of the domain of interest of the data integration system: integrity constraints are expressed on such a schema to enhance its expressiveness, thus improving its capability of representing the real world.

Since sources are in general autonomous, the data provided by the sources are likely not to satisfy the constraints on the global schema. Integrity constraints have to be taken into account during query processing; otherwise, the system

may return incorrect answers to the user [Fagin *et al.*, 2003; Calì *et al.*, 2002].

Another significant issue is that the sources may not provide exactly the data that satisfy the corresponding portion of the global schema; in particular, they may provide either a subset or a superset of the data satisfying the mentioned portion, and the mapping is to be considered *sound* or *complete* respectively. Mappings that are both sound and complete are called *exact*.

In this paper, we restrict our analysis to the GAV approach, which is the most used in the context of data integration. In particular, we study a relational data integration framework in which key dependencies (KDs) and inclusion dependencies (IDs) are expressed on the global schema, and the mapping is considered sound. The main contributions of this paper are the following:

1. After showing that query answering in the general case is undecidable, we provide a sound and complete query rewriting technique first for the case of IDs alone, and then for the case of KDs together with the maximal class of IDs for which the problem is decidable, called *non-key-conflicting IDs*, or simply NKCIDs (Section 3).
2. Since it is likely that data retrieved at different, autonomous sources violate the KDs, we introduce a novel semantics that is a “relaxation” of the sound semantics, and that allows minimal repairs of the data (Section 4). We then present a sound and complete query rewriting technique in the case where KDs and NKCIDs are expressed on the global schema (Section 5).
3. Finally, we present decidability and complexity results of the (decision) problem of query answering in the different cases (Section 6).

2 Formal framework for data integration

In this section we define a logical framework for data integration, based on the relational model with integrity constraints.

Syntax We consider to have an infinite, fixed alphabet Γ of constants (also called values) representing real world objects, and will take into account only databases having Γ as domain. We adopt the so-called *unique name assumption*, i.e., we assume that different constants denote different objects.

Formally, a data integration system $\mathcal{I}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

1. $\mathcal{G}$ is the *global schema* expressed in the relational model with integrity constraints. In particular, $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, where (i) Ψ is a set of relations, each with an associated arity that indicates the number of its attributes. The attributes of a relation r of arity n are represented by the integers $1, \dots, n$. (ii) Σ_I is a set of *inclusion dependencies* (IDs), i.e. a set of assertions of the form $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$, where r_1, r_2 are relations in Ψ , $\mathbf{A} = A_1, \dots, A_n$ ($n \geq 0$) is a sequence of attributes of r_1 , and $\mathbf{B} = B_1, \dots, B_n$ is a sequence of attributes of r_2 . (iii) Σ_K is a set of *key dependencies* (KDs), i.e., a set of assertions of the form $key(r) = \mathbf{A}$, where r is a relation in the global schema, and $\mathbf{A} = A_1, \dots, A_n$ is a sequence of attributes of r such that for each $i \in \{1, \dots, n-1\}$ $a_i < a_{i+1}$. We assume, without loss of generality, that the attributes in $\mathbf{A}$ are the first n attributes of r . Moreover, we assume that at most one KD is specified for each relation.
2. $\mathcal{S}$ is the *source schema*, constituted by the schemas of the various sources that are part of the data integration system. We assume that the sources are relational, and that integrity constraints expressed on $\mathcal{S}$ are satisfied data at the sources. Hence, we do not take such constraints into account in our framework.
3. $\mathcal{M}$ is the *mapping* between the global and the source schema. In our framework the mapping is defined in the GAV approach, i.e., each relation in Ψ is associated with a *view*, i.e., a query, over the sources. We indicate the mapping as a set of assertions of the form $\langle r, V \rangle$, where r is a relation and V is the associated view over the source schema. We assume that the language used to express queries in the mapping is *positive Datalog* [Abiteboul *et al.*, 1995], over the alphabet of the relation symbols in $\mathcal{S}$. A Datalog query (or program) q of arity n is a collection of rules of the form $h(\vec{x}) \leftarrow conj(\vec{x}, \vec{y})$, where $conj(\vec{x}, \vec{y})$ is a set of atoms whose predicate symbols are either relation symbols in $\mathcal{S}$ or the head symbol h , and involve $\vec{x} = X_1, \dots, X_n$ and $\vec{y} = Y_1, \dots, Y_m$, where X_i and Y_j are either variables or values of Γ . We call $h(\vec{x})$ the *head* of the rule, and $conj(\vec{x}, \vec{y})$ the *body*.

Finally, a *query* over the global schema q is a formula that is intended to extract a set of tuples of elements of Γ . The language used to express queries over $\mathcal{G}$ is *union of conjunctive queries* (UCQ) [Abiteboul *et al.*, 1995], i.e., a Datalog program such that each rule head uses the same predicate of the same arity, and only relation symbols of $\mathcal{G}$ occur in each rule body.

Semantics A *database instance* (or simply *database*) $\mathcal{C}$ for a relational schema $\mathcal{DB}$ is a set of facts of the form $r(t)$ where r is a relation of arity n in $\mathcal{DB}$ and t is an n -tuple of values of the domain alphabet Γ . We denote as $r^{\mathcal{C}}$ the set $\{t \mid r(t) \in \mathcal{C}\}$; moreover, given a Datalog query q , we denote as $q^{\mathcal{C}}$ the evaluation of q over $\mathcal{C}$, i.e., the minimal fixpoint model of q and $\mathcal{C}$ [Abiteboul *et al.*, 1995].

In order to specify the semantics of a data integration system $\mathcal{I}$, we start by considering a *source database* for $\mathcal{I}$, i.e., a database $\mathcal{D}$ for the source schema $\mathcal{S}$. Based on $\mathcal{D}$, we now specify which is the information content of the global schema $\mathcal{G}$. We call *global database* for $\mathcal{I}$ any database for $\mathcal{G}$. For-

mally, given a source database $\mathcal{D}$ for $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, the semantics of $\mathcal{I}$ wrt $\mathcal{D}$, denoted $sem(\mathcal{I}, \mathcal{D})$, is a set of global databases for $\mathcal{I}$, where a global database $\mathcal{B}$ is in $sem(\mathcal{I}, \mathcal{D})$ if:

1. $\mathcal{B}$ is consistent with $\mathcal{G}$, i.e., it satisfies the IDs in Σ_I and the KDs in Σ_K . More formally: (i) $\mathcal{B}$ satisfies an inclusion dependency $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ if for each tuple t_1 in $r_1^{\mathcal{B}}$ there exists a tuple t_2 in $r_2^{\mathcal{B}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{B}]$, where $t[\mathbf{A}]$ is the projection of the tuple t over $\mathbf{A}$. If $\mathcal{B}$ satisfies all inclusion dependencies expressed on $\mathcal{G}$ we say that $\mathcal{B}$ is consistent with Σ_I ; (ii) $\mathcal{B}$ satisfies a key dependency $key(r) = \mathbf{A}$ if for each $t_1, t_2 \in r^{\mathcal{B}}$ with $t_1 \neq t_2$ we have $t_1[\mathbf{A}] \neq t_2[\mathbf{A}]$. If $\mathcal{B}$ satisfies all key dependencies expressed on $\mathcal{G}$ we say that $\mathcal{B}$ is consistent with Σ_K .
2. $\mathcal{B}$ satisfies the mapping $\mathcal{M}$ wrt $\mathcal{D}$, i.e., it satisfies each pair $\langle r, V \rangle$ in $\mathcal{M}$ wrt $\mathcal{D}$. In particular, we say that $\mathcal{B}$ satisfies the pair $\langle r, V \rangle$ wrt $\mathcal{D}$, if all the tuples satisfying V in $\mathcal{D}$ satisfy r in $\mathcal{B}$, i.e. $V^{\mathcal{D}} \subseteq r^{\mathcal{B}}$. Note that the above definition amounts to consider any view V as *sound*, which means that the data retrieved from sources satisfy the global schema, but are not necessarily complete.

By simply evaluating each view over the source database $\mathcal{D}$, we obtain a global database, called *retrieved global database* $ret(\mathcal{I}, \mathcal{D})$, that actually satisfies the sound mapping (but that is not necessarily consistent with $\mathcal{G}$).

We give now the semantics of queries. Formally, given a source database $\mathcal{D}$ for $\mathcal{I}$ we call *answers* to a query q of arity n w.r.t. $\mathcal{I}$ and $\mathcal{D}$, the set $ans(q, \mathcal{I}, \mathcal{D})$ defined as follows: $ans(q, \mathcal{I}, \mathcal{D}) = \{\langle c_1, \dots, c_n \rangle \mid \text{for each } \mathcal{B} \in sem(\mathcal{I}, \mathcal{D}), \langle c_1, \dots, c_n \rangle \in q^{\mathcal{B}}\}$.

In this paper, we address the query answering problem, that is the problem of computing the set $ans(q, \mathcal{I}, \mathcal{D})$. To this aim, we make use of query rewriting techniques, i.e., we exploit the mapping $\mathcal{M}$ to reformulate the query q into another query q_r , the *rewriting*, that can be evaluated on the source database $\mathcal{D}$. We say that q_r is a *perfect rewriting* of q w.r.t. $\mathcal{I}$ if $q_r^{\mathcal{D}} = ans(q, \mathcal{I}, \mathcal{D})$ for each $\mathcal{D}$. Furthermore, with regard to decidability and complexity results, we will refer to the decision problem associated to query answering, that is, given a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, a source database $\mathcal{D}$, a query q of arity n over $\mathcal{G}$ and a n -tuple $\vec{t}$ of values of Γ , to establish whether $\vec{t} \in ans(q, \mathcal{I}, \mathcal{D})$.

Example 2.1 Consider a data integration system $\mathcal{I}_0 = \langle \mathcal{G}_0, \mathcal{S}_0, \mathcal{M}_0 \rangle$, referring to the context of football teams. The global schema $\mathcal{G}_0$ consists of the relation predicates $player(Pname, Pcountry, Pteam)$ and $team(Tacronym, Tname, Tleader)$, and the following constraints: $key(player) = \{Pname\}$, $key(team) = \{Tacronym\}$, $team[Tleader] \subseteq player[Pname]$.

The source schema $\mathcal{S}_0$ consists of the schemas of three sources comprising the relation s_1 of arity 4, and the relations s_2 and s_3 , both of arity 3. Finally, the mapping $\mathcal{M}_0$ is defined by the two assertions

$$\begin{aligned} \langle player, & \quad player(X, Y, Z) \leftarrow s_1(X, Y, Z, W) \rangle \\ \langle team, & \quad team(X, Y, Z) \leftarrow s_2(X, Y, Z) \\ & \quad team(X, Y, Z) \leftarrow s_3(X, Y, Z) \rangle \end{aligned}$$

Consider the source database $\mathcal{D}_0 = \{s_1(\text{Totti}, \text{ITA}, \text{RM}, 27), s_1(\text{Beckham}, \text{ENG}, \text{MU}, 28), s_2(\text{RM}, \text{Roma}, \text{Totti}), s_3(\text{MU}, \text{Man.Utd.}, \text{Giggs})\}$. Then, $\text{ret}(\mathcal{I}_0, \mathcal{D}_0) = \{\text{player}(\text{Totti}, \text{ITA}, \text{RM}), \text{player}(\text{Beckham}, \text{ENG}, \text{MU}), \text{team}(\text{RM}, \text{Roma}, \text{Totti}), \text{team}(\text{MU}, \text{Man.Utd.}, \text{Giggs})\}$. Notice that the facts in $\text{ret}(\mathcal{I}_0, \mathcal{D}_0)$ together with the foreign key constraint $\text{team}[\text{Tleader}] \subseteq \text{player}[\text{Pname}]$ impose that *Giggs* is a player. Since the views are sound, the semantics for the integration system has to account for all the global databases that provide the country and the team of the player. Hence, $\text{sem}(\mathcal{I}_0, \mathcal{D}_0)$ contains all database instances that can be obtained by adding to $\text{ret}(\mathcal{I}_0, \mathcal{D}_0)$ (among others) at least one fact of the form $\text{player}(\text{Giggs}, \alpha, \beta)$, where α and β are values of the domain Γ . Given the query $q(X) \leftarrow \text{player}(X, Y, Z)$, we have that $\text{ans}(q, \mathcal{I}_0, \mathcal{D}_0) = \{\text{Totti}, \text{Beckham}, \text{Giggs}\}$. ■

3 Query rewriting

In this section we present algorithms for computing the perfect rewriting of a UCQ query in GAV integration systems with KDs and IDs. We first study the case in which only IDs are expressed on the global schema, then we deal with the simultaneous presence of both IDs and KDs.

Query rewriting under IDs only We start by studying query rewriting when only IDs are expressed on the global schema. To this aim, we need some preliminary definitions.

Given a conjunctive query q , we say that a variable X is *unbound* in q if it occurs only once in q , otherwise we say that X is *bound* in q . Notice that variables occurring in the head of the query are necessarily bound, since each of them must also occur in the query body. A *bound term* is either a bound variable or a constant.

In the following, we assume that all unbound variables in the query q are represented by the special term ξ .

Definition 3.1 Given an atom $g = s(X_1, \dots, X_n)$ and an inclusion $I = r[i_1, \dots, i_k] \subseteq s[j_1, \dots, j_k]$, we say that I is *applicable to g* if, for each ℓ such that $1 \leq \ell \leq n$, if $X_\ell \neq \xi$ then there exists h such that $j_h = \ell$. Moreover, we denote with $gr(g, I)$ the atom $s(Y_1, \dots, Y_m)$ (m is the arity of s in Ψ) where for each ℓ such that $1 \leq \ell \leq m$, $Y_\ell = X_{j_h}$ if there exists h such that $j_h = \ell$, otherwise $Y_\ell = \xi$.

Roughly speaking, an inclusion I is applicable to an atom g if the relation symbol of g corresponds to the symbol in the right-hand side of I and if all the attributes for which bound terms appear in g are propagated by the inclusion I . When I is applicable to g , $gr(g, I)$ denotes the atom obtained from g by using I as a rewriting rule whose direction is right-to-left.

Definition 3.2 Given an atom $g_1 = r(X_1, \dots, X_n)$ and an atom $g_2 = r(Y_1, \dots, Y_n)$, we say that g_1 and g_2 *unify* if for each i such that $1 \leq i \leq n$, either $X_i = Y_i$ or $X_i = \xi$ or $Y_i = \xi$. Moreover, if g_1 and g_2 unify, we denote as $U(g_1, g_2)$ the atom $r(Z_1, \dots, Z_n)$ where, for each i , if $X_i = Y_i$ or $Y_i = \xi$ then $Z_i = X_i$, otherwise $Z_i = Y_i$.

Informally, two atoms unify if they can be made equal through a substitution of each instance of the special symbol ξ with other terms.

Below we define the algorithm **ID-rewrite** to compute the perfect rewriting of a union of conjunctive queries Q . Informally, the algorithm computes the closure of the set of conjunctive queries Q with respect to the following two rules:

(i) if there exists a query $q \in Q$ such that $\text{body}(q)$ contains two atoms g_1 and g_2 that unify, then the algorithm computes the query $\text{reduce}(q, g_1, g_2)$, which is obtained from q by replacing g_1 and g_2 with $U(g_1, g_2)$ in the query body, and then by applying the substitution obtained in the computation of $U(g_1, g_2)$ to the whole query. Such a new query is then transformed by the function τ , which replaces with ξ each variable symbol X such that there is a single occurrence of X in q . The use of τ is necessary in order to guarantee that each unbound variable is represented by the symbol ξ . Such a query is then added to Q .

(ii) if there exists an inclusion I and a query $q \in Q$ containing an atom g such that I is applicable to g , then the algorithm adds to Q the query obtained from q by replacing g with $gr(g, I)$ in its body (denoted in the algorithm as $q[g/gr(g, I)]$). Namely, this step adds new conjunctions obtained by applying inclusion dependencies as rewriting rules (applied from right to left).

The above rules correspond respectively to steps (a) and (b) of the algorithm.

Algorithm ID-rewrite(Ψ, Σ_I, q)

Input: relational schema Ψ , inclusion dependencies Σ_I , union of conjunctive queries Q

Output: perfect rewriting of Q

$Q' := Q$;

repeat

$Q_{aux} := Q'$;

for each $q \in Q_{aux}$ **do**

 (a) **for each** $g_1, g_2 \in \text{body}(q)$ **do**

if g_1 and g_2 unify

then $Q' := Q' \cup \{\tau(\text{reduce}(q, g_1, g_2))\}$;

 (b) **for each** $g \in \text{body}(q)$ **do**

for each $I \in \Sigma_I$ **do**

if I is applicable to g

then $Q' := Q' \cup \{q[g/gr(g, I)]\}$

until $Q_{aux} = Q'$;

return Q'

Termination of the algorithm is immediately implied by the fact that the number of conjunctions that can be generated by the algorithm is finite, since the maximum length of a generated conjunction is equal to the maximum length of a conjunction in the body of the initial query Q , and the number of different atoms that can be generated by the algorithm is finite, since the alphabet of relation symbols used is finite (and corresponds to the relation symbols occurring in Q and in Σ_I), as well as the set of terms used (corresponding to the set of variable and constant names occurring in the query Q plus the symbol ξ).

Henceforth, we denote as Π_{ID} the UCQ returned by **ID-rewrite**(Ψ, Σ_I, Q). Moreover, we define the Datalog program $\Pi_{\mathcal{M}} = \{V | \langle r, V \rangle \in \mathcal{M}\}$.

Theorem 3.3 Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be an integration system and let Q be a UCQ query over $\mathcal{G}$. Then, $\Pi_{ID} \cup \Pi_{\mathcal{M}}$ is a perfect rewriting of Q w.r.t. $\mathcal{I}$.

Query rewriting under KDs and IDs Now we address the problem of query rewriting in the case where KDs and IDs are defined on the global schema. Unfortunately, KDs and IDs interact reciprocally so that the (decision) problem of query answering in this setting becomes undecidable. The following theorem is a consequence of a similar property proved in [Calì *et al.*, 2003] in the context of a single database.

Theorem 3.4 Consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, where Σ_I and Σ_K are sets of IDs and KDs respectively. Given a source database for $\mathcal{I}$, a query q over $\mathcal{G}$, and a tuple $\bar{t}$ of values of Γ , the problem of calculating $\text{ans}(q, \mathcal{I}, \mathcal{D})$ is undecidable.

Undecidability of calculating the certain answers to a query immediately implies undecidability of calculating the perfect rewriting [Calì *et al.*, 2003]. The problem of query answering becomes decidable if we restrict the IDs to be in a particular class, so that they do not interact with KDs.

Definition 3.5 Consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$. An ID $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$ is a *non-key-conflicting ID (NKCID)* w.r.t. $\mathbf{K}$ if either: (i) no KD is defined on r_2 ; (ii) the KD $\text{key}(r_2) = \mathbf{K}$ is in Σ_K and $\mathbf{A}_2$ is not a strict superset of $\mathbf{K}$, i.e., $\mathbf{A}_2 \not\supset \mathbf{K}$. If all IDs in Σ_I are NKCIDs w.r.t. Σ_K , the system $\mathcal{I}$ is said *non-key-conflicting (NKC)*.

We point out that the class of NKC IDs comprises the well-known class of *foreign key dependencies*, which correspond to IDs of the form $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$ such that $\text{key}(r_2) = \mathbf{A}_2$.

The most important feature of a NKC data integration system is the *separation* between the IDs and the KDs; in such a case, we can take IDs into account as if the KDs were not expressed on $\mathcal{G}$.

Theorem 3.6 (Separation) Given a NKC data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, let $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M} \rangle$, with $\mathcal{G}' = \langle \Psi, \Sigma_I, \emptyset \rangle$, the system obtained by $\mathcal{I}$ by eliminating the KDs of $\mathcal{G}$; let $\mathcal{D}$ be a source database for $\mathcal{I}$ and $\mathcal{I}'$. Moreover, let q be a query of arity n over $\mathcal{G}$ and $\mathcal{G}'$, and $\bar{t}$ an n -tuple of values. We have that $\bar{t} \notin \text{ans}(q, \mathcal{I}, \mathcal{D})$ iff $\mathcal{D}$ is consistent with Σ_K and $\bar{t} \notin \text{ans}(q, \mathcal{I}', \mathcal{D})$.

Proof (sketch). We say that $\mathcal{D}$ is consistent with Σ_K iff $\text{ret}(\mathcal{I}, \mathcal{D})$ is consistent with Σ_K . An important result, immediately derived from [Johnson and Klug, 1984], states that $\text{ans}(q, \mathcal{I}, \mathcal{D})$ is obtained by evaluating q over a (possibly infinite) database, called *chase* and denoted with $\text{chase}(\text{ret}(\mathcal{I}, \mathcal{D}))$. The chase is obtained by adding tuples to $\text{ret}(\mathcal{I}, \mathcal{D})$ in a way that the added tuples repair violations of IDs. The chase satisfies the IDs over $\mathcal{G}$, and it is a representative of all databases in $\text{sem}(\mathcal{I}, \mathcal{D})$ [Calì *et al.*, 2002].

“ $\Rightarrow$ ” Since by hypothesis $\bar{t} \notin \text{ans}(q, \mathcal{I}, \mathcal{D})$, there exists a global database $\mathcal{B} \in \text{sem}(\mathcal{I}, \mathcal{D})$ such that $\bar{t} \notin q^{\mathcal{B}}$. A fortiori, $\mathcal{B}$ satisfies Σ_I , therefore $\mathcal{B} \in \text{sem}(\mathcal{I}', \mathcal{D})$. The claim follows immediately.

“ $\Leftarrow$ ” By hypothesis, $\mathcal{D}$ is consistent with Σ_K and $\bar{t} \notin \text{ans}(q, \mathcal{I}', \mathcal{D})$. Therefore, $\bar{t} \notin q^{\text{chase}(\text{ret}(\mathcal{I}', \mathcal{D}))}$; note that $\text{chase}(\text{ret}(\mathcal{I}', \mathcal{D})) = \text{chase}(\text{ret}(\mathcal{I}, \mathcal{D}))$. It can be shown, by induction on the number of added tuples in the construction of $\text{chase}(\text{ret}(\mathcal{I}, \mathcal{D}))$, that if $\mathcal{I}$ is a NKC system, and $\text{ret}(\mathcal{I}, \mathcal{D})$ satisfies Σ_K , also $\text{chase}(\text{ret}(\mathcal{I}, \mathcal{D}))$ satisfies Σ_K .

It follows that $\text{chase}(\text{ret}(\mathcal{I}, \mathcal{D}))$ is a representative for all databases in $\text{sem}(\mathcal{I}, \mathcal{D})$, and $\text{ans}(q, \mathcal{I}, \mathcal{D}) = q^{\text{chase}(\text{ret}(\mathcal{I}, \mathcal{D}))}$. The claim follows straightforwardly. $\square$

We now go back to query rewriting. In the case of a NKC data integration system, we can apply the same technique developed for IDs alone, provided that we take into account the KDs with suitable rules. Indeed, observe that if $\text{ret}(\mathcal{I}, \mathcal{D})$ violates Σ_K , any tuple is in the answer to any query. Therefore, with regard to this issue, we first introduce a unary global relation *val*; the idea is that *val* stores all values occurring in $\mathcal{D}$. We construct a set of rules Π_{val} as follows: denoting with $\{r_1, \dots, r_{N_r}\}$ the set of all relations in $\mathcal{G}$,

$$\text{val}(X_j) \leftarrow r_i(X_1, \dots, X_{n_i})$$

with $1 \leq i \leq N_r$ and $1 \leq j \leq n_i$. Then, consider a KD of the form $\text{key}(r) = \mathbf{K}$; without loss of generality, we suppose that r has arity m and $\mathbf{K} = \{1, \dots, k\}$ ($k < m$). We introduce a set of $m - k$ rules; in particular, for $k + 1 \leq i \leq m$:

$$q(Y_1, \dots, Y_n) \leftarrow \begin{aligned} &r(X_1, \dots, X_k, X_{k+1}, \dots, X_m), \\ &r(X_1, \dots, X_k, X'_{k+1}, \dots, X'_m), \\ &X_i \neq X'_i, \text{val}(Y_1), \dots, \text{val}(Y_n) \end{aligned}$$

We denote with Π_{KD} the set of rules introduced as described above. From the results of Section 3 and from the above observation, we derive the following result.

Theorem 3.7 Consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and a query q of arity n over $\mathcal{G}$. Then, $\Pi_{\text{ID}} \cup \Pi_{\text{KD}} \cup \Pi_{\text{val}} \cup \Pi_{\mathcal{M}}$ is a perfect rewriting of q .

4 Semantics for inconsistent data sources

In the sound semantics, violations of IDs are treated “automatically” because of the nature of the semantics; instead, the violation of a single KD leads to the non-interesting case in which $\text{sem}(\mathcal{I}, \mathcal{D}) = \emptyset$.

According to a common approach in the literature on inconsistent databases [Fagin *et al.*, 1983; Lin and Mendelzon, 1998; Arenas *et al.*, 1999], we now introduce the *loosely-sound* semantics, opposed to the previous one (that we will call *strictly-sound*), in which the soundness assumption is suitably relaxed. The intuition is that in the “relaxed” semantics we are allowed to delete tuples from $\text{ret}(\mathcal{I}, \mathcal{D})$ to repair violations of KDs, as long as we “minimize” such deletions; violations of IDs are treated as in the sound semantics.

Given a source database $\mathcal{D}$, we define the following ordering $\gg_{(\mathcal{I}, \mathcal{D})}$ over the global databases for $\mathcal{I}$ that are consistent with $\mathcal{G}$. Given two such global databases $\mathcal{B}_1$ and $\mathcal{B}_2$, we write $\mathcal{B}_1 \gg_{(\mathcal{I}, \mathcal{D})} \mathcal{B}_2$, iff $\mathcal{B}_1 \cap \text{ret}(\mathcal{I}, \mathcal{D}) \supset \mathcal{B}_2 \cap \text{ret}(\mathcal{I}, \mathcal{D})$. That is, the portion of $\text{ret}(\mathcal{I}, \mathcal{D})$ contained in the global database is greater in $\mathcal{B}_1$ than in $\mathcal{B}_2$, i.e., $\mathcal{B}_1$ approximates the sound mapping better than $\mathcal{B}_2$.

We call *maximal* w.r.t. $(\mathcal{I}, \mathcal{D})$ a global database $\mathcal{B}$ for $\mathcal{I}$ consistent with $\mathcal{G}$, such that there exists no global database $\mathcal{B}'$ consistent with $\mathcal{G}$ such that $\mathcal{B}' \gg_{(\mathcal{I}, \mathcal{D})} \mathcal{B}$. Based on this notion, we define the *loosely-sound* semantics sem_l as follows: $\text{sem}_l(\mathcal{I}, \mathcal{D}) = \{\mathcal{B} \mid \mathcal{B} \text{ is consistent with } \mathcal{G} \text{ and } \mathcal{B} \text{ is maximal w.r.t. } (\mathcal{I}, \mathcal{D})\}$. Finally, we denote with $\text{ans}_l(q, \mathcal{I}, \mathcal{D})$ the set of answers to queries under the loosely-sound semantics.

Example 2.1 (cont.) Consider now the source database $\mathcal{D}'$ obtained by adding to $\mathcal{D}_0$ the fact $s_2(RM, Roma, Beckham)$. Then, $ret(\mathcal{I}_0, \mathcal{D}') = ret(\mathcal{I}_0, \mathcal{D}_0) \cup \{team(RM, Roma, Beckham)\}$. We have now that the tuples in $ret(\mathcal{I}_0, \mathcal{D}')$ violate also the KD $key(team) = \{Tacronym\}$; hence, $sem_I(\mathcal{I}_0, \mathcal{D}')$ contains the databases of the forms $\{player(Totti, ITA, RM), player(Beckham, ENG, MU), team(MU, Man.Utd., Giggs), team(RM, Roma, Totti), player(Giggs, \alpha, \beta)\}$ and $\{player(Totti, ITA, RM), player(Beckham, ENG, MU), team(MU, Man.Utd., Giggs), team(RM, Roma, Beckham), player(Giggs, \alpha, \beta)\}$, for each $\alpha, \beta \in \Gamma$. Notice that for the query $q(X) \leftarrow player(X, Y, Z)$ $ans_I(q, \mathcal{I}_0, \mathcal{D}') = ans(q, \mathcal{I}_0, \mathcal{D})$. On the other hand, given the query $q'(X, Z) \leftarrow team(X, Y, Z)$ we have that $ans_I(q', \mathcal{I}_0, \mathcal{D}') = \{\langle MU, Giggs \rangle\}$ whereas $ans(q', \mathcal{I}_0, \mathcal{D}) = \{\langle MU, Giggs \rangle, \langle RM, Totti \rangle\}$. ■

It is immediate to verify that, if $sem(\mathcal{I}, \mathcal{D}) \neq \emptyset$, then $sem_I(\mathcal{I}, \mathcal{D}) = sem(\mathcal{I}, \mathcal{D})$, i.e., if there exists a global database that satisfies both the constraints on $\mathcal{G}$ and the mapping assertions in $\mathcal{M}$ w.r.t. a source database $\mathcal{D}$, then the strictly-sound and the loosely-sound semantics coincide.

5 Query rewriting in loosely-sound semantics

We now address the problem of computing answers to a query under the loosely-sound semantics. Specifically, we present a rewriting technique to compute answers to queries posed to NKC systems under the loosely-sound semantics.

Our method relies on Theorem 3.6 stating that for NKC systems it is possible to “separately” deal with inclusion and key dependencies: actually, for the first ones we exploit the algorithm $ID\text{-}rewrite(\Psi, \Sigma_I, Q)$ presented in Section 3, whereas for the second ones we make use of Datalog⁻ under stable model semantics, a well-known extension of Datalog that allows for using negation in the body of program rules [Kolaitis and Papadimitriou, 1991].

More specifically, we define a Datalog⁻ program Π_{IKD} that allows us to compute the maximal subsets of $ret(\mathcal{I}, \mathcal{D})$ that are consistent with Σ_K . Π_{IKD} is obtained by taking, for each relation $r \in \mathcal{G}$, the rules

$$\begin{aligned} r(\vec{x}, \vec{y}) &\leftarrow r_{\mathcal{D}}(\vec{x}, \vec{y}), \text{ not } \bar{r}(\vec{x}, \vec{y}) \\ \bar{r}(\vec{x}, \vec{y}) &\leftarrow r_{\mathcal{D}}(\vec{x}, \vec{y}), r(\vec{x}, \vec{z}), Y_1 \neq Z_1 \\ &\dots \\ \bar{r}(\vec{x}, \vec{y}) &\leftarrow r_{\mathcal{D}}(\vec{x}, \vec{y}), r(\vec{x}, \vec{z}), Y_m \neq Z_m \end{aligned}$$

where: in $r(\vec{x}, \vec{y})$ the variables in $\vec{x}$ correspond to the attributes constituting the key of the relation r ; $\vec{y} = Y_1, \dots, Y_m$ and $\vec{z} = Z_1, \dots, Z_m$.

