

The 19th European Conference on Artificial Intelligence

Thomas Roth-Berghofer, Nava Tintarev, David B. Leake, Daniel Bahls (Eds.)

Explanation-aware Computing ExaCt 2010

ECAI 2010 Workshop, University of Lisbon, Portugal,
August 16, 2010
Proceedings of the 5th International ExaCt workshop

Preface

Both within AI systems and in interactive systems, the ability to explain reasoning processes and results can substantially affect system usability. For example, in recommender systems good explanations may help to inspire user trust and loyalty, increase satisfaction, make it quicker and easier for users to find what they want, and persuade them to try or buy a recommended item.

Current interest in mixed-initiative systems provides a new context in which explanation issues may play a crucial role. When knowledge-based systems are partners in an interactive socio-technical process, with incomplete and changing problem descriptions, communication between human and software systems is a central part. Explanations exchanged between human agents and software agents may play an important role in mixed-initiative problem solving.

Other disciplines such as cognitive science, linguistics, philosophy of science, psychology, and education have investigated explanation as well. They consider varying aspects, making it clear that there are many different views of the nature of explanation and facets of explanation to explore. Within the field of knowledge-based systems, explanations have been considered as an important link between humans and machines. There, their main purpose has been to increase the confidence of the user in the system's result, by providing evidence of how it was derived. Additional AI research has focused on how computer systems can themselves use explanations, for example to guide learning.

This volume contains the papers presented at the ECAI 2010 workshop on Explanation-aware Computing (ExaCt2010) held on August 16, 2010, in Lisbon, Portugal. The major goal of the workshop was to bring researchers, scientists from both industry and academia, and representatives from such different communities and fields as Informatics, Philosophy, and Sociology, together to study, understand, and explore the aspects of explanation in IT-applications. The papers presented in this volume illustrate some of the variety of perspectives on explanation and recent progress.

Atzmüller and Roth-Berghofer present a variety of dimensions in the Mining and Analysis Continuum of Explaining (MACE) that need to be considered when building datamining applications, and discuss the application of the MACE in the context of social software.

Forcher and Roth-Berghofer extend their approach on Explanation-Aware Software Design (EASD), which generally aims at making software systems smarter in interactions with their users. In their paper they present an abstract model for explanation generation and describe the realisation of that approach for the semantic search engine Koios using Semantic Web technology.

Kapetanakis et al. develop an approach for providing explanation to the intelligent diagnosis and monitoring of business workflows based on operation data in the form of temporal log data.

Možina et al. use Argument-Based Machine Learning (ABML) to address the problem that the justifications provided by machine learning algorithms that learn simple symbolic models—which are often seen as an advantage of those approaches—are frequently hard to understand.

Finally, Sonntag and Theobald present a generic dialogue shell that can automatically infer previously unknown knowledge (facts) and provide explanations for the inference steps involved. The system also generates explanation graphs, which can be displayed in the GUI of the dialogue shell and help the user understand the underlying reasoning processes.

There were 7 submissions to ExaCt 2010. Each submission was reviewed by at least three programme committee members, and 5 papers were accepted. This volume was produced using the EasyChair system¹.

ExaCt 2010 continued a series of workshops begun with a AAAI Fall symposium in 2005², a AAAI-07 workshop³, an ECAI-08 workshop⁴, and an IJCAI-09 workshop⁵. The workshop series aims to draw on multiple perspectives on explanation, to examine how explanation can be applied to further the development of robust and dependable systems and to illuminate system processes to increase user acceptance and feeling of control. The presentations and discussions at this workshop have shown the value of sharing perspectives to clarify and advance research in explanation-aware computing.

Readers who would like to participate in further discussions on this topic or like to receive further information about future workshops might consider joining the Yahoo!-group `explanation-research`⁶. More information on explanation research is available at the website `on-explanation.net`⁷.

August 2010

Thomas Roth-Berghofer
Nava Tintarev
David B. Leake
Daniel Bahls

Acknowledgements

ExaCt 2010 was partially funded by EPSRC grant EP/E011764/1 and the UK Resource Centre for Women in Science Engineering and Technology (UKRC).

¹ <http://www.easychair.org>

² <http://exact2005.workshop.hm>

³ <http://exact2007.workshop.hm>

⁴ <http://exact2008.workshop.hm>

⁵ <http://exact2009.workshop.hm>

⁶ <http://tech.groups.yahoo.com/group/explanation-research/>

⁷ <http://on-explanation.net>

Workshop Organisation

Chairs

Thomas Roth-Berghofer, TU Kaiserslautern / DFKI GmbH, Germany
Nava Tintarev, University of Aberdeen, United Kingdom
David B. Leake, Indiana University, USA
Daniel Bahls, MIMOS BERHAD, Kuala Lumpur, Malaysia

Programme Committee

Agnar Aamodt, Norwegian University of Science and Technology (NTNU)
David W. Aha, Navy Center for Applied Research in AI, Washington DC, USA
Martin Atzmüller, University of Würzburg, Germany
Patrick Brézillon, LIP6, France
Jörg Cassens, University of Lübeck, Germany
Li Chen, Hong Kong Baptist University
Babak Esfandiari, Carleton University, Canada
Ashok Goel, Georgia Tech University, Atlanta, GA, USA
Pierre Grenon, KMI, The Open University, UK
Anders Kofod-Petersen, SINTEF, Norway
Deborah L. McGuinness, Rensselaer Polytechnic Institute, Troy, NY, USA
Hector Muñoz-Avila, Lehigh University, USA
Paulo Pinheiro da Silva, University of Texas, El Paso, USA
Enric Plaza, IIIA-CSIC, Spain
Michael M. Richter, University of Calgary, Canada
Christophe Roche, University of Savoie, France
Sven Schwarz, DFKI GmbH, Germany
Gheorghe Tecuci, George Mason University, Fairfax, VA, USA
Douglas Walton, University of Windsor, Canada

External Reviewer

Gunnar Aastrand Grimnes, DFKI GmbH, Germany

Table of Contents

Towards Applying the MACE in Social Software	1
<i>Martin Atzmueller, Thomas Roth-Berghofer</i>	
Explanation-Aware System Design of the Semantic Search Engine Koios .	13
<i>Björn Forcher, Thomas Roth-Berghofer</i>	
Providing Explanations for the Intelligent Monitoring of Business Workflows Using Case-Based Reasoning	25
<i>Stelios Kapetanakis, Miltos Petridis, J Ma, Liz Bacon</i>	
Learning to Explain with ABML	37
<i>Martin Mozina, Matej Guid, Jana Krivec, Aleksander Sadikov, Ivan Bratko</i>	
Explanations in Dialogue Systems through Uncertain RDF Knowledge Bases	49
<i>Daniel Sonntag, Martin Theobald</i>	

Towards Explanation-Aware Social Software: Applying the Mining and Analysis Continuum of Explaining

Martin Atzmueller¹ and Thomas Roth-Berghofer^{2,3}

¹ University of Kassel,
Knowledge and Data Engineering Group
Wilhelmshöher Allee 73, 34121 Kassel, Germany

² Knowledge Management Department,
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Straße 122, 67663 Kaiserslautern, Germany

³ Knowledge-Based Systems Group, Department of Computer Science,
University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern

atzmueller@cs.uni-kassel.de, thomas.roth-berghofer@dfki.de

Abstract. Data mining methods build patterns or models. When presenting these, all or part of the result needs to be explained to the user in order to be understandable and for increasing the user acceptance of the patterns. In doing that, a variety of dimensions in the Mining and Analysis Continuum of Explaining (MACE) needs to be considered, e.g., from concrete to more abstract explanations. This paper discusses the application of the MACE in the context of social software. We consider applications of the proposed approaches in three social software systems, and show how the data mining results can seamlessly be analysed on the presented continuous dimensions and levels.

1 Introduction

Social software systems, e.g., advanced social resource sharing systems such as *Bibsonomy*⁴, or popular social networking applications such as *LinkedIn*⁵ and *Facebook*⁶ are in wide-spread use. Additionally, more and more ubiquitous systems include the user in a more comprehensive way, e.g., [2]. During the utilisation of these systems, a lot of data is acquired that can be applied for improving the user experience of the systems. Interest-driven recommendations, for example, intelligent notifications, or improved ranking of resources are common application areas. For those, data mining methods are usually applied. In order to enable a transparent and trust-enabled approach, explanation capabilities provide a viable supplement: Concerning actions of the system, e.g., recommendation, notification, or ranking, often explanations for these actions, relating to the used models, patterns, or parts thereof are requested. If these cannot be fulfilled, acceptance of the actions and the trust in the system is hard to achieve. Appropriate

⁴ www.bibsonomy.org

⁵ www.linkedin.com

⁶ www.facebook.com

explanation techniques in data mining and analysis are therefore crucial for an effective data mining approach. This is especially relevant for *semantic data mining* and related approaches [4, 19], where background knowledge incorporated by these provides further explanation capabilities, and enables in turn an iterative refinement of the respective explanation knowledge base. Also, appropriate privacy-preserving measures accompanied by suitable explanation techniques can significantly increase the trust in social systems applying data mining techniques. In [5], we present the **Mining and Analysis Continuum of Explaining (MACE)** providing several explanation dimensions and levels in the context of data mining and analysis. This paper focuses on the application of the MACE in the context of social software, that is, requirements and exemplary questions that can be tackled with corner principles of the MACE. We describe three application scenarios, and discuss the elements of the MACE in this context.

The rest of the paper is structured as follows: Section 2 briefly introduces explanation-aware software design and computing. After that, Section 3 outlines the MACE, including explanation-aware mining and analysis, general explanation goals and kinds, and the other elements of the continuum. Section 4 presents three application scenarios of the new approach in ubiquitous and social environments. Finally, Section 5 concludes the paper with a summary and discusses further interesting options for future research.

2 Explanation-Aware Software Design and Computing

Software systems need the ability to explain reasoning processes and their results as those abilities substantially affect their usability and acceptance. Explanation-aware computing (ExaCt) is the vision of software systems being smart in interactions with their users and Explanation-aware Software Design (EASD) aims at making software systems smarter in this regard. EASD looks at ways to guide software designers and engineers to a purposeful explanation-aware software system by making their designers and engineers explanation-aware. The long-term goal is to provide the respective methods and tools for engineering and improving such explanation capabilities. Here we focus on bringing explanation-awareness techniques to data mining.