Informally, for each relation r , Π_{IKD} contains (i) a relation $r_{\mathcal{D}}$ that represents $r^{ret(\mathcal{I}, \mathcal{D})}$; (ii) a relation r that represents a subset of $r^{ret(\mathcal{I}, \mathcal{D})}$ that is consistent with the KD for r ; (iii) an auxiliary relation $\bar{r}$. The above rules force each stable model M of Π_{IKD} to be such that r^M is a maximal subset of tuples from $r^{ret(\mathcal{I}, \mathcal{D})}$ that are consistent with the KD for r .

Then, we consider the Datalog⁻ program $\Pi_{IKD} \cup \Pi_{ID} \cup \Pi_{\mathcal{MD}}$, where Π_{ID} is obtained through $ID\text{-}rewrite(\Psi, \Sigma_I, Q)$, and $\Pi_{\mathcal{MD}}$ is obtained from $\Pi_{\mathcal{M}}$ by replacing each symbol r with $r_{\mathcal{D}}$.

Theorem 5.1 Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a NKC system, and Q be a UCQ of arity n over $\mathcal{G}$. Then, $\Pi_{IKD} \cup \Pi_{ID} \cup \Pi_{\mathcal{MD}}$ is a perfect rewriting of Q w.r.t. $\mathcal{I}$.

6 Summary of complexity results

Strictly-sound semantics. Query answering is undecidable even if we allow a slightly more general class of IDs than the NKCIDs; let us define a 1-key-conflicting (1KC) data integration system as a system such that for each ID $r_1[A_1] \subseteq r_2[A_2]$, A_2 can be a strict superset of $key(r_2)$ (if defined), but containing at most one attribute more than $key(r_2)$.

Theorem 6.1 The problem of query answering in 1KC integration systems, under the strictly-sound semantics, is undecidable.

In the strictly-sound semantics, the complexity of the decision problem of query answering is immediately derived from the rewriting of Section 3.

Theorem 6.2 The problem of query answering in NKC integration systems, under the strictly-sound semantics, is in PTIME in data complexity.

Proof. Trivial, since the perfect rewriting $\Pi_{ID} \cup \Pi_{KD} \cup \Pi_{val} \cup \Pi_{\mathcal{M}}$ can be evaluated in PTIME w.r.t. $\mathcal{D}$. □

Loosely-sound semantics Since, as we already said, when $sem(\mathcal{I}, \mathcal{D}) \neq \emptyset$ the strict semantics and the loose ones coincide, it is easy to see that the above properties of query answering under the strictly-sound semantics can be easily generalized.

Theorem 6.3 The problem of query answering in 1KC integration systems, under the loosely-sound semantics, is undecidable.

We now characterize the problem of query answering under the loosely-sound semantics in NKC systems.

Theorem 6.4 The problem of query answering in NKC integration systems, under the loosely-sound semantics, is coNP-complete in data complexity.

Proof (sketch). Membership in coNP follows from Theorem 5.1, and from the fact that query answering in Datalog⁻ is coNP-complete in data complexity, while coNP-hardness can be easily proved by a reduction of the 3-COLORABILITY problem to our problem. □

The summary of the results we have obtained is reported in the table in Figure 1, which presents the complexity of query answering for both the strictly-sound and the loosely-sound semantics. Each row corresponds to a different class of dependencies (specified in the first two columns), while each cell of the table reports data complexity and combined complexity¹ of query answering for UCQs: for each decidable case, the complexity of the problem is complete w.r.t. the class reported. In the second column of the table, FK stands for “foreign key dependencies” (a well-known class of IDs) while GEN stands for “general IDs”. We have marked with the symbol ♠ the cells corresponding either to already known results or to results immediately implied by known results.

¹The results for combined complexity, which we cannot present in detail due to space limitations, hold under the assumption that the mapping is expressed in terms of UCQs.

KDs	IDs	strictly-sound	loosely-sound
no	GEN	PTIME/PSPACE [♣]	PTIME/PSPACE [♣]
yes	no	PTIME/NP [♣]	coNP/ Π_2^P [♣]
yes	FK	PTIME/PSPACE	coNP/PSPACE
yes	NKC	PTIME/PSPACE	coNP/PSPACE
yes	IKC	undecidable	undecidable
yes	GEN	undecidable [♣]	undecidable [♣]

Figure 1: Complexity of query answering (decision problem)

7 Discussion and related work

In this paper we have presented techniques for query rewriting in data integration systems with integrity constraints, and analyzed the complexity of query answering. To this aim, we have exploited formalisms and methods both from the traditional database theory and from computational logic.

Several works in the literature address the problem of data integration under constraints on the global schema. In this respect, query rewriting under integrity constraints has been first studied in the LAV setting. In particular, [Duschka and Genesereth, 1997] presents a method for query rewriting under functional dependencies in LAV systems, which is able to compute the perfect rewriting in the case of queries and mapping expressed through conjunctive queries, and “maximally contained” rewritings in the case of recursive mappings.

Then, [Gryz, 1999] analyzes query rewriting under inclusion dependencies in LAV systems, and presents a method which is able to deal simultaneously with acyclic IDs and functional dependencies, based on an algorithm for computing the rewriting of a conjunctive query in a database with inclusion dependencies. The algorithm is based on a very interesting idea: obtaining query rewriting by computing the rewriting of each atom in a way “almost independent” of the other atoms. This can be obtained if the body of the initial query q is preliminarily “minimized”. However, we have found out that Gryz’s algorithm does not actually compute the perfect rewriting, in the sense that some conjunctions of the perfect rewriting are missing. Our algorithm ID-rewrite presented in Section 3 is certainly inspired by Gryz’s main intuitions, but overcomes the above mentioned incompleteness through a new technique for generating the rewriting.

Complexity of query answering in GAV under IDs alone is immediately derived by the results in [Johnson and Klug, 1984]; in this work, the same problem is solved for a restricted class of KDs and IDs, which, however, is significantly less general than the one treated in this paper. More recently, integration under constraints in GAV systems has been addressed in [Calì et al., 2002], which presents a method for query rewriting in the presence of KDs and foreign key dependencies, under a semantics analogous to our strictly-sound semantics. Thus, the method does not deal with data inconsistencies w.r.t. KDs. Moreover, [Fagin et al., 2003] presents an approach for dealing with integrity constraints in a GLAV setting (a generalization of LAV and GAV).

In single database settings, [Arenas et al., 1999; Greco et al., 2001] propose methods for consistent query answering in inconsistent databases, which are able to deal with universally quantified constraints. The semantics adopted in these works

is different from the ones considered in the present paper.

Acknowledgments This research has been supported by the Projects INFOMIX (IST-2001-33570) and SEWASIE (IST-2001-34825) funded by the EU, and by the Project D2I funded by MIUR (Ministero per l’Istruzione, l’Università e la Ricerca). We thank Maurizio Lenzerini and Jarek Gryz for precious discussions about this material.

References

- [Abiteboul et al., 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachusetts, 1995.
- [Arenas et al., 1999] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proc. of PODS’99*, pages 68–79, 1999.
- [Calì et al., 2002] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. In *Proc. of CAiSE 2002*, volume 2348 of *LNCS*, pages 262–279. Springer, 2002.
- [Calì et al., 2003] Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS 2003*, 2003. To Appear.
- [Duschka and Genesereth, 1997] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proc. of PODS’97*, pages 109–116, 1997.
- [Duschka and Levy, 1997] Oliver M. Duschka and Alon Y. Levy. Recursive plans for information gathering. In *Proc. of IJCAI’97*, pages 778–784, 1997.
- [Fagin et al., 1983] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. In *Proc. of PODS’83*, pages 352–365, 1983.
- [Fagin et al., 2003] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proc. of ICDT 2003*, pages 207–224, 2003.
- [Greco et al., 2001] Gianluigi Greco, Sergio Greco, and Ester Zamparo. A logic programming approach to the integration, repairing and querying of inconsistent databases. In *Proc. of ICLP’01*, volume 2237 of *LNAI*, pages 348–364. Springer, 2001.
- [Gryz, 1999] Jarek Gryz. Query rewriting using views in the presence of functional and inclusion dependencies. *Information Systems*, 24(7):597–612, 1999.
- [Johnson and Klug, 1984] David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences*, 28(1):167–189, 1984.
- [Kolaitis and Papadimitriou, 1991] Phokion G. Kolaitis and Christos H. Papadimitriou. Why not negation by fixpoint? *J. of Computer and System Sciences*, 43(1):125–144, 1991.
- [Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.
- [Lin and Mendelzon, 1998] Jinxin Lin and Alberto O. Mendelzon. Merging databases under constraints. *Int. J. of Cooperative Information Systems*, 7(1):55–76, 1998.

Integrating Multiple Internet Directories by Instance-based Learning

Ryutaro Ichise, Hiedeaki Takeda, Shinichi Honiden

National Institute of Informatics

2-1-2 Hitotsubashi Chiyoda-ku, Tokyo, 101-8430 Japan

{ichise,takeda,honiden}@nii.ac.jp

Abstract

Finding desired information on the Internet is becoming increasingly difficult. Internet directories such as Yahoo!, which organize web pages into hierarchical categories, provide one solution to this problem; however, such directories are of limited use because some bias is applied both in the collection and categorization of pages. We propose a method for integrating multiple Internet directories by instance-based learning. Our method provides the mapping of categories in order to transfer documents from one directory to another, instead of simply merging two directories into one. We present herein an effective algorithm for determining similar categories between two directories via a statistical method called the κ -statistic. In order to evaluate the proposed method, we conducted experiments using two actual Internet directories, Yahoo! and Google. The results show that the proposed method achieves extensive improvements relative to both the Naive Bayes and Enhanced Naive Bayes approaches, without any text analysis on documents.

1 Introduction

The World-Wide Web (WWW) is now used not only by computer specialists, but by all sorts of people, from children to businessmen and businesswomen, which has resulted in an enormous quantity of web pages available on the Internet. This makes finding pages containing the desired information rather difficult. Search engines are requisite for finding the desired information. In general, current search engines perform their functions via one of two methods: a keyword search and a directory search. A keyword search engine performs searches by means of user-specified keywords. Keyword-based search engines like Google can reliably find pages containing the specified keywords. However, if the user does not have a thorough knowledge of his/her search domain and cannot choose the appropriate keywords, the search engine is useless. In such a situation, directory-based search engines do a good job. In a directory-based search engine, pages are evaluated and organized by humans before

being registered in the search engine's archive. Directory-based search engines provide knowledge of WWW navigation. Users reach the desired information by going up and down the directories.

Although pages are carefully selected and well-organized, a single Internet directory is not sufficient because the Internet directory tends to have some bias in both collecting and categorizing pages. In order to solve these problems, we herein propose a method that coordinates multiple Internet directories by estimating directory similarities. The proposed method does not simply merge multiple directories into a larger directory, but instead determines the relationship between directories. Many public Internet directories exist. Some are designed to cover wide domains, while others focus on special domains. Such Internet directories are difficult to merge. Moreover, these directories should not be integrated, because the existing differences in concept hierarchies among the directories is important when selecting and using directory-based search engines. In this paper, we propose a method to solve this problem by determining the rules for mapping categories in a directory to those in another directory. Our solution can be applicable not only to the Internet directory problem, but also to the integration of web marketplace catalogs [Agrawal and Srikant, 2001]¹ and ontology integration in general.

The remainder of this paper is organized as follows. In Section 2, we define the problem of coordinating multiple Internet directories. In Section 3, we discuss related studies. In Section 4, we propose a new machine learning method for the above-mentioned problem. Next, in Section 5, we compare the performance of the proposed method to that of the Enhanced Naive Bayes approach [Agrawal and Srikant, 2001] for an integration problem using real Internet directories. Finally, in Section 6, we present our conclusions.

2 Integration of Multiple Internet Directories

In order to state the problem, we introduce a model for the Internet directories we intend to integrate. We assume there are two Internet directories: a *source* directory and a *target* directory. The documents in the source Internet directory are

¹For example, the integration of a distributor's catalog and a web marketplace.

expected to be assigned to categories in the target Internet directory. This produces a *virtually* integrated Internet directory in which the documents in the source directory are expected to be members of both the source and target directories. This integrated directory inherits the categorization hierarchy from the target Internet directory.

The Internet directory model for the source and target is as follows:

- The source Internet directory, S_D contains a set of categories, $C_{s1}, C_{s2}, \dots, C_{sn}$, that are organized into an “is-a” hierarchy. Each category can also contain documents.
- The target Internet directory T_D contains a set of categories, $C_{t1}, C_{t2}, \dots, C_{tm}$ that are organized into an “is-a” hierarchy. Each category can also contain documents.

The proposed model permits documents to be assigned to intermediate categories. This model is similar to the catalog model in [Agrawal and Srikant, 2001], except that the categories are organized into a conceptual hierarchy. Since the catalog model ignores hierarchy structure and therefore cannot assign documents to an intermediate category, our Internet directory model is more general than the catalog model.

The problem addressed in this paper is finding an appropriate category C_t in the target directory T_D for each document D_{si} in the source directory S_D . An example is shown as the mapping of a black box D_X in Figure 1, where the black circles indicate categories and the hollow boxes indicate documents. What we need to do is determine an appropriate category in T_D for a document which appears in S_D but *not* in T_D , because mapping is not necessary if the document is included in both the source and the target directories. Documents D_1 and D_2 in Figure 1 are examples of such documents. This mapping can have several possibilities, e.g., D_X can be mapped to an upper left category or to a lower left category, etc.

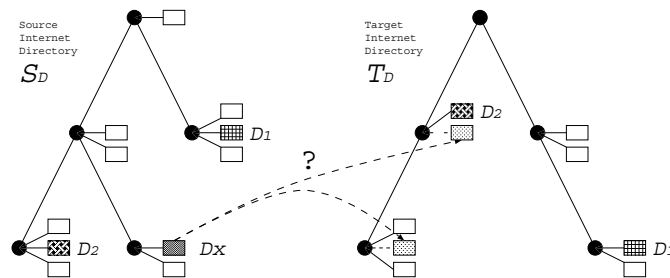


Figure 1: Problem statement.

3 Related Work

One popular approach to this kind of problem is to apply standard machine learning methods [Langley, 1996]. This requires a flattened class space having one class for every leaf node. The problem can then be considered to be a normal classification problem for documents. Naive Bayes (NB) [Mitchell, 1997] is an established method used for this type of document classification framework. A classifier

is constructed using the words in the documents. However, this classification scheme ignores the hierarchical structure of classes and, moreover, cannot use the categorization information in the source directory. Enhanced Naive Bayes [Agrawal and Srikant, 2001], hereafter referred to as E-NB, is a method which does use this information. E-NB will be discussed in the next section. GLUE [Doan *et al.*, 2002] is another type of system employing NB. To improve accuracy, GLUE combines NB and a constraint optimization technique called relaxation labeling. However, the general performance of the system depends on that of NB. Unlike NB, the systems in [Koller and Sahami, 1997], [Wang *et al.*, 1999] classify documents into hierarchical categories, and these systems use the words in the documents for classification rules. However, these systems cannot use the categorization information in the source directory.

Another type of approach is ontology merging/alignment systems. These systems combine two ontologies, which are represented in a hierarchical categorization. Chimaera [McGuinness *et al.*, 2000] and PROMPT [Noy and Musen, 2000] are examples of such systems and assist in the combination of different ontologies. However, such systems require human interaction for merging or alignment. In addition to this requirement, these systems are based on the similarity between words, which introduces instability. The dictionaries used for such systems often have word similarity bias. FCA-MERGE [Stumme and Madche, 2001] is another type of ontology merging method. It uses the attributes of concepts to merge different ontologies. As a result, it creates a new concept without regarding the original concepts in both ontologies. Calvanese *et al.* [Calvanese *et al.*, 2001] also discussed an ontology integration framework with respect to global ontology and local ontology.

As mentioned before, we also tackled the similar problem of catalog integration. An approach, besides E-NB, is to construct the abstract-level structure of two hierarchies [Omelayenko and Fensel, 2001]. This approach does not direct the transformation of source and target information, but transforms via the abstract-level structure. It is relatively easy to transfer information through many hierarchical structures, but it is hard to create a common structure for those hierarchies.

The bookmark-sharing systems of SiteSeer [Rucker and Polanco, 1997] and Blink [Blink, 2000] also deal with a similar problem. These systems attempt to share URL information which appears in the source bookmark but not in the target bookmark. These systems flatten the categorization of bookmarks in the same manner as NB and determine the mapping of categories in each bookmark, based on the shared URL information. Such systems are problematic when a given URL does not fit into an exact category. kMedia [Takeda *et al.*, 2000] is another bookmark-sharing system that uses hierarchical structures explicitly, but is dependent on the similarity of the words within pages.

In the context of information retrieval, Stuckenschmidt [Stuckenschmidt, 2002] has coordinated the formal model of hierarchies with multiple classification hierarchies. In our paper, since the upper and lower boundaries of concept mapping can be determined by subsumption relationships, we can infer the boundary of valid mapping and also check

whether or not the given mapping is consistent. In our problem, since we have only partial evidence of classes and do not know the definitions of classes, we can not apply this method directly.

4 Learning the Relationship between Categories of Two Internet Directories

In this section, we explain our method to determine the relationship between categories in two Internet directories. One characteristic of the proposed method is to use the hierarchical structure “as is.” We use all categories including intermediate categories and leaf categories, because information for categorization can also be obtained from these categories. Another characteristic of the proposed method is the exclusive reliance on the categorization structure of both directories, i.e., with no reliance on the semantic information of the documents. We can determine the relationship between categories of the two directories by statistically comparing the membership of the documents to the categories.

4.1 Basic Concept

Although E-NB was developed for a very similar problem, it is missing an important feature: categorization hierarchy. According to [Agrawal and Srikant, 2001], the initial definition of the problem is identical to that of this paper. However, the previous paper assumes that “any documents assigned to an interior node really belong to a conceptual leaf node that is a child of that node,” and concludes from this assumption that “we can flatten the hierarchy to a single level and treat it as a set of categories.” However, the conclusion overlooks the categorization hierarchies. The categorizations are not independent of each other. The categorization hierarchies are usually structured using an “is-a” or another relationship. If a document is categorized in a lower category in the categorization hierarchy, then the document should also be categorized in the upper categories². If we flatten the categories, such information regarding the relationships between categories will be lost.

Another problem associated with NB is its sensitivity to the words in documents. For example, if a document contains the word *bank*, the category for the document could be a financial category; however, the category could also be a construction category. The quality of categorization with NB is thus not stable according to the contents of documents, due to the natural language techniques of processing words. It is critical to integrate Internet directories because mixing reliable Internet directories maintained by human editors with less reliable and less stable machine-generated classifiers will confuse users and decrease their confidence. In addition to the problem of sensitivity, analysis of a document to extract words is expensive.

Our method focuses on the similarity of the way of categorization, not the similarity of documents. Then, how do we measure the similarity of categorization? We utilize shared documents in both the source and target Internet directories

²Note that the reverse is not always true, because documents can be categorized into intermediate categories.

as our measurement standard. If many documents in category C_{si} also appear in category C_{tj} at the same time, we consider these two categories to be similar, because the ways of categorization in C_{si} and C_{tj} are supposed to be similar, i.e., if another document D comes in C_{si} , it is likely that D will be also included in C_{tj} . This method avoids both the keyword extraction process and the handling of word meaning. The example shown in Figure 2 illustrates that documents in the bottom category in the source Internet directory can be transferred to a similar category in the target Internet directory.

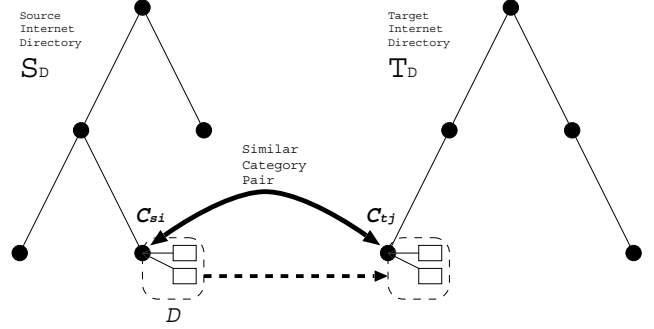


Figure 2: Document transfer from source Internet directory to target Internet directory.

4.2 κ -Statistic

The remaining problem is how to determine the pairs of similar categories. We adopt a statistical method to determine the degree of similarity between two categorization criteria. The κ -statistic method [Fleiss, 1973] is an established method for evaluating the similarity between two criteria. Suppose there are two categorization criteria, C_{si} in S_D and C_{tj} in T_D . We can determine whether or not a particular document belongs to a particular category³. Consequently, documents are divided into four classes, as shown in Table 1. The symbols N_{11} , N_{12} , N_{21} and N_{22} denote the numbers of documents for these classes. For example, N_{11} denotes the number of documents which belong to both C_{si} and C_{tj} . We may logically assume that if categories C_{si} and C_{tj} have the same criterion of categorization, then N_{12} and N_{21} are nearly zero, and if the two categories have a different criterion of categorization, then N_{11} and N_{22} are nearly zero. The κ -statistic method uses this principle to determine the similarity of categorization criteria.

		Category C_{tj}	
		belong	not belong
Category C_{si}	belong	N_{11}	N_{12}
	not belong	N_{21}	N_{22}

Table 1: Classification of documents by two categories.

In the κ -statistic method, we calculate the probability P ,

³Remember that categorization in a directory is determined using the nodal structure of the categorization hierarchy.

which denotes the percentage of coincidence of the conceptual criteria, and the probability P' , which denotes the percentage of coincidence of the conceptual criteria by chance.

$$P = \frac{N_{11} + N_{22}}{N_{11} + N_{12} + N_{21} + N_{22}}$$

$$P' = \frac{(N_{11} + N_{12})(N_{11} + N_{21}) + (N_{21} + N_{22})(N_{12} + N_{22})}{(N_{11} + N_{12} + N_{21} + N_{22})^2}$$

As such, the value of the κ -statistic is represented as the following equation:

$$\kappa = \frac{P - P'}{1 - P'}$$

Next, we examine whether we can assume the $\kappa = 0$, which indicates that the percentage of coincidence for the two conceptual criteria is zero. We therefore calculate the test statistic Z according to the following equation.

$$Z = \kappa \sqrt{\frac{(N_{11} + N_{12} + N_{21} + N_{22})(1 - P')}{P'}} \quad (1)$$

The value Z follows a normal distribution. A null hypothesis is considered to occur when the percentage of coincidence of the concept criteria is zero. When we assume a significance level of 5% and the following equation is satisfied, we can dismiss the null hypothesis.

$$Z \geq 1.64486$$

When the null hypothesis can be dismissed, the criteria are determined to be the same.

4.3 Determination of Pairs of Similar Categories

The relationship between the two categorization criteria is examined from top to bottom. This algorithm is shown in Figure 3. First, the most general categories in the two categorization hierarchies are compared using the κ -statistic. If the comparison confirms that the two categories are similar, then the algorithm outputs this pair of categories. At the same time, the algorithm generates all possible pairs of their children categories. It generates two lists of categories each of which is a list of the confirmed category and its children categories. Then, the algorithm picks one category from each list and makes a pair, except for the original pair. This new pair is then evaluated recursively using the κ -statistic method. When a similar pair is not generated, the algorithm outputs the pairs of similar concepts between the two categorization hierarchies. They are used as mapping rules from the source directory to the target directory⁴.

This top-down comparison approach attempts to reduce the exploration space, using the structure of both directories as a guide. Assuming that similar categories include similar sub-categories, we can skip the combination of categories which

⁴The rules do not guarantee the consistency of the mappings, because category hierarchies have the possibility of inconsistency with regard to similar relationships.

```

main_algorithm
Input:    $C_{s1}$ , // Top category in  $S_D$ 
         $C_{t1}$ , // Top category in  $T_D$ 
Output:   $R$ ;    // A set of similar category pairs
begin
   $t := 1$ ;
   $R := \phi$ ;
   $X_1 := \{[C_{s1}, C_{t1}]\}$ ;
  while  $X_t \neq \phi$ 
     $X_{t+1} := \phi$ ;
    while  $X_t \neq \phi$ 
       $[C_1, C_2] := \text{any element in } X_t$ ;
      if  $C_1$  and  $C_2$  is similar
         $X_{t+1} := X_{t+1} + \text{make\_comb}(C_1, C_2)$ ;
         $R := R + [C_1, C_2]$ ;
      fi;
       $X_t := X_t - [C_1, C_2]$ ;
    end;
     $t := t + 1$ ;
  end;
  return  $R$ ;
end;

make_comb
Input:    $C_s$ , // Category in  $S_D$ 
         $C_t$ , // Category in  $T_D$ 
Output:   $S$ ;    // A set of category pairs in  $S_D$  and  $T_D$ 
begin
   $S := \{[C_{si}, C_{tj}] \mid C_{si} \text{ is either } C_s \text{ or a child of } C_s, \\ C_{tj} \text{ is either } C_t \text{ or a child of } C_t\} - [C_s, C_t]$ ;
  return  $S$ ;
end;

```

Figure 3: Algorithm to determine similar category pair.

is likely unnecessary⁵. This is another benefit of using the “as is” hierarchy, in contrast to the flattening approach.

4.4 Application Policy of Rules

Since the proposed method uses categorization similarity, if a category in the source Internet directory does not have a similar category in the target directory, then those documents cannot be categorized in the target Internet directory. In order to avoid this problem, we once again use the categorization hierarchy. If the system cannot find a similar category pair for the category containing the document, it applies the rule generated for the parent category instead.

⁵This assumption may sometimes be too strong, e.g., a child category of a category is not so relevant, whereas its child category (a grandchild category of the original category) is relevant. In such cases, we should relax this constraint when exploring future candidates.

4.5 Computational Complexity

In this section, we discuss the computational complexity of the proposed algorithm shown in Figure 3. Suppose that two concept hierarchies S_D and T_D are trees of width b (the number of subcategories for each category) and height d (the number of categories from top to bottom) for the sake of simplicity. In this case, we calculate the total number of categories N using the following equation:

$$N = b^0 + b^1 + \dots + b^d$$

Comparison of the similarity among all categories requires $N \times N$ comparisons. As a result, the order of computational complexity is $O(b^{2d})$.

Next, we consider the cost of our algorithm. Assuming that after the similarity tests using κ -statistic have been conducted, $n\%$ of the tests are confirmed to be similar to other categories. The number of pairs produced by *make_comb* is $(b+1)^2 - 1$ because it makes pairs from the two lists, each pair of which consists of the category itself and its child categories, except for the original pair. The number of expected combination pairs after processing of the κ -statistic is $n(b^2 + 2b) + (1-n) \cdot 0$. The second half of the formula $(1-n) \cdot 0$ is applied because no similar pair is produced when the κ -statistic does not confirm the similarity. We finally obtain the total number of nodes M to be compared as the product of the number of examined nodes and the expected number of nodes. M is given by the following equation:

$$M = b^0 + n^1(b^2 + 2b)^1 + n^2(b^2 + 2b)^2 + \dots + n^d(b^2 + 2b)^d$$

The order of complexity is then obtained as $O(n^d b^{2d}) = O((nb^2)^d)$. Since n denotes the percentage of category pairs confirmed to be similar, $0 \leq n \leq 1$ is satisfied. As a result, we obtain less computational complexity using the algorithm in Figure 3. In other words, although the depth remains the same, the search algorithm can reduce computation by constraining the breadth size.

5 Experiments Using Internet Directories

5.1 Experimental Settings

In order to evaluate the proposed algorithm, we conducted experiments using data collected from the Yahoo! [Yahoo!, 2001] and Google [Google, 2001]⁶ Internet directories. The data was collected in the fall of 2001. In order to compare the proposed method to that in [Agrawal and Srikant, 2001], we selected the same locations in Yahoo! and Google for the experimental data. The locations are as follows:

- Yahoo! : Recreation / Automotive
Google : Recreation / Autos
- Yahoo! : Entertainment / Movies_and_Film
Google : Arts / Movies
- Yahoo! : Recreation / Outdoors
Google : Recreation / Outdoors

⁶Since the data in Google is constructed by the data in dmoz [dmoz, 2001], we collected data through dmoz.

- Yahoo! : Arts / Visual_Arts / Photography
Google : Arts / Photography
- Yahoo! : Computers_and_Internet / Software
Google : Computers / Software

Table 2 shows the numbers of categories, the links in each Internet directory and the links included in both Internet directories. Links are considered to be the same when the URLs in both directories are identical.

	Yahoo!		Google		shared links
	categories	links	categories	links	
Autos	782	5200	583	7909	1002
Movies	4001	16947	4623	21244	2735
Outdoors	2247	13932	825	13725	716
Photography	375	4173	170	3999	536
Software	997	4686	2142	35628	952

Table 2: Statistics on the experimental data.

We conducted ten-fold cross validations for the shared links. The shared links were divided into ten data sets; nine of these sets were used to construct rules, and the remaining set was used for testing. Ten experiments were conducted for each data set, and the average accuracy is shown in the results. In order to compare our proposed method to the E-NB approach, the classifiers are assumed to correctly assign documents when the document is categorized either in the same category as the test data or in the parent categories of the test data. E-NB constructs classifiers for only the first-level categories in the experimental domain, whereas the proposed method uses all of the categories from top to bottom. The significance level for the κ -statistic was set at 5%.

5.2 Experimental Results

The experimental results are shown in Tables 3 and 4. Table 3 shows the results obtained using Yahoo! as the source Internet directory and Google as the target Internet directory, and Table 4 shows the results obtained using Google as the source Internet directory and Yahoo! as the target Internet directory. For comparison, these tables also include the results of [Agrawal and Srikant, 2001]. E-NB denotes the method of [Agrawal and Srikant, 2001] and SBI denotes the similarity-based integration method that is proposed in this paper. The data obtained for the proposed method and that presented in [Agrawal and Srikant, 2001] are not truly comparable because the collection date is different⁷.

The proposed algorithm did well compared to E-NB, performing more than 10% better in accuracy on the averages. In the Movies domain, the proposed algorithm performs much better in accuracy than E-NB. One reason for this is that the pages related to movies contain various words that indicate numerous movie settings, making classification using word-based systems difficult. On the other hand, in the Outdoors domain, performance is similar for both methods because the

⁷In addition, differences may have occurred due to data selection. For example, Yahoo! has links that use @ (the at mark). The treatment of such links was not discussed in [Agrawal and Srikant, 2001]. In the present study, we did not use such links.

Dataset	Accuracy		Improvement
	E-NB	SBI	
Autos	76.2	88.0	11.8
Movies	42.6	77.9	35.3
Outdoors	77.8	74.0	-3.8
Photography	72.8	79.7	6.9
Software	62.4	73.4	11.0
Average	64.4	78.4	14.0

Table 3: Results of Yahoo! as the source Internet directory and Google as the target Internet directory.

Dataset	Accuracy		Improvement
	E-NB	SBI	
Autos	73.1	83.6	10.5
Movies	46.2	68.1	21.9
Outdoors	65.4	68.9	3.5
Photography	51.3	60.4	9.1
Software	58.6	74.0	15.4
Average	58.9	71.0	12.1

Table 4: Results of Google as the source Internet directory and Yahoo! as the target Internet directory.

Outdoors domain treats pages using special outdoors-related words, and no classification method for the Outdoors domain has been established for human readers.

As mentioned earlier, in the experiments described in [Agrawal and Srikant, 2001], the classifiers induced by the system have the ability to classify the documents into only the first-level categories of the test domain in the target Internet directory. The proposed classifiers can classify the documents into sub-categories as well. For example, we used 4,623 categories for document assignment in the Movies domain of Yahoo! as the source Internet directory and Google as the target Internet directory, whereas their system used only 40 categories. Therefore, the classifiers obtained by the proposed method are more reliable and useful than those obtained by other methods.

Next, we compare our method with GLUE [Doan *et al.*, 2002]. The basic steps of GLUE are as follows. First, the user trains learners for each concept for target taxonomies. Learners can be a combination of different strategies like name- or contents-based learning. Then, they determine that the source concept and target concept are similar, if the application of learners for the source concept yields good enough. Finally, the system applies the relaxation labeling method for similar concept pairs. The method does not use the hierarchical structures in learning very well, and relies mainly on the performance of the learning method, namely NB. On the other hand, our method does use hierarchical structures in learning very well. So, we do not need a semantical analysis in learning. They report an accuracy of 65-95% which is equivalent to our results, but they use only a few categories, i.e., 40-363, while we use 170-4623 categories. We can say from both theoretical and practical viewpoints that our method is more appropriate in learning for large hierarchical structures.

6 Conclusions

In this paper, a statistical-based technique was proposed for integrating multiple Internet directories by determining the relationship between these directories. The proposed method uses the κ -statistics to find similar category pairs, and transfers the document categorization from a category in the source Internet directory to a similar category in the target Internet directory. The proposed method has an advantage in document treatment in that it relies on the category structure only, and not on words or word similarity in a document. The performance of the proposed method was tested using actual Internet directories, and the results of these tests show that the performance of the proposed method was more than 10% better in accuracy than that of previous methods, although the number of categories were by far larger for our proposed method. In addition, our problem modeling is more general than the other models in terms of assignment documents on intermediate categories. From this, we can conclude our approach shows great promise in comparison to systems employing NB, such as E-NB [Agrawal and Srikant, 2001], GLUE [Doan *et al.*, 2002], and so on.

Although the present results are encouraging, much has yet to be done. The limitation of the proposed method is that this method is based on the existence of shared links. In other words, the proposed method is less reliable if there are fewer shared links. To improve our proposed method, various methods might be used to obtain semantic information in order to increase the number of shared links, e.g., to regard similar pages as the same links. Another option for this problem might be to combine semantic information or other information about the documents. If we could establish good combination of our proposed method and such information, it is possible that we could greatly improve our method. Moreover, the present method can find only one-to-one mapping rules. If the system is able to find one-to-many or many-to-many mappings, it would work more correctly in categorization. For the development of these mappings, we might invent probabilistic mapping representations and also introduce a subsumption technique [Stuckenschmidt, 2002]. Finally, the proposed method should be expanded so that it can apply to more than three concept hierarchies. In such a case, despite the conflict between several concept hierarchies, we would expect more information to be obtained. The aforementioned tasks for improvement will be investigated in future studies.

References

- [Agrawal and Srikant, 2001] Rakesh Agrawal and Ramakrishnan Srikant. On integrating catalogs. In *Proceedings of the Tenth International World Wide Web Conference (WWW-10)*, pages 603–612, 2001.
- [Blink, 2000] Blink. <http://www.blink.com/>, 2000.
- [Calvanese *et al.*, 2001] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. A framework for ontology integration. In *Proceedings of the First Semantic Web Working Symposium*, pages 303–316, 2001.
- [dmoz, 2001] dmoz. <http://dmoz.org/>, 2001.

- [Doan *et al.*, 2002] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the 11th International World Wide Web Conference*, 2002.
- [Fleiss, 1973] J. L. Fleiss. *Statistical Methods for Rates and Proportions*. John Wiley & Sons, 1973.
- [Google, 2001] Google. <http://directory.google.com/>, 2001.
- [Koller and Sahami, 1997] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In Douglas H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 170–178, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.
- [Langley, 1996] Pat Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996.
- [McGuinness *et al.*, 2000] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Proceedings of the Conference on Principles of Knowledge Representation and Reasoning (KR-00)*, pages 483–493, S.F., April 11–15 2000. Morgan Kaufman Publishers.
- [Mitchell, 1997] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [Noy and Musen, 2000] Natalya Fridman Noy and Mark A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, pages 450–455, Menlo Park, 2000. AAAI Press.
- [Omelayenko and Fensel, 2001] Borys Omelayenko and Dieter Fensel. An analysis of b2b catalogue integration problems. In *Proceedings of the International Conference on Enterprise Information Systems*, pages 945–952, 2001.
- [Rucker and Polanco, 1997] James Rucker and Marcos J. Polanco. Siteeer: Personalized navigation for the web. *Communications of the ACM*, 40(3):73–75, 1997.
- [Stuckenschmidt, 2002] Heiner Stuckenschmidt. Approximate information filtering with multiple classification hierarchies. *International Journal of Computational Intelligence and Applications*, 2(3):295–302, 2002.
- [Stumme and Madche, 2001] G. Stumme and A. Madche. Fca-merge: Bottom-up merging of ontologies. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 225–230, 2001.
- [Takeda *et al.*, 2000] Hideaki Takeda, Takeshi Matsuzuka, and Yuichiro Taniguchi. Discovery of shared topics networks among people. In R. Mizoguchi and J. Slaney, editors, *Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence (PRICAI-2000)*, volume 1886 of *LNAI*, pages 668–678, Berlin, 2000. Springer.
- [Wang *et al.*, 1999] Ke Wang, Senqiang Zhou, and Shi-ang Chen Liew. Building hierarchical classifiers using class proximity. In Malcolm Atkinson, Maria E. Orlowska, Patrick Valduriez, Stan Zdonik, and Michael Brodie, editors, *Proceedings of the 25th international Conference on Very Large Data Bases*, pages 363–374, Los Altos, CA 94022, USA, 1999. Morgan Kaufmann Publishers.
- [Yahoo!, 2001] Yahoo! <http://www.yahoo.com/>, 2001.

AI AND THE INTERNET

A semantic framework for multimedia document adaptation

Jérôme Euzenat, Nabil Layaïda, Victor Dias

INRIA Rhône-Alpes

655 avenue de l'Europe, 38330 Montbonnot Saint-Martin, France

{Jerome.Euzenat,Nabil.Layaïda}@inrialpes.fr

Abstract

With the proliferation of heterogeneous devices (desktop computers, personal digital assistants, phones), multimedia documents must be played under various constraints (small screens, low bandwidth). Taking these constraints into account with current document models is impossible. Hence, generic source documents must be transformed into documents compatible with the target contexts. Currently, the design of transformations is left to programmers. We propose here a semantic framework, which accounts for multimedia document adaptation in very general terms. A model of a multimedia document is a potential execution of this document and a context defines a particular class of models. The adaptation should then retain the source document models that belong to the class defined by the context if such models exist. Otherwise, the adaptation should produce a document whose models belong to this class and are “close” to those of the source documents. We focus on the temporal dimension of multimedia documents and show how adaptation can take advantage of temporal reasoning techniques. Several metrics are given for assessing the proximity of models.

1 Introduction

The multiplication of execution contexts for multimedia documents requires the adaptation of document specifications to the particularities of the contexts. Adaptation is not very precisely defined and it is currently specified through programming. We propose a semantic approach to multimedia documents (§2). It does not deal with the semantics of document content, but with that of their composition. The approach allows the definition of adaptation in very general semantic terms independent from the multimedia objects (§3). We then investigate the temporal dimension of multimedia documents specified qualitatively (§4) and propose metrics for finding the “best” adaptations. We discuss then the limitations of current multimedia specifications that prohibit a better adaptation (§5).

We first introduce the characteristics of multimedia documents (§1.1) and adaptation (§1.2).

1.1 Multimedia documents

A multimedia document is a digital document composed of objects of different nature: text, sound, image, animation, etc. These objects and their compositions are called multimedia objects. Multimedia documents are traditionally analysed following four dimensions [Layaïda, 1997]:

- logical (organisation into chapters, shots, etc.),
- spatial (graphic layout),
- hypermedia (relations between documents and document fragments),
- temporal (temporal ordering of the multimedia objects).

These dimensions are not totally independent and require a combined processing.

This paper primarily focuses on the adaptation of multimedia documents along their temporal dimension. In a temporal multimedia document, the presentation of the multimedia objects is scheduled over time. Such a document is presented in Figure 1. Time is displayed on the horizontal axis. The example presented is the introduction of a slideshow made of different panels composed of graphic objects that can be presented simultaneously. The first panel displays the title, authors and outline of the speech; each of these objects are represented by a segment whose begin and end points correspond to the beginning and ending of their presentation on screen.

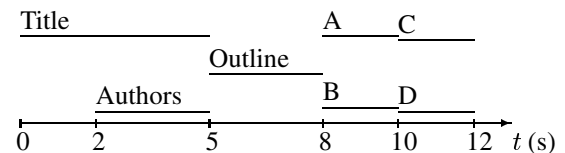


Figure 1: Temporal dimension of a multimedia document.

The Title object starts at second 0 and ends at second 5, while the Author object starts at second 2 and ends at second 5. Between seconds 5 and 8 the Outline object is presented, etc. Such a description is exact and quantitative since it defines exactly the beginning and ending instants of each multimedia object. This information is sufficient for playing the document: to one exact quantitative representation corresponds only one possible execution of the document (within a fixed temporal reference).

Specifying a multimedia document in an exact manner is like writing a paper directly in PostScript instead of using L^AT_EX. Multimedia documents are not often specified in an exact way because it is more convenient for the author to leave the interpretation of the specification to the machine as soon as the will of the author is clearly expressed. The author can concentrate on the creative part of it or her work instead of characterising the exact position of each object.

Non-exact specifications can be achieved by expressing the qualitative relations between multimedia objects. There are several languages for specifying multimedia documents with different ways of expressing the temporal dimension: SMIL [W3C, 1998] expresses the positioning of multimedia objects with parallel and sequence operators on intervals; Magic [Dalal *et al.*, 1996] and Madeus [Layaïda, 1997] use a restriction of the Allen algebra of temporal intervals.

The document of Figure 1 can be expressed qualitatively. For instance, the Authors object starts after and finishes with the Title object; the Authors object meets the Outline object. From such a specification, the multimedia presentation system (or the Player) computes a plan (called “scenario”) that can be executed. This function is called temporal formatting.

1.2 Adapting multimedia documents

A server delivers a multimedia document to be played by a client. Clients and servers can be different machines with different capabilities. Different contexts of multimedia presentations introduce different constraints on the presentation itself. For instance, bandwidth limitations between the client and the server can result in preventing the client from playing two bandwidth-demanding videos at the same time. Display limitations can produce similar constraints. Other constraints may also be introduced by user preferences, content protection or terminal capabilities. The constraints imposed by a client are called a profile.

Profiles can be expressed in terms of a restriction of the language used for specifying target documents or in terms of additional constraints imposed on the objects. For instance, if the device features only a screen with limited capabilities, it can be impossible to display two images simultaneously on the same screen.

For satisfying these constraints, multimedia documents must be adapted before being played. From the profile and the source document, the adaptation must provide a document satisfying the constraints expressed in the profile. Qualitative specifications are central to this process as they enable more efficient adaptation by providing more flexibility. This adaptation is usually performed by a program transforming the document [Villard, 2001; Lemlouma and Layaïda, 2001].

For the purpose of characterising the adaptation process, we introduce a semantics of multimedia documents and illustrate it on the temporal dimension (§2). The semantic definition of adaptation (§3) leads to distinguish refining adaptation (in which the models of the adapted document are models of the source documents) from transgressive adaptation (in which the models are as close as possible to those of the source document). Section 4 illustrates the notion of closeness for the temporal dimension. The limits of our approach are then presented (§5).

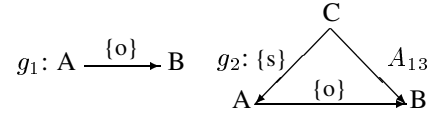


Figure 2: Relation graph (inverse arcs are not displayed).

2 A semantic approach to multimedia documents

We describe the specification of multimedia documents (§2.1) before defining their semantics (§2.2) used finally for defining adaptation (§3).

2.1 Specifications

We assume that the qualitative specifications of multimedia documents use the temporal interval algebra [Allen, 1983] for representing the temporal relationships between multimedia objects. So, the temporal extent of the multimedia objects will be a temporal interval I whose beginning and ending time are identified by I^- and I^+ . The specification will relate each pair of multimedia objects by a subset of the set A_{13} of temporal relations (presented in Table 1).

relation (r): $x \ r \ y$	x / y	converse: $y \ r^{-1} \ x$
before (b)	— —	(bi) after
meet (m)	— —	(mi) met by
during (d)	— —	(di) contains
overlaps (o)	— —	(oi) overlapped by
starts (s)	— —	(si) started by
finishes (f)	— —	(fi) finished by
equals (e)	— —	(e)

Table 1: The 13 relationships between temporal intervals.

Definition 1 (Specification). A specification $s = \langle O, C \rangle$ of a document is made of a set of objects O and a set of constraints C between these objects (i.e., a relation between several objects). The set of all specifications will be noted S .

Example 1 (Temporal specifications).

$$s_1 = \langle \{A, B\}, \{A\{o\}B\} \rangle$$

$$s_2 = \langle \{A, B, C\}, \{A\{o\}B, C\{s\}A\} \rangle$$

In the remainder, the constraints will be considered as binary. The temporal specification can then be represented as a relation graph [van Beek, 1992]. This representation will be used for describing models.

Definition 2 (Relation graph). A specification $s = \langle O, C \rangle$ can be represented as a complete direct labelled graph $g_s = \langle N, E, \lambda \rangle$ such that the elements of O are in bijection with those of N and $\lambda : E \rightarrow 2^{A_{13}}$ is a total function from the arcs to temporal relations such that for each $x \ r \ y \in C$, $\lambda(\langle x, y \rangle) \subseteq r$.

Definition 3 (Resolved relation graph). A relation graph is resolved iff all the labels are singletons.

2.2 Semantics of a specification

The specification of a multimedia document is interpreted as the set of its potential executions. A model of a multimedia document (in the sense of model theory) is an execution of the document satisfying the specification.

Definition 4 (Interpretation). An interpretation of a specification is a pair $\langle I, D \rangle$ such that D is the domain of interpretation and I is a function from O to D and from C to $D \times D$ (i.e., such that a constraint applied to two elements of the domain of interpretation is either true or false).

Example 2 (Temporal interpretation). In order to interpret the temporal aspects of multimedia documents, we consider the interpretations such that the objects in O are interpreted as intervals of the positive real numbers and the constraints are interpreted as the corresponding relations in the temporal interval algebra. For instance, A can be interpreted as the interval $[10\ 20]$, B as $[12\ 30]$ and C as $[10\ 30]$, o is true if its first argument begins before the second one and ends during it, $\{b\ m\ o\}$ is true if its first argument begins before the second one and ends before the end of the second one.

A model is defined in the usual way:

Definition 5 (Model). A model of a specification $\langle O, C \rangle$ is an interpretation $\langle I, D \rangle$ such that for each $o\ r\ o' \in C$, $\langle I(o), I(o') \rangle \in I(r)$ is true. The set of models of a specification s is noted $\mathcal{M}_s$.

Example 3 (Temporal model). The interpretation presented in Example 2 is a model of s_1 but not of s_2 .

In the following, we will always consider that there exists at least one model of the source specification (which is thus consistent).

These models correspond faithfully to the execution of the multimedia documents. However, the formatter will consider executions as equal if they only differ by a translation factor and the adaptation will consider two executions as equal if they only differ in duration, preserving topology and ordering. We introduce qualitative representations of models as abstractions of models.

Definition 6 (Qualitative representation of a model). The qualitative representation of a model $\langle I, D \rangle$ of a specification $\langle O, C \rangle$ is a complete direct labelled graph $\langle N, E, \lambda \rangle$ such that the elements of O are in bijection with those of N and $\lambda : E \rightarrow 2^{A_{13}}$ is a total function from the arcs to temporal relations such that for each $I(x)\ r\ I(y)$, $\lambda(\langle x, y \rangle) = r$.

Since the Allen relations are exclusive and exhaustive, qualitative representations of a model correspond to resolved relation graphs.

Example 4. $q_{s_2}^1 = \langle \{A, B, C\}, \{A\{o\}B, C\{s\}A, C\{m\}B\} \rangle$ is a qualitative representation of one of the three models of s_2 .

3 Semantics of adaptation

The adaptation of a multimedia document is constrained by the profile. The profile defines constraints that must be satisfied by the document to be played.

3.1 Adaptation constraints

Definition 7 (Adaptation constraint). An adaptation constraint a determines a set of possible executions $\mathcal{M}_a$. The set of adaptation constraints will be noted $\mathcal{A}$.

The example above introduced a constraint prohibiting more than one image to be displayed at once on a screen. This can be expressed by a MSO constraint.

Example 5 (Maximum Simultaneous Objects). The constraint $\text{MSO}_T(n)$ (Maximum Simultaneous Objects) is a global constraint prohibiting the display of more than n objects belonging to the set T simultaneously. It thus determines the set of interpretations $\langle I, D \rangle$ of a specification $s = \langle O, C \rangle$, such that $\forall i \in \mathbb{R}, |\{o \in T; I(o)^- \leq i \leq I(o)^+\}| \leq n$. In the remainder, MSO will be used instead of MSO_O .

A profile p is a set of such constraints. It determines a class of qualitative models (those who satisfy the constraints). The role of adaptation is thus to determine if there exist models of the initial specification belonging to that class. Otherwise, it is convenient to alter the specification by finding, among the set of models satisfying the profile, those that are “semantically closer” to the source specification.

Definition 8 (Classification of adaptation). Three types of adaptation can be identified in function of the value of $\mathcal{M}_s \cap \mathcal{M}_p$ (inducing three different constraints on the model selection function α):

Compliant specification $\mathcal{M}_s \cap \mathcal{M}_p = \mathcal{M}_s$: the source document satisfies the adaptation constraints (all models of $\mathcal{M}_s$ satisfy the adaptation constraints, so α is identity).

Refining adaptation $\emptyset \subset \mathcal{M}_s \cap \mathcal{M}_p \subset \mathcal{M}_s$: there exists some models of s satisfying the adaptation constraints ($\alpha(\mathcal{M}_s) = \mathcal{M}_s \cap \mathcal{M}_p$).

Transgressive adaptation $\mathcal{M}_s \cap \mathcal{M}_p = \emptyset$: no model of s satisfies the adaptation constraints (α will then select some models of $\mathcal{M}_p$ closest to those of the specification s).

If the constraints of the profile can be expressed as a formula of the specification language, the two first cases are characterised by the consistency of $s \cup p$. The $\text{MSO}_T(1)$ constraint can be expressed by the relation graph in which all the labels of arcs connecting two nodes in T are subsets of $\{b\ m\ mi\ bi\}$. Filtering it can be efficiently performed.

3.2 Problems

One of the benefits of the approach is to be able to clearly provide criteria that an adaptation function τ must meet. These criteria are expressed here as a set of problems.

The first one is that the adapted specification must satisfy the adaptation constraints.

Problem (soundness). $\mathcal{M}_{\tau(s)} \subseteq \mathcal{M}_p$: do the models of the adapted specification satisfy the adaptation constraints?

Moreover, if there exists a possible execution of the document satisfying the adaptation constraints, this execution must be preserved in the adapted specification.

Problem (refining-completeness). if $\mathcal{M}_s \cap \mathcal{M}_p \neq \emptyset$ then $\mathcal{M}_{\tau(s)} \supseteq \mathcal{M}_s \cap \mathcal{M}_p$

In such a case, the adaptation should not authorise models that were not models of the source specification.

Problem (refining-parsimony). if $\mathcal{M}_s \cap \mathcal{M}_p \neq \emptyset$ then $\mathcal{M}_{\tau(s)} \subseteq \mathcal{M}_s$.

Unfortunately, no guarantee is given that the languages used for expressing the specifications and the adaptation constraints allow the expression of a specification satisfying these requirements.

Problem (representability). $\forall s \in \mathcal{S}; \forall p \subseteq \mathcal{A}$, Does $\tau(s) \in \mathcal{S}$ exist such that $\mathcal{M}_{\tau(s)} = \mathcal{M}_s \cap \mathcal{M}_p$.

Moreover, one constraint that should be achieved by a semantic approach is that the result of adaptation must not depend on the syntactic form of the specification.

Problem (syntax independence). $\forall s, s' \in \mathcal{S}; \mathcal{M}_s = \mathcal{M}_{s'} \Rightarrow \mathcal{M}_{\tau(s)} = \mathcal{M}_{\tau(s')}$

Taking the semantic approach to multimedia document adaptation allows the characterisation of adaptation in a very general way depending only on model theoretic considerations. In particular, these definitions are totally independent from the language used for expressing documents and profiles as well as the multimedia object and constraint types.

This characterisation clearly emphasises the constraints that a refining adaptation must meet and that can be overlooked when programming the transformation.

Transgressive adaptation is more difficult to characterise and this is considered in the next section.

4 Transgressive adaptation in the temporal dimension

The goal of transgressive adaptation is to find a specification $\tau(s)$ as close as possible to the source specification s . Semantically, this amounts to find the specifications whose models are the closest possible to those of the source specification. Figure 3 shows the set of models satisfying MSO(1). A distance Δ must be defined between two sets of models in order to find which models to select.

Problem (compactness). if $\mathcal{M}_s \cap \mathcal{M}_p = \emptyset$ then $\tau(s) = \mu_{\Delta(\mathcal{M}_{s'}, \mathcal{M}_s)} \{s' \in \mathcal{S}; \mathcal{M}_{s'} \subseteq \mathcal{M}_p\}$

Example 6. For the specification $s_1 = \langle \{A, B\}, \{A\{o\}B\} \rangle$ and the constraint of displaying only one graphic object at once (MSO(1)), adaptation must be transgressive. The 13 possible models correspond to the 13 possible Allen relations. Among these models, only those built from before ($\{b\}$), meets ($\{m\}$), met by ($\{mi\}$) and after ($\{bi\}$) satisfy the adaptation property MSO(1). These possible combinations are presented in Figure 3.

Applying a semantic approach to transgressive adaptation can be compared to the use of the semantic approach for knowledge base revision [Dalal, 1988]. This will be the first step taken here. But we will show that the simple distance used for comparing models is not sufficient for adapting multimedia documents. Measures depending on the kind of multimedia objects are required.

The usual way to compute the distance between sets of models is function of a distance d between two models and a method of aggregation F .

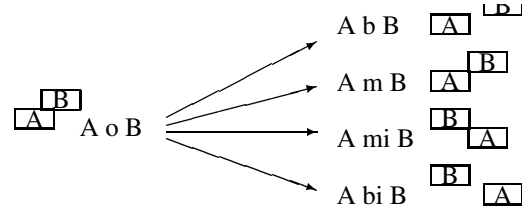


Figure 3: The four solutions corresponding the linearization of an overlapping ($\mathcal{M}_{MSO(1)}$).

Definition 9 (Distance between sets of models).

$$\Delta(\mathcal{M}, \mathcal{M}') = F_{m \in \mathcal{M}, m' \in \mathcal{M}'} d(m, m')$$

Various methods exist for aggregating distances (e.g., single linkage, full linkage, Hausdorf distance). The single linkage (i.e., the distance between two sets of models is the smallest distance between a pair of elements of both sets) seems better for semantics because the designer of the specification always has a privileged model in mind.

The second element is the distance d that is considered hereafter¹.

4.1 Distance on the qualitative models

The first distance that comes to mind consists of counting the relations between two objects that differ between models. It is comparable to the Hamming distance (i.e., the cardinal of the symmetric difference between two sets) counting the propositional atoms that do not have the same truth-value in propositional logic [Dalal, 1988]. Because qualitative models correspond to resolved relation graphs, the distance is easily computed by counting the labels which differ between each pair of objects. The distance is defined on graphs (and more precisely on their labelling functions).

Definition 10 (Distance between resolved relation graphs).

$$d(\lambda, \lambda') = \sum_{n, n' \in N} \begin{cases} 1 & \text{if } \lambda(\langle n, n' \rangle) \neq \lambda'(\langle n, n' \rangle) \\ 0 & \text{otherwise} \end{cases}$$

Example 7. Concerning s_1 , the four models are all at the same distance from the model of the source specification because they all differ by two relations (the one between A and B and its converse).

In order to find more precise results that discriminate between the four models of the example, the relations between intervals can be transcribed into relations between their begin and end points and the same sort of distance can be used.

4.2 Point-based distance on the qualitative models

The relations of the interval algebra can be represented by relations of the instant algebra between the endpoints of the intervals. For instance, the meet relation (m) will be represented by $\gamma(m) = \langle <, <, =, < \rangle$. The distance between two intervals based on their endpoints will be the number of positions in the 4-uple that differ.

¹The metrics presented satisfy all the properties of distances [Monjardet, 1981]. This is not discussed due to space constraints.

Definition 11 (Distance between interval relations based on endpoints).

$$\delta(r, r') = \sum_{i=1}^4 \begin{cases} 1 & \text{if } \gamma(r)[i] \neq \gamma(r')[i] \\ 0 & \text{otherwise} \end{cases}$$

The distance between models is the sum of the distance between each interval relation.

Definition 12 (Distance between models based on endpoints).

$$d(\lambda, \lambda') = \sum_{n, n' \in N} \delta(\lambda(\langle n, n' \rangle), \lambda'(\langle n, n' \rangle))$$

Example 8. Table 2 exhibits the relations on endpoints preserved by each of the four models of s_1 satisfying the adaptation constraint with regard to the model of the source specification. This distance discriminates better the models for

	A^-/B^-	A^-/B^+	A^+/B^-	A^+/B^+	δ
A o B	<	<	>	<	0
A b B	<	<	<	<	1
A m B	<	<	=	<	1
A mi B	>	=	>	>	3
A bi B	>	>	>	>	3

Table 2: Relations preserved by the linearization of an overlap

adapting situation s_1 . Two models (before and meets) are clearly preferred over the others (met by and after). But intuitively, it seems that meet is a better solution because it reduces the distance between the two objects which were previously overlapping. We show that it is possible to find a distance conforming to this intuition.

4.3 Conceptual distance in the interval algebra

The problem with the former distance is that it does not take into account the topological structure of temporal relations (i.e., it only counts differing relations on endpoints or intervals without consideration for a proximity between the disagreeing relations). To take this proximity into account, we take advantage of the notion of conceptual neighbourhood [Freksa, 1996] and the shortest path distance in its graph (see Figure 4).

Conceptual neighbourhood attempts at capturing the proximity between qualitative relations by observing the effects of transforming the related objects.

Definition 13 (Conceptual neighbourhood). The conceptual neighbourhood relation is a binary relation N_Γ^X between elements of a set of relations Γ such that $N_\Gamma^X(r, r')$ iff the continuous transformation X of a situation involving two individuals x and y can transform $r(x, y)$ into $r'(x, y)$ without transiting by a third relation.

The conceptual neighbourhood relation for the transformation that moves one endpoint without affecting the others is given in Figure 4.

A distance between relations can be directly computed from the graph.

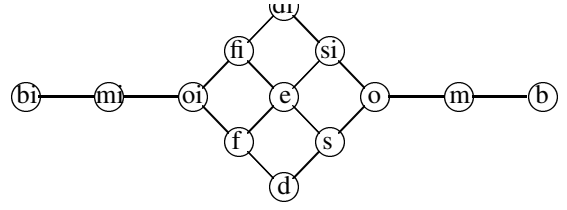


Figure 4: Conceptual neighbourhood graph (N_{13}^A).

Definition 14 (Conceptual distance). The conceptual distance δ' between two relations is the length of the shortest path between r and r' in the graph of N_Γ^X .

Then the distance between models can be expressed by summing up the conceptual distances between the relationships used in both models.

Definition 15 (Conceptual distance between models).

$$d(\lambda, \lambda') = \sum_{n, n' \in N} \delta'(\lambda(\langle n, n' \rangle), \lambda'(\langle n, n' \rangle))$$

Example 9. Concerning s_1 , the models satisfying the adaptation constraint MSO(1) are at different conceptual distances from the source specification: before is at a distance of 2, meet is at a distance of 1, met by is at a distance of 5 and after is at a distance of 6. So the closest solution to serializing the overlap relation is meet. This corresponds to the intuition.

5 Limitations

Extending the presented work to the spatial dimension does not look very difficult. The logical dimension is even easier because it provides a very structured organisation of the document that, we conjecture, can yield direct adaptation. So the proposed approach is able to cope with adaptation in each dimension of the document.

Real difficulties arise when hypermedia and temporal and spatial dimensions are considered together. As a matter of fact, the presence of hypermedia links which, when triggered by the users, jump to other parts of the presentation, introduce non-determinism in the interpretation of documents [Dalal *et al.*, 1996; Fargier *et al.*, 1998]. This non determinism does not easily fit with the conceptual neighbourhood approach which favours continuity.

A further analysis shows that the temporal information contained in specifications is not sufficient for a good adaptation. For instance, considering two panels composed of two pictures each (AB and then CD, like in Figure 1) and the MSO(1) constraint, the closest models linearizing the presentation are ABCD and ABDC with a conceptual distance of 18. However, if both panels aim at comparing two objects O1 (right) and O2 (left) on the basis of two features (one by panel), preserving the parallelism (which suggests the comparison) imposes the choice of ABCD. The absence of information about the comparison is missing from the specification resulting in lower quality adaptation. Some authors [Rutledge *et al.*, 2000] have proposed to use rhetorical structures [Mann and Thompson, 1987] in order to choose a better presentation at the formatting stage. This could be useful for the adaptation stage as well.

6 Related work

The most related work is that of [Dalal *et al.*, 1996], which describes the generation of multimedia presentations through the negotiation of the temporal constraints. Like the work presented here, the temporal specifications are expressed by Allen relations. The approach differs because we consider an existing specification to be adapted where the authors generate schedules and preferences among them on the fly. So there is no alteration of already existing constraints based on the semantic characterisation, but a satisfiability check and negotiation of constraints when inconsistency is detected.

The transgressive adaptation can be compared with the revision in knowledge bases [Gärdenfors, 1992]: the addition of a new (adaptation) constraint leads to inconsistency. It is necessary to find a new specification satisfying this constraint and not too different from the source specification. One difference is that adaptation constraints are not always formulas of the specification language. Having several constraints raises problems similar to incremental revision: since the constraints are not provided in a sequence but in a set, it is important that the adaptation does not depend on some order of presentation constraints. Although transgressive adaptation is neither revision (it does not correspond to some change in our knowledge) nor update (it cannot be compared to the acquisition of a new information), more generic techniques developed for revision could be used in the context of multimedia adaptation.

7 Conclusion

This paper applied a semantic approach to multimedia documents and their adaptation. This allows for a precise definition of what is expected from the adaptation of these documents and the comparison of the results given by hand-made transformation with what was expected. It proposes a model-based distinction between compliant documents, refining adaptation and transgressive adaptation. This framework has been applied to the temporal dimension of the documents providing measures for sharply discriminating the possible transgressive adaptations.

As discussed above, there remains more work to be carried out for covering all the aspects of multimedia documents and for deepening the specification of documents and adaptation constraint so that the adaptation produces quality results.