The term explanation has been widely investigated in different disciplines such as cognitive science, artificial intelligence, linguistics, philosophy of science, and teaching. All these disciplines consider certain aspects of the term and make clear that there is not only one such concept but a variety of concepts. Explanations are in some sense always answers to questions, may the questions be raised explicitly or not. Explanations are an important vehicle to convey information to understand one another in everyday conversations. They support humans in their decision-making [14].

In a general explanation scenario (Figure 1) we distinguish three main participants [13]: the *user* who is corresponding with the software system via its user interface (UI), the *originator*, i.e., the problem solver or ‘reasoning’ component, which provides the functionality for the original task of the software, and the *explainer*. Both, originator and explainer, have their own knowledge container to support their tasks as indicated by the two database symbols. EASD models important aspects for understanding the originator and its application domain. Originator and explainer need to be tightly coupled to help the explainer provide knowledge about the inner workings of the originator.

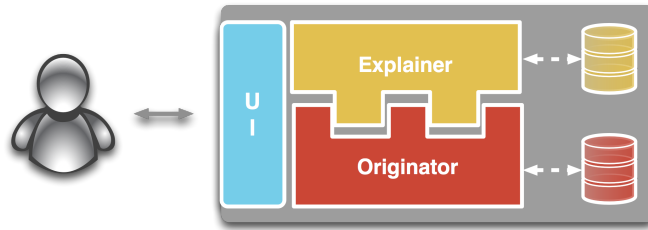


Fig. 1. Communication participants in general explanation scenario [13]

3 The MACE

The mining and analysis continuum of explaining (MACE) provides different perspectives on the same problem. It considers

1. Different explanation goals
2. Different kinds of explanation
3. Modes of presentation
4. Level of detail of explanation: concrete vs. abstract
5. Utilisation of different knowledge container.
6. Privacy: Which data/information or knowledge from the different knowledge containers is actually revealed to the user?

In the following, we first describe the data mining foundations of the MACE in some detail, before we outline its explanation dimensions, and discuss the issues involved.

3.1 Explanation-Aware Mining and Analysis

In the mining process, explanation features are involved before, during, and after the respective data mining main step, i.e., the *modelling* step. Therefore, we take a broad view regarding the data mining system as the originator, and provide explanation capabilities for each of the datamining steps. In short, the involved mechanisms can be described as follows: The input of the system is given by a (descriptive) specification of the process, the (source) data, and optional background knowledge. The system output is given by a data mining model, e.g., a set of patterns. The output is then accompanied by a “description” of the elementary mining steps, i.e., traces and/or logs of the respective events and steps of the process. The output can then be explained in terms of the input data, additional background knowledge and the intermediate results (trace). Additionally, setting up the specification itself is often a difficult task, for which appropriate explanation features are crucial.

3.2 Explanation Continuum

In the following, we discuss the different dimensions of explanation in the mining and analysis context (Fig. 2), beginning with goals and kinds of explanation, which are

immediately useful classifications and a good starting point for explanation-aware application development [12]. In designing a software system knowing about goals and kinds of explanations helps with structuring available knowledge and deciding which knowledge further is required for exhibiting certain explanation capabilities.

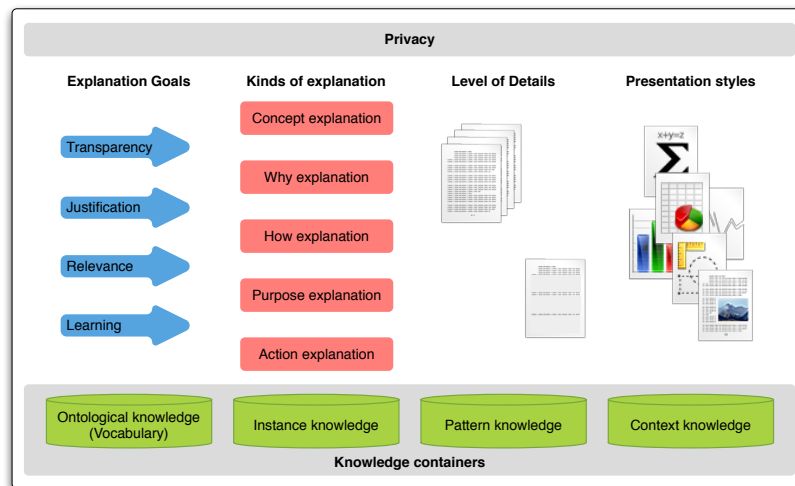


Fig. 2. Overview of the explanation dimensions

Kinds of explanation There are several useful kinds of explanations for knowledge-based systems, i.e., *concept*, *purpose*, *why*, *how*, and *action explanations* [18]. Concept explanations map concepts unknown to the user to already known concepts. The goal of purpose explanations is to describe the purpose of a fact or object. Why explanations justify a fact or an event. How explanations, a special case of why explanations, describe processes in general that lead to an event by providing a causal chain. They provide information about the function of a device. Action explanations explain or predict the behaviour of ‘intelligent systems’ on the basis of ‘known’ goals, beliefs, constraints, and rationality assumptions built into the system. Action explanations also answer how questions, but in a concrete context.

The user and/or application goals relate mainly to the type of explanation. During data mining, a data-driven approach starts with the (intermediate/final) results of the mining step. Then, reconstructive explanations [21] are provided by analysing the trace of the system – for why, how, purpose and action explanations. Concept explanations do not necessarily need the analysis of the trace, if they can directly apply the ontological knowledge. Helpful explanation is also provided by adaptation suggestions - comparing the current specification to stored experiences.

Explanation goals help software designers focus on user needs and expectations towards explanations and help to understand what the system has to be able to explain and when to explain something. Sørmo et al. [16, 17] suggest a respective set of explanation goals, i.e., *transparency*, *justification*, *relevance*, and *learning*.

The transparency goal aims at imparting an understanding of how a system found an answer, allowing users to examine the way the system reasons and allows them to look for explanations for why the system has reached a surprising or anomalous result.

The justification goal aims at increasing confidence in the advice or solution offered by a system by giving some kind of support for the conclusion suggested by the system. This goal allows for a simplification of the explanation compared to the actual process the system goes through to find a solution.

An explanation adhering to the relevance goal justifies the strategy pursued by the respective system. This is in contrast to the previous two goals that focus on the solution. The reasoning trace type of explanations may display the strategy of the system implicitly, but it does not argue why it is a good strategy.

Users do not always understand the terms used by a system, especially when starting to use a system. This may be because the user is a novice in the domain, but also because different people can use terms differently or organise knowledge in different ways. It may not be clear, even to an expert, what the system means when using a specific term, and he or she may want to get an explanation of what the system means when using it.

Contextual explanation provides information about the context of the results with respect to already known knowledge, e.g., considering deviations, or contradictions. The transparency of the results can be significantly increased using contextual, why, how, or purpose explanations. The explainer should then show connections between the knowledge containers involved and increase the validity/acceptance of the results using contextual/concept explanation. Also, explanation can be used for improving the modelling of the data mining step, e.g., for documenting and justifying design decisions.

Detail The *detail dimension* concerns the level of detail of the explanation session. Then, the explanation session can make use of various detail abstractions that mostly concern aggregation levels of the data. Abstract explanations, for example, can be enabled using high-level concepts contained in the ontological knowledge container, while finer-grained explanations are provided by applying elements of the low-level presentation dimensions, for example, using textual documents contained in the support knowledge container. Unstructured information can be seen as a very basic form of low-level but very specific and helpful information, consider medical discharge letters or electronic patient records, for example.

Presentation The *presentation dimension* of explaining, i.e., the explanation session information, needs to be performed in an appropriate way, e.g., using textual information, aggregation such as tables or visualisations for more aggregation and abstraction. The design issues involved here are also strongly connected to the *detail dimension*, since the level of detail needs to be reflected by the presentation options and the presentation modes need to be compatible with the detail level.

In the continuum, the presentation dimension provides seamless drill-down/roll-up capabilities similar to OLAP [9] techniques connected with the detail dimension. For instance, information can first be presented using diagrams, then a chunk of knowledge can be inspected using (aggregated) tables, and individual cells can then be analysed on the (input) data, e.g., textual data, or other tables or rows/columns of these.

Containers In [12], Roth-Berghofer and Cassens outline the combination of explanation goals and kinds of explanations in the context of Case-Based Reasoning, and examine the contribution of the four CBR *knowledge containers* for modelling the necessary knowledge [8, 11]. Casting this idea on the field of data mining, the MACE distinguishes the following containers that include explicit knowledge for explaining:

- The *ontological knowledge container* provides *vocabulary*, i.e., the applied concepts of the application context. In the data mining context, this relates to the used attributes. In addition, it contains information about the used concepts, their relations and their properties. In that sense, the vocabulary forms the basis of the other three containers. Ontological knowledge provides that basis for a lot of explanation options cf. [12]. The ontological knowledge is mainly contained in the knowledge base of the *originator* since it also forms the basis of the data mining process.
- The *pattern knowledge container* contains information about the relations, i.e., the mined patterns. Annotations and categorisation for these, for example, such as *known* or *suspected* relations form the basis of explaining the mined patterns in context. The pattern knowledge container has a strong connection to the support knowledge and the ontological knowledge container, since the patterns can always be linked to the ontology and also be supplemented with unstructured information, e.g., by comments or other textual (unstructured) information. The pattern knowledge can both be contained in the *originator* and the *explainer* depending on the application context. If the pattern knowledge is being utilised throughout the data mining session, then the knowledge needs to be embedded into the knowledge base of the originator. On the other hand, if the pattern knowledge is mainly used for enhancing the explanation, e.g., by setting the patterns into context, then the knowledge is usually located in the knowledge base of the explainer.
- The *instance knowledge container*, i.e., the *case base* presents the set of structured cases/instances that are used for data mining. The cases/instances provide the basis for very specific and concrete explanations and require appropriate selection and filtering options. Since the instance knowledge is operational knowledge for the originator, it is included in the respective knowledge base.
- The *context knowledge container* provides additional information about the domain and its concepts, e.g., unstructured information containing the respective concepts. In the medical domain, for example, this could be given by medical records for individual patients. The context knowledge container covers all types for supporting, optimising, or restricting explanations. For the latter case, context knowledge can include information, e.g., about privacy issues of the data, such that certain features of an explanation can be suppressed, if necessary. Since context knowledge is operational knowledge for explaining it is provided by the explanation knowledge base.