References

- [Allen, 1983] James Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
- [Dalal *et al.*, 1996] Mukesh Dalal, Steven Feiner, Kathleen McKeown, Shimei Pan, Michelle X. Zhou, Tobias Hollerer, James Shaw, Yong Feng, and Jeanne Fromer. Negotiation for automated generation of temporal multimedia presentations. In *ACM Multimedia conference*, pages 55–64, Boston (MA US), 1996.
- [Dalal, 1988] Mukesh Dalal. Investigations into a theory of knowledge base revision: preliminary report. In *Actes 7th*
- AAAI, Philadelphia (PA US)*, pages 475–479, Philadelphia (PA US), 1988. AAAI.
- [Fargier *et al.*, 1998] Hélène Fargier, Muriel Jourdan, Nabil Layaïda, and Thierry Vidal. Using temporal constraints networks to manage temporal scenario of multimedia documents. In *Proceedings of ECAI 98 Workshop on Spatial and Temporal Reasoning*, Brighton (UK), 1998.
- [Freksa, 1996] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1–2):199–227, 1996.
- [Gärdenfors, 1992] Peter Gärdenfors, editor. *Belief revision*. Cambridge university press, Cambridge (GB), 1992.
- [Layaïda, 1997] Nabil Layaïda. *Madeus: Système d'édition et de présentation de documents structurés multimédia*. Thèse d'informatique, Université Grenoble 1, 1997.
- [Lemlouma and Layaïda, 2001] Tayeb Lemlouma and Nabil Layaïda. The negotiation of multimedia content services in heterogeneous environments. In *Proc. 8th International Conference on Multimedia Modeling (MMM01)*, pages 187–206, Amsterdam (NL), 2001.
- [Mann and Thompson, 1987] William Mann and Sandra Thompson. Rhetorical structure theory: A theory of text organization. Technical Report RS-87-190, International Computer Science Institute, Marina Del Rey, CA, 1987.
- [Monjardet, 1981] Bernard Monjardet. Metrics on partially ordered sets — a survey. *Discrete mathematics*, 35(1):173–184, 1981.
- [Rutledge *et al.*, 2000] Lloyd Rutledge, Jim Davis, Jacco van Ossenbruggen, and Lynda Hardman. Inter-dimensionnal hypermedia communicative devices for rhetorical structure. In *Proc. 7th International Conference on Multimedia Modeling (MMM00)*, pages 89–105, Nagoya (JP), 2000.
- [van Beek, 1992] Peter van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58(1–2):297–326, 1992.
- [Villard, 2001] Lionel Villard. Authoring transformations by direct manipulation for adaptable multimedia presentations. In *Proc. ACM Symposium on Document Engineering (DocEng'01)*, pages 125–134, Atlanta (US), 2001.
- [W3C, 1998] W3C. *SMIL 1.0 Specification*, 1998. <http://www.w3.org/TR/REC-smil/>.

An Ontology-based Architecture for Cooperative Information Agents

Frederico L. G. Freitas

Universidade Católica de Santos
R. Carvalho de Mendonça, 144, Vila Mathias
11.070-906 Santos, SP, Brazil
fred@unisantos.br

Guilherme Bittencourt

Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Cx. Postal 476, 88.040-900, Florianópolis, Brazil
gb@das.ufsc.br

Abstract

In the Web, extractor agents process classes of pages (like ‘call for papers’ pages, researchers’ pages, etc), neglecting the relevant fact that some of them are interrelated forming clusters (e.g., science). We propose here an architecture for cognitive multi-agent systems to retrieve and classify pages from these clusters, based on data extraction. To enable cooperation, two design requirements are crucial: (a) a Web vision coupling a vision for contents (classes and attributes to be extracted) to a functional vision (the role of pages in information presentation); (b) explicit representation of agents’ knowledge and abilities in the form of ontologies, both about the cluster’s domain and agents’ tasks. Employing this Web vision and agents’ cooperation can accelerate the retrieval of useful pages. We got encouraging results with two agents for the page classes of scientific events and articles. A comparison of results to similar systems comes up with two requirements for such systems: functional categorization and a thoroughly detailed ontology of the cluster.

1 Introduction

Although the terms “Web information agents” and “cooperative information gathering” (CIG) are in fashion, there are scarce examples of Web systems endowed with some sort of cooperation or integration among tasks related to text processing, viz classification, retrieval, extraction and summarization. When defining the main aspects of CIG [Oates et al 1994], information extraction was mentioned as a key technology that addresses information acquisition in complex environments, suggesting implicitly task integration for CIG.

Indeed, there is room for cooperation, in particular for extractor agents on the Web. These systems are being designed to process classes of pages with similar structuring and contents (e.g., call for papers, scientific articles, ads, etc). However, the relevant fact that many of these classes are interrelated forming *clusters* (e.g., Science) has been neglected so far. Cooperation among

extractors could increase the retrieval of useful pages, taking advantage of these relations.

Aiming at task integration and cooperation, we designed an architecture for cognitive multi-agent systems that retrieve and classify pages from clusters of information in the Web, extracting slots of data from it. Two design requirements are crucial. One is a Web vision coupling a vision for contents (classes and attributes to be extracted) to a functional vision (the role of pages in information presentation, i.e., whether a page is a list, message, class instance or garbage). The other, necessary to enable cooperation, consists of explicit knowledge representation in the form of ontologies, both about the cluster’s domain and agents’ tasks.

Experiments held with two agents, that treated the classes of scientific events and articles, produced promising results, leading to two conclusions: (a) functional categorization can be profitable; (b) key requirements for success in Web CIG systems are this categorization and a detailed ontology of the cluster.

The article is organized as follows: Section 2 describes the proposed Web vision. Section 3 introduces the architecture, its components and many types of reuse enabled. Section 4 outlines a case study: MASTER-Web (Multi-Agent System for Text Extraction, classification and Retrieval over the Web), applied to the Science cluster, with the two agents cited above. The results on contents and functional classifications are also shown. Section 5 compares MASTER-Web to similar systems. Section 6 brings future work and conclusions.

2 A Proposed Web Vision for CIG

Web extractors seek pages that are instances of pages’ classes (e.g., call for papers pages, researchers’, etc). These pages share many common attributes (slots), like date and place of a conference. Pages’ classes outline a Web division *by contents*. The slots usually found in a class are *discriminant*, once they help distinguish class members. This fact supports the use of extraction in class categorization. Researchers’ pages, for instance, usually contain typical data, such as projects, interest areas, etc.

Most links in classes’ pages point to pages containing data from a few other classes. A set of interrelated

classes and their relations gather a body of knowledge about a specific domain (e.g., science, tourism, etc). They form a *cluster* of classes. In researchers' pages, e.g., we often find links to papers, calls for papers and other classes from the scientific cluster.

Another view of the Web, based on the work of Pirolli et alii [1996], focus on *functionality*, dividing pages by the role played in linkage and information storage. We separate them into five functional groups: *content* pages (class members), resource directories (*lists* of content pages, e.g., lists of researchers or scientific articles), *messages* about content pages, *recommendations* (pages that are members of other classes) and garbage.

Joining these two visions, we can accurately identify not only the information to be extracted from page classes, but also instances of relations among classes in a cluster. Thus, it can improve the search of useful pages.

3 Proposed Architecture for CIG

We propose an architecture for cognitive multi-agent system (MAS) to retrieve and extract data from Web pages belonging to classes of a cluster. The core idea of employing a MAS resides on taking advantage from the relations among classes through agents' cooperation. The architecture overview is illustrated in figure 1.

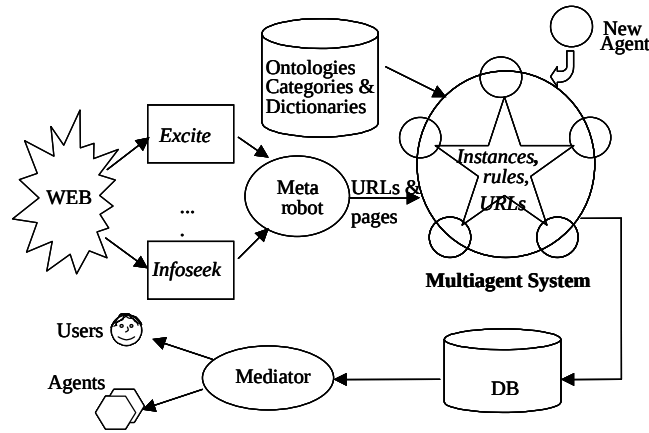


Figure 1: Architecture overview.

Each agent, represented as a circle in the figure, recognizes, filters and classifies pages, extracting attributes (slots) from them. The pages are supposed to belong to the class of pages that the agent processes. For instance, CFP pages are processed by the CFP agent, and papers pages by an articles' agent, in the Science cluster. The MAS is based on Distributed Problem Solving, where agents cooperate without overlapping functions.

Each agent relies on a meta-robot that can be connected to multiple search engines - like AltaVista, Excite, etc. The meta-robot queries the search engines with keywords that assure recall for the agent's page class (e.g., the terms 'call for papers' and 'call for participation' for the CFP agent). Due to the lack of precision, the resultant

URL set from the queries is characterized by a wide variation of functional groups, containing many lists, messages, content pages from its class and from others agents' classes, and garbage. The retrieved URLs feeds a low priority queue of URLs to be processed.

An agent continuously accesses this queue and another one, assigned with high priority, which stores URLs sent as recommendations by other agents of the MAS or taken from pages considered as lists. These links are considered as "hot hints", because they were found under a safer context. Therefore, they are expected to attain higher precision. Cooperation pays off if these suggestions contain less garbage than search engine results do.

An agent processes URLs by extracting data and classifying the pages functionally and by contents. The extracted data and the results are kept in a database together. A mediator facilitates database access to users or agents, from the system or external, offering simpler reduced non-normalized database views.

When an agent joins the system, it announces itself and sends instances and rules to the other agents. This knowledge will be employed by them on the recognition of links or pages likely to belong to the page class processed by the sender agent. So, the agents update their knowledge bases and send to the new agent, in return, their own recognition instances and rules. When a link or page fires any other agent's recognition rule, the agent sends the link or page to that agent as a recommendation.

3.1 Agents' tasks

An agent performs the following steps for each URL. Figure 2 depicts an agent in detail.

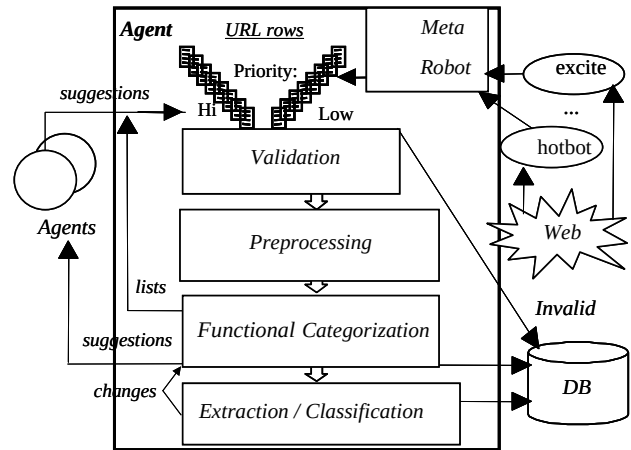


Figure 2: An agent and its processing steps.

Validation. Non-html, non-http, inaccessible and pages already stored in the database are ruled out in this step.

Preprocessing. Representation elements - such as contents with and without html tags, centroid, title, links and e-mails, among other elements - are generated from each valid page applying IR techniques, like stop-lists, stemming and tagging. If necessary, shallow natural

language representations can be used. The representation elements are passed to the agent's inference engine.

Functional classification. An agent deduces to what functional group the page fits to, whether a list, message, garbage or class member dealt by the agent or by another.

It is also here that links in a page are suggested to other agents when any of their identification rules about the anchor or URL fires. For example, an anchor or URL with the word "conference" is useful for a CFP agent. Suggestions may trigger scanning on directory structure prefixes, like /staff/ for a researchers' agent. These directories usually keeps plenty of content pages.

Extraction and contents classification. These two tasks are quite correlated; Riloff [1994] has already demonstrated the applicability of extraction for filtering. In our architecture, extraction assists classification, since each agent manages many subclasses, e.g., the CFP agent treats conferences, workshops and journals. After categorizing a page according to its semantic contents, the remaining slots, which weren't discovered yet, are searched and extracted. In case contradictions or strange facts appear during extraction, the functional class assigned to the page can be modified. For instance, a supposed CFP page is retracted to be a list or garbage, if dates farther than a year are met in the top.

Using as many representation elements as required, each slot of data is extracted by a union of templates, cases and rules. When a slot is part of a predefined list, it is gleaned by matching terms from the dictionaries (e.g. checking the presence of acronyms to find US states).

3.2 Agents' knowledge

Ontologies play a central role in the architecture, serving not only as vocabulary in agent messages, but also defining proper concepts, slots, instances, restrictions and relations. The architecture encompasses five ontologies:

Cluster ontology. The main ontology ought to focus on the original domain of the cluster (e.g. Science), aiming at a rich and comprehensive representation of its classes, even if they don't appear in the cluster. There are some reasons for this. First, reuse possibilities by any application dealing with the domain are enhanced. Moreover, it provides a suitable structure to extract slots. To sum up, a detailed domain ontology assures performance in contents categorization (see section 5.1).

Web ontology. Classes here contain representations of the pages, chosen according to their adequacy in the required tasks. They comprise HTTP protocol data, Information Retrieval (IR) page representations (such as terms and frequencies, contents with and without tags, links, e-mails, etc), and Natural Language Processing (NLP) representations, suitable for unstructured classes.

CIG ontology. Operational knowledge classes (and instances) for the tasks of functional categorization, contents classification and extraction. It includes:

- Templates to extract slots (e.g. deadline of a CFP), to recognize functional categories (such as garbage, messages, recommendations, etc), and to classify content pages (e.g. conference or workshop CFP);
- Auxiliary classes like dictionaries concepts (with keywords and synonyms), agents and abilities, etc;
- Functions to determine regions, extract, convert, format, check consistency and dismiss extracted data;
- Complex and expressive cases specifying conditions, concepts and sets of slots whose presence or absence characterize a content page class. An example below:

```
([Articles-with-author-org-and-place] of Case
  (All-of-Concepts-Keywods TRUE)
  (Absent-Concepts-in-the-Beginning [thesis])
  (Concepts-in-the-Beginning [abstract])
  (Slots-in-the-Beginning [Author-Name]
    [Organization-Name] [Location]))
```

This case detects articles. If, in the beginning of a page, an author name, an organization and a country (or US state) were extracted, terms similar to "abstract" were found, but not terms related to the concept "thesis" (like "partial fulfillment"), a rule bound to this case is fired. The attribute All-of-Concepts-Keywods determines whether all the slots have to appear in the page; if it was false only one would be necessary. Next, a Class-Recognizer instance (displayed below) and a rule creates an instance of the class associated with the case.

```
([Part-Publication] of Class-Recognizer
  (Cases [Articles-with-author-org-and-place])
  (Class [Part-Publication]))
```

The agents first categorize pages into abstract classes (which can't have instances, as, in the example, Part-Publication), to assign later the exact class the pages fit to (e.g. Conference-Article). Cases, recognizers, concepts and rules are also applied on slot extraction and search of links and URLs to be suggested to other agents.

Auxiliary ontologies. Comprise ontologies for NLP (like WordNet [Miller 1995]), time and places (with countries, states, etc), and other specific ontologies from other subject areas, which can be helpful for an agent or domain (like bibliographic data, for the "articles" agent).

Indeed, a knowledge-based approach is adequate to permit not only high-level intentional cooperation, but also massive reuse of code, search engines' services, DB definitions - once agents' tables are the same (pages not recognized, dictionaries, search engines and their queries), except the particular table where an agent stores its extracted data (for instance, CFPs to the CFP agent) - and knowledge. This latter is the most important reuse and can happen in several ways. Ontologies can be downloaded from repositories, like Ontolingua. A cluster ontology serves to all agents in a cluster, as well as most knowledge from dictionaries and rules. Thus, a new agent is quickly built, needing only new rules and instances of concepts, cases and templates, given that knowledge acquisition has already been accomplished.

4 Case Study: the Science domain

A model of agent was developed in Java, and a mediator, in Delphi and HTML. We utilized KQML for agents' communication. Ontologies were written using Protégé¹. Rules were specified in the inference engine Jess². The search engines AltaVista, NorthernLight and Goggle were queried for retrieval. NLP wasn't implemented yet. The data to be extracted was identified to support the classifications, but not extracted

We reused an ontology of science from the project (KA)² [Benjamins et al, 1998], introducing many refinements. A key one was the creation of abstract classes grouping classes with common features, in order to improve classification. For example, class Scientific-Event derived abstract subclasses Live-Sc-Event - with concrete subclasses Conference and Workshop - and Sc-Publication-Event, with subclasses Journal and Magazine

Two agents from the Science cluster were developed:

CFP Agent. Handles any CFP pages. Extraction templates for 21 slots were defined (e.g. Place, Deadline, Editor-in-Chief) from 8 scientific events subclasses (the ones already cited plus Generic-Live-Sc-Event, Generic-Sc-Publication-Event and Special-Issues for Journal and Magazine), classified by 28 cases and templates.

Scientific articles agent. The search engines were queried with terms that appear often in articles: "abstract", "introduction", "related work", "conclusion", "discussion", and others. The agent is responsible for 10 classes: workshop, conference, journal and magazine articles, book chapters and generic articles, as well as thesis, dissertations, technical and project reports. 8 slot templates, 16 cases and 36 templates were created.

Three tests were executed with each agent. The first two treated *corpi* obtained from the search engines' queries; one was applied for knowledge acquisition, to specify the cases, templates and rules, and the other, for blind test. The third test was made directly from the Web.

4.1 Results

Table 1 displays tests' results. In the table, recognition stands for an evaluation if the agent correctly identified member pages. Note that a fourth test was run with CFP agent, in which it benefited from lists of content pages. Lists of CFPs on a focused subject are usually maintained by researchers, departments and organizations.

CFP Agent. More than 70% of content pages were conference CFPs, mostly recognized by the presence, in the top of the pages, of the concept CFP, and the slots initial date and location. The few false content pages were very long and about communities of users (Linux, XML, etc), containing many of the slot triggering concepts. The agent failed to recognize CFPs with little data or not using the jargon ("deadline", etc). Lists were detected

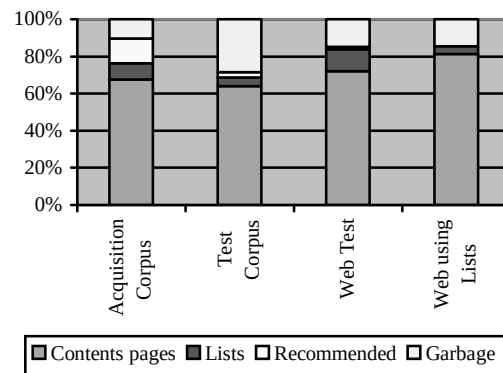
when many anchors contained terms related to events, or when lots of time intervals in a page (e.g., 1-4 December) were present. Special care should be taken to lists, once false positives can queue bad links as "hot hints".

	"Call for Papers" Agent				Articles Agent		
	Acquisition Corpus	Test Corpus	Web Test	Web using Lists	Acquisition Corpus	Test Corpus	Web Test
Recognition	97,1	93,9	96,1	96,3	93,1	82,7	87,8
Functional Categ.	93,8	93,9	93,8	95,7	96,8	94,0	95,1
Contents Classification	94,9	93,3	92,9	91,7	97,0	93,0	81,4
Nr. of Pages Processed	244	147	129	188	190	150	184

Table 1: Agents' performances.

The lists increased substantially the retrieval of content pages. The set of pages retrieved in the fourth test was clearly more focused. Pages from other categories, like lists, messages and recommendation to the fictitious Organizations agent and Dividable-Publication agent (process Proceedings), were substituted by content pages in the form of frames. In the other tests, only one frame appeared. In this test, the number of contents pages rose between 13 and 22% (see Figure 3), due to frames pages.

Figure 3: Functional categories distribution in the CFP agent



Even the garbage patterns changed: instead of false content pages, caused by search engines' lack of precision, garbage were graphical, initial pages of CFPs. It is also remarkable that some scientific events found in lists were not discovered in the other tests. These facts demonstrate that functional categorization, and, in particular, the use of lists, pays off.

Scientific articles agent. Recognition errors happened in articles with few data, data in the end, or with affiliations in unknown companies. An article must have at least three of the slots Organization-Name

¹ <http://protege.stanford.edu>

² <http://herzberg.ca.sandia.gov/Jess>

Author-Name, Location, e-mail and Department. The agent missed articles containing only the authors' names, but this is normal in other similar systems like CiteSeer.

Contents classification was based on information about articles' publication, usually put in the top of papers. More than half of the articles didn't bring this data.

Cooperation. We executed one more test, in which the agents cooperated. The CFP agent asked the articles' agent for anchors in top of papers containing concepts like "conference", "journal", etc. The articles' agent transformed these requests into rules to suggest anchors to the CFP agent. Although it worked and no suggestion was wrong, only three anchors were sent.

To prove the usefulness of cooperation, the CFP agent scanned anchors in Program Committees seeking for suggestions to a future "researchers" agent. No extraction technique (e.g. proper names) or dictionary was present, only simple heuristics. The agent recommended 30 correct and 7 wrong anchors, a promising rate, given that researchers' pages are less structured and, therefore, harder to be retrieved by search engines queries.

5 Related Work and Discussion

Many subject areas are involved with this project: information, retrieval, extraction, classification, multi-agents, information agents, NLP and ontologies, at least. Instead of trying to review these areas, we will focus on comparing and discussing solutions similar to ours.

5.1 WebKB

This system [Craven et al, 1999] integrates extraction and classification as separate tasks, supported by machine learning. Pages from computer science departments' sites in universities are classified against a domain ontology previously defined, with classes and relations. The pages were represented using IR techniques. Classification and extraction rules were learnt from annotated *corpi*.

The decision of relying on machine learning depends upon some issues. The first one is a comparison between the costs of annotating *corpi* against inspecting them for knowledge acquisition, as we did. ML brings gains such as speed and adaptability. However, there are drawbacks, like readability and ontological engagement of the learned rules (which tends to be quite specific), difficulties to include a priori knowledge and to capture some rules without adding lots of features, and even to generalize through a huge number of features or classes.

WebKB's ontology was composed of four basic classes: activities – with subclasses projects and courses – persons – student, faculty and *staff* –, departments and other. Relations among classes were defined, as in our project: project members, advisors, instructors, etc.

Discussion. The authors evaluated contents' classifications only by the false positives, reporting rates between 73 and 83 %, except for the classes "staff member" and "other". Nonetheless, if false negatives were computed, the class "other" reaches 93,6%,

"student" 43% and the remaining 6 classes less than 27%, lowering the average rate to around 50%.

Other drawbacks are the huge incidence in the class "other", almost three times the sum of the other classes together, 65% of them wrong. This class severely affected performance, since 90% of the mistakes of the tests came from it. Anyway, even if the classification was correct, the utility and costs of such system would be questionable, since more than 73% of the pages would have to be categorized as "other".

This deluge of garbage pages seems to stem from two facts: (a) The absence of functional categorization, and (b) the role of ontologies was underestimated in WebKB, since its ontology doesn't seem to be comprehensive enough. On the contrary, MASTER-Web's ontology incorporates classes not useful yet, such as projects and products, because agents to process these classes can be available in the future. Moreover, current agents apply these definitions to manage their own classes.

On the other hand, the learning process may face problems on generalization with an ontology with many classes. Much more annotated pages would probably be needed as well. Therefore, any solution shall rely on cooperation, with learning or inference agents.

5.2 CiteSeer and DEADLINER

These systems perform efficient retrieval, filtering and extraction over the Web, exploiting statistical and learning methods joined with a priori knowledge.

CiteSeer. One of the most accessed software in the search of scientific articles [Bollacker et al, 1999]. They are found monitoring mailing lists, newsgroups and publishers, or querying search engines with keywords "papers", "publications" and "postscript". An article is recognized by the existence of a reference section.

The system extracts bibliographic data from each article and from its reference section, which plays the role of the functional category of list, helping to locate other articles. The number of citations of an article by others represents a measure of its relevance. Databases of authors and journals, as well as complex techniques, are applied to identify co-references for authors and articles.

DEADLINER. This system [Kruger et al 2000] searches the Web for scientific events. It scans newsgroups and broadcast e-mails, and has its own meta-robot. From each CFP, it extracts initial and final date, deadline, program committee, and affiliation of each committee member, themes, events' name and country. Extraction is carried out by a sequence of positional filters integrated using automatic learning. Extraction performance achieves more than 70% for all the slots. For program committee extraction, the system benefits and updates CiteSeer's researchers' database.

Recognition performance is similar to ours (more than 95%), but its definition of event is more restrictive: all the slots are required, except country, plus some submission information. MASTER-Web's CFP agent

offers more recall and flexibility, accepting announcements of book chapters, magazines, contests, workshops and journals. Moreover, the requirements are expressed in cases, which are much more flexible. CFP agent was designed bearing in mind that users are interested in any CFPs for their research works.

Discussion. Although developed by the same research group, CiteSeer and DEADLINER nowadays are not able to cooperate directly. They handle anchors and pages that interest each other, only cooperating in the construction of the researchers' database.

This is a common problem for extractor agents on the Web: the simplicity, adaptability and speed of wrapping, learning and finite-state methods prevent these systems to cooperate in an intentional knowledge-based manner. Thus, they couldn't ask for data of a certain type, employing, as shared vocabulary for communication, ontologies about the domain, their abilities, desired pages, features among other pieces of knowledge.

As the number of Web extractors grows, each with its robot collector, strain on bandwidth can be. Conversely, cooperation could turn out to be even more profitable than search engines results, constituting the only starting way to look for unstructured classes, like researchers' pages. Furthermore, cooperation can take place also at the *level of data*: the CFP agent can warn a researchers' agent that a certain researcher was chair of an event, a fact that could be missing in her homepage.

This stems from the knowledge representation. In statistical and finite-state approaches, knowledge is hidden behind algorithms. It can be neither reasoned nor communicated at knowledge level, blocking cooperation among extractors and integrated extraction. On the other hand, declarative systems can change their behavior dynamically in favor of the cluster's processing, communicating in an elegant manner. Another advantage of these systems resides on the fact that, with an approach like CiteSeer and DEADLINER to process a new class, most of the new system has to be made from scratch. With our architecture, reuse is massive, and only part of the knowledge has to be acquired and specified.

6 Future Work and Conclusions

This project has a long road ahead. New agents for the Science cluster have to be deployed, like a researchers' agent, to make cooperation actually effective. We must extend our tests to other clusters formed by interrelated page classes, such as a tourism cluster - linking events, hotels, and transport pages.

Learning and NLP techniques must be accommodated to deal with less structured page classes. The agents should also check data duplicity, and documents written in other formats (like .ps and XML) must be covered too.

With this work, we tried to fertilize the field of CIG with some new issues and principles, with cooperation as the guiding idea: task integration, declarative inferential extractors capable of intentional cooperation, a Web

vision coupling functional and contents aspects, mapping of semantic relations among page classes and identification of linkage patterns. A key requirement for CIG systems came up with the tests: a detailed ontology is needed to process the richness of a cluster's domain.

On the practical side, we designed a fully reusable cognitive MAS to process data from whole regions of the Web, and got encouraging results from it. We claim that keyword-based search engines can be a basis for more accurate ontology-based domain-restricted cooperative information agents in a near future.

Acknowledgments

The authors thank Tercio Sampaio and Rafael Teske who helped with implementation, and the referees.

References

- [Benjamins et al, 1998] R. Benjamins, D. Fensel and A. G. Pérez. *Knowledge Management through Ontologies*. Proc. of the 2nd International Conf. on Practical Aspects of Knowledge Management, Basel, Switzerland. 1998.
- [Bollacker et al, 1998] K. Bollacker, S. Lawrence and C. L. Giles. *CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications*. Proceedings of the 2nd International ACM Conference on Autonomous Agents, USA. 1998.
- [Craven et al, 1999] M. Craven, A. McCallum, D. DiPasquo, T. Mitchell, D. Freitag, K. Nigam and S. Slattery. *Learning to Extract Symbolic Knowledge from the World Wide Web*. Technical Report CMU-CS-98-122. School of Computer Science. CMU, USA. 1999.
- [Kruger et al, 2000] A. Kruger, C. L. Giles, F. Coetze, E. Glover, G. Flake, S. Lawrence and C. Omlin. *DEADLINER: Building a new niche search engine*. Conference on Information and Knowledge Management, Washington DC, November, 2000.
- [Miller, 1995] George Miller. *WordNet: A Lexical Database for English*. Communications of the ACM. 38(11):39-41. USA. 1995.
- [Oates et al, 1994] T. Oates, M. Prasad and V. Lesser, *Cooperative Information Gathering: A Distributed Problem Solving Approach*. Computer Science Technical Report 94-66-v2. Univ. of Mass, Amherst, USA. 1994.
- [Pirolli et al, 96] P. Pirolli, J. Pitkow and R. Rao, *Silk from a Sow's Ear: Extracting Usable Structures from Web*. http://www.acm.org/sigchi/chi96/proceedings/papers/Pirolli_2/pp2.html. 1996.
- [Riloff, 1994] Ellen Riloff. *Information Extraction as a Basis for Portable Text Classification Systems*. PhD. thesis. Department of Computer Science. University of Massachusetts at Amherst. USA. 1994.

Web Page Cleaning for Web Mining through Feature Weighting

Lan Yi

School of Computing
National University of Singapore
3 Science Drive 2
Singapore 117543
yilan@comp.nus.edu.sg

Bing Liu

Department of Computer Science
University of Illinois at Chicago
851 S. Morgan Street
Chicago, IL 60607
liub@cs.uic.edu

Abstract

Unlike conventional data or text, Web pages typically contain a large amount of information that is not part of the main contents of the pages, e.g., banner ads, navigation bars, and copyright notices. Such irrelevant information (which we call *Web page noise*) in Web pages can seriously harm Web mining, e.g., clustering and classification. In this paper, we propose a novel feature weighting technique to deal with Web page noise to enhance Web mining. This method first builds a *compressed structure tree* to capture the common structure and comparable blocks in a set of Web pages. It then uses an information based measure to evaluate the importance of each node in the compressed structure tree. Based on the tree and its node importance values, our method assigns a weight to each word feature in its content block. The resulting weights are used in Web mining. We evaluated the proposed technique with two Web mining tasks, Web page clustering and Web page classification. Experimental results show that our weighting method is able to dramatically improve the mining results.

1 Introduction

The rapid expansion of the Internet has made Web a popular place for disseminating and collecting information. However, useful information on the Web is often accompanied by a large amount of noise such as banner ads, navigation bars, copyright notices, etc. Although such information items are functionally useful for human viewers and necessary for Web site owners, they can seriously harm automated information collection and mining on the Web, e.g., Web page clustering, Web page classification, and information retrieval. Web noise can be grouped into two categories based on their granularities:

Global noise: It refers to redundant objects with large granularities, which are no smaller than individual pages. Global noise includes mirror sites, duplicated Web pages and old versioned Web pages to be deleted, etc.

Local (intra-page) noise: It refers to irrelevant items within a Web page. Local noise is usually incoherent

with the main content of the page. Such noise includes banner ads, navigational guides, decoration pictures, etc. This work focuses on dealing with local noise in Web pages. We call it *Web page cleaning*. The work is challenging because it is a non-trivial task to decide which part of a Web page is meaningful and which is noisy. Despite its importance, relatively little research has been done on Web page cleaning so far (see Section 2). In this paper, we propose an effective feature weighting technique to clean Web pages with the purpose of improving Web mining. Our method does not decide and physically remove those noisy blocks of a page. Instead, it assigns low weights to the features in those possibly noisy blocks. The advantage of our method is that there is no need to use a threshold to determine whether a block is noisy or not. Setting a suitable threshold is very hard in practice.

Our cleaning technique is based on the following observation. In a typical commercial Web site, Web pages tend to follow some fixed layouts and presentation styles as most pages are automatically generated. Those parts of a page that also appear in many other pages in the site are more likely to be the noise, and those parts of the page that are quite different from other pages are usually the main content of the page. To capture the common structure and comparable blocks among Web pages, we introduce the *compressed structure tree* to compress or to merge a set of Web pages from a Web site. We then propose a measure that is based on information theory [Shannon, 1948] to evaluate each element node in the compressed structure tree to determine the importance of the node. Based on the tree and node importance values, our method assigns a weight to each word feature in its content block. These feature weights are then directly used in Web mining.

Our experimental results based on two popular Web mining tasks, i.e., Web page clustering and classification, demonstrate that the proposed cleaning technique is able to boost the mining accuracy significantly. For example, in classification, the average classification accuracy over all our datasets increases from 0.785 before cleaning to 0.951 after cleaning.

2 Related Work

Although Web page cleaning is an important problem,

relatively little work has been done to deal with it. In [Lin and Ho, 2002], a method is proposed to detect informative blocks in Web pages. Their concept of informative blocks is similar to our concept of main contents of a page. However, the work in [Lin and Ho, 2002] is limited by the following two assumptions: (1) the system knows *a priori* how a Web page can be partitioned into coherent content blocks; and (2) the system knows *a priori* which blocks are the same blocks in different Web pages. As we will see, partitioning a Web page and identifying corresponding blocks in different pages are actually two critical problems in Web page cleaning. Our system is able to perform these tasks automatically. Besides, their work views a Web page as a flat collection of blocks, and each block is viewed as a collection of words. These assumptions are often true in news Web pages, which is the domain of their applications. In general, these assumptions are too strong.

In [Bar-Yossef and Rajagopalan, 2002], Web page cleaning is defined as a frequent template detection problem. They propose a frequency based algorithm to detect templates or patterns. Their work is not concerned with the context of a Web site, which can give useful clues for page cleaning. Moreover, in their work, the partitioning of a Web page is pre-fixed, which is not suitable for a Web site because a Web site typically has common layouts and presentation patterns. We can exploit these common patterns to partition Web pages and to clean them.

Other related work includes data cleaning for data mining and data warehousing (e.g., [Lee, *et al.*, 2000]). However, they are only concerned with structured data. Web page data are semi-structured, and thus require different cleaning techniques.

Web page cleaning is also related to feature selection in traditional text learning (see [Yang and Pedersen, 1997] for a survey of feature selection techniques). In feature selection, features are individual words or attributes. However, items in Web pages have some structures, which are reflected by their nested HTML tags. Hence, different methods are needed in the context of the Web.

[Kushmerick, 1999] proposes some learning mechanisms to recognize banner ads, redundant and irrelevant links of Web pages. However, these techniques are not automatic. They require a large set of manually labeled training data and also domain knowledge to generate classification rules.

[Kao *et al.*, 2002] enhances the HITS algorithm of [Kleinberg, 1998] by using the entropy of anchor text to evaluate the importance of links. It focuses on improving HITS in order to find informative structures in Web sites. Although it segments Web pages into content blocks to avoid unnecessary authority and hub propagations, it does not detect or eliminate noisy contents in Web pages.

Our work is also related to segmentation of text documents, which has been studied extensively in information retrieval. Existing techniques roughly fall into two categories: lexical cohesion methods [Beeferman *et al.*, 1997; Eichman *et al.*, 1999; Kaufmann, 1999; Reynar, 1999] and multi-source methods [Allan *et al.*, 1998; Beeferman *et al.*,

1999]. The former identifies coherent blocks of text with similar vocabulary. The latter combines lexical cohesion with other indicators of topic shift, such as relative performance of two statistical language models and cue words. In [Hearst and Plaunt, 1993], Hearst and Plaunt discussed the merits of imposing structure on full-length text documents and reported good results of using local structures for information retrieval. Instead of using pure text, which is unstructured, we process semi-structured data. We can make use of the semi-structures present in Web pages to help our cleaning task.

Finally, our feature weighting method is different from feature weighting methods used in information retrieval. One of the popular methods used in text information retrieval for feature weighting is the TFIDF scheme [Salton and McGill, 1983; Baeza-Yates and Bibeiro-Neto, 1999]. This scheme is based on individual word (feature) counts within a page and among all the pages. It is, however, not suitable for Web pages because it does not consider Web page structures in determining the importance of each content block and consequently the importance of each word feature in the block. For example, a word in a navigation bar is usually noisy, while the same word occurring in the main part of the page can be very important.

3 The Proposed Technique

The proposed cleaning technique is based on analysis of both layouts (structures) and contents of the Web pages in a given Web site. Thus, our first task is to find a suitable data structure to capture and to represent common layouts or presentation styles in a set of pages of the Web site. We propose the *compressed structure tree (CST)* for this purpose. Below, we start by giving an overview of the DOM (Document Object Model)¹ tree and showing that it is insufficient for our task. We then present the compressed structure tree, which is followed by our entropy measures for evaluating the nodes in the compressed structure tree for noise detection.

3.1 DOM tree

Each HTML page corresponds to a DOM tree where tags are internal nodes and the actual text, images or hyperlinks are the leaf nodes. Figure 1 shows a segment of HTML codes and its corresponding DOM tree. In the DOM tree, each solid rectangle is a *tag node*. The shaded box is the actual content of the node, e.g., for the tag IMG, the actual content is “src=image.gif”. The order of child tag nodes is from left to right. Notice that our study of HTML Web pages begins from the BODY tag nodes since all the viewable parts are within the scope of BODY. Each tag node is also attached with the display properties of the tag. Display properties defined by CSS (Cascading Style Sheet)² are treated as normal display properties if CSS is available in the page source (i.e., if CSS is not external). For convenience of analysis, we add a virtual root node

¹ <http://www.w3.org/DOM/>

² <http://www.w3.org/Style/CSS/>

without any attribute as the parent tag node of BODY tag node in each DOM tree.

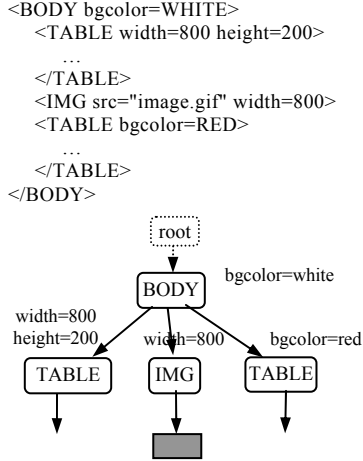


Figure 1: A DOM tree example (lower level tags are omitted)

Although a DOM tree is sufficient for representing the structure of one HTML page and it has been used in many applications (e.g., [Chakrabarti *et al.*, 2001]), it cannot represent the common structure of a set of Web pages. We aim to compress individual DOM trees of Web pages into a single tree (*compressed structure tree*) which captures the commonalities of the pages in a Web site. Based on this tree, we can easily study some statistical properties of the structures and the contents of the Web pages.

3.2 Compressed structure tree (CST)

Before defining the compressed structure tree, we first define the presentation style of a tag node in a DOM tree.

Definition (presentation style): The *presentation style* of a tag node T in a DOM tree, denoted by S_T , is a sequence $\langle r_1, r_2, \dots, r_n \rangle$, where r_i is a pair (*Tag*, *Attr*) specifying the i th child tag node of T .

- *Tag* is the tag name, e.g., “TABLE” and “IMG”;
- *Attr* is the set of display attributes of *Tag*, e.g., bgcolor = RED, width = 100, etc.

n is the length of the style. For example, in Figure 1, the presentation style of tag node BODY is $\langle (TABLE, \{width=800, height=200\}), (IMG, \{width=800\}), (TABLE, \{bgcolor=red\}) \rangle$.

We say two presentation styles $S_a: \langle r_{a1}, r_{a2}, \dots, r_{am} \rangle$ and $S_b: \langle r_{b1}, r_{b2}, \dots, r_{bn} \rangle$ are equal, i.e., $S_a = S_b$, iff $m = n$ and $r_{ai}.Tag = r_{bi}.Tag$ and $r_{ai}.Attr = r_{bi}.Attr$, $i = 1, 2, \dots, m$.

An element node (the basic information unit) of a compressed structure tree (CST) is defined as follows:

Definition (element node): An *element node* E represents a set of merged tag nodes in the DOM tree. It has 5 components, denoted by (*Tag*, *Attr*, *TAGs*, *STYLEs*, *CHILDS*), where

- *Tag* is the tag name;
- *Attr* is the set of display attributes of *Tag*.
- *TAGs* is the set of actual tag nodes in the original DOM trees that are compressed (or merged) in E .

- *STYLEs* is the set of presentation styles merged into E .
- *CHILDS* is the set of pointers pointing to the child element nodes of E in CST.

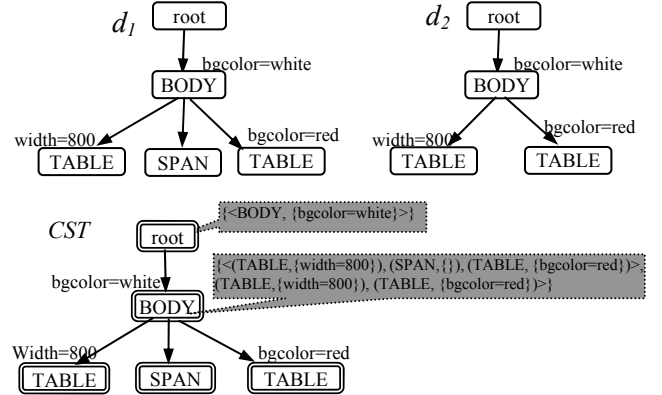


Figure 2: DOM trees and the compressed structure tree (lower levels of trees are omitted)

An example of compressed structure tree is given in Figure 2, which compresses the DOM trees d_1 and d_2 . We observe that, except for the tag node SPAN, all the tag nodes in d_1 are merged with corresponding tag nodes in d_2 . So, an element node in CST actually denotes a set of merged tag nodes, which represents a set of logically comparable blocks in different DOM trees.

Tag nodes in different DOM trees cannot be merged randomly. We need to ensure that the merged tag nodes are the same logical blocks in different Web pages. We build a CST of a set of Web pages from a Web site by merging their DOM trees from top to bottom in following steps:

1. All root tag nodes of the DOM trees are merged to form the first (the top most) element node of the CST. We have $Tag = root$, $Attr = \{\}$, and $TAGs$ being the set of all the root tag nodes in the DOM trees of the Web pages.
2. We compute *STYLEs* of the element node E passed from the previous step. $E.STYLEs$ is the set of presentation styles of all the tag nodes in the DOM trees covered by E . Note that common presentation styles are combined.
3. All the corresponding child tag nodes of those (tag nodes) in $E.TAGs$ with the same presentation style are merged, which form the initial child element nodes of E . However, we want to further merge these initial child element nodes. For any pair of initial child element nodes E_1 and E_2 , if their respective *Tags* and *Attrs* are the same, we compare their textual contents to see whether they are similar and can be merged. Let the characteristic feature (or word) set of $E_j.TAGs$ be $I_j = \{feature_k | freq(feature_k) \geq \gamma, feature_k \text{ occurs in textual contents of } E_j.TAGs\}$, where $j = 1, 2$. $freq(feature_k)$ is the document frequency of $feature_k$ within $E_j.TAGs$ and γ is a predefined constant between 0 and 1. If $|I_j| > 0$ ($j = 1, 2$) and $|I_1 \cap I_2| / |I_1 \cup I_2| \geq \lambda$, then E_1 and E_2 are merged to form a new element node (E_1 and E_2 are deleted). The new element node is inserted into the set of initial

child element nodes. The merging step ends when no pair of child element nodes can be merged. In our experiments, we set $\gamma = 0.85$ and $\lambda = 0.85$, which perform very well.

4. If no child element node is created in step 3, stop; else for each child element node from step 3, goto step 2.

After the CST is built, it is used to compute a weight for each word feature in each block of a Web page.

3.3 Weighting policy

Intuitively, if an element node contains many different presentation styles, it is more likely to be important and hence should be assigned a high weight. Otherwise, it will be assigned a low weight, i.e., it is more likely to be noisy.

We use the entropy of presentation styles to encode the importance of an element node E in the CST.

Definition: For an internal element node E in CST, let $l = |E.STYLES|$ and $m = |E.TAGS|$. The *importance* of E , denoted by $NodeImp(E)$, is

$$NodeImp(E) = \begin{cases} 1 & \text{if } m = 1 \\ -\sum_{i=1}^l p_i \log_m p_i & \text{if } m > 1 \end{cases} \quad (1)$$

where p_i is the probability that a tag node in $E.TAGS$ uses the i th presentation style.

For example, in CST of Figure 2, by equation 1, we obtain $NodeImp(root) = -1\log_2 1 = 0$ since the root element node has only one presentation style. For the BODY element node, $NodeImp(BODY) = -(0.5\log_2 0.5 + 0.5\log_2 0.5) = 1$.

A leaf node is treated differently from an internal node since it contains the actual words or features without any tag. We define its importance to be the averaged importance of the actual word features in it.

Definition: For a leaf element node E in CST, let N be the number of features present in E , the *importance* of E is:

$$NodeImp(E) = \frac{\sum_{i=1}^N (1 - H_E(a_i))}{N} = 1 - \frac{\sum_{i=1}^N H_E(a_i)}{N} \quad (2)$$

where a_i is a feature of the content in E and $H_E(a_i)$ is the information entropy of a_i within E , which is

$$H_E(a_i) = \begin{cases} 0 & \text{if } m = 1 \\ -\sum_{j=1}^m p_{ij} \log_m p_{ij} & \text{if } m > 1 \end{cases} \quad (3)$$

where $m = |E.TAGS|$, and p_{ij} is the probability that a_i appears in the j th tag node of $E.TAGS$.

$NodeImp(E)$ only evaluates local importance of E . In order to weigh a feature contained in a leaf node of a CST, we use the cumulative importance of the path from root to the node containing the feature. We call this cumulative importance of E the *path importance*, denoted by $PathImp(E)$. $PathImp(E)$ measures the importance of the structure from root to E in CST.

Since the path importance is a cumulative importance value, it should meet the following requirement:

- For any two element nodes E_1 and E_2 in CST, if E_1 is an

ancestor of E_2 , then $1 \geq PathImp(E_2) \geq PathImp(E_1) \geq 0$. We define the path importance of an element node E in a CST as follows.

Definition: For an element node E in a CST, the *path importance* of E , denoted by $PathImp(E)$, is:

$$PathImp(E) = 1 - \prod_{E_i \in Ancestor(E) \cup \{E\}} (1 - NodeImp(E_i)) \quad (4)$$

where E_i is an ancestor of E or E itself in the CST.

Our weighting policy considers both the structure and content context of the features. The weight of each feature in a particular block (or under a particular tag) of the Web page is computed as follow.

Definition: For a feature a_i in a leaf element node E of a CST, the weight of a_i under the tag node T_j of the DOM tree of a Web page (i.e., $T_j \in E.TAGS$), denoted by $W_E(a_i, T_j)$, is defined by

$$W_E(a_i, T_j) = PathImp(E) \times (1 - H_E(a_i)) \times f_{ij} \quad (5)$$

where f_{ij} is the frequency of a_i under tag T_j of the page.

In our implementation, we do not use the actual leaf tag nodes in a page as they overly fragment the page. Instead, we use the grandparent tag nodes of the actual leaf tag nodes in the DOM tree as the (virtual) leaf tag nodes in building CST, and in computing feature weights.

After a weight to each feature in every block of a Web page is assigned, we add the weights of the same feature in the page. All the feature weights of the page together form a feature vector of the page, which is used as input to Web mining, e.g., clustering and classification.

4 Empirical Evaluation

This section evaluates the proposed cleaning technique. Since the purpose of our data cleaning is to improve Web mining, we performed two mining tasks, i.e., clustering and classification, to test our system. By comparing the mining results before cleaning and after cleaning, we show that our Web cleaning technique is able to boost mining accuracy dramatically.

Below, we first describe the datasets used in our experiments and the evaluation measures. We then present our experimental results of clustering and classification.

4.1 Datasets and evaluation measures

Our empirical evaluation is done using Web pages from 5 commercial Web sites, Amazon³, CNet⁴, PCMag⁵, J&R⁶ and ZDnet⁷. These sites contain many introduction or overview pages of different kinds of products. In order to guide users and/or to show advertisements, these Web pages all contain a large amount of noise. We will see that the noise mislead mining algorithms to produce poor re-

³ <http://www.amazon.com/>

⁴ <http://www.cnet.com/>

⁵ <http://www.pcmag.com/>

⁶ <http://www.jandr.com/>

⁷ <http://www.zdnet.com/>

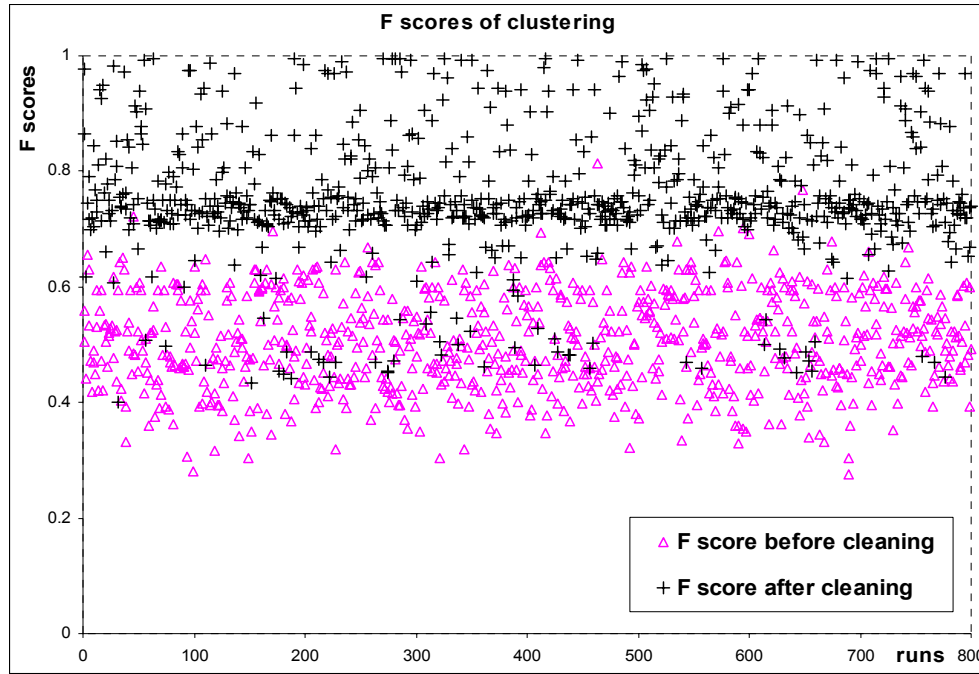


Figure 3: The distribution of F scores of clustering

sults (in both clustering and classification).

All the five Web sites contain Web pages of many categories or classes of products. We choose the Web pages that focus on the following categories of products: Digital Camera, Notebook, TV, Printer and Mobile Phone. Table 1 lists the number of documents that we downloaded

Web sites	Ama-	CNet	J&R	PCMag	ZDnet
Camera	402	219	80	137	151
Notebook	434	480	51	144	143
Mobile	45	109	9	43	97
Printer	767	500	104	107	80
TV	719	449	199	0	0

Table 1: Web pages and their classes from the 5 sites from each Web site, and their corresponding classes.

Since we test our system using clustering and classification, we use the popular F score measure in information retrieval to evaluate the results before and after cleaning. F score is defined as follows:

$$F = 2p*r/(p+r),$$

where p is the precision and r is the recall. F score measures the performance of a system on a particular class, and it reflects the average effect of both precision and recall. We will also include the accuracy results for classification.

4.2 Experimental results

We now report our experimental results before and after cleaning. Cleaning is done on all the Web pages from each Web site separately using the proposed technique.

Clustering

For clustering, we use the popular k-means algorithm

[Anderberg, 1973]. We put all the 5 categories of Web pages into a big set, and use the k -means clustering algorithm to cluster them into 5 clusters. Since the k -means algorithm selects the initial seeds randomly, we performed a large number of experiments (800) to show the behaviors of k -means before and after page cleaning. The F score for each of the 800 runs is drawn in Figure 3, where X-axis shows the experiment number and Y-axis shows the F score. F score for each run is the average value of the 5 classes, which is computed as follow: By comparing the pages' original classes and the clustering results, we find the optimal assignment of class to each cluster that gives the best average F score for the 5 classes.

From Figure 3, we can clearly observe that after cleaning the results are drastically better. Over the 800 runs, the average F score for the noise case is 0.506, while the average F score for the clean case is 0.756, which is a remarkable improvement.

After cleaning, 84.38% of the 800 results have F scores higher than 0.7, and only 4.63% lower than 0.5. Before cleaning, only 0.5% of the 800 results (4 out of 800) have F scores higher than 0.7, and 47.63% lower than 0.5. Thus, we can safely conclude that our feature weighting technique is highly effective for clustering.

Classification

For classification, we use Support Vector Machine (SVM), which has been shown to perform very well for text classification by many researchers (e.g., [Joachims 1997]). We used the SVMlight package [Joachims, 1999] for all our experiments.

In each experiment, we build classifiers based on two classes of pages from different Web sites. For example, the two classes can be camera pages from Amazon and

C ₁	C ₂	F(N)	F(C)	A(N)	A(C)
notebook	camera	0.751	0.946	0.804	0.949
notebook	mobile	0.517	0.817	0.790	0.927
notebook	printer	0.733	0.933	0.763	0.944
notebook	TV	0.746	0.909	0.766	0.918
camera	mobile	0.841	0.962	0.784	0.941
camera	printer	0.715	0.967	0.779	0.975
camera	TV	0.759	0.982	0.808	0.983
mobile	printer	0.529	0.848	0.810	0.956
mobile	TV	0.649	0.851	0.833	0.942
printer	TV	0.712	0.972	0.716	0.971
Average		0.695	0.919	0.785	0.951

Table 2: F scores and accuracies for classification: before cleaning (N) and after cleaning (C)

notebook pages from CNet. Then we use the classifier to classify the combined set of camera and notebook pages from the other Web sites. We have experimented with all possible two class combinations of the 5 sites. Table 2 gives the averaged F scores and accuracies before and after cleaning. In the table, F(N) and A(N) stand for F score and accuracy before cleaning respectively, while F(C) and A(C) stand for F score and accuracy after cleaning. From the results, we can see that after cleaning, the classifiers perform much better. On average over all the experiments, both the F score and accuracy improve by a large margin.

5 Conclusions

In this paper, we proposed an effective technique to clean Web pages for Web mining. Observing that Web pages in a Web site usually share some common layouts and presentation styles, we propose a new tree structure, called compressed structure tree to concisely capture the commonalities of a Web site. This tree provides us with rich information for analyzing both the structure and the contents of the Web pages. An information based measure is also proposed to evaluate each tree node to determine its importance, which is then used to assign weights to features in their corresponding Web pages. Using the resulting feature weights, we can perform Web mining tasks. Experimental results with two popular Web mining tasks show that the proposed technique is highly effective.

References

- [Allan *et al.*, 1998] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study: final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998.
- [Anderberg, 1973] Michael R. Anderberg. *Cluster analysis for applications*, Academic Press, New York, 1973.
- [Baeza-Yates and Bibeiro-Neto, 1999] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
- [Bar-Yossef and Rajagopalan, 2002] Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications, *WWW-2002*, 2002.
- [Beeferman *et al.*, 1997] Doug Beeferman, Adam Berger, and John Lafferty. A model of lexical attraction and repulsion. *ACL-1997*, 1997.
- [Beeferman *et al.*, 1999] Doug Beeferman, Adam Berger and John Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177-210, 1999.
- [Chakrabarti *et al.*, 2001] Soumen Chakrabarti, Mukul M. Joshi and Vivek B. Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. *SIGIR-2001*.
- [Eichman *et al.*, 1999] David Eichmann, Miguel Ruiz, Padmini Srinivasan, Nick Street, Chris Cul and Filippo Menczer. A cluster-based approach to tracking, detection and segmentation of broadcast news. In *Proceedings of the DARPA Broadcast News Workshop*, 1999.
- [Hearst and Plaunt, 1993] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. *SIGIR-93*, 1993.
- [Joachims 1997] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. *ECML-1997*, 1997.
- [Joachims, 1999] Thorsten Joachims. Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.
- [Kao *et al.*, 2002] Hung-Yu Kao, Ming-Syan Chen Shian-Hua Lin, and Jan-Ming Ho, Entropy-Based Link Analysis for Mining Web Informative Structures. *CIKM-2002*, 2002.
- [Kaufmann, 1999] Stefan Kaufmann. Cohesion and collocation: Using context vectors in text segmentation. *ACL-1999*, 1999.
- [Kleinberg, 1998] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.
- [Kushmerick, 1999] Nicholas Kushmerick. Learning to remove Internet advertisements. *Agnets-1999*, 1999.
- [Lee, *et al.*, 2000] Mong Li Lee, Tok Wang Ling, Wai Lup Low. Intelliclean: A knowledge-based intelligent data cleaner. *SIGKDD-2000*, 2000.
- [Lin and Ho, 2002] Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from Web documents. *SIGKDD-2002*, 2002.
- [Reynar, 1999] Jeffrey C. Reynar. Statistical Models for Topic Segmentation. *ACL-99*, 1999
- [Salton and McGill, 1983] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
- [Shannon, 1948] Shannon, C. A Mathematical Theory of Communication. *Bell System Technical Journal*, Vol 27, pp.379-423 and 623-656, July and October, 1948.
- [Yang and Pedersen, 1997] Yiming Yang, Jan O. Pedersen. A comparative study of feature selection in text categorization. *ICML-97*, 1997.

ART AND CREATIVITY

A Learning-Based Jam Session System that Imitates a Player's Personality Model

Masatoshi Hamanaka^{1), 2)}, Masataka Goto^{3), 2)}, Hideki Asoh²⁾, and Nobuyuki Otsu^{2), 4)}

1) Research Fellow of the Japan Society for the Promotion of Science,

2) National Institute of Advanced Industrial Science and Technology (AIST),

3) "Information and Human Activity," PRESTO, JST, 4) University of Tokyo

Mbox 0604, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan

m.hamanaka@aist.go.jp

Abstract

This paper describes a jam session system that enables a human player to interplay with virtual players which can imitate the player personality models of various human players. Previous systems have parameters that allow some alteration in the way virtual players react, but these systems cannot imitate human personalities. Our system can obtain three kinds of player personality models from a MIDI recording of a session in which that player participated – a reaction model, a phrase model, and a groove model. The reaction model is the characteristic way that a player reacts to other players, and it can be statistically learned from the relationship between the MIDI data of music the player listens to and the MIDI data of music improvised by that player. The phrase model is a set of player's characteristic phrases; it can be acquired through musical segmentation of a MIDI session recording by using Voronoi diagrams on a piano-roll. The groove model is a model that generates onset time deviation; it can be acquired by using a hidden Markov model. Experimental results show that the personality models of any player participating in a guitar trio session can be derived from a MIDI recording of that session.

1 Introduction

Our goal is to create a jam session system in which virtual players react as if they were actual human players. We want to make it possible for a human player to interact, whenever they like, with a virtual player that can imitate whoever the human player wishes to perform, for example, with a familiar, professional, or deceased player, or even with themselves. What is most important in imitating players is to acquire the *player's personality models* of a target human player.

Previous session systems have not been able to imitate a human player's personality. Some systems [Aono *et al.*, 1995] have been designed to follow the performance of a human soloist, but without considering the individual character of the virtual player. Although JASPER [Wake

et al., 1994] has a set of rules that determine the system's reactions and VirJa Session [Goto *et al.*, 1996] has parameters for altering how it reacts, these systems cannot develop player personality models of an actual human player.

To realistically imitate a human player, a system must be able to acquire player personality models of that player. The imitating virtual player can then improvise according to the models. The Neuro-Musician [Nishijima and Kijima, 1989; Nishijima and Watanabe, 1992] can learn the relationship between 30 sets of an 8-bar-length input pattern and an output pattern by using neural networks. However, it is only capable of dealing with the limited style of a jam session where a solo part must be changed in 8-bar rotation. In other words, a virtual player and a human player cannot both play a solo part in the same time. Moreover, the Neuro-Musician must prepare a training set of 8-bar-length input-output data to enable neural network learning. In an actual jam session, a player does not always play an 8-bar solo to the 8-bar solo of the other players. Therefore, we cannot acquire the player models from a MIDI session recording by using the Neuro-Musician method.

On the other hand, the Band-OUT-of-a-Box (BoB), which deals with a problem similar to ours [Thom, 2001a; Thom, 2001b], indicates that machine learning techniques provide a useful approach to acquire a player's models. However, BoB can only react to a human performance of an immediately previous four bars. It has a fixed relationship in which the human player is the leader and the virtual player is the follower.

Our jam system allows us to acquire player personality models of a target human player from the MIDI¹ recording of a session in which that player participated. The main advantage of our approach is that we do not have to directly evaluate the target player: all we need to build the models is session recording data.

2 A Guitar Trio Session System

Our system deals with constant-tempo 12-bar blues performed by a guitar trio. Figure 1 shows a jam session

¹ MIDI stands for Musical Instrument Digital Interface.



Figure 1: Session model in which either human or computer can be selected as each player.

model in which either a human or the computer can be selected to perform the part of each player. We can imagine sessions in which all players will be human players just as we can imagine sessions in which all players are computer players. The three players take the solo part one after another without a fixed leader-follower relationship.

We obtain three kinds of player personality model from a MIDI session recording – a *reaction model*, a *phrase model*, and a *groove model*.

The system has two modes, a *learning mode* and a *session mode*. In the learning mode (discussed in Sections 3, 4, and 5), the system acquires player personality models in non-real time. These models are stored in a database and different personality models can be assigned to the two virtual players before session play (Figure 2). In the session mode (discussed in Section 6), a human player can interact with the virtual players in real time.

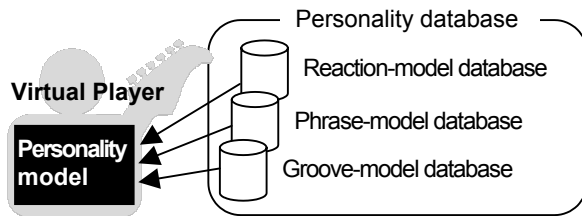


Figure 2: Virtual player and personality databases

3 Learning a reaction model

A reaction model is the characteristic way that a player reacts to other players. Acquiring an actual player's individual reaction model is necessary to create a virtual player that reacts as that actual human player does. As shown in Figure 3, each virtual player listens to the performances of all the players (including its own) and uses the reaction model to determine what its next reaction (output performance) will be. The main issue in deriving

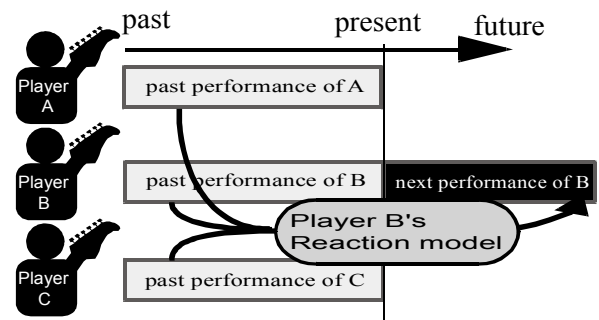


Figure 3: Player's reaction model

the reaction model is to learn the relationship between the input and the output of the target player in MIDI recordings. This can be formulated as a problem of obtaining a mapping from the input to the target player's output. However the direct MIDI-level learning of this mapping is too difficult because the same MIDI-level sequence rarely occurs more than once and the mapping itself is too sparse. We therefore have introduced two intermediate subjective spaces: an *impression space* and an *intention space* (Figure 4).