Thus, only the context knowledge container includes unstructured information, i.e., textual documents, while all other knowledge container provide structured knowledge for the explanation process.

Privacy Whenever data is collected from heterogeneous sources, aggregation of the data can reveal a lot more information than the single data sources. Consider the shopping domain, for example: In this domain, a lot of information is collected by both the shop and, e.g., a credit card company. All shopping data together can reveal a lot (more) about the life of a single customer, her interests and habits – compared to ‘isolated’ data. Especially when data from distributed environments and data sources is combined this provides special requirements for privacy-preserving methods, e.g., [1, 20].

Privacy becomes an even more important issue with the availability and use of Linked (Open) Data. The Linked Open Data project is a Semantic Web effort using the Web (i.e., using standards like HTTP, URIs and RDF) to connect related data that was not previously linked together [7]. One of the most visible examples for Linked Open Data is DBPedia, “a community effort to extract structured information from Wikipedia and to make this information available on the Web”.⁷ Governments and organisations start to provide Linked Data in vast amounts.⁸ These data sets will have a strong impact on data mining solutions.

Therefore, the privacy dimension in explaining is a very important and sensitive issue: If the explanation for a user only concerns his data, then the privacy implications are mostly a minor concern. However, whenever other (personal and) privacy-sensitive data is considered during the explanation session, then appropriate measures need to be considered ranging from a privacy-preserving to an open view on the data, information, and knowledge. Appropriate data aggregation or anonymisation, for example, can be used for ensuring the privacy of the collected data.

4 Ubiquitous and Social Contexts

In the following, we describe three application scenarios of the MACE in social software. The first scenario considers the *Social Resource and Metadata Hub ALOE*. Second, we consider the application of the MACE for the social resource sharing system Bibsonomy. Finally, we analyse the explanation continuum in an ubiquitous scenario. In all scenarios, we will revisit the elements of the MACE and provide illustrative examples in the respective application contexts.

In summary, the main benefit of the MACE is its ability to provide more complex and structured explanations at the same time. Consider a video recommendation from your favourite video portal, for instance. In this case, a video is only recommended based on the previously seen videos, however, usually only a simple explanation is provided, e.g., given by one video that is relevant to the recommended video. Utilising the

⁷ <http://dbpedia.org/About>

⁸ <http://esw.w3.org/TaskForces/CommunityProjects/LinkingOpenData/DataSets>

MACE considers more sophisticated explanation options, based on specialised knowledge which could include *interests* or *preferences* in our example. Then, e.g., more advanced justifications can be provided.

4.1 ALOE

The Social Resource and Metadata Hub ALOE⁹ is a Web 2.0 resource sharing platform designed, initially, for learning content of arbitrary format [10]. ALOE supports shar-

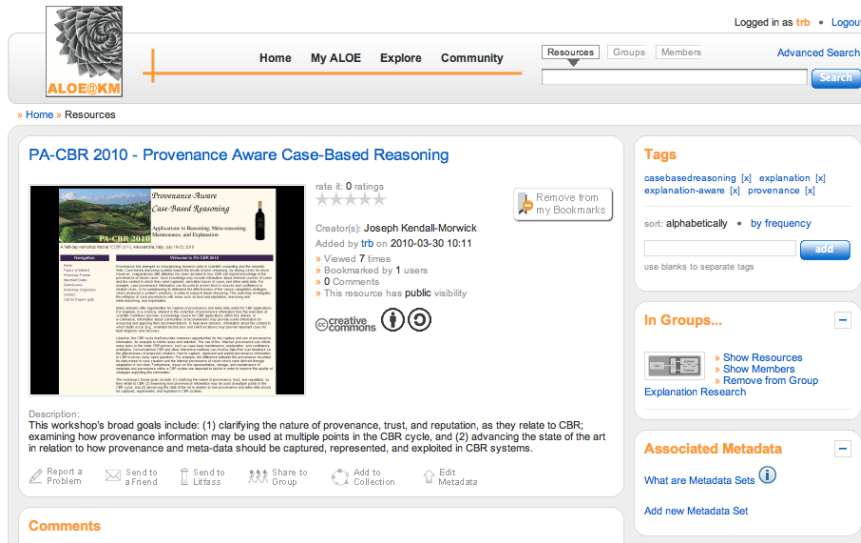


Fig. 3. Screenshot of the ALOE system

ing of files and bookmarks, provides tagging, commenting and rating functionalities and offers various search facilities. A group concept enables users to contact and exchange resources with other users that share similar topics of interest. The ALOE system enables users to share content according to personal interests and to be notified of relevant new resources. Whenever users add resources to their portfolio they have to annotate it with a title and tags, supported by information extraction components. Optionally a description as well as author and licensing information may be added. Several deployments of ALOE are in daily use, e.g., at DFKI's Knowledge Management department.

The design of the ALOE platform focused on understandability and ease-of-use. From the beginning transparency was the most important explanation goal to pursue in order to gain the trust of its users. “Concept explanations” and “how” explanations were deemed most suitable for providing transparency in this resource sharing scenario. For Terms that are most likely not self-explanatory concept explanations are provided

⁹ <http://aloe-project.de/>

(indicated by a circled ‘i’). For example, in Fig. 3, new users may not know about what metadata sets are. By clicking on the respective link (lower right corner) provides a brief explanation.

C-LINK, a conference organisation system built on top of the ALOE platform is a social sharing tool allowing conference participants to exchange, for instance, resources related to their talks [15]. A content-based recommender system provides event recommendations as well as recommendations of potentially interesting users based on the user’s research topics and shared resources. “Why” and “action” explanations help the users to make sense of the recommended resources, events or conference participants.

4.2 BibSonomy

Bibsonomy [6] is a social resource sharing system for managing publications and bookmarks. Besides this basic functionality, Bibsonomy also includes a recommendation framework, user profiles (via a curriculum vitae feature), arbitrary user groups, and links between users implemented by the “friend” functionality. In this way, Bibsonomy exceeds the typical basic features of a resource sharing system by advanced Web 2.0 and social software principles. However, these features do only complement the resource sharing functionality, which is still at the core of the system.

Explanation features concern the system as the originator of, for example, recommendations, rankings, or “related” user suggestions. Explanations can then be utilised “as is”, for enhancing the transparency of the system, but also for adaptations of user settings, and for a general customisation of the (behaviour of) the system.

Concerning the above examples, explanation for recommendations concerns “how” and “why” explanations. This is also related to the ranking features, for which the system should also explain the ranking explicitly via action explanations. Purpose explanations are important for group recommendations, for which concept explanations are also helpful. Finally, user profiles should also be connected to the explanation component, especially if they are involved in the recommendation, or if users are notified about publications of other (interesting) users.

Essentially, the system provides a lot of (social) features and tries to hide these at the same time in order to enable easy access to its features for novice users. Therefore, concept explanations can be applied on the interface meta-level for guiding the user when using the system. Thus, the learning goal can be implemented for enhancing the usability of the system in general. Furthermore, a useful complement for the interface explanations is given by (automatic) adaptive explanations for system changes, e.g., whenever an upgrade of the system is performed.

4.3 UBICON

A broad ubiquitous context is given by the UBICON framework¹⁰, cf. Fig. 4, developed in the context of the VENUS research cluster, that is concerned with the design of social and technological networking issues in situated ubiquitous systems. Therefore, the platform includes ubiquitous and social applications and environments.

¹⁰ <http://www.ubicon.eu>

First steps of the project are an application using sensor tags for inter-person sensing and localisation, called *Peer Radar* similar to the *Live Social Semantics* approach [2]. The goal of the application is to integrate data from the social semantic web, e.g., from social portals like Facebook, LinkedIn, Delicious, Flickr, lastFM, and from social systems like BibSonomy and KNOWTA in order to integrate those with data from the system, i.e., data about the social interactions and behavioural patterns of the user. Then, context-dependent actions can be applied for a user, i.e., providing helpful information about currently present other users, recommendations for actions, and additional information similar to the paradigm of augmented reality.

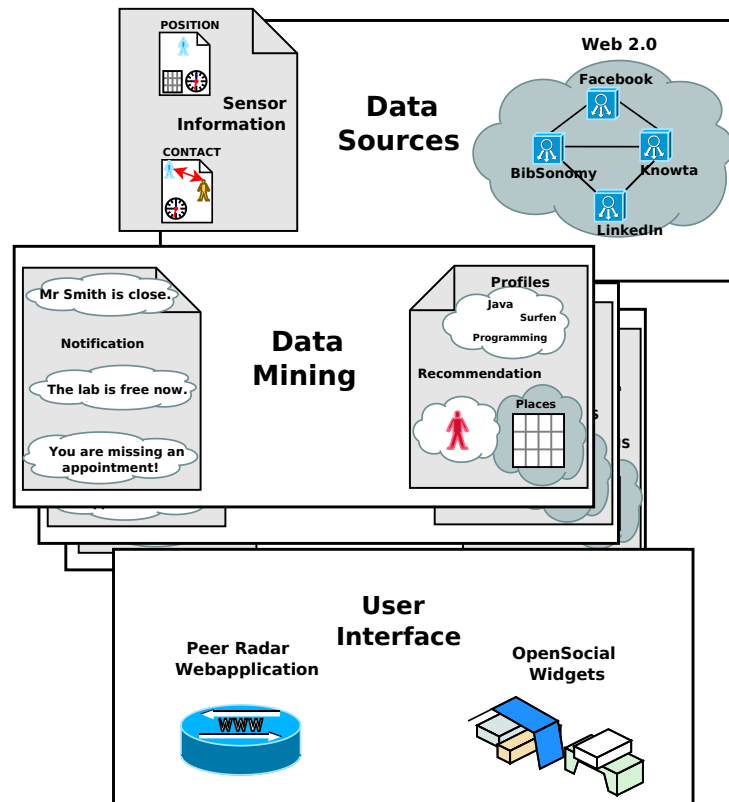


Fig. 4. Overview on Peer Radar System implemented using the UBICON framework for connecting ubiquitous and social environments

In this scenario, the system (as the originator) provides information to the user. In turn it can be explained by the explainer in order to increase the transparency and validity of the information, with the final goal of increasing trust in the system. Suitable recommendations include, for example, *interesting* contacts, topics, context-specific reminders for tasks or persons. These often motivate *actions*, e.g., contacting other users,

or working with certain resources. In such cases, the explanation of the proposed mined results is of ultimate importance. In this way, the trust in the actions of the system (and thus in the system itself), can be significantly increased.