3.1 Impression space

The impression space represents the subjective impression derived from the MIDI input. By applying principal components analysis (PCA) to the results of subjective evaluations of various MIDI performances, we determined three coordinate axes for the impression space. PCA is a statistical method for reducing the number of dimensions while capturing the major variances within a large data set. While listening to a performance, a subject subjectively evaluated it by using ten impression words to rank the performance's impression on a scale of one to seven. The three selected axes of the impression space represent qualities that can be described as *appealing*, *energetic*, and *heavy*. To obtain a vector in this space, an *impression vector* corresponding to the MIDI input, we use canonical correlation analysis (CCA). This analysis maximizes the correlation between various low-level features of the MIDI input (such as pitch, note counts, tensions, and pitch bend) and the corresponding subjective evaluation. Since an impression vector is obtained from an individual player's performance, we have at every moment three impression vectors (Figure 4).

The impression space is necessary for learning the relationship between various input performances and the corresponding output performances. If we represent the input performances as short MIDI segments without using the impression space, the same MIDI segments will not be repeated in different sessions. The impression space enables the abstracting of subjective impressions from input MIDI data and those impressions can be repeated. Even if two segments of the input MIDI data differ, they can be represented as a similar vector in the impression space as long as they give the same impression.

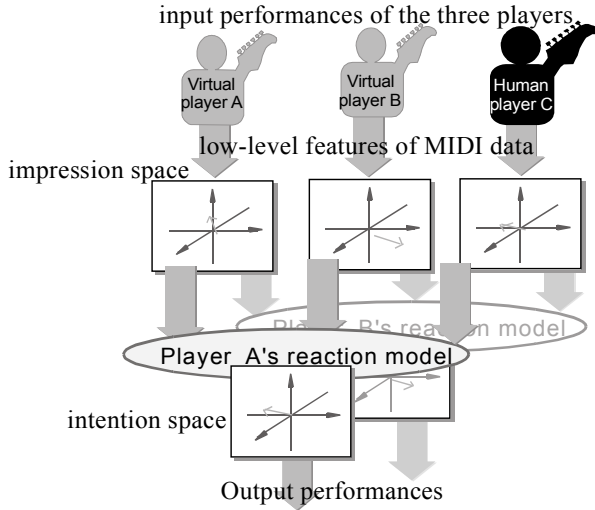


Figure 4: Player architecture.

Figure 5 shows the transition of the rated values for the impression word “appealing.” The black line represents the value calculated by the system and the gray line represents the value as evaluated by a human listener. For 92 percent of the performance, the calculated and subjectively evaluated values do not differ by more than 1.

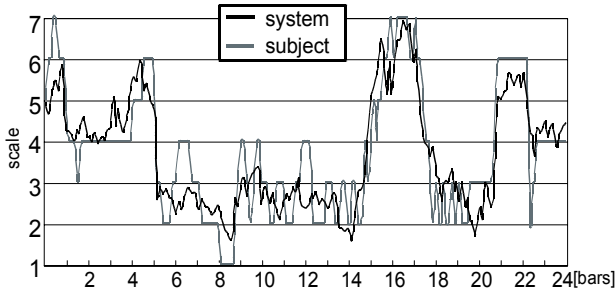


Figure 5: Transition of rated values calculated by system and evaluated by subject (for impression word “appealing”).

3.2 Intention space

The intention space represents the intention of the player improvising the output. A vector in this space, an *intention vector*, determines the feeling of the next output. It is used to select short MIDI phrases from a phrase database, and connecting the selected phrases generates the output MIDI performance.

Without the intention space, learning the relationship between impression vectors and output MIDI data is difficult because in actual MIDI recordings various outputs can occur when the input data gives the same impression. The intention space makes it easier to learn the player’s reaction model.

The intention space is constructed by using multidimensional scaling (MDS) [Kruskal and Wish, 1978] such that intention vectors are distributed with proximities proportional to subjective similarities of short phrases corresponding to those vectors. Based on MDS results, we determined the three dimensions of this

we determined the three dimensions of this space. Because the number of the short phrases is limited, those phrases are sparsely placed in the intention space. When generating the output, the system selects the output phrase close to the determined intention vector: an appropriate phrase can be selected even if the phrase database does not have a phrase that is exactly placed on the intention vector.

3.3 Acquiring a reaction model

We can regard the mapping from the impression space to the intention space as the reaction model. To derive this mapping function statistically, we obtain various training sets from the target session recordings. These sets are pairs of impression vectors obtained from the three players during a sequence of the past twelve bars and the corresponding next intention vector. For this learning we use Gaussian radial basis function (RBF) networks [Chen *et al.*, 1991]. The RBF networks have one hidden layer with nonlinear inputs, and each node in the hidden layer computes the distance between the input vector and the center of the corresponding radial basis function.

The input of the RBF network is impression vector $\mathbf{x}$ (L dimensions) derived from three players and the output of the RBF network is intention vector $\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \tilde{\mathbf{y}}_3$ (3 dimensions). Training data of the RBF network is Nth (11(=12-1) chorus x 12 bars x 4 beats = 528) couples of past impression vectors and corresponding intention vectors $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$. Therefore, training data is $(\mathbf{x}^{(k)}, \mathbf{y}_j^{(k)})$ ($k = 1, 2, \dots, N, j = 1, 2, 3$).

The RBF network has Mth ($M \leq N$) units. Each unit indicates a center vector on an Lth dimensional space. The output of a unit is represented by equation (1).

$$\phi(r) = \exp\left(-\frac{r^2}{\sigma^2}\right) \quad \sigma \neq 0, r \geq 0 \quad (1)$$

$$r = \|\mathbf{x}^{(k)} - \mathbf{c}_i\|$$

The output of the RBF network $\tilde{\mathbf{y}}_j^{(k)}$ is the sum of products of unit outputs $\phi(r)$ and weights $a_j(i)$ represented in equation (2).

$$\tilde{\mathbf{y}}_j^{(k)} = \sum_{i=1}^M a_j(i) \phi(\|\mathbf{x}^{(k)} - \mathbf{c}_i\|) \quad (2)$$

By using modeling residual $\mathbf{e}_j^{(k)}$, the relationship of the network input and output can be represented by equation (3).

$$\mathbf{y}_j = \mathbf{P}(\mathbf{x})\mathbf{a}_j + \mathbf{e}_j \quad (3)$$

$$\mathbf{P}(\mathbf{x}) = \begin{bmatrix} p_1^{(1)} & p_2^{(1)} & \cdots & p_M^{(1)} \\ p_1^{(2)} & p_2^{(2)} & \cdots & p_M^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ p_1^{(N)} & p_2^{(N)} & \cdots & p_M^{(N)} \end{bmatrix}$$

$$p_i^{(k)} = \phi(\|\mathbf{x}^{(k)} - \mathbf{c}_i\|) \quad (k = 1, 2, \dots, N, i = 1, 2, \dots, M)$$

$$\mathbf{y}_j = \begin{bmatrix} y_j^{(1)} \\ y_j^{(2)} \\ \vdots \\ y_j^{(N)} \end{bmatrix}, \mathbf{a}_j = \begin{bmatrix} a_j(1) \\ a_j(2) \\ \vdots \\ a_j(M) \end{bmatrix}, \mathbf{e}_j = \begin{bmatrix} e_j^{(1)} \\ e_j^{(2)} \\ \vdots \\ e_j^{(N)} \end{bmatrix}$$

The RBF networks have good generalization ability and can learn whichever nonlinear mapping function we are dealing with.

4 Learning a phrase model

A phrase model is a set of player's characteristic phrases. To create a virtual player that performs using phrases as an actual human player does, acquiring the actual player's individual phrase model is necessary. This can be done through musical segmentation of a MIDI session recording (Figure 6).

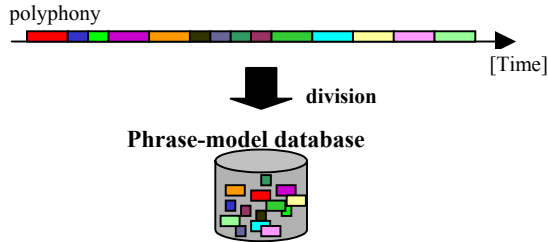


Figure 6: Acquiring phrase model.

Two kinds of grouping appear in polyphony, one in the direction of pitch interval and the other in the direction of time. Grouping in the pitch interval direction divides polyphony into multiple homophony (Figure 7a). In the time direction, notes are grouped from time gap to time gap (Figure 7b).

To segment a MIDI session recording into phrases, we need to automatically divide the polyphony notes into groups. The generative theory of tonal music (GTTM) [Lerdahl and Jackendoff, 1983] includes a grouping concept, and thus can be used to derive a set of rules for the division of notes into groups. We think that GTTM is the most promising theory of music in terms of computer implementation; however, no strict order exists for applying the rules of GTTM. This may lead to ambiguities in terms of analysis results. The implementation of GTTM as a computer system has been attempted [Ida et al., 2001], but the resulting system was only capable of dealing with a limited polyphony made up of two monophonies.

In this paper, we propose a method of grouping based on applying the Voronoi diagram. We have developed a method of grouping rather than naively implementing GTTM so that a result obtained using our method is equifinal to one obtained with the GTTM approach. We

a: Grouping in pitch interval direction.



b: Grouping in time direction.



Figure 7: Examples of grouping.

compare the results of grouping by our method with the results of grouping by a human according to the GTTM.

4.1 Generative Theory of Tonal Music

The generative theory of tonal music is composed of four modules, each of which is assigned to a separate part of the structural description of a listener's understanding of music. The four GTTM modules are the *grouping structure*, the *metrical structure*, the *time-span reduction*, and the *prolongational reduction*.

The grouping structure is intended to formalize the intuition that tonal music is organized into groups, which are in turn composed of subgroups. There are two kinds of rules for GTTM grouping: *grouping well-formedness rules* and *grouping preference rules*. Grouping well-formedness rules are necessary conditions for the assignment of a grouping structure and restrictions on the generated structures. When more than one structure may satisfy the grouping well-formedness rules, grouping preference rules only suggest the superiority of one structure over another; they do not represent a deterministic procedure. This can lead to the problem of ambiguity mentioned above.

4.2 Use of Voronoi diagrams for grouping

To overcome the ambiguity problem, we propose a method of grouping based on the use of Voronoi diagrams. The GTTM result is a binary tree that indicates the hierarchical structure of a piece of music. In our method, Voronoi diagrams on a piano-roll represent the hierarchical structure of a piece of music.

Voronoi diagram

A Voronoi diagram partitions a plane into cells (Figure 8) designated by the set of points P – that is, $\{p_1, p_2, \dots, p_n\}$ – such that for each cell corresponding to point p_i , all points q in that cell are nearer to p_i than to any other point in P [Aurenhammer, 1991]:

$$\text{dist}(q, p_i) < \text{dist}(q, p_j) \quad (p_i, p_j \in P, j \neq i). \quad (4)$$

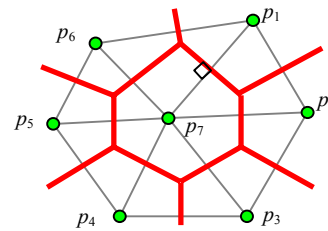


Figure 8: Voronoi diagram.

We can form the Voronoi diagram for a given set of points in a plane as a connected set of segments of half-plane boundaries, where each of the half-planes is formed by partitioning the plane into two half-planes, one on either side of the bisector of the line between each adjacent pair p_i and p_j .

Voronoi diagram for two notes

Our method uses the piano-roll format as a score, and thus notes are expressed as horizontal line segments on a piano-roll. To construct a Voronoi diagram on the score, we need to consider the Voronoi diagram for multiple horizontal line segments, which will be constructed of linear segments and quadratic segments.

When two notes sound at the same time or no note sounds, the corresponding part of the Voronoi diagram is a linear segment (Figures. 9a and 9c). When a single note sounds, the Voronoi diagram becomes a quadratic segment (Figure 9b).

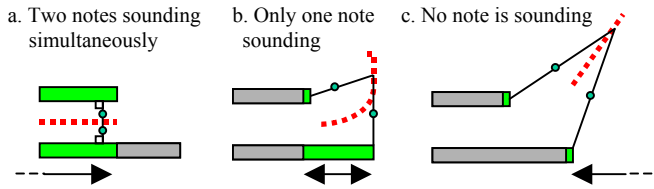


Figure 9: Voronoi diagrams for pairs of notes.

Voronoi diagram for more than two notes

To construct a Voronoi diagram for more than two notes, we construct the Voronoi diagrams for all note pairs and delete the irrelevant segments. For example, to construct a Voronoi diagram for notes a, b, and c, we construct three Voronoi diagrams (Figure 10). The boundaries in the three diagrams then intersect at a point that is equidistant from each note. The Voronoi diagram for notes a and b is divided into two half-lines at the intersection. We then delete the half-line that is closer to c than to a or b.

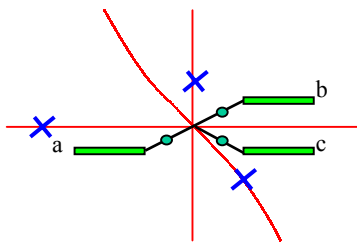


Figure 10: Voronoi diagram for more than two notes.

4.3 Making groups

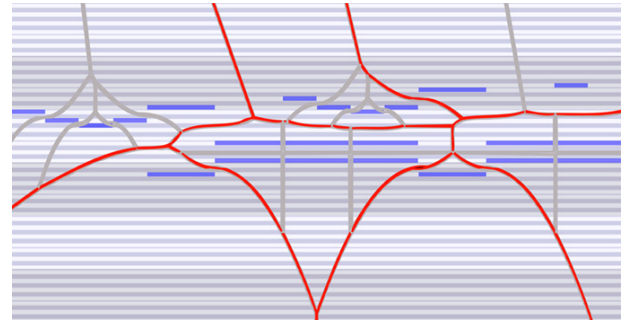
Hierarchical groups of notes were constructed by converging adjacent notes iteratively. Here, we introduce the following principle for making groups; the smallest Voronoi cell is first merged to an adjacent group.

We have implemented our grouping method (Figure 11a) and have compared the results with correct data obtained by a human (Figure 11b). We evaluated grouping performance in terms of a correctness rate defined as

$$\text{Correctness rate} = \frac{\text{The number of notes grouped correctly}}{\text{The number of notes}} \quad (5)$$

When we ran the program, we calculated that the correctness rate was 78.5 percent. The tune used as MIDI data in this experiment was the *Turkish March*.

a. Result obtained using our method.



b. Result obtained using GTTM.

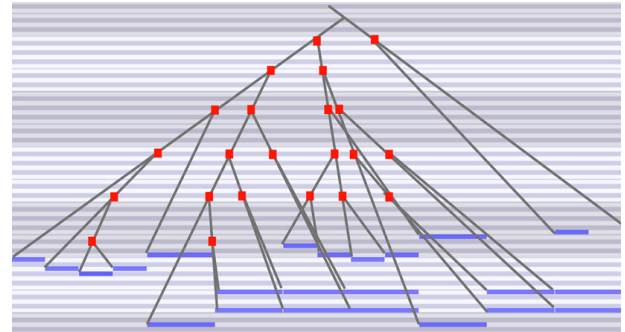


Figure 11: Results obtained using our method and GTTM.

5 Learning a groove model

A groove model is a model generating the deviation of onset times. Acquiring an actual player's individual groove model is necessary to create a virtual player that performs with musical expression as that human player does. A human player, even when repeating a given phrase on a MIDI-equipped instrument, rarely produces exactly the same sequence of onset notes because the onset times deviate according to performer's actions and expressions. We can model the process generating that deviation by using a probabilistic model.

5.1 Formulation of the hidden Markov models

Let a sequence of intended (normalized) onset times be θ and a sequence of performed onset times (with deviation) be y . Then, a model for generating the deviation of onset times can be expressed by a conditional probability $P(y|\theta)$ (Figure 12). Using this conditional probability and the prior probability $P(\theta)$ can be formulated as a hidden Markov model (HMM), which is a probabilistic model that generates a transition sequence of hidden states as a Markov chain. Each hidden state in the state transition sequence then generates an observation value according to an observation probability.

Modeling of performance

Target in modeling

We modeled the onset time of a musical note (i.e. the start time of the note) and introduced a new model of distribution of onset times. While the duration- time-based model

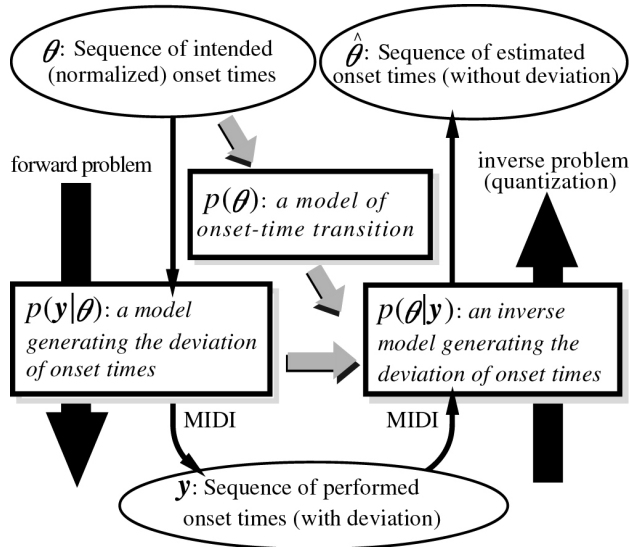


Figure 12: Model to generate onset time deviation.

used in [Otsuki *et al.*, 2002] is limited, our onset-time-based model is suitable for treating polyphonic performances, such as those including two-hand piano voicing and guitar arpeggio.

Unit in modeling

We use a quarter note (beat) as the unit of each HMM: the temporal length corresponding to each HMM is a quarter note. The reason we use the quarter-note unit is to distinguish between eighth triplets and sixteenth notes within the scope of quarter notes. The three notes of eighth triplets are located on three equi-divided positions in a quarter note, while the four sixteenth notes are located on four equi-divided positions in a quarter note. An actual per-

formance consisting of a sequence of quarter notes can be modeled by concatenating the quarter-note-length HMMs. This quarter-note modeling has the advantages of reducing calculation time and facilitating the preparation of the large data sets used for training the model.

Unit of quantization

We introduce two different discrete temporal indices, k and i . The unit of k is a quantization unit to describe performed onset time and is $1/480$ of a quarter note, which is often used in commercial sequencing software. The i unit is a quantization unit to describe the intended onset time and is one-twelfth of a quarter note. It can describe both eighth triplets and sixteenth notes.

Quarter-note hidden Markov model

Figure 13 shows the HMM used in our study. We model a sequence of onset times within a quarter note (beat) by using the HMM. All the hidden states of the HMM correspond to possible positions of intended onset times, and an observed value that comes from a hidden state corresponds to a performed onset time with deviation. Onset times in a beat are quantized into 12 positions for hidden states and into 480 positions for observation values. That is, each component of the HMM is interpreted as follows. Hidden state i : intended onset time. ($i = 1, \dots, 12$) Observation k : performed onset time. ($k = 1, \dots, 480$) Transition probability a_{ij} : probability that intended onset time j follows intended onset time i . Observation probability $b_i(k)$: probability that performed onset time is k and intended onset time is i . A state-transition sequence begins with a dummy state “Start” and ends with a state “End” (Figure 14).

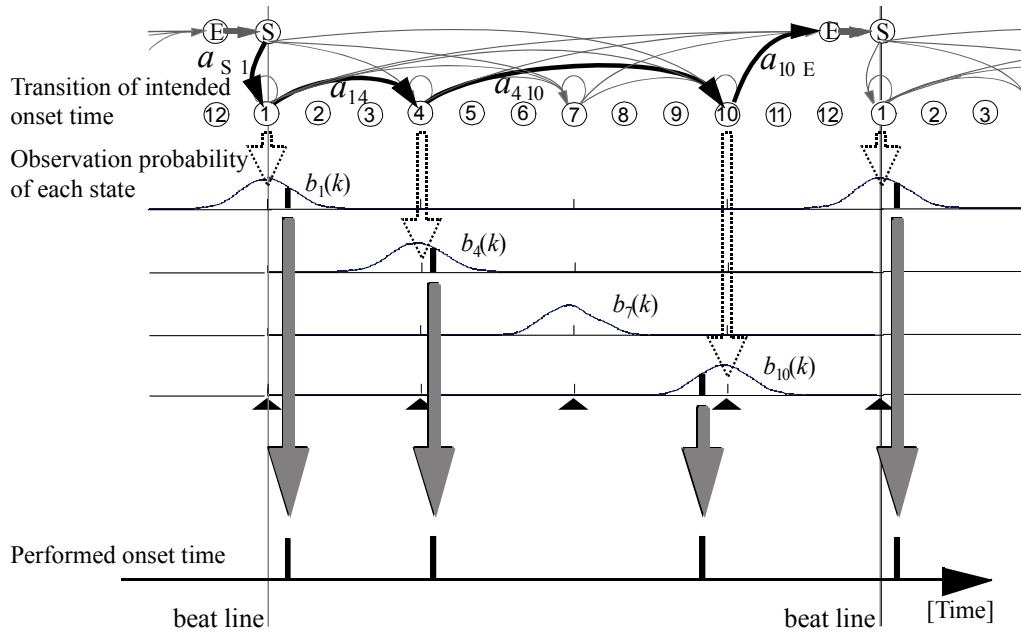


Figure 13: Overview of quarter-note hidden Markov model

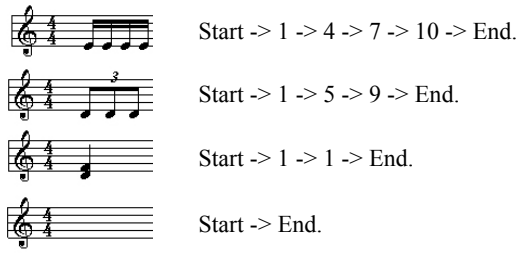


Figure 14: Simple example of state sequences.

5.2 Learning model parameters

The HMM parameters a_{ij} and $b_i(k)$ were derived from a MIDI session recording by using the Baum-Welch algorithm. Figure 15 shows a $b_i(k)$ distribution obtained from a human performance in a MIDI session recording.

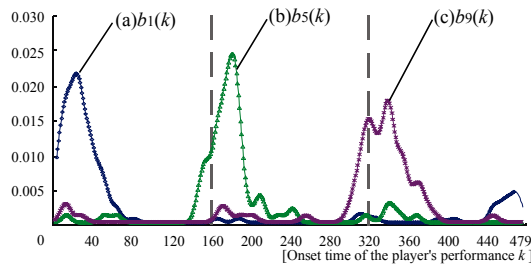


Figure 15: $b_1(k)$, $b_5(k)$, $b_9(k)$ obtained from human-performance data.

6 Session mode

Using the personality models acquired in the learning mode, each virtual player improvises while reacting to the human player and the other virtual player. The processing flow of each virtual player can be summarized as follows:

1. The low-level features of the MIDI performances of all the players are calculated at every 1/48 bar.
2. At every 1/12 bar, the three impression vectors are obtained from the low-level features.
3. At the beginning of every bar, the intention vector is determined by feeding the reaction model the past impression vectors.
4. The output performance is generated by connecting short phrases selected from a phrase-model database. Each phrase is selected, according to the determined intention vector, by considering its fitness for the chord progression. A virtual player can start a solo performance at any bar.
5. The deviation of the onset times is generated according to the groove model.

Note that the reaction model can predict the next intention vector from the impression vectors gathered during the past twelve bars in real time: a virtual player thus does not fall behind the other players.

7 Experimental Results

We have implemented the proposed system on a personal computer (with a Pentium IV 2.8GHz processor); Figure

16 shows a screen snapshot of the system output. As shown, there are three columns (called *player panels*), each corresponding to a different player. The toggle switch on the top of each panel indicates whether the panel is for a virtual player or a human player, and each panel contains two boxes representing three-dimensional spaces: the upper box is the impression space and the lower box is the intention space. The sphere in each box indicates the current value of the impression or intention vector.

In our experiments, after recording a session performance of three human guitarists playing MIDI guitars, we first made the system learn the reaction model, the phrase model and the groove model of each. We used a metronome sound to maintain the tempo (120 MM) when recording, and the total length of this recording session was 144 bars. We then let five human guitarists individually use the system in session mode. The system indeed enabled each human guitarist to interact with two virtual guitarists, each with a different reaction model.

To find out how well a virtual player could imitate a human player, we asked a human player to perform with virtual player A imitating him and with virtual player B imitating a different player. The human player and the virtual player imitating him tended to take a solo at almost the same time and to perform phrases that felt similar. Figure 17 shows the transition of intention vectors of three players during 48 bars where the intention vectors of the virtual player A and the human player are particularly similar. Examining all the values of the intention vectors during the session, we compared the distances between the intention vectors of the virtual players and the human player. Over 144 bars the average distance between the intention vectors of the human player and the virtual player imitating him was significantly smaller than that between the intention vectors of the human player and the virtual player imitating a different player. We think that this means the virtual player's RBF networks actually predicted the human player's intention.

We also tested whether a virtual player could imitate the target human player by applying the Turing test format. As subjects, we used three guitarists (A, B, and C) who had played in the same band for more than a year and so un-

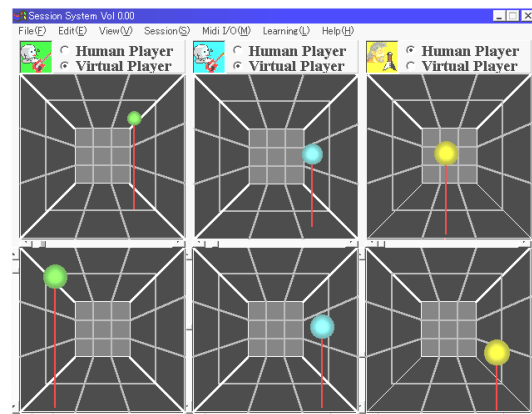


Figure 16: Screen snapshot of system output.

derstood each other's musical personalities. We prepared a reaction model, a phrase model, and a groove model for each of these subjects. We then used different personality models to prepare 27 ($= 3 \text{ players} \times 3 \text{ models}$) kinds of virtual player. The subjects evaluated each virtual player with regard to whose models it was based upon. Subjects were told in advance that the player was a virtual player imitating either player A, B, or C. We found that a virtual player having the three personality models acquired from the same human player was correctly identified as such. We calculated that the success rate was 100 percent. The subjects could thus distinguish when the virtual player was based on the personality models from one human player or from multiple players.

Furthermore, five guitarists who performed with the system remarked that each virtual player performed characteristically. In particular, a human player who participated in a jam session with a virtual player that was imitating him remarked that he was uncomfortable playing with that virtual player because he felt that he was being mimicked. These results show that our system successfully derived and applied personality models from MIDI session recordings.

8 Conclusion

We have described a jam session system in which a human player and virtual players interact with each other. The system is based on the learning of three types of personality model – the reaction, the phrase, and the groove models. Our experimental results show that our system can imitate the musical personalities of human players. We plan to extend the system so that it can be applied to other musical instruments, such as piano and drums.

Acknowledgments

We thank Daisuke Hashimoto, Shinichi Chida, and Yasuhiro Saita, who cooperated with the experiment as players.

References

- [Aono *et al.*, 1995] Yushi Aono, Haruhiro Katayose and Seiji Inokuchi. An improvisational accompaniment system observing performer's musical gesture, Proceedings of the 1995 International Computer Music Conference, pp. 106-107, September 1995.
- [Wake *et al.*, 1994] Sanae Wake, Hirokazu Kato, Naoki

- Saiwaki, and Seiji Inokuchi. Cooperative Musical Partner System Using a Tension- Parameter: JASPER (Jam Session Partner), Transaction of IPS Japan, Vol. 35, No. 7, pp. 1469-1481, July 1994. (in Japanese)
- [Goto *et al.*, 1996] Masataka Goto, Isao Hidaka, Hideaki Matsumoto, Yousuke Kuroda, and Yoichi Muraoka. A Jazz Session System for Interplay among All Players -VirJa Session (Virtual Jazz Session System), Proceedings of the 1996 International Computer Music Conference, pp. 346-349, August 1996.
- [Nishijima and Kijima, 1989] Masako Nishijima and Yuji Kijima. Learning on Sense of Rhythm with a Neural Network-The NEURO DRUMMER, Proceedings of the 1989 International Conference on Music Perception and Cognition, pp78-80, October 1989.
- [Nishijima and Watanabe, 1992] Masako Nishijima and Kazuyuki Watanabe. Interactive music composer based on neural networks, Proceedings of the 1992 International Computer Music Conference, pp.53-56, October 1992.
- [Thom, 2001a] Belinda Thom. Bob: An Improvisational Music Companion, Doctoral Thesis, School of Computer Science, Carnegie Mellon University, July 2001.
- [Thom, 2001b] Belinda Thom. Machine Learning Techniques for Real-time Improvisational Solo Trading, Proceedings of the 2001 International Computer Music Conference, pp.439-446, September 2001.
- [Kruskal and Wish, 1978] Joseph B. Kruskal, and Myron Wish. Multidimensional Scaling, Sage Publications, 1978.
- [Chen *et al.*, 1991] S. Chen, C.F.N. Cowan and P.M. Great. Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks, IEEE Transactions on Neural Networks, Vol. 4, pp. 246-257, March 1991.
- [Lerdahl and Jackendoff, 1983] Fred Lerdahl and Ray Jackendoff. A Generative Theory of Tonal Music, The MIT Press, 1983.
- [Ida *et al.*, 2001] Kentarou Ida, Keiji Hirata, and Satoshi Tojo. An Attempt at Automatic Analysis of the Grouping Structure and Metrical Structure Based on GTTM, Transaction of IPS Japan, Vol. 2001, No. 42, October 2001. (in Japanese)
- [Aurenhammer, 1991] Franz Aurenhammer. "Voronoi Diagrams - a Survey of Fundamental Geometric Data Structure," ACM Computing Surveys, Vol. 23, 1991.
- [Otsuki *et al.*, 2002] Tomoshi Otsuki, Naoki Saitou, Mitsuru Nakai, Hiroshi Shimodaira, and Shigeki Sagayama. Musical Rhythm Recognition Using Hidden Markov Model, Journal of Information Processing Society of Japan, Vol.43, No.2, February 2002. (in Japanese)

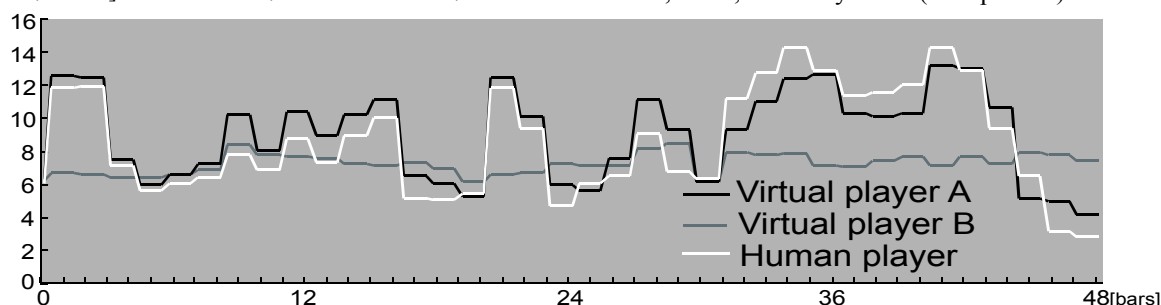


Figure 17: Transition of intention vectors of three players. (This Figure charts one component of intention vectors that have three components.)