Concerning the end-user, the ubiquitous social context provides several examples for explanation, in order to reassess decisions proposed by the system, by asking the system for justifications, for the relevancy of recommendations, and for the transparency of the mining steps. All kinds of explanations can, for example, be embedded into the recommendation explanation task but also in the reminder task. The final goal is to increase the trust in the system which is crucial for the continuous application in the ubiquitous and social context. Therefore, transparent explanations are of high importance.

For constructing the explanations, the explainer can utilise the social structures, social contacts and social knowledge implemented in both the social networks and the social resource sharing system. Using these, complex and structured explanations can be obtained according to the principles of the MACE, e.g., justifications for recommendations according to different details of interested users and/or interest profiles.

5 Conclusions

This paper discussed the application of the continuum of explaining for data mining and analysis in the context of social software: It described how data mining results can be analysed on several continuous dimensions and levels. Then, the mined patterns and data mining models can be inspected in context depending on the level of detail and the concrete perspective. Three application scenarios in ubiquitous and medical applications presented exemplary contexts for the presented approach and demonstrated its capabilities.

For future work, we want to investigate ontological explanations in more detail, especially in applications of ubiquitous and social environments. These provide ample options for mining and analysis, enable the inclusion of structured and unstructured data and helpful background knowledge. Furthermore, appropriate tool support is necessary, especially regarding the presentation dimensions. Therefore, we want to investigate advanced explanation-aware presentation techniques, e.g., in the context of the KNOWTA [3, 4] system and the UBICON framework, focusing on the concrete explanation-enhancing design issues.

Acknowledgements

This work has been partially supported by the VENUS research cluster at the interdisciplinary Research Center for Information System Design (ITeG) at Kassel University.

References

1. Agrawal, R., Srikant, R.: Privacy-Preserving Data Mining. In: Proc. ACM SIGMOD Conference on Management of Data. pp. 439–450. ACM Press (2000)
2. Alani, H., Szomszor, M., Cattuto, C., den Broeck, W.V., Correndo, G., Barrat, A.: Live Social Semantics. In: International Semantic Web Conference. pp. 698–714 (2009)

3. Atzmueller, M., Haupt, F., Beer, S., Puppe, F.: Knowta: Wiki-Enabled Social Tagging for Collaborative Knowledge and Experience Management. In: Proc. International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS). vol. CEUR-WS (2009)
4. Atzmueller, M., Lemmerich, F., Reutelshoefer, J., Puppe, F.: Wiki-Enabled Semantic Data Mining - Task Design, Evaluation and Refinement. In: Proc. International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS). vol. CEUR-WS (2009)
5. Atzmüller, M., Roth-Berghofer, T.: Ready for the MACE? The Mining and Analysis Continuum of Explaining uncovered. Research Report RR-10-02, Deutsches Forschungszentrum für Künstliche Intelligenz (2010), ISSN 0946-008X
6. Benz, D., Hotho, A., Jäschke, R., Krause, B., Mitzlaff, F., Schmitz, C., Stumme, G.: The Social Bookmark and Publication Management System BibSonomy - A Platform for Evaluating and Demonstrating Web 2.0 Research. VLDB Journal Submitted. (2010)
7. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
8. Clancey, W.J.: The Epistemology of a Rule-Based Expert System: A Framework for Explanation. Artificial Intelligence 20, 215–251 (1983)
9. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd Edition. Morgan Kaufmann, San Francisco, USA (2006)
10. Memmel, M., Schirru, R.: Aloe - a socially aware learning resource and metadata hub. In: Wolpers, M., Klamma, R., Duval, E. (eds.) Proceedings of the EC-TEL 2007 Poster Session at (EC-TEL-2007), September 17-20, Crete, Greece. vol. 280 (2007)
11. Richter, M.M.: The knowledge contained in similarity measures. Invited Talk at the First International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra, Portugal (1995)
12. Roth-Berghofer, T.R., Cassens, J.: Mapping goals and kinds of explanations to the knowledge containers of case-based reasoning systems. In: Muñoz-Avila, H., Ricci, F. (eds.) Case-Based Reasoning Research and Development, ICCBR 2005, Chicago, IL, USA, August 2005, Proceedings. pp. 451–464. No. 3620 in LNAI, Springer Verlag, Heidelberg (2005)
13. Roth-Berghofer, T.R., Richter, M.M.: On explanation. Künstliche Intelligenz 22(2), 5–7 (May 2008)
14. Schank, R.C.: Explanation Patterns: Understanding Mechanically and Creatively. Lawrence Erlbaum Associates, Hillsdale, NJ (1986)
15. Schirru, R., Baumann, S., Memmel, M., Dengel, A.: Extraction of contextualized user interest profiles in social sharing platforms. submitted to Journal of Universal Computer Science
16. Sørmo, F., Cassens, J.: Explanation goals in case-based reasoning. In: Gervás, P., Gupta, K.M. (eds.) Proceedings of the ECCBR 2004 Workshops. pp. 165–174. Departamento de Sistemas Informáticos y Programación, Universidad Complutense Madrid (2004)
17. Sørmo, F., Cassens, J., Aamodt, A.: Explanation in case-based reasoning – perspectives and goals. Artificial Intelligence Review 24(2), 109–143 (2005)
18. Spieker, P.: Natürlichsprachliche Erklärungen in technischen Expertensystemen. Dissertation, University of Kaiserslautern (1991)
19. Svátek, V., Rauch, J., Ralbovský, M.: Ontology-Enhanced Association Mining. In: Semantics, Web and Mining. LNCS, vol. 4289, pp. 163–179 (2005)
20. Vaidya, J., Clifton, C.: Privacy-Preserving Data Mining: Why, How, and When. IEEE Security and Privacy 2, 19–27 (2004)
21. Wick, M.R., Thompson, W.B.: Reconstructive expert system explanation. Artif. Intell. 54(1-2), 33–70 (1992)

Explanation-Aware Software Design of the Semantic Search Engine *KOIOS*

Björn Forcher¹, Thomas Roth-Berghofer^{1,2},
Michael Sintek², and Andreas Dengel^{1,2}

¹ Knowledge Management Department,
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Straße 122, 67663 Kaiserslautern, Germany

² Knowledge-Based Systems Group, Department of Computer Science,
University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern

`{first.lastname}@dfki.de`

Abstract. The provision of explanations in knowledge based systems has a long tradition in computer science. Regarding expert systems, several interesting contributions were proposed, but an overall method for explanation generation is missing. Furthermore, various approaches have never been realised or have never been followed up with respect to current technologies. Explanation-Aware Software Design (EASD) aims at making software systems smarter in interactions with their users. The long-term goal is to develop methods and tools for engineering and improving such capabilities. In this paper we present the idea of EASD including an abstract model for explanation generation. We describe in detail the realisation of that approach for the semantic search engine *KOIOS* using Semantic Web technology.³

Key words: explanation-awareness, explanation generation, semantic search

1 Introduction

Software systems need the ability to explain ‘reasoning processes’ and their results as those abilities can substantially affect their usability and acceptance. Explanation-aware Software Design (EASD) aims at making software systems smarter in interactions with their users [1]. The long-term goal is to develop methods and tools for engineering and improving such capabilities.

*KOIOS*⁴ is a semantic search engine that enables keyword-based search on graph-shaped RDF data. It first computes a set of relevant SPARQL queries allowing users to select the most appropriate ones. Subsequently, a selected query

³ This research work was supported in part by the research program THESEUS in the MEDICO project, funded by the German Federal Ministry of Economics and Technology (01MQ07016). Responsibility for this publication lies with the authors.

⁴ <http://projects.dfki.uni-kl.de/KoiosWebUI/>

is used to search in a respective triple store. Since SPARQL queries are sometimes hard to read, it is necessary to prepare the queries in a more user-friendly way. In addition, it is helpful to know how the keywords relate to the SPARQL queries. System developers, for instance, may want to test the search engine. In this context, explanations are interesting when the system presents unexpected results. It may turn out that the implementation of the search algorithm is incorrect. Hence, explanations can help to correct a system or improve it. Non-expert users are not interested in the exact implementation of the search algorithm. Instead, they just need simple support to choose the most useful query.

For addressing these issues, we developed and integrated an explanation facility into *KOIOS*. The facility is used to justify computed search results by revealing a connection between keywords and queries. The justifications are depicted as semantic networks in an attempt to make search results more plausible for users. The facility is based on the concept of EASD including a complex model for explanation generation. That means that the provision of explanations is an integral part of the development of *KOIOS* the mentioned user groups and their explanation needs.

The paper is structured as follows. The next section gives a brief overview of relevant research on explanation generation. Sections 3 and 4 presents the concept of EASD and our current model for computing explanations. Section 5 describes the semantic search engine *KOIOS* and motivates the need for explanations. Finally, we illustrate how we realised the explanation component of *KOIOS* regarding EASD. We conclude the paper with a summary and outlook.

2 Related Work

Explanation facilities were important components of Expert Systems (ES) supporting the user's needs and decisions. In those early systems, explanations were often nothing more than (badly) paraphrased rules that lacked important aspects, or too much information was given at once [2].