Getting Serious about the Development of Computational Humor

Oliviero Stock and Carlo Strapparava

ITC-irst, I-38050 Povo Trento Italy

{stock, strappa}@itc.it

Abstract

Society needs humor, not just for entertainment. In the Web age, presentations become more and more flexible and personalized and they will require humor contributions for electronic commerce developments (e.g. product promotion, getting selective attention, help in memorizing names, etc...) more or less as it happened in the world of broadcasted advertisement. Even if deep modeling of humor in all of its facets is not something for the near future, there is something concrete that has been achieved and that can help in providing attention to the field. The paper refers to the results of HAHACRONYM, a project devoted to humorous acronym production, a circumscribed task that nonetheless requires various generic components. The project opens the way to developments for creative language, with applications in the world of advertisement.

1 Introduction

Future human-machine interaction, according to a widespread view, will emphasize naturalness and effectiveness and hence the incorporation of models of possibly all human cognitive capabilities, including the handling of humor. There are many practical settings where computational humor will add value. Among them: business world applications (e-commerce to name one), general computer-mediated communication and human-computer interaction [Morkes *et al.*, 1999], educational and edutainment systems. There are important prospects for humor also in automatic information presentation. In the Web age presentations become more and more flexible and personalized and they will require humor contributions for electronic commerce developments (e.g. product promotion, getting selective attention, help in memorizing names, etc...) more or less as it happened in the world of broadcasted advertisement.

Yet deep modeling of humor in all of its facets is not something for the near future; the phenomena are too complex, humor is one of the most sophisticated forms of human intelligence. It is *AI-complete*: the problem of modeling it is as difficult to solve as the most difficult Artificial Intelligence problems. But some steps can be followed to achieve results. In the general case, in order to be successfully humorous,

a computational system should be able to: recognize situations appropriate for humor; choose a suitable kind of humor for the situation; generate an appropriately humorous output; and, if there is some form of interaction or control, evaluate the feedback.

We are concerned with systems that automatically produce humorous output (rather than systems that appreciate humor). Some of the fundamental competencies are within the range of the state of the art of natural language processing. In one form or in another humor is most often based on some form of incongruity. For verbal humor this means that different interpretations of utterances must be possible (and must not be detected before the culmination of the humorous process) or must cause perception of specific forms of opposition. Natural language processing research has often dealt with ambiguity in language. A common view is that ambiguity is an obstacle for deep comprehension. Most current text processing systems attempt to reduce the number of possible interpretations of the sentences, and a failure to do so is seen as a weakness of the system. From a different point of view, the potential for ambiguity can be seen as a positive feature of natural language. Metaphors, idioms, poetic language and humor use all the multiple senses of texts to suggest connections between concepts that cannot, or should not, be stated explicitly.

The work presented here is based on various resources for natural language processing, adapted for humor. It is a small step, but aiming at an appreciable concrete result. It has been developed within HAHACRONYM¹, a project devoted to computational humor. A visible and evaluable result was at the basis of the deal. We proposed a situation that is of practical interest, where there is no domain restriction and many components are present, but simpler than in more extended scenarios. The goal is a system that makes fun of existing acronyms, or, starting from concepts provided by the user, produces a new acronym, constrained to be a word of the given language. And, of course, it has to be funny.

The project was meant to convince about the potential of computational humor, through the demonstration of a working prototype and an assessment of the state of the art and

¹European Project IST-2000-30039. HAHACRONYM has been the first EU project devoted to computational humor. The consortium included ITC-irst, as coordinator, and the University of Twente. See <http://haha.itc.it>.

of scenarios where humor can add something to existing information technologies. The results of the project put us in a better position to move forwards in introducing computational humor in more complex scenarios.

2 AI and Computational Humor

So far only very limited effort has been put on building computational humor prototypes. Indeed very few working prototypes that process humorous text and/or simulate humor mechanisms exist, and mostly they worked in very restricted domains.

There has been a considerable amount of research on linguistics of humor and on theories of semantics and pragmatics of humor [Attardo, 1994; Attardo and Raskin, 1991; Giora and Fein, 1999; Attardo, 2002]; however, most of the work has not been formal enough to be used directly for computational humor modeling. An effort toward formalization of forced reinterpretation jokes has been presented by Ritchie [2002].

Within the artificial intelligence community, most writing on humor has been speculative [Minsky, 1980; Hofstadter *et al.*, 1989]. Minsky made some preliminary remarks about how humor could be viewed from the artificial intelligence/cognitive science perspective, refining Freud's notion that humor is a way of bypassing our mental "censors" which control inappropriate thoughts and feelings. Utsumi [1996] outlines a logical analysis of irony, but this work has not been implemented. Among other works: De Palma and Weiner [1992] have worked on knowledge representation of riddles, Ephratt [1990] has constructed a program that parses a limited range of ambiguous sentences and detects alternative humorous readings. A formalization, based on a cognitive approach (the belief-desire-intention model), distinguishing between real and fictional humor has been provided by Mele [2002].

Probably the most important attempt to create a computational humor prototype is the work of Binsted and Ritchie [1997]. They have devised a formal model of the semantic and syntactic regularities underlying some of the simplest types of punning riddles. A punning riddle is a question-answer riddle that uses phonological ambiguity. The three main strategies used to create phonological ambiguity are syllable substitution, word substitution and metathesis.

Almost all the computational approaches try to deal with the incongruity theory at various level of refinement [Raskin, 1985; Attardo, 1994]. The incongruity theory focuses on the element of surprise. It states that humor is created out of a conflict between what is expected and what actually occurs in the joke. This accounts for the most obvious features of a large part of humor phenomena: ambiguity or double meaning.

Specific workshops concerned with Computational Humor have taken place in recent years and have drawn together most of the community active in the field. The proceedings of the most comprehensive events are [Hulstijn and Nijholt, 1996] and [Stock *et al.*, 2002]. Ritchie [2001] has published a survey of the state of the art in the field.

3 HAHACRONYM

3.1 Resources

In order to realize the HAHACRONYM prototype, we have refined existing resources and we have developed general tools useful for humorous systems. As we will see, a fundamental tool is an incongruity detector/generator, that makes the system able to detect semantic mismatches between word meaning and sentence meaning (i.e. in our case the acronym and its context). For all tools, particular attention was put on reusability.

The starting point for us consisted in making use of some "off-the-shelf" resources, such as WORDNET DOMAINS [Magnini *et al.*, 2002] (an extension of the well-known English WORDNET) and standard parsing techniques. The tools resulting from the adaptation will be reusable for other applications, and are portable straightforwardly to other languages (e.g. WORDNET DOMAINS is multilingual).

Wordnet and Wordnet Domains

WORDNET is a thesaurus for the English language inspired by psycholinguistics principles and developed at the Princeton University by George Miller [Fellbaum, 1998]. It has been conceived as a computational resource, therefore improving some of the drawbacks of traditional dictionaries, such as circularity of definitions and ambiguity of sense references. Lemmata (about 130,000 for version 1.6) are organized in synonym classes (about 100,000 *synsets*). WORDNET can be described as a "lexical matrix" with two dimensions: a dimension for *lexical relations*, that is relations holding among words, and therefore language specific, and a dimension for *conceptual relations*, which hold among senses (the synsets) and that, at least in part, we consider independent from a particular language. A synset contains all the words by means of which it is possible to express the synset meaning: for example the synset {horse, Equus_caballus} describes the sense of "horse" as an animal, while the synset {knight, horse} describes the sense of "horse" as a chessman. The main relations present in WORDNET are *synonymy*, *antonymy*, *hyperonymy-hyponymy*, *meronymy-holonymy*, *entailment*, *troponymy*.

Augmenting WordNet with Domain information

Domains have been used both in linguistics (i.e. Semantic Fields) and in lexicography (i.e. Subject Field Codes) to mark technical usages of words. Although this is useful information for sense discrimination, in dictionaries it is typically used for a small portion of the lexicon. WORDNET DOMAINS is an attempt to extend the coverage of domain labels within an already existing lexical database, WORDNET (version 1.6). The synsets have been annotated with at least one domain label, selected from a set of about two hundred labels hierarchically organized. (Figure 1 shows a portion of the domain hierarchy.)

We have organized about 250 domain labels in a hierarchy (exploiting Dewey Decimal Classification), where each level is made up of codes of the same degree of specificity: for example, the second level includes domain labels such as BOTANY, LINGUISTICS, HISTORY, SPORT and RELIGION, while at the third level we can find specialization such as

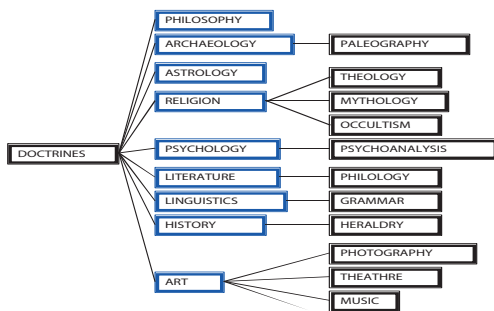


Figure 1: A portion of the domain hierarchy

AMERICAN_HISTORY, GRAMMAR, PHONETICS and TENNIS.

Information brought by domains is complementary to what is already present in WORDNET. First of all a domain may include synsets of different syntactic categories: for instance MEDICINE groups together senses from Nouns, such as *doctor#1* and *hospital#1*, and from Verbs such as *operate#7*. Second, a domain may include senses from different WORDNET sub-hierarchies. For example, SPORT contains senses such as *athlete#1*, deriving from *life_form#1*, *game_equipment#1*, from *physical_object#1*, *sport#1* from *act#2*, and *playing_field#1*, from *location#1*.

Opposition of semantic fields

On the basis of well recognized properties of humor accounted for in many theories (e.g. incongruity, semantic field opposition, apparent contradiction, absurdity) we have modelled an independent structure of domain opposition, such as RELIGION vs. TECHNOLOGY, SEX vs. RELIGION, etc... We exploit these kind of opposition as a basic resource for the incongruity generator.

Adjectives and Antonymy Relations

Adjectives play an important role in modifying and generating funny acronyms. WORDNET divides adjectives into two categories. *Descriptive adjectives* (e.g. big, beautiful, interesting, possible, married) constitute by far the largest category. The second category is called simply *relational adjectives* because they are related by derivation to nouns (i.e. electrical in electrical engineering is related to noun electricity). To relational adjectives, strictly dependent on noun meanings, it is often possible to apply similar strategies as those exploited for nouns. Their semantic organization, though, is entirely different from the one of the other major categories. In fact it is not clear what it would mean to say that one adjective "is a kind of" (ISA) some other adjective. The basic semantic relation among descriptive adjectives is antonymy. WORDNET proposes also that this kind of adjectives is organized in clusters of synsets associated by semantic similarity to a focal adjective. Figure 2 shows clusters of adjectives around the direct antonyms *fast/slow*.

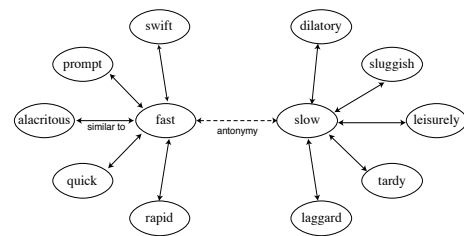


Figure 2: An example of adjective clusters linked by antonymy relation

Exploiting the hierarchy

It is possible to exploit the network of lexical and semantic relations built in WORDNET to make simple ontological reasoning. For example, if a noun or an adjective has a geographic location meaning, the pertaining country and continent can be inferred.

Rhymes

The HAHACRONYM prototype takes into account word rhymes and the rhythm of the acronym expansion. To cope with this aspect we got and reorganized the CMU pronouncing dictionary (<http://www.speech.cs.cmu.edu/cgi-bin/cmudict>) with a suitable indexing. The CMU Pronouncing Dictionary is a machine-readable pronunciation dictionary for North American English that contains over 125,000 words and their transcriptions.

Its format is particularly useful for speech recognition and synthesis, as it has mappings from words to their pronunciations in the given phoneme set. The current phoneme set contains 39 phonemes; vowels may carry lexical stress.

Parser, grammar and morphological analyzer

Word sequences that are at the basis of acronyms are subject to a well-defined grammar, simpler than a complete noun phrase grammar, but complex enough to require a nontrivial analyzer. We have decided to use a well established non-deterministic parsing technique. As far as the dictionary is concerned, we use the full WORDNET lexicon, integrated with an ad-hoc morphological analyzer.

Also for the generation part we exploit the grammar as the source for syntactic constraints.

All the components are implemented in Common Lisp augmented with nondeterministic constructs.

Other resources

An "a-semantic" dictionary is a collection of hyperbolic/epistemic/deontic adjective/adverbs. This is a last resource, that some time can be useful in the generation of new acronyms. Some examples are: *abnormally*, *abstrusely*, *adorably*, *exceptionally*, *exorbitantly*, *exponentially*, *extraordinarily*, *voraciously*, *weirdly*, *wonderfully*. This resource is hand-made, using various dictionaries as information sources.

Other lexical resources are: a euphemism dictionary, a proper noun dictionary, lists of typical foreign words commonly used in the language with some strong connotation.

3.2 Architecture and Implementation

To get an ironic or “profaning” re-analysis of a given acronym, the system follows various steps and strategies. The main elements of the algorithm can be schematized as follows:

- acronym parsing and construction of a logical form
- choice of what to keep unchanged (typically the head of the highest ranking NP) and what to modify (e.g. the adjectives)
- look up for possible substitutions
- exploitation of semantic field oppositions
- granting phonological analogy: while keeping the constraint on the initial letters of the words, the overall rhyme and rhythm should be preserved (the modified acronym should sound similar to the original as much as possible)
- exploitation of WORDNET antonymy clustering for adjectives
- use of the a-semantic dictionary as a last resource

Figures 3 and 4 show a sketch of the system architecture.

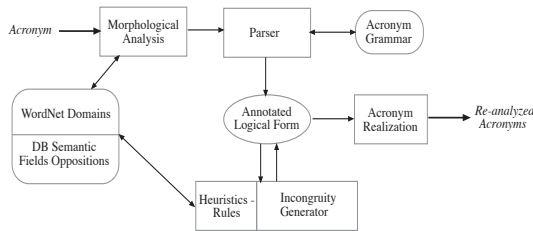


Figure 3: Acronyms Reanalysis

In our system, making fun of existing acronyms amounts to an ironic rewriting, desecrating them with some unexpectedly contrasting, but otherwise consistently sounding expansion.

As far as acronym generation is concerned, the problem is more complex. To make the task more attractive - and difficult - we constrain resulting acronyms to be words of the dictionary (APPLE would be good, IBM would not). The system takes in input concepts (actually synsets, possibly the result of some other process, for instance sentence interpretation) and some minimal structural indication, such as the semantic head. The primary strategy of the system is to consider as potential acronyms words that are in ironic relation with the input concepts. By definition acronyms have to satisfy constraints - to include the initial letters of some lexical realization of the inputs words synsets, granting that the sequence of initials satisfy the overall acronym syntax. In this primary strategy, the ironic reasoning comes mainly at the level of acronym choice in the lexicon and in the selection of the fillers of the *open slots* in the acronym.

For example, giving as input “fast” and “CPU”, we get static, torpid, dormant. (Note that the complete synset for “CPU” is {processor#3, CPU#1, central_processing_unit#1, mainframe#2}. So

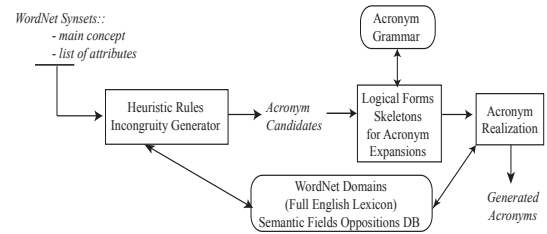


Figure 4: Acronyms Generation

we can use in the acronym expansion a synonym of “CPU”. The same happens for “fast”). Once we have an acronym proposal, a syntactic skeleton has to be filled to get a correct noun phrase. For example given in input “fast” and “CPU”, the system selects TORPID with the possible syntactic skeleton:

$\langle \text{adv} \rangle_T \langle \text{adj} \rangle_O$ Rapid Processor $\langle \text{prep} \rangle \langle \text{adj} \rangle_I \langle \text{noun} \rangle_D$
or

$\langle \text{adj} \rangle_T \langle \text{adj} \rangle_O$ Rapid Processor $\langle \text{prep} \rangle \langle \text{noun} \rangle_I \langle \text{noun} \rangle_D$

where “rapid” and “processor” are synonyms respectively of “fast” and “CPU” and the notation $\langle \text{Part.of.Speech} \rangle_{\text{Letter}}$ means a word of that particular part.of.speech with *Letter* as initial. Then the system fills this syntactic skeleton with strategies similar to those described for re-analysis.

The system is fully implemented in Common Lisp, exploiting CLOS and the Meta-Object protocol to import WORDNET DOMAINS and to implement the navigation/reasoning algorithms.

3.3 Examples

Here below some examples of acronym re-analysis are reported. As far as semantic field opposition is concerned we have slightly tuned the system towards the domains FOOD, RELIGION and SEX. We report the original acronym, the re-analysis and some comments about the strategies followed by the system.

ACM - Association for Computing Machinery
→ Association for Confusing Machinery
FBI - Federal Bureau of Investigation
→ Fantastic Bureau of Intimidation

The system keeps all the main heads and works on the adjectives and the PP head, preserving the rhyme and/or using the a-semantic dictionary.

CRT - Cathodic Ray Tube
→ Catholic Ray Tube
ESA - European Space Agency
→ Epicurean Space Agency
PDA - Personal Digital Assistant
→ Penitential Demoniactal Assistant
→ Prenuptial Devotional Assistant
MIT - Massachusetts Institute of Technology
→ Mythical Institute of Theology

Some re-analyses are RELIGION oriented. Note the rhymes.

As far as generation from scratch is concerned, a main concept and some attributes (in terms of synsets) are given as input to the system. Here below we report some examples of acronym generation.

Main concept: *processor* (in the sense of CPU);
Attribute: *fast*

OPEN - On-line Processor for Effervescent Net

PIQUE - Processor for Immobile Quick Uncertain Experimentation

TORPID - Traitorously Outstandingly Rusty Processor for Inadvertent Data processing

UTMOST - Unsettled Transcendental Mainframe for Off-line Secured Tcp/ip

We note that the system tries to keep all the expansions of the acronym coherent in the same semantic field of the main concept (COMPUTER_SCIENCE). At the same time, whenever possible, it exploits some incongruity in the lexical choices.

3.4 Evaluation

Testing the humorous quality of texts is not an easy task. There have been relevant studies though, such as [Ruch, 1996]. For HAHACRONYM, a simpler case, an evaluation was conducted under the supervision of Salvatore Attardo at Youngstown University, Ohio. Both reanalysis and generation have been tested according to criteria of success stated in advance and in agreement with the European Commission, at the beginning of the project.

The participants in the evaluation were forty students. They were all native speakers of English. The students were not told that the acronyms had been computer generated. No record was kept of which student had given which set of answers (the answers were strictly anonymous). No demographic data were collected. However, generally speaking, the group was homogeneous for age (traditional students, between the ages of 19 and 24) and mixed for gender and race.

The students were divided in two groups. The first group of twenty was presented the reanalysis and generation data. We tested about 80 reanalyzed and 80 generated acronyms (over twice as many as required by the agreement with the European Commission). Both the reanalysis module and the generation module were found to be successful according to the criteria spelled out in the assessment protocol. The acronyms reanalysis module showed roughly 70% of acronyms having a score of 55 or higher (out of a possible 100 points), while the acronym generation module showed roughly 53% of acronyms having a score of 55 or higher. The thresholds for success established in the protocol were 60% and 45%, respectively.

A set of randomly generated acronyms were presented to a different group of twenty students. A special run of the system was performed in which the semantic filters and heuristics had been disabled, while only the syntactical constraints were operational. (If the syntactical rules had been disabled as well the output would have been gibberish and it would be difficult to compare with the regular system production). In

this test, less than 8% of the acronyms passed the 55 points score test, we conclude that the output of HAHACRONYM is significantly better than random production of reanalysis.

Acronyms	scored > 55	Success Thresholds
Generation	52.87%	45%
Reanalysis	69.81%	60%
Random	7.69%	

Table 1: Evaluation Results

A curiosity that may be worth mentioning: HAHACRONYM participated to a contest about (human) production of best acronyms, organized in December 2002 by RAI, the Italian National Broadcasting Service. The system won a jury's special prize.

4 Prospects for advertisement

Humor is the healthy way of creating 'distance' between one's self and the problem, a way of standing back and looking at the problem with perspective. Humor reveals new aspects, disarms and relaxes. It is also infectious and it is an important way to communicate ideas. On the cognitive side humor has two very important properties:

- it helps getting and keeping people's attention. Type and rhythm of humor may vary and the time involved in building the effect may be different in different cases: some times there is a context - like joke telling - that from the beginning let you expect for the humorous climax, which may occur after a long while; other times the effect is obtained in almost no time, with one perceptive act - for instance in static visual humor, funny posters or in cases when some well established convention is reversed with an utterance;
- it helps remembering. For instance it is a common experience to connect in our memory some knowledge we have acquired to a humorous remark or event. In a foreign language acquisition it may happen that an involuntary funny situation is created because of so called "false friends" - words that sound similar in two languages and may have the same origin but have a very different meaning. The humorous experience is conducive to remembering the correct use of the word.

No wonder that humor has become one of the favorite strategies used by creative people involved in the advertising business. In fact among the various fields in which creative language is used, advertising is probably the area of activity in which creativity (and hence humor) is practiced with most precise objectives.

From an applied AI point of view, we believe that an environment for proposing solutions to advertising professionals can be a realistic practical development of computational humor and one of the first attempts in dealing creative language.

Some examples of the huge array of opportunities that language offers and that existent NLP techniques can cope with: rhymes, wordplays, popular sayings or proverbs, quotations, alliterations, triplets, chiasmus, neologism, non sequitur,

adaptation of existing expressions, adaptation of proverbs, personification, synaesthesia (two or more senses combined).

An intelligent system devoted to advertisement, based on an apparatus that extends the one we have described, can play with these expressions, alter them, change the context, use them as a platform for completely new ideas (e.g. create the slogan: *Thirst come, thirst served* for a soft drink).

In the longer run the aim is to go for individual-oriented advertisement. Let us leave aside issues of privacy for the moment, though we know this is a critical theme. The availability of individual profiles, automatic inferencing of interest and behavior models, information about the location of the individual, combined with reasoning on the offer side (what is advisable to promote in a given context of business) will all provide the basic input for advertising things worth getting the interest of a specific individual in a given context. Commercial advertisement will be just one case along this line - one can think of influencing the individual behaviour on the basis of the availability of any possible resources such as, for instance, cultural goods. A prediction is that when such situation-oriented advertisements will become widespread, the same features that prevail now with broadcasted material will be sought after for the new case. So, forms of humorous advertisement for the individual (even playing on personal aspects) will become a plus.

An important aspect to be taken into account is how humor is appreciated by different individuals. Personality studies regarding this specific theme give important indications [Ruch, 2002]. One option will also be to develop humor for conversational systems, based on embodied agents. The work of Nijholt [2002] Andr  and Rist [2000] and Cassell [2001] could provide the starting point for introducing dynamic humor.

For obtaining good results (there are not many things that are worse than bad humor!) for these longer term objectives, much work will be needed, especially further integration with ontologies, common sense reasoning and pragmatics.

References

- [Andr  and Rist, 2000] E. Andr  and T. Rist. Presenting through performing: On the use of multiple lifelike characters in knowledge-based presentation systems. In *Proc. of the Second International Conference on Intelligent User Interfaces (IUI 2000)*, pages 1–8, 2000.
- [Attardo and Raskin, 1991] S. Attardo and V. Raskin. Script theory revis(it)ed: joke similarity and joke representation model. *Humour*, 4(3):293–347, 1991.
- [Attardo, 1994] S. Attardo. *Linguistic Theory of Humor*. Mouton de Gruyter, Berlin, 1994.
- [Attardo, 2002] S. Attardo. Formalizing humor theory. In [Stock et al., 2002], 2002.
- [Binsted and Ritchie, 1997] K. Binsted and G. Ritchie. Computational rules for punning riddles. *Humor*, 10(1), 1997.
- [Cassell, 2001] J. Cassell. Embodied conversational agents: Representation and intelligence in user interface. *AI Magazine*, 22(3):67–83, 2001.
- [Ephratt, 1990] M. Ephratt. What’s in a joke. In M. Golumbic, editor, *Advances in AI: Natural Language and Knowledge Based Systems*, pages 406–450. Springer Verlag, 1990.
- [Fellbaum, 1998] C. Fellbaum. *WordNet. An Electronic Lexical Database*. The MIT Press, 1998.
- [Giora and Fein, 1999] R. Giora and O. Fein. Irony: Context and salience. *Metaphor and Symbol*, 14:241–257, 1999.
- [Hofstadter et al., 1989] D. Hofstadter, L. Gabora, V. Raskin, and S. Attardo. Synopsis of the workshop on humor and cognition. *Humor*, 2(4):293–347, 1989.
- [Hulstijn and Nijholt, 1996] Joris Hulstijn and Anton Nijholt, editors. *Proc. of International Workshop on Computational Humour (TWLT 12)*, University of Twente, Enschede, 1996.
- [Magnini et al., 2002] B. Magnini, C. Strapparava, G. Pezulo, and A. Gliozzo. The role of domain information in word sense disambiguation. *Journal of Natural Language Engineering*, 8(4):359–373, 2002.
- [Mele, 2002] F. Mele. Real and fictional ridicule. In [Stock et al., 2002], 2002.
- [Minsky, 1980] M. Minsky. Jokes and the logic of the cognitive unconscious. Technical report, MIT Artificial Intelligence Laboratory, 1980. AI memo 603.
- [Morkes et al., 1999] J. Morkes, H. Kernal, and C. Nass. Effects of humor in task-oriented human-computer interaction and computer-mediated communication. *Human-Computer Interaction*, 14:395–435, 1999.
- [Nijholt, 2002] A. Nijholt. Embodied agents: A new impetus to humor research. In [Stock et al., 2002], 2002.
- [Palma and Weiner, 1992] P. De Palma and E. J. Weiner. Riddles: accessibility and knowledge representation. In *Proc. of the 15th International Conference on Computational Linguistics (COLING-92)*, Nantes, 1992.
- [Raskin, 1985] V. Raskin. *Semantic Mechanisms of Humor*. Dordrecht/Boston/Lancaster, 1985.
- [Ritchie, 2001] G. Ritchie. Current directions in computational humour. *Artificial Intelligence Review*, 16(2):119–135, 2001.
- [Ritchie, 2002] G. Ritchie. The structure of forced interpretation jokes. In [Stock et al., 2002], 2002.
- [Ruch, 1996] W. Ruch. Special issue: Measurement approaches to the sense of humor. *Humor*, 9(3/4), 1996.
- [Ruch, 2002] W. Ruch. Computers with a personality? lessons to be learned from studies of the psychology of humor. In [Stock et al., 2002], 2002.
- [Stock et al., 2002] O. Stock, C. Strapparava, and A. Nijholt, editors. *Proceedings of the The April Fools Day Workshop on Computational Humour (TWLT20)*, Trento, 2002.
- [Utsumi, 1996] A. Utsumi. Implicit display theory of verbal irony: Towards a computational theory of irony. In [Hulstijn and Nijholt, 1996], 1996.

Automated Generation of Graphic Sketches by Example

Michelle X. Zhou

Min Chen

IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
Phone: (914)784-7000
{mzhou, minchen}@us.ibm.com

Abstract

Hand-crafting effective visual presentations is time-consuming and requires design skills. Here we present a case-based graphic sketch generation algorithm, which uses a database of existing graphic examples (cases) to automatically create a sketch of a presentation for a new user request. As the first case-based learning approach to graphics generation, our work offers three unique contributions. First, we augment a similarity metric with a set of adequacy evaluation criteria to retrieve a case that is most similar to the request and is also usable in sketch synthesis. To facilitate the retrieval of case fragments, we develop a systematic approach to case/request decomposition when a usable case cannot be found. Second, we improve case retrieval speed by organizing cases into hierarchical clusters based on their similarity distances and by using dynamically selected cluster representatives. Third, we develop a general case composition method to synthesize a new sketch from multiple retrieved cases. Furthermore, we have implemented our case-based sketch generation algorithm in a user-system cooperative graphics design system called IMPROVISE+, which helps users to generate creative and tailored presentations.

1 Introduction

Automated graphics generation systems promise to aid users in creating effective visual presentations (e.g., charts and diagrams) [Mackinlay, 1986; Zhou, 1999]. Upon a user request (e.g., displaying sales data), these systems directly provide users with the final presentation (e.g., a barchart). To better tailor a presentation to user preferences, we are building a user-system cooperative graphics generation system, called IMPROVISE+. IMPROVISE+ generates a presentation in two steps: sketch generation and sketch refinement. Here a *sketch* is an outline of a presentation without all visual details (Figure 1). A generated sketch is first presented to users for their feedback. Depending on the feedback, IMPRO-

WISE+ may redesign the sketch or refine it to create a final illustration (e.g., setting the exact layout of Figure 1). By allowing users to critique a sketch *first*, IMPROVISE+ can save the cost of fine-tuning an undesirable design.

Our focus here is on a case-based learning approach to sketch generation. Given a user request, from a database of existing graphic examples (cases) our approach uses a similarity metric to retrieve the case that is most similar to the request. The retrieved case is then directly reused for or adapted to the new situation (e.g., new data). Instead of using a rule-based approach as most graphics generation systems do [Mackinlay, 1986; Zhou, 1999], our decision of using case-based learning is two-fold. First, it is difficult to hand-extract a complete and consistent set of graphics design rules, while existing graphic examples are abundant [Zhou et al., 2002a]. Second, case-based learning is efficient for sketch generation, where we focus on learning overall visual structures instead of precise visual arrangements (e.g., exact positions and scales) ¹.

Although case-based learning has been applied to various design problems [Borner, 2001], it has never been applied to graphics design. As the first case-based learning

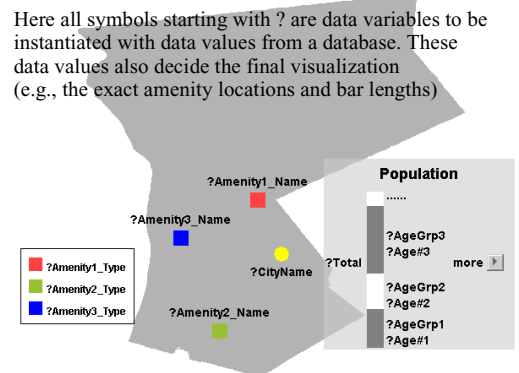


Figure 1. A sketch of a cartogram generated by IMPROVISE+.