Wick and Thompson [3] implemented the Reconstructive Explainer (REX) that implements the concept of *reconstructive explanations* for expert systems. REX transforms a trace, *i. e.*, a line of reasoning, into a plausible explanation story, *i. e.*, a line of explanation. The transformation is an active, complex problem-solving process using additional domain knowledge. The degree of coupling between the trace and the explanation is controlled by a filter that can be set to one of four states regulating the transparency of the filter. The more information of the trace is let through the filter, the more closely the line of explanation follows the line of reasoning. We took up the theme of (re-)constructing explanations in our current work.

In [4], Ducassé and Noyé describe a stratified model for user-oriented program analysis including explanations. It contains three steps to explain solutions of logic programs, namely *Extraction*, *Analysis*, and *Visualisations*. The model uses several sources of input such as source codes, execution information and other specifications which are integrated by the extraction step. The result of the

extraction is the trace that is analysed and abstracted in the following step. The resulting data of the second step is presented to the user. Ishizaka and Lusti [5] extend that model by defining a trace model as first step.

The Semantic Web community also addresses the issue of explainability. The Inference Web effort [6] realises an explanation infrastructure for complex Semantic Web applications. Inference Web includes the *Proof Markup Language* (PML) for capturing explanation information. PML can be used as interlingua and offers constructs to represent where information came from (provenance) or how it was manipulated (justifications). The Inference Web comprises different tools and services in order to manipulate and edit PML models including various interaction modes such as exploration or filtered explanation views. We also consider different explanation views in order to provide useful kinds of interaction for realising understandable explanatory dialogues. However, the current work is concerned with explaining system processes. As PML is especially designed for representing justifications of reasoning results it is not part of the explanation facility of *KOIOS*.

3 Explanation-Aware Software Design

In this section we present our current research about *Explanation-Aware Software Design* (EASD). The concept comprises two key aspects regarding development and runtime of the system. In the following, we will elaborate on some general aspects regarding integrating explanations in knowledge-based systems. Subsequently, we present an abstract architecture and a multi-layered explanation model for corresponding facilities.

The first key aspect aims at integrating explanation capabilities into the entire system development process. Swartout and Moore [7] formulated the five desiderata *fidelity*, *understandability*, *sufficiency*, *low construction overhead*, and *efficiency*. Explanations respect fidelity if they build on the same knowledge the system uses for reasoning. Understandability comprises such factors as *user sensitivity* and *feedback*. User sensitivity addresses the user's goals and preferences but also her knowledge of the system and respective domain. Feedback is important as users do not necessarily understand a given explanation. The system should offer certain kinds of dialogue and different perspectives so that users can get a better understanding. This includes that the explanation is sufficient. Finally, integrating such explanation capabilities should not be too complicated and the explanation component should not affect the performance of the system.

We consider the understandability and sufficiency aspects as highly relevant for EASD. In general, the system is used by various kinds of users such as end-users, experts or developers. Each user group has different requirements and comes with different a priori knowledge with respect to the system itself and according domains. In this context, the explanation need of the system functionality must be addressing goals and preferences of users [8], such as *Transparency* and *Justification*. The first goal aims at imparting an understanding of how a system found an answer. This allows users to check the system by examining

the way it reasons and allows them to look for explanations for why the system has reached a surprising or anomalous result. The justification goal aims at increasing confidence in the advice or solution offered by a system by giving some kind of support for the conclusion suggested by the system. This goal allows for a simplification of the explanation compared to the actual process the system goes through to find a solution.

Depending on the goal, a particular explanation method is indicated. Here, the term ‘method’ denotes the dialogue and presentation of the explanation in a graphical or textual way, but also the kind of explanation. Spieker [9] distinguishes several useful kinds, among them *concept explanations* and *action explanations*. Concept explanations concern questions such as *What is (the semantics of) ... ?*. The purpose of action explanations is to explain the cause or provide a justification for a fact or a current situation.

Obviously, it is not possible to foresee all conceivable explanations with respect to all user groups. However, this is not the claim of EASD. The main contribution is not to add the explanation capability after a system is developed or to provide arbitrary explanations as it happened in many expert systems [2]. Instead, EASD intends to give developers the means to integrate suitable and understandable explanations into the design process of the entire system. For instance, one of our long-term goals is to extend UML in order to provide explanations in Java programs.

As mentioned before, the second key aspect of EASD becomes important at runtime of the system. The system has to be *aware* of the explanation scenario itself enabling a goal-oriented dialogue. In general, an explanation problem cannot be solved by a single explanation. Often, only an explanatory dialogue [10] can satisfy the explanation need.

We consider three participants in any explanation scenario [11]. We call the system or agent that provides the solution to some problem, a technical device, a plan, or a decision to be explained the *originator*. This agent is interested in which way the user reacts after receiving the explanation. The *user* is the second participant. He is the addressee of the explanation. Finally, the *explainer* presents the explanation to the user. This agent is interested in transferring the intention of the originator to the user as correctly as possible. The explainer selects the style of the explanation and is responsible for the computational aspects as well as for organising a dialogue if needed. The user applies the explanation in some way. With this application, in principle, a utility is connected, either for the user or for the originator. It depends on the application which utility dominates. Sometimes the user is mainly interested in reacting properly and sometimes the primary interest is on the side of the originator.

4 Multi-Layered Explanation Model

We propose an abstract architecture and a multi-layered explanation model for providing explanations as depicted in Fig. 1. The mentioned scenario implies that the originator has to provide detailed information about its behaviour

and solutions regarding the fidelity desideratum. Therefore it is necessary that the originator prepares some kind of log or trace representing the initial starting point for the explainer to generate explanations. Regarding (potential) user questions, this information is step-by-step transformed into an adequate explanation including representational information.

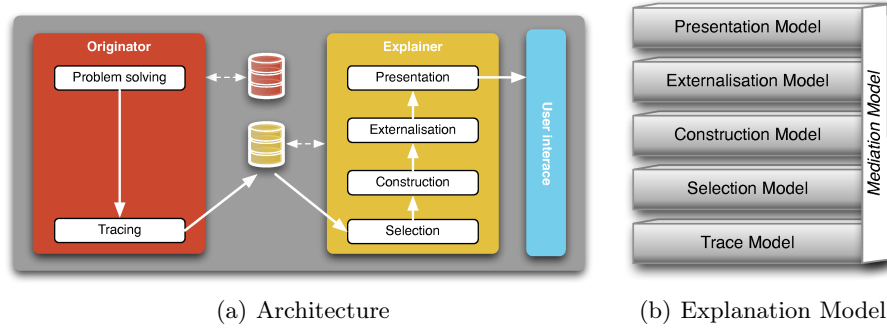


Fig. 1. Explanation-Aware Software Design

Depending on the coupling, originator and explainer share information sources that are required for explanation generation. However, the originator can have access to information which is hidden from the explainer and vice versa. At least, they have to share the tracing information.

As its name implies, the *tracing process* collects all information with respect to the behaviour of the originator for constructing the *trace model*. As it is not useful to collect virtually everything (efficiency desideratum), the information need must be predefined in the system design with respect to the explanation need of users. However, this contradicts the low construction overhead desideratum which should also be taken into account.

Any kind of explanation concerns only certain parts of the entire trace and thus, these parts must be identified. The respective transformation process, called *selection process*, extracts information from the trace with respect to user questions and dialogue situation representing basic explanation information. Hence, the selection represents a filter for the trace model.

This information does not necessarily represent complete explanation information. For this purpose, the selection model must be adapted to the current user requirements and context. The corresponding process is called *construction process* which is similar to the concept of reconstructive explanations as presented in Sect. 2. Here, supporting information can be added or confusing information can be removed. The construction process is used to span the entire explanation space comprising several possible explanation alternatives. Hence, the *construction model* contains the content of the explanation including domain knowledge and context depended information.

Explanation is information that is communicated by text, charts, tables, *etc.* Each communication form has different application possibilities in an explanation scenario. Text can describe complicated conceptions whereas charts can reveal qualitative connections in a simple way [12]. The *externalisation process* transforms the content model into a formal description for communicating explanations which is named as *externalisation model*. In this work, we put a special emphasis on semantic networks based on mathematical graphs for depicting explanations. However, we will consider further formal descriptions of tables, lists or natural language in the future.

The result of that process is not intended to represent the final explanation information because presentation information is missing. For instance, a mathematical graph model does not contain any layout information or written natural language uses a certain font. However, layout and style are important aspects of explanations influencing readability and understandability. For instance, highlighting important concepts in a textual explanation can serve as substantial and helpful support. The final transformation process transforms the externalisation model into the *presentation model* that can be leveraged by rendering engines to depict the explanation.

The main difference between this and existing approaches as presented in Sect. 2 is strict separation between content and form of an explanation. This allows depicting one and the same explanation in different forms which plays an important role with respect to user preferences [13].

A further difference is the explicit integration of the externalisation process. Each kind of externalisation offers various kinds of user interactions. For instance, lists can be sorted by means of a certain criterion or hyperlinks in texts can be followed. Regarding mathematical graphs, different exploration methods are conceivable such as closing of certain explanation paths. However, the most interesting aspect may be the application of various graph algorithms, such as shortest path algorithms. In case the construction model contains several possible explanation alternatives in form of paths the *shortest* explanation can be presented [14]. If users doubt that one, the next shortest path can be offered. Hence, the externalisation and presentation model play an important role to realise the understandability desideratum.

A further difference to other approaches is the concept of explanation-awareness at runtime of the system. For being *explanation-aware* the explainer has to trace his own transformation process including the results of the single process steps. Consequently, the explainer is aware of how content and presentation correspond to each other. In addition, it is aware of why an explanation is given in the current explanation scenario enabling explanatory dialogues and analysis of given explanations. This kind of information is stored in the *mediation model* addressing the sufficiency desideratum. For instance, if the explainer knows that a particular node of a graph-based explanation represents a domain concept, it can offer a respective concept explanation.

In the next sections we show how we realised the presented approach in the semantic search engine *KOIOS*.

5 The Semantic Search Engine *KOIOS*

KOIOS is a semantic search engine that enables keyword-based search on graph-shaped RDF data. *KOIOS* first computes a set of relevant SPARQL queries allowing users to select the most appropriate ones. Consequently, a selected query is used to search in a respective triple store. The main advantage is that users do not need explicit knowledge about the query syntax or the underlying ontologies. The presented approach is based on the work of [15] and [16].

For computing queries, the keywords are mapped to elements of input RDF data in a first step. Based on these *keyword elements*, a search is performed on the graph data to determine a *connecting element*. As its name implies, it is a particular graph element connecting the keyword elements. The paths between the keyword elements and the connecting elements are analysed to create a *matching subgraph*. For each subgraph, a conjunctive query is constructed by mapping the graph elements to query elements. To achieve a scalable search, the input data is preprocessed obtaining two data indices. The *keyword index* is used to realise the keyword mapping. For searching, a *graph index* is derived which represents a kind of summary of the input data containing structural patterns only. At runtime, this index is extended with the keyword elements obtained by means of the keyword index. Therefore, the extended graph comprises sufficient information to compute the SPARQL query.

The presentation of search results in common search engines always includes some hint why a particular result is relevant with respect to the keywords. Google, for instance, shows title, text snippets and URI of retrieved documents and highlights keywords contained therein. This information represents some kind of explanation or justification which describes a connection between search and result. In the end, it can help the user to open only relevant documents.

In case of *KOIOS*, explaining search results for end-users is bit more complicated. The justification has to contain the information how keywords are mapped to keyword elements and how the elements are connected. Finally, the interpretation as query has to be explained. In general, the same applies for developers when they debug the system. However, they need more detailed information and further intermediate results.

In the next section we describe how we used Semantic Web technology to realise some basic explanations.

6 Explanation Generation in *KOIOS*

In this section we present the realisation of the explanation facility of *KOIOS* using Semantic Web technology. For this purpose, we developed three RDFS ontologies which are used to describe the models of the explanation component. In addition, we implemented for each process a set of rules specifying how to transform a certain model.

As presented in Sect. 3 the starting point of the explanation generation is a kind of log provided by the originator. For that purpose, we developed a simple

process ontology in RDFS which comprises primitive concepts and relations for representing the *KOIOS* process called *KOIOS Process Language (KPL)*. It provides different specialisations of the basic class *Process* such as *KeywordMapping*, *GraphSearch* or *QueryConstruction*. In addition, KPL offers several classes for input and output of these processes. For instance, the class *IndexHit* is used to capture the mapping from keywords to a specific element of the input graph. As *KOIOS* is implemented in JAVA we used AspectJ⁵ to design a logging module. The module instantiates the trace model of a particular search and provides it for the explainer for further processing. Here, the benefit of using aspect-oriented programming is that the logging module can be completely decoupled from the originator *KOIOS*.

For realising a graph based view on the process model we also developed the Mathematical Graph Language (MGL). In a nutshell, the MGL ontology includes primitive concepts and relations for representing knowledge about simple mathematical graphs and is described using RDFS. It contains three main constructs, namely *Graph*, *Node* and *Edge* indicating the corresponding mathematical concepts. In addition, the class *Markup* and its two differentiations *Labelling* and *Weighting* can be used to annotate graph elements. All related markups are part of a certain *Marker* which also has two subclasses, i.e., *Labeller* and *Weighter*.

MGL is used to describe the externalisation model but it does not contain any information of visualising a graph. Although an intuitive visualisation of a mathematical graph and labelling information is possible, the layout of the graph is important. Especially the sequence of *KOIOS* processes should be arranged properly for clarity reasons. Thus, we conceived a third ontology for visualising graph based information called VGL. The ontology provides elementary constructs for structuring graph nodes in a grid or circle layout. In addition, VGL offers a variety of node visualisations by means of symbols or icons including labels. Hence, the VGL is used to describe the visualisation model.

So far, we described the necessary ontologies to describe the different layers of the explanation model. For constructing the layers we used *HEPHAISTOS* which is a transformation language for the Semantic Web. It is based on Horn logic but is especially designed for querying and transforming RDF models. Although the formal semantics of RDF (open world) and Horn logic (closed world) are not conforming, *HEPHAISTOS* is a useful tool particularly with respect to explanation generation. The name of the language is inspired by the Greek god of forge, Hephaistos, who ‘transformed’ workpieces into other forms. Similar to such approaches as TRIPLE [17] or OntoBroker⁶, the core language of *HEPHAISTOS* supports RDF primitives, i. e., name spaces, resources, and statements. It also includes the representation of contexts of statements which extends statements by a fourth dimension to quadruples. The core language can be compiled into Horn logic programs and executed by Prolog systems. For integrating the Java framework RDF2Go⁷ with *HEPHAISTOS* we implemented

⁵ <http://www.eclipse.org/aspectj/>

⁶ <http://www.ontoprise.de/en/home/products/ontobroker/>

⁷ <http://semanticweb.org/wiki/RDF2Go>

a JNI interface to the Prolog system XSB.⁸ The rule in Fig. 2 illustrates a simplified rule of the externalisation process which maps instances of the KPL class ‘IndexHit’ to instances of the MGL class ‘Node’ including labels.

```

01: <r1>
02:
03: true(H,uri('rdf','type'),uri('kpl','IndexHit'),
      uri('kpl','ConstContext')),
04: true(H,uri('kpl','element'),E,uri('kpl','ConstContext')),
05: X = true(H,uri('kpl','keywords'),K,uri('kpl','ConstContext')),
06: map([H],N),
07: map([X],L)
08:
09: ->
10:
11: true(N,uri('rdf','type'),uri('mgl','Node'),uri('kpl','GraphContext')),
12: true(L,uri('mgl','about'),N,uri('kpl','GraphContext')),
13: true(L,uri('mgl','label'),K,uri('kpl','GraphContext')),
14: true(L,uri('rdf','type'),uri('mgl','Labeling'),uri('kpl',
      'GraphContext')),
15: true(H,uri('kpl','isMappedTo'),N,uri('kpl','MediationContext')).

```

Fig. 2. Simplified rule of the externalisation process

The rule comprises three sections, namely rule declaration, input pattern and output pattern. The first section *<r1>* simply names the rule and has no further function. The following section describes the input pattern with respect to the construction model. It also includes instructions how elements of the construction model are mapped to elements of the externalisation model. In addition, it is also possible to add further constraints for separating similar rules from one another. The last section only contains the output pattern of the rule. It comprises quads for the externalisation and the mediation model as well (line 15). The rule contains three important predicates. The *true* predicate is used to represent either triples or quads depending on the number of arguments. The example only uses quads whereas the fourth argument represents the context or the URI of the explanation model. An URI is expressed by means of the *uri* predicate. In case the predicate has two arguments, the first argument represents an abbreviation of a namespace and the second argument a local name. Finally, the *map* predicate is used to ‘construct’ new resources out of existing ones (lines 6-7). The first argument is always a list of resources whereas the new resource is represented by the second argument. For mapping triples or quads, the equal operator has to be used (line 5). Every fact is assigned a unique ID. The equal operator employs the ID to construct an URI and assigns it to a variable. As a consequence, it is possible to map a set of triples to one resource.

⁸ <http://xsb.sourceforge.net/>

For explaining search results of *KOIOS* we provided for each explanation need of end-users and developers a set of rules that transforms step-by-step the trace model into a visual model. The information of the visual model is leveraged by a particular renderer returning a Java JPanel with the entire explanation.

Fig. 3 shows the graphical user interface of *KOIOS*. In this case, the underlying graph-shaped data corresponds to YAGO⁹ representing a huge knowledge base that is derived from Wikipedia¹⁰ and WordNet.¹¹ The figure depicts a search for *Barack Obama* and everything this person *wrote*. The white left panel contains one of five computed SPARQL queries whereas the right white panel presents the respective result of the query. The interaction panel below is used to browse through the SPARQL queries, to change the interaction with the SPARQL queries, and to request certain kinds of explanations. In this version of *KOIOS* we realised only two kinds of explanation. Clicking on the earth symbol above the question mark symbol the user can obtain conceptual explanations of the instances and classes contained in a query. While maintaining the same externalisation mode the connections to other elements are revealed.

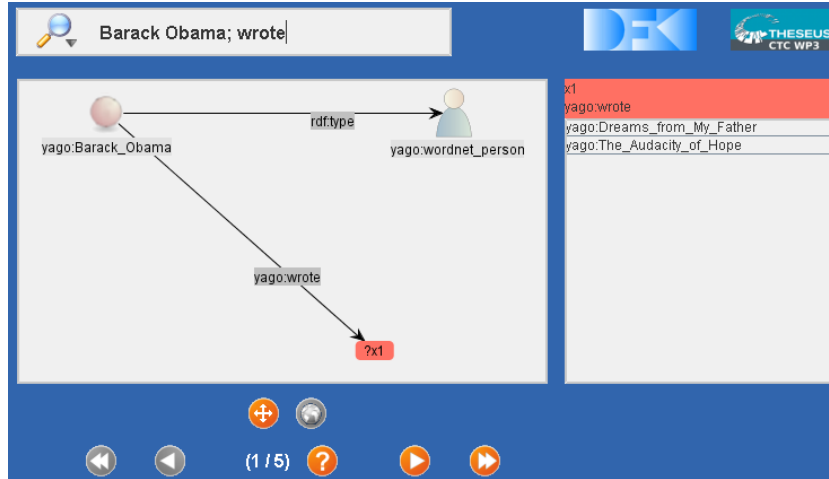


Fig. 3. Graphical User interface of *KOIOS*

The second kind of explanation is retrieved by clicking on the question mark symbol. Thereupon a new panel is opened containing action explanations. The explanations are intended for developers and try to justify the keyword-element mappings (justification goal). Regarding the example from Fig. 3 the developer may be interested in why the query contains a *yago:wordnet_person* instead of a *yago:wordnet_politician*. Fig. 4 provides the

⁹ <http://www.mpi-inf.mpg.de/yago-naga/yago/>

¹⁰ <http://en.wikipedia.org/>

¹¹ <http://wordnet.princeton.edu/>

hint that the resource *yago:Barack_Obama* is typed with *yago:wordnet_person* and *yago:wordnet_politician* as well. However, the justification also provides the weights of both classes in the summary graph which is indicated by the float value in the node label. Hence, the class *yago:wordnet_person* is considered most in the computation process addressing the transparency goal.