¹ Due to the complexity in graphics rendering, it is impractical to learn the values of visual parameters, such as scales and positions, from existing cases. We compute the parameters during the sketch refinement stage.

approach to graphics design, our work offers three unique contributions. 1) In case retrieval, we apply a set of adequacy evaluation criteria in addition to a similarity metric to ensure that the retrieved case is usable in sketch synthesis. To handle situations where a usable case may not be found, we decompose cases/requests into sub-cases/sub-requests to facilitate the retrieval of case fragments. 2) We improve case retrieval speed by organizing cases into hierarchical clusters based on their similarity distances and by using dynamically selected cluster representatives. 3) We synthesize a new sketch from multiple retrieved cases through case generalization and visual decoration inference (e.g., inferring decorations such as coordinate axes and legends).

Starting with a brief discussion of related work, we then present our case-based learning algorithm, highlighting the three unique features mentioned above. Finally we use a concrete example to demonstrate how a new sketch is created.

2 Related Work

Unlike rule-based graphics systems [Mackinlay, 1986; Andre and Rist, 1993; Chuah et al., 1997; Zhou, 1999], our work is the first to apply a general case-based learning technique to graphics design. Although one system, SAGE, has an example-based generation component, it only reuses examples created by its own rule engine [Chuah et al., 1997]. Compared to SAGE, IMPROVISE+ uses a much more fine-grained representation to capture the semantic and syntactic features of each graphic example [Zhou et al., 2002b]. As a result, without using any rules IMPROVISE+ can create new graphic sketches by directly learning from a wide variety of graphic examples.

Differing from a programming by demonstration system [Myers et al., 1994], where users must supply the desired examples, our approach uses the graphic examples stored in a database.

There are many case-based systems developed for other domains, e.g., engineering design [Sycara et al., 1992]. The closest to ours are known as case-based structure generation systems [Borner, 2001]. However we have gone beyond existing approaches to address specific challenges in graphics design. In particular, we support a systematic, multi-level case/request decomposition to achieve a more accurate case-request matching. In contrast, existing systems either ignore case/request decomposition [Borner, 2001] or simplify it (e.g., only leaf-level decomposition [Michelen and Sycara, 1994]). We also develop a general case composition

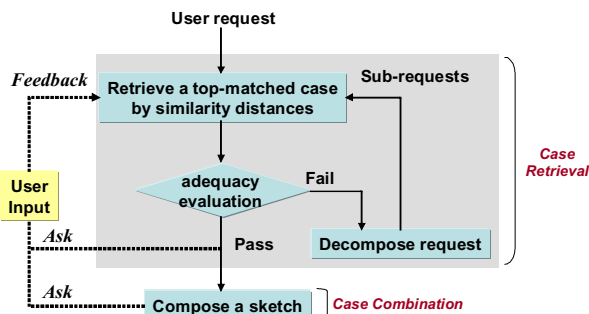


Figure 2. Flow diagram of our sketch generation algorithm.

method to synthesize a new sketch from multiple cases, while existing systems only allow limited case composition (e.g., combining only highly similar cases [Borner, 2001]).

3 Example-based Sketch Generation

Figure 2 gives an overview of our case-based sketch generation algorithm. Our algorithm uses a database of existing information graphics (cases) to suggest a visual design for a new user request. Here each stored case is described by its data content D and visual representation V . Each request is presented by specifying the data D' and its desired visualization V' , which may be partially or not specified at all. Given such a request, our algorithm first uses a similarity metric to retrieve the top-matched case by computing the similarity distances between the request and existing cases. A *top-matched case* is the case that has the shortest similarity distance to the request. If the top-matched case fails our adequacy test, the current request is decomposed into a set of sub-requests. The whole process is recursively called to find the top-matched case for each sub-request. Depending on the retrieval result, a new sketch (a fully specified V') may be constructed from the visual encoding (V) of a single matched case or composed from multiple cases. Upon completion, the user is presented with a rendered graphic sketch (e.g., Figure 1). We also involve users at different design stages (Figure 2). For example, our studies have shown that users may express their preferences to retrieve more desired cases [Zhou et al., 2002b] or propose new visual compositions during sketch synthesis.

Before discussing our algorithm, we first briefly describe the representations of our cases and user requests.

3.1 Representation of Cases and User Requests

We employ a labeled graph structure to represent our cases and requests. Since we have described in detail how to model and represent various semantic and syntactic features of an existing graphic example previously [Zhou et al., 2002b], we summarize the representation here.

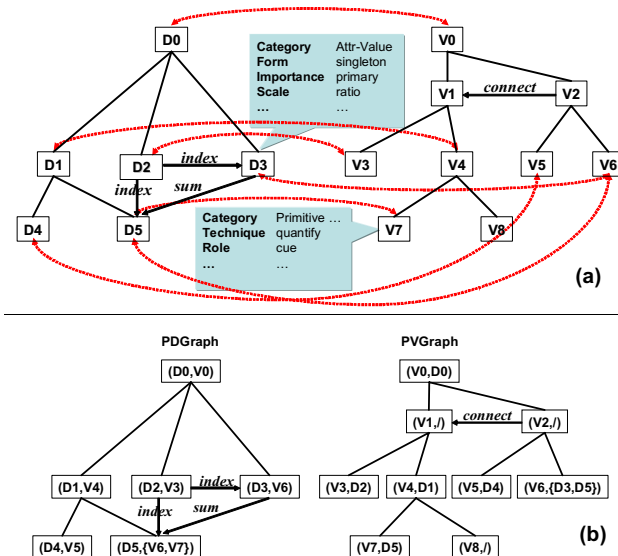


Figure 3. A labeled graph representation of a case.

Visual Database of Cases

Our case base contains an assortment of graphic designs collected from a wide variety of sources [Zhou et al., 2002a]. Using a labeled graph structure [Zhou et al., 2002b], each case is described in XML as a graph (Figure 3a), which expresses a complex mapping between a data hierarchy (D0–D5) and a visual hierarchy (V0–V8). Within each hierarchy, a data/visual node is described by a set of features (e.g., D3 and V7). There are two types of links in the graph: intra-hierarchy links for data/visual node relationships (e.g., D3 is indexed by D2 and all undirected edges implying parent-child relations), and inter-hierarchy links (red dotted lines) for data-visual mappings (e.g., D5 to V7). To facilitate case retrieval, we also index each case using two graphs (Figure 3b): PDGraph organizes data-visual mapping pairs along the data hierarchy, while PVGraph arranges the mapping pairs by the visual hierarchy. Note that a data/visual node may be mapped to multiple visual/data nodes (e.g., D5 and V6) or none (denoted by “/”). Our bi-graph indexing not only preserves all the information captured in the original complex graph (Figure 3a), but also partitions one complex graph into two simpler sub-graphs (Figure 3b), which can greatly simplify the similarity measuring process.

When loading all cases from the database to memory, IMPROVISE+ parses the XML document of each case and automatically builds all case indices (PDGraphs and PVGraphs).

User Request

A user request submitted in XML format is also described as a graph similar to the case representation. Figure 4 outlines a request for presenting the information of a city. Specifically, it asks to display the relevant county (CountyBoundary), basic information of the city (Name and Location), the city gazetteer (Population, AgeInfo for 6 age groups, and an arrow button indicating MoreInfo is available upon request), and 3 city amenities (Name, Location, and Type) such as community golf courses and swimming pools. Since users may not know every presentation aspect, the representation of a user request is often a partially specified graph with the majority of the visual elements left unspecified. For example, Figure 4 does not specify a visual encoding for any data nodes except MoreInfo.

Based on the characteristics of data relations, our algorithm automatically assigns *matching priorities* to different data relations in a user request to indicate that matching certain data relations well (e.g., index in Figure 5) is more important than matching others (e.g., has-a). As described below, matching priorities aid us in evaluating the usability of a retrieved case and in selecting visual candidates. Currently, we assign priorities by relation type. For example, a higher priority is given to presentation relations like index than to semantic relations like has-a.

3.2 Case Retrieval

The success of an interactive case-based system like ours, depends largely on the quality and speed of the case retrieval process. To ensure retrieval quality, we augment similarity measuring with *adequacy evaluation* that tests whether a retrieved case is usable in sketch synthesis. If a usable case cannot be found, we decompose cases and requests into sub-cases and sub-requests to facilitate the

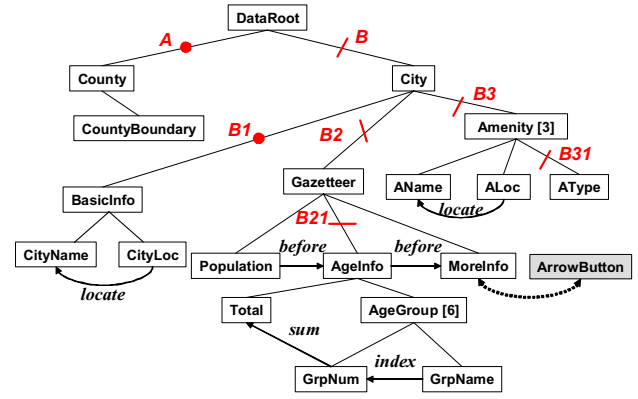


Figure 4. A sample user request.

retrieval of case fragments. In addition, we use a hierarchically structured case base to improve our retrieval speed.

Adequacy-Guided Retrieval

Using a quantitative similarity metric that we have developed [Zhou et al., 2002b], our algorithm first retrieves the top-matched case for a request. As a result, each data node and data relation in the request are associated with a computed similarity distance in the range of [0, 1]. Since our similarity model stresses the overall structural similarity between the request and the existing examples, a data node D' in the request may or may not acquire a match (D , V) from the top-matched case. Here D is the data that matches D' , and V (the visual mapping of D) is a potential *visual candidate* for encoding D' in the new sketch. If no match is found for D' , the recorded distance is 1.0. However, the top-matched case may be inadequate for creating an effective sketch for the following three reasons. Accordingly, we formulate three adequacy criteria to evaluate whether a retrieved case is usable in sketch synthesis.

First, a top-matched case may produce a good overall match but poor or no matches for certain data nodes in the request. This implies that certain data or a sub-set of data expressed in the request may not be well visualized in the new sketch as they could be. To ensure a good match for every data node (i.e., a short similarity distance), our first criterion requires that the similarity distance for every data node of a request be below a threshold. After conducting a series of case retrieval experiments, we currently set the threshold to 0.3 on a [0, 1] scale, which proves to be a good indicator for creating a quality new sketch.

Second, suppose that a top-matched case passes the above criterion. By the matching priorities set in the request, however the distances for more important data relations may be larger than those of less important ones. Since matching priorities are used to select visual candidates during case composition, using such a matching result may alter the original intention of the request and produce an undesired

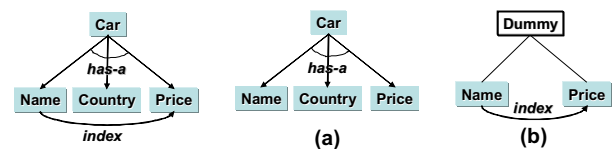


Figure 5. A decomposition by data relations.

sketch. Therefore our second criterion states that for every data relation in a request, the higher its matching priority is, the shorter its associated similarity distance must be.

Third, even though a top-matched case passes the above two criteria, it may still not be adequate for synthesizing a new sketch. Our synthesis starts with the visual candidates of data leaf nodes in a request and composes the higher-level visual nodes from bottom up using the lower-level visual nodes. However our top-down graph matching is a partial matching, which may not guarantee that every data leaf node in a request acquire a visual candidate [Zhou et al., 2002b]. Our synthesis would then fail due to a lack of basic building blocks. Thus our third criterion requires that every data leaf node in a request obtain a visual candidate. It is not required to find visual candidates for intermediate data nodes from case retrieval, since theirs can be composed from those of their children.

Case and Request Decomposition

It is rare to find an exact match for a user request from the case base, but it is quite common that a fragment of the user request matches well with a fragment of a case [Mitchell, 1997]. To facilitate the retrieval of case fragments, we support case/request decomposition.

Case decomposition. To avoid creating a large search space, our challenge is to determine the granularity of sub-cases. Based on the data and visual characteristics of a graphic example [Zhou et al., 2002b], we develop four heuristics to guide our case decomposition.

1). We extract independent and meaningful *visual structures*, which are schematic illustrations for conveying concepts and processes [Winn, 1987]. For example, the table showing the city information (Figure 6a) is considered as a sub-case. Within a case, such a structure can be easily identified by the value (VisualStructure) of its feature Category.

2). We turn a case (e.g., Figure 6b) into a sub-case by trimming all its decorations (e.g., coordinate axes and legends). These simplified sub-cases are good matching candidates for most user requests, which normally do not specify the data for creating visual decorations.

3). We extract all visual leaf objects along with their data mappings from cases (e.g., V7 and D5 in Figure 3b) to form a visual dictionary [Zhou and Feiner, 1997]. The dictionary is used to find matches for user requests that contain a single node.

4). We decompose a case by data relations (has-a and index) encoded in Figure 6(b), which lead to two sub-cases (Figure 5a–b). In Figure 5(b), a Dummy node is also added to preserve a rooted graph

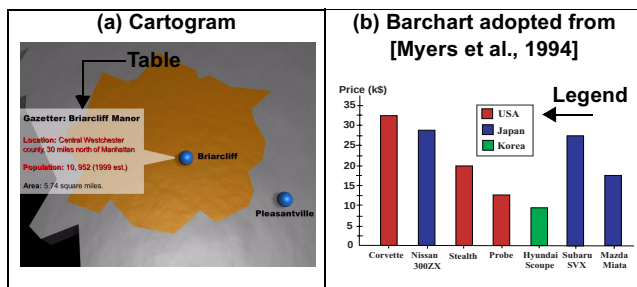


Figure 6. Annotated decomposable graphic examples.

structure for a fast graph matching (see below).

For the sake of performance, we automatically extract all sub-cases during the case loading stage.

Request decomposition. Unlike case decomposition, which is done once during the case loading stage, request decomposition occurs whenever a retrieved top-matched case fails our adequacy evaluation. To ensure the retrieval quality without incurring the expense of rematching, we extract only the failed fragments as sub-requests, while retaining the results for succeeded ones. Here a *failed fragment* is the biggest possible rooted sub-graph that contains failed nodes excluding the root of the current request. Suppose that every node in fragment A (Figure 4) passes our evaluation, but a node (e.g., AName) in fragment B fails. We then create a sub-request containing the whole fragment B with City as the root, while keeping the matching results of fragment A. Depending on the matching results, the decomposition may be repeated until sub-requests contain a single node. The matched cases for these single-node requests can be found from our visual dictionary.

Since sub-requests break up the original structure of a request, it is always desirable *not* to produce too many sub-requests. Request decomposition however enables us to find desired case fragments from a set of heterogeneous cases as we have, where finding a maximal common subgraph may fail [Borner, 2001]. Unlike a static request decomposition used by other researchers [Michelena and Sycara, 1994], our decomposition is dynamically performed based on our adequacy evaluation results.

Improve Retrieval Speed

As the number of cases grows, the cost of searching for a desired case increases. Moreover, finding a graphic example that matches a user request best is a computationally intensive graph-matching process itself. Specifically, we need to match two pairs of PDGraphs and PVGraphs between a case and a user request. By exploiting the rooted hierarchical structure of PD/PVGraph, we perform an ordered, top-town graph matching [Zhou et al., 2002b] to accelerate otherwise an arbitrary graph matching process (e.g., matching two complex graphs similar to Figure 3a). To further improve case retrieval speed, we reduce the search space by using a hierarchically structured case base.

Instead of searching the case base linearly, we exploit the organization of the cases. We use a hierarchical clustering algorithm [Duda and Hart, 1973] to arrange all cases by their pair-wise distances computed using a similarity metric [Zhou et al., 2002b]. Figure 7 shows such a cluster hierarchy. Starting with the two outmost clusters in the hierarchy (e.g., clusters 1–2 in Figure 7), at each level we search only one

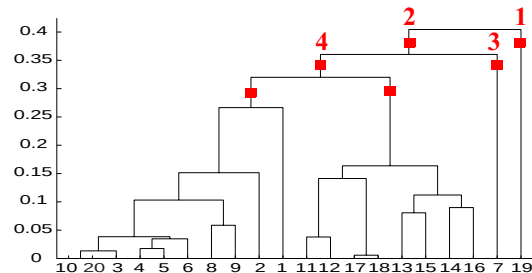


Figure 7. A cluster hierarchy of 20 cases.

cluster that is most likely to contain the top-matched case.

To find a cluster to follow, we first select a representative for each cluster using a quick approximation. A *representative* is a case closest to the request by three meta properties of its PDGraph: the total number of nodes, the average number of levels, and the average number of nodes at each level. We then select the cluster whose representative produces the shortest similarity distance to the request by our similarity metric. The rationale here is that the top-matched case for a request is *likely* to be in the same cluster of cases that match the request well by both meta properties and our similarity metric.

Following the selected cluster (e.g., cluster 2 in Figure 7) down the hierarchy (e.g., clusters 3–4), our algorithm repeats the above process until it explores a leaf cluster. In our experiments, this cluster-guided search improves the performance over a linear search by a factor of 3. Due to the approximation used, our method however is not guaranteed to find the top-matched case. Unlike other structured case search, where cluster representatives are pre-selected in advance [Borner, 2001], we dynamically compute representatives for each request to achieve a more accurate retrieval.

3.3 Case Composition: Sketch Synthesis

As the result of a successful retrieval, each data leaf node of the request acquires at least one visual candidate. Starting with the visual candidates of the leaves, our algorithm synthesizes a sketch from bottom up by creating visual candidates for higher-level data nodes and finally for the root. In this section, we address three challenges arising in sketch synthesis. First, we resolve visual candidate conflicts when a data node in the request acquires multiple candidates. Second, to compose multiple retrieved cases, we use decision-tree learning to generalize existing visual compositions and to verify new compositions. Third, we automatically infer visual decorations from existing cases, such as the coordinate axes and legends, to complete a visual sketch design.

Conflict Resolution of Visual Candidates

Within a user request, a data node may acquire multiple visual candidates from different case matching. For example, data nodes Name and Price appear in two sub-requests (Figure 5a–b). It is most likely for both items to obtain multiple visual candidates as the two sub-requests are matched to different cases. To select a proper visual candidate, we currently use both the matching priority set in the user request and the distance calculated during similarity measuring. Specifically, we first retain candidates that are acquired through matching the data relations with a higher priority. In the above example, we will keep the candidates for Name and Price acquired through matching the sub-request in Figure 5(b), since the priority for matching relation index precedes that of relation has-a (Figure 5a). If there are multiple candidates by the same matching priority, we then select candidates that have produced the shortest distance during the match. Should there still be multiple candidates left, our algorithm would choose a candidate randomly.

Generalizing and Verifying Visual Composition

Our sketch synthesis uses visual candidates acquired for the lower-level data nodes to create visual candidates for the

higher-level nodes in the request. Since visual candidates may be retrieved from different cases, their compositions may never exist before and new compositions are needed to piece them together. Here we denote a visual composition ($\Rightarrow$) of N visual elements as:

$$E_1, \dots, E_N \Rightarrow rC,$$

where E_i is the i th element, r is the composition relation, and C is the composed visual object. Below is an example of a cartogram (a map-based presentation) composition:

$$\text{Position, Map} \Rightarrow \text{overlay_Cartogram}.$$

It composes a Cartogram by overlaying a Position element on top of a Map element. In our approach, a new composition for an intended data node (e.g., Amenity in Figure 4) is proposed using the composition information recorded in its children (e.g., AName) during the case retrieval. Specifically, in each data node that acquires a visual candidate V from the retrieved case (e.g., V7 from Figure 3a), we record the composition relation and category specified in V 's parent (e.g., V4). As a result, for each retrieved visual candidate our algorithm records the possible compositions that the visual candidate has participated in.

However a proposed composition may not always be valid. Here *validity* means that a visual composition must produce an effective visual design [Mackinlay, 1986]. For example, the above cartogram composition is a valid composition, but composing a visual object by juxtaposing two horizontal position elements is not. Without hand-crafting rules or pre-defining connectors [Michelena and Sycara, 1994], we use a decision-tree learning technique to automatically induce a set of classification rules from 200 visual composition samples extracted from our cases. By our visual composition definition, we describe each sample using N input features specifying N visual element categories (e.g., Position and Map) and one target specifying a combined composition relation and category (e.g., overlay_Cartogram). Currently N is set to 3 since most of our samples contain 3 elements. If a sample contains fewer than 3 elements, the remaining features are set to null. We train C4.5 [Quinlan, 1993], a decision-tree learning algorithm, on the 200 samples using 5-fold cross-validation, a standard procedure when the amount of data is limited. We obtain 14 generalized composition rules with an overall classification error of 22%. We then use these rules to verify a new composition.

To compensate for the situation where there is a lack of visual composition samples, we introduce *negative samples* that are known invalid compositions to help identify *invalid* new compositions. Similar to the above process, we use classification rules induced from the negative samples to verify invalid new compositions to be eliminated. After both positive and negative verifications, if there are still multiple proposed compositions we use the confidence factors generated in the decision-tree learning to select the most probably valid composition. Without a sufficient number of composition samples, our approach may not always verify a composition correctly. That is why we involve users in the design process to help IMPROVISE+ in its decision-making (Figure 2).

Inferring Visual Decoration

A sketch is incomplete without the necessary *visual decorations*, such as coordinate axes and legends, which can guide users to comprehend the information encoded in the

graphic [Wilkinson, 1999]. However in a request a user rarely specifies the data for creating such decorations, which our algorithm must infer.

Currently our inference is based on an assumption that visual decorations can be created using only data leaves in a request. According to visual psychology studies, a person tends to first perceive the global structure of a graphic (e.g., a spatial map) then examine the details (a particular color or a position) [Goldsmith, 1984]. Only when a person attempts to interpret the meanings of visual primitives (e.g., the color code in Figure 6b), visual decorations are needed (e.g., the legend in Figure 6b). Hence visual decorations normally encode leaf data or their transformations.

Specifically, for each leaf L and its match (D, V) from a retrieved case C , we trace the visual mappings of the matched data node D in C . Within case C , if data D is used to create visual decorations, our algorithm would create a visual decoration for L in the user request. More precisely, if there exists a data-visual mapping pair (D', V') in case C , where D' is D or a transformation of D , and V' is a visual decoration; V' becomes a visual decoration candidate for the new sketch. However, L may need to be transformed for creating V' , since D' is often a transformation of D . Suppose that a user asks to present the depth of different lakes (LakeDepth), which happens to acquire a match (Price, VertLength) from a retrieved case as in Figure 6(b). Here Price (a *domain*) has been transformed into a *range* [minPrice, maxPrice] in creating the Y axis. Similarly, in the new sketch a Y axis will be created for encoding the range of LakeDepth converted from the LakeDepth domain.

4 An Example

Here we demonstrate how a specific user request (Figure 4) is fulfilled. For this request, our algorithm first retrieves the top-matched case, which fails our adequacy test, as the similarity distance (0.5803) for node Gazetteer in fragment B is above the required threshold (0.3). While keeping the matches for fragment A, we extract the whole fragment B as a sub-request with node City as its root, since we always attempt to find an overall good match for the bigger sub-graph (e.g., the sub-graph covering B instead of B2). Next, the top-matched case for fragment B is retrieved, where

nodes in B1 pass the evaluation, but those in B2–B3 do not. Subsequently, our algorithm treats fragments B2–B3 as two new sub-requests and finds matches for them. Except nodes under fragments B21 and B31, all other data nodes have now acquired acceptable matches. Eventually desired matches are found for the last two sub-requests generated for fragments B21 and B31. Since B31 contains only one node, its match is retrieved from our visual dictionary.

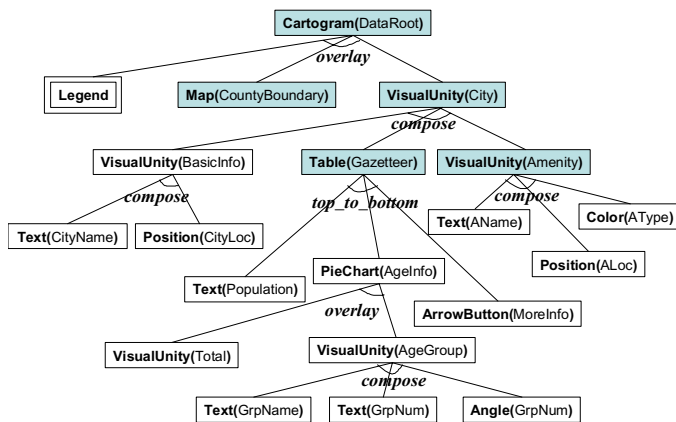
Using the retrieved results, our algorithm synthesizes a new sketch from bottom up (Figure 8a). It proposes new compositions to create visual candidates for shaded data nodes, while reusing retrieved candidates for others. Specifically, a new composition is proposed for Gazetteer, since the visual candidates for its children Population, AgeInfo, and MoreInfo, are retrieved from different cases. In this case, a top-to-bottom composition is recorded for Population during the retrieval. According to C4.5, this is a valid composition, thus it is used to create a Table for conveying Gazetteer with 3 items arranged from top to bottom: a text, a pie chart, and an arrow button. Similarly, visual candidates are created for Amenity, then for City, at last for the root. As a result, a new sketch is created for this request with an inferred legend for interpreting the type of amenities (Figure 8b).

If the user chooses to use a different set of cases, an alternative can be created for the same request (Figure 1). Once a new sketch is created, it can be directly stored back in our case base as a new addition, since the generated sketch has a representation similar to other existing cases.

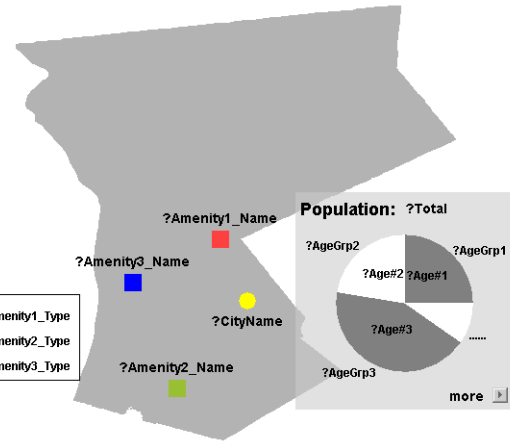
5 System Implementation and Performance

IMPROVISE+ is implemented using Java and C++. We use Java to implement all design components, including our case-based sketch generation algorithm. We have implemented two rendering components, one in Java2D for 2D graphics and the other in C++/Open Inventor/OpenGL for 3D graphics. IMPROVISE+ currently runs on Win32/Linux/SGL. On a PC with a 1.13 GHz Pentium III processor, it takes about 5 seconds to create the sketch shown in Figure 8(b) from about 100 cases, each of which contains about 30 data nodes and 30 visual nodes on average.

We have also conducted several experiments to test the effectiveness of our approach. According to our user feed-



(a) Graph structure of a composed sketch



(b) Rendered graphic sketch

Figure 8. A composed sketch and its rendering for the user request in Figure 5.

back, our case-based approach can provide more versatile and “creative” design suggestions (e.g., Figure 1) than a rule-based approach can [Zhou, 1999]. Moreover, involving users in proper design stages helps create a more tailored visual presentation. For example, by specifying different retrieval preferences [Zhou et al., 2002b], users can choose among different design alternatives (Figure 1 vs. Figure 8b).

6 Conclusions and Future Work

In this paper, we have presented a case-based graphic sketch generation algorithm with an emphasis on its three unique features and how they facilitate an efficient and effective sketch generation. First, we present an adequacy-guided case retrieval method, which augments a similarity metric with a set of adequacy evaluation criteria to retrieve a top-matched case that is also usable in sketch synthesis. To facilitate the retrieval of case fragments, we also describe how to systematically decompose a case/request when a usable case cannot be found. Second, we explain how to enhance case retrieval speed by organizing cases into hierarchical clusters based on their similarity distances and by using dynamically selected cluster representatives. Third, we show how to construct a new sketch through case composition, including case generalization and visual decoration inference (e.g., inferring coordinate axes and legends).

Currently, we are working in two areas to improve IMPROVISE+. First, we are creating a GUI that allows users to easily specify a request by “drawing” a graph similar to Figure 4 without writing an XML document. Second, we are developing more sophisticated interaction support, where IMPROVISE+ can automatically engage users in interaction based on context. For example, if IMPROVISE+ cannot find a valid visual composition using decision-tree learning, it may decide to solicit user inputs.

Acknowledgments

We would like to thank Keith Houck and Alison Lee for proofreading the earlier version of this paper. We would also like to thank Shimei Pan for useful discussions on case-based learning.

References

- Andre, E. and Rist, T. (1993). The design of illustrated documents as a planning task. In Maybury, M., editor, *Intelligent Multimedia Interfaces*, chapter 4, pages 94–116. AAAI Press/The MIT Press, Menlo Park, CA.
- Borner, K. (2001). Efficient case-based structure generation for design support. *AI Review*, 16(2):87–118.
- Chuah, M., Roth, S., and Kerpedjiev, S. (1997). Sketching, searching, and customizing visualizations: A content-based approach to design retrieval. In Maybury, M., editor, *Intelligent Multimedia Information Retrieval*, pages 83–111. AAAI Press/The MIT Press.
- Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. Wiley.
- Goldsmith, E. (1984). *Research into Illustration: An Approach and a Review*. Cambridge University Press, Cambridge, England.
- Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Trans. on Graphics*, 5(2):110–141.
- Michelena, N. and Sycara, K. (1994). Physical synthesis in case-based design. In *Proc. of ASME DTM '94*, pages 273–284.
- Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
- Myers, B., Goldstein, J., and Goldberg, M. (1994). Creating charts by demonstration. In *CHI '94*, pages 106–111.
- Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufman.
- Sycara, K., Guttal, R., Koning, J., Narasimhan, S., and Chandra, D. (1992). Cadet: A case-based synthesis tool for engineering design. *Intl. J. of Expert Systems*, 4:157–188.
- Wilkinson, L. (1999). *The Grammar of Graphics*. Springer.
- Winn, W. (1987). Charts, graphs, and diagrams in educational materials. In Willows, D. and Houghton, H., editors, *The Psychology of Illustration: Basic Research*, volume 1, chapter 5, pages 152–198. Springer-Verlag.
- Zhou, M. (1999). Visual planning: A practical approach to automated visual presentation. In *Proc. IJCAI '99*, pages 634–641.
- Zhou, M., Chen, M., and Feng, Y. (2002a). Building a visual database for example-based graphics generation. In *Proc. IEEE InfoVis '02*, pages 23–30.
- Zhou, M. and Feiner, S. (1997). The representation and use of a visual lexicon for automated graphics generation. In *Proc. IJCAI '97*, pages 1056–1062, Nagoya, Japan.
- Zhou, M., Ma, S., and Feng, Y. (2002b). Applying machine learning to automated information graphics generation. *IBM Sys. J.*, 41(3):504–523.