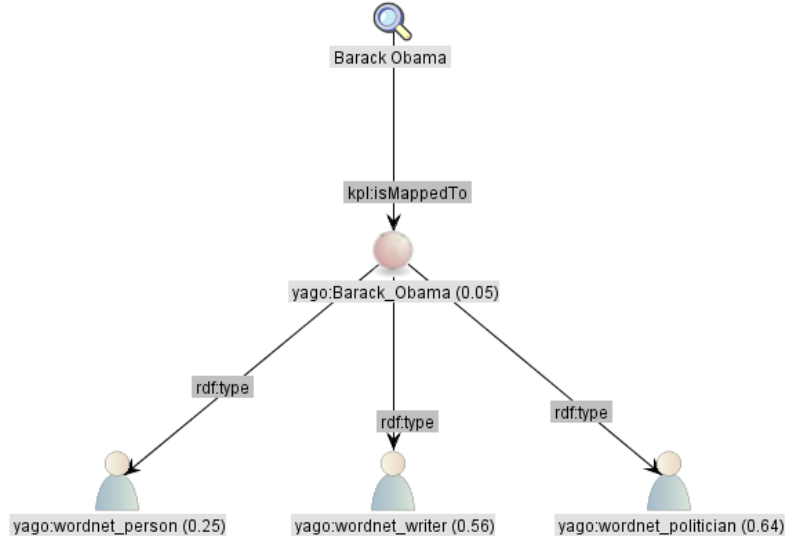


Fig. 4. Justification of keyword-element mapping

7 Summary and Outlook

In this paper we presented the explanation facility of the semantic search engine *KOIOS*. The facility was developed making use of Explanation-Aware Software Design (EASD) principles, which advocate the integration of explanation capability into the entire system development process. For that purpose, the explanation need of select users is collected implying particular explanation goals such as learning or transparency. Depending on such goals well defined explanation methods ensure that explanations fulfil user requirements. EASD explanation generation is based on a multi-layered explanation model and an iterative transformation process. In case of *KOIOS*, a process model is stepwise transformed into a visualisation model with respect to a particular user question. The constructed explanation model allows different views on explanations and enables various explanation methods such as sophisticated explanatory dialogues or the provision of alternative explanations.

The current process ontology of *KOIOS* is dedicated to its search algorithm. The next step of our work is to generalise the ontology for using it in other

knowledge-based systems. In addition, the provision of transformation rules for each explanation need is quite difficult due to syntax errors. Here, reusable rule patterns seem to be indicated.

References

1. Roth-Berghofer, T.: ExaCt manifesto: Explanation-aware computing (2009) http://on-explanation.net/content/ExaCt_Manifesto.html.
2. Richards, D.: Knowledge-based system explanation: The ripple-down rules alternative. In: Knowledge and Information Systems. Volume 5. (2003) 2–25
3. Wick, M.R., Thompson, W.B.: Reconstructive expert system explanation. *Artif. Intell.* **54**(1-2) (1992) 33–70
4. Ducassé, M., Noyé, J.: Logic programming environments: Dynamic program analysis and debugging. *J. Log. Program.* **19/20** (1994) 351–384 model of user-oriented program analysis.
5. Ishizaka Alessio, L.M.: How to program a domain independent tracer for explanations. *Journal of Interactive Learning Research* **17**(1) (2006) 57–69
6. McGuinness, D.L., Ding, L., Glass, A., Chang, C., Zeng, H., Furtado, V.: Explanation interfaces for the semantic web: Issues and models. In: Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI'06). (2006)
7. Swartout, W.R., Moore, J.D.: Explanation in second generation expert systems. *Second generation expert systems* (1993) 543–585
8. Sørmo, F., Cassens, J.: Explanation Goals in Case-Based Reasoning. In Gervás, P., Gupta, K.M., eds.: Proceedings of the ECCBR 2004 Workshops. Number 142-04 in Technical Report of the Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid, Madrid (2004) 165–174
9. Spieker, P.: Natürlichsprachliche Erklärungen in technischen Expertensystemen. Dissertation, University of Kaiserslautern (1991)
10. Du, G., Richter, M.M., Ruhe, G.: An explanation oriented dialogue approach and its application to wicked planning problems. *Computers and Artificial Intelligence* **25**(2-3) (2006)
11. Roth-Berghofer, T.R., Richter, M.M.: On explanation. *Künstliche Intelligenz* **22**(2) (2008) 5–7
12. Wright, P., Reid, F.: Written information: Some alternatives to prose for expressing the outcomes of complex contingencies. *Journal of Applied Psychology* **57** (2) (1973) 160–166
13. Kemp, E.A.: Communicating with a knowledge-based system. In Brezillon, P., ed.: Improving the Use of Knowledge-Based Systems with Explanation. (1992)
14. Roth-Berghofer, T., Forcher, B.: Improving the understandability of semantic search explanations. *International Journal of Knowledge Engineering and Data Mining (IJKEDM)* (2010) to appear.
15. Wang, H., Zhang, K., Liu, Q., Tran, T., Yu, Y.: Q2semantic: A lightweight keyword interface to semantic search. In: In Proceedings of the 5th International Semantic Web Conference (ESWC'08). (2008)
16. Tran, D.T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: Proc. of the 25th Intern. Conference on Data Engineering (ICDE'09), Shanghai, China (2009)
17. Sintek, M., Decker, S.: TRIPLE – a query, inference, and transformation language for the semantic web. In: ISWC '02: Proc. of the First International Semantic Web Conference on The Semantic Web, London, UK, Springer-Verlag (2002) 364–378

Providing Explanations for the Intelligent Monitoring of Business Workflows Using Case-Based Reasoning

Stelios Kapetanakis, Miltos Petridis, Jixin Ma, Liz Bacon

School of Computing and Mathematical Sciences, University of Greenwich, Maritime Greenwich
Campus, Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, UK
{s.kapetanakis, m.petridis, j.ma, e.bacon}@gre.ac.uk

Abstract. This paper presents an approach for providing explanation to the intelligent diagnosis and monitoring of business workflows based on operation data in the form of temporal log data. The representation of workflow related case knowledge in this research using graphs is explained. Workflow cases are represented in terms of events and their corresponding temporal relationships. The matching and CBR retrieval mechanisms used in this research are explained and the architecture of an integrated intelligent monitoring system is shown. The paper contains an illustration and evaluation of the approach based on experiments on real data from a university quality assurance exam moderation system. It is shown that a graph matching based similarity measure is capable to diagnose problems within business workflows and the results of this reasoning process can be explained to workflow managers and users with the use of graph visualisation techniques.

Keywords: Case-Based Reasoning, Explanation, Business Workflows, Temporal Reasoning, Graph Similarity.

1 Introduction

In modern organisations most activities are organised in terms of business workflows that allow for the timely and efficient coordination of people and systems to achieve organisational objectives. Increasingly, business workflows are defined, orchestrated, facilitated and monitored electronically by computer systems. This provides organisations with an opportunity to monitor and manage business workflows more efficiently and a challenge to do so in an auditable and trustworthy way that allows managers to take operational corrective actions and make useful business process improvements.

The Business Process Modelling Notation (BPMN) developed by the Business Process Management Initiative (BPMI) and Object Management Group (OMG) provides a standard for the graphical representation of workflow based business processes[1]. Workflow based business process representation is possible with standards covering the definition, orchestration and choreography of business processes.

Over the last few years, a number of standards have emerged and are widely accepted and supported by mainly Service Oriented Architecture (SOA) based enterprise technologies and systems. The OASIS Business Process Execution Language (BPEL), short

for Web Services BPEL (WS-BPEL) is a key orchestration technology [2]. The Workflow Management Coalition (WfMC) backed XML Process Definition Language (XPDL) is a format standardised to interchange Business Process definitions between different workflow products and systems [3].

Using a graphical representation of business processes such as UML or BPMN allows business process designers, managers and participants in business workflows to understand the definitions and context of business workflows as these are designed to be viewed by humans. However, in monitoring business workflows, systems produce a large number of monitoring data, usually in terms of a number of temporally ordered logs of events that cannot be easily monitored, understood and acted upon easily by human operators.

Humans faced with the task of monitoring business workflows have to deal with the challenge of sifting through a large number of events appearing in a typical workflow event log and the complexity of reasoning with temporal relationships between events. Furthermore, effective monitoring may need the combination of more than one logs and/or correlation of workflow events to other events that may not be captured by the systems. Finally, contextual knowledge about the workflow execution may be incomplete or uncertain. This is due to the fact that workflows are very often adapted or manually overridden by managers in order to deal with unanticipated problems and changes in the operating environment, especially in the aspects of workflows that interact with human roles. Workflows are liable to change as the business requirements change and may involve business processes from different parts of an organisation, or between collaborating organisations which may cause conflicts.

It has become increasingly difficult for humans to monitor business workflows bringing the need for the intelligent monitoring, diagnosis and intervention into business workflows by software systems.

A software system attempting the automatic interpretation and monitoring of workflows requires the ability to provide explanations that can allow managers and stakeholders in business workflows to identify, understand and act on problems identified on the workflow data. In order for this to be efficient, the system should be able to provide adequate explanation on why a particular workflow has been tagged as requiring addition and on what the identified problem may be, as well as providing some explanation on a remedial action it may require.

This paper presents an approach to providing explanations for the monitoring of business workflows using the Case-Based Reasoning (CBR) approach[4]. A specific issue with this approach is the fact that a workflow monitoring CBR system does not attempt to build an explicit model of the knowledge associated with workflows, but reasons on the fundamental assumption that similar problems have similar solutions. A similarity measure is defined and used to identify possible solutions for a particular target case workflow case problem by finding similar neighbouring workflow cases from within a case base and reusing the solutions of these cases to provide a solution to the target workflow case problem. The explanation required for such a system is one that shows the system user why the system retrieved particular workflow cases as most similar and how the retrieved similar cases can be used to provide a solution to the problem in hand.

In many typical CBR systems, the similarity and relevance of the contribution of neighbouring cases can be simple to understand. However, due to the structural and

temporal complexity of workflow related knowledge providing a suitable and trustworthy explanation is a challenge.

This paper presents an approach and a system that attempts to provide explanations to support the CBR monitoring of business workflows. Section 2 presents the CBR approach and its application to a real workflow monitoring problem. Section 3 discusses the workflow similarity measures used in this work that support the visualisation of workflow logs and the similarity between workflows used for explanation. Section 4 presents the architecture of a system used for the monitoring of business processes that incorporates an explanation module. Section 5 provides a simple evaluation of the proposed approach based on the example workflow monitoring case study.

2 The Case-Based Reasoning Approach for Monitoring Business workflows

Case-Based Reasoning (CBR) has been proposed as a natural approach to the recall, reuse and adaptation of workflows and knowledge associated with their structure. Minor et al [5] proposed a CBR approach to the reuse and adaptation of agile workflows based on a graph representation of workflows and structural similarity measures. Dijkman et al[6] have investigated algorithms for defining similarities between business processes focused on tasks and control flow relationships between tasks. Van der Aalst et al [7] compare process models based on observed behaviour in the context of Petri nets. Weber et al[21] demonstrates a tool for adaptive workflow management through conversational CBR. The definition of similarity measures for structured representations of cases in CBR has been proposed [8] and applied to many real life applications requiring reuse of domain knowledge associated with rich structure based cases [9],[10].

A new approach for the intelligent monitoring of business workflows using CBR has been proposed and has been shown to be able to monitor effectively real business workflows when compared to human domain experts[11],[12]. This approach can deal with the uncertainty aspect of the workflow monitoring problem. It requires the definition of similarity measures informed from knowledge discovery of norms and known problems from past operation. It uses a simple graph based representation of cases based on events, actions, intervals and their temporal relationships. An architecture capable to deal with the definition and orchestration of business processes as well as capable of providing monitoring and control services over workflows is key to allowing for the intelligent management of business processes within an organisation.

2.1 The Exam Moderation Business Process Workflows

In order to evaluate the approach proposed in this research, it was decided to use a typical formal business process comprising of defined workflows and involving a number of actors interacting and communicating securely with a workflow monitoring system and involving artefacts tracked and changed by the workflow. The University of Greenwich, School of Computing and Mathematical Science exam moderation system is such a business process for which anonymised operational logs were made available to this research project. The

exam moderation system is an automated web enabled secure system that allows course (module) coordinators, course moderators, exam drafters (typically senior managers), admin staff and external examiners to upload, modify, approve and lock student exam papers and orchestrates and captures the associated actions and communications. The system automates the whole process and provides an audit trail of events generated by workflow stakeholders and the system. The system orchestrates a formal process made up of workflows. The process can be defined and displayed formally in terms of a BPMN diagram. The system tracks most workflow actions as timed log events(fig. 1). Most of these actions generate targeted email communications to workflow stakeholders, some for information and others requiring specific further actions from these stakeholders.

For example, the action of a new exam version upload from a course coordinator is notified to the moderator, drafter and admin staff. This can prompt the moderator to approve the uploaded version or upload a new version. However, the coordinator can also upload a new version and admin staff may also decide to format the uploaded version and upload it as a newer version. The system captures all versions, workflow actions, emails sent. There is also a facility to initiate and record free form comments associated to particular document versions and/or workflow actions.

AUDIT TRAIL
Examination document Exam_COMPXX70_ver1_May_2009.DOC has been uploaded by Michael Peterson at 03/04/2009 12:21:43. Email sent to Moderator: mcXX Office: cms-exams
Examination document Exam_MS_COMPXX70_ver1_May_2009.DOCX has been uploaded by Michael Peterson at 04/04/2009 00:06:03. Email sent to Moderator: mcXX Office: cms-exams
Examination document Exam_COMPXX70_ver4_May_2009.DOCX has been uploaded by Ch&&&g XX at 07/04/2009 13:15:22. Email sent to Coordinator: pmXX Office: cms-exams Head of Department: pmXX
Examination comment has been added by Ch&&&g Ma at 07/04/2009 13:18:32. Email sent to Coordinator: pmXX Office: cms-exams Head of Department: pmXX
Moderator C&&&g XX has signed off the Exam and Marking Scheme at 08/04/2009 00:07:17. No Emails sent
Drafter SXtX McKXXXie has removed the Moderator Sign Off for the Exam and Marking Scheme at 08/04/2009 11:40:49. Email sent to Moderator: mcXX Coordinator: pmXX Office: cms-exams Head of Department: pmXX

Fig. 1. An example excerpt of an event log for the exam moderation process (partially obfuscated)

Discussions with workflow monitoring managers showed that patterns of events indicated, but not defined uniquely the current context and state of a workflow. Managers were generally able to guess from looking at the workflow events and communications audit what the context and current state of a workflow was and point to possible problems. However, this is a very time consuming and error-prone process. The number of workflows monitored by a manager is more than one hundred and it is practically impossible to keep monitoring these continuously for problems. Most problems occur due to human

misunderstanding of the current state and confusion with roles and responsibilities and usually result to the stalling of a workflow. Managers will then try to restart the process by adding comments to the system, or initiate new actions and communications. However, this depends on managers realizing that such a problem has occurred in the first place and on understanding the nature of the problem.

The CBR Workflow Intelligent Monitoring System (CBR-WIMS) is a system that was built to provide an automatic monitoring system that will notify managers and stakeholders of potential problems with the workflow and provide advice on actions that can remedy a perceived problem. The monitoring system works based on experience of past event/action temporal sequences and the associated contextual knowledge and classification in a Case-Based Reasoning system. Similarity measures allow the retrieval of close matches and their associated workflow knowledge. This allows the classification of a sequence as a particular type of problem that needs to be reported to the monitoring system. Additionally, it is intended that any associated knowledge or plan of action can be retrieved, adapted and reused in terms of a recommendation for remedial action on the workflow.

3 Workflow and Event Log Representation and Similarity Measures

In CBR-WIMS similarity measures between workflow representations can be defined using a graph representation of workflow processes using an exhaustive graph similarity search algorithm based on the Maximum Common Subgraph [10].

3.1 Representation of events in business workflows

The representation of events in the workflow event log uses a general time theory, based on intervals [13]. In the theory used here, the temporal relationships have been reduced from the ones proposed by Allen [14] to just one, the “meets” relationship.

The general time theory takes both points and intervals as primitive. It consists of a triad $(T, \text{Meets}, \text{Dur})$, where:

- T is a non-empty set of time elements;
- Meets is a binary order relation over T ;
- Dur is a function from T to R_0^+ , the set of non-negative real numbers.

A time element t is called an interval if $\text{Dur}(t) > 0$; otherwise, t is called a point.

This approach has been shown to be suitable for defining temporal similarity measures in the context of a CBR system based on the graph representation of events and intervals and their temporal relationships and similarity measures. This can be achieved using graph matching techniques such as the Maximum Common Subgraph (MCSG [10]. Additionally, such a graph can be checked for consistency of temporal references using linear programming techniques [15] and it has been shown that it can provide the basis of a system that can derive explanations from partial temporal information[16].

For example, consider a scenario with a temporal reference (T, M, D) , where:

$$T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\};$$

$M = \{\text{Meets}(t_1, t_2), \text{Meets}(t_1, t_3), \text{Meets}(t_2, t_5), \text{Meets}(t_2, t_6), \text{Meets}(t_3, t_4), \text{Meets}(t_4, t_7),$
 $\text{Meets}(t_5, t_8), \text{Meets}(t_6, t_7), \text{Meets}(t_7, t_8)\};$
 $D = \{\text{Dur}(t_2) = 1, \text{Dur}(t_4) = 0.5, \text{Dur}(t_6) = 0, \text{Dur}(t_8) = 0.3\}$
 The graphical representation of temporal reference (T, M, D) is shown in Fig. 2:

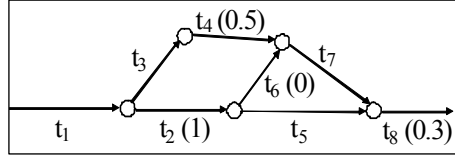


Fig. 2. Graph representation of temporal relationships

3.2 Similarity between graph representation of workflows

To determine the similarity between workflows the WIMS-CBR system uses a number of measures. The main one is based on the Maximum Common Subgraph (MCSG) between two graphs representing workflows.

The Maximum Common Subgraph similarity between two such graphs can be defined as:

$$S(G, G') = \frac{\left(\sum_{\substack{\text{matches} \\ C, C' \\ \text{in} \\ \text{MCSG}}} \sigma(C, C') \right)^2}{\text{count}(G) \cdot \text{count}(G')} \quad (1)$$

where $\text{count}(G)$ represents the number of edges in graph G and $\sigma(C, C')$ is the similarity measure, $0 \leq \sigma(C, C') \leq 1$, between two individual edges (intervals or events) C and C' .

In the case of time stamped events produced by the workflow event log, the duration of each interval can be calculated, so the graphs are collapsed into a single timeline. In this case, the similarity measure is easier to calculate as the MCSG is a common segment made up of events and intervals in a given order in each of the compared workflow logs. In this common graph segment each edge (event or interval) has a similarity measure to its counterpart in the other log that exceeds a given threshold value ϵ . Eq. 1 above can be used to provide the overall similarity between the two workflows. Other branches of the graph can represent contextual temporal information necessary for the interpretation of a sequence of events. This other temporal information could be, for example, the proximity to a deadline, or reminder communications broadcast to staff by managers outside the system (i.e. using normal direct email).

3.3 Providing explanation on temporal similarity between workflows

Tools specialized on log visualisation are common in the computer Systems and networking area. Snort View [17] works in a similar way analysing data from network

attack logs. LogView [18] conducts application log analysis after searching within the various web traffic data that were produced from log clusters. Tools use different visualization techniques to present log data. SnortView uses 2-D plots to represent the log events whereas LogView uses Treemaps. Treemaps [19] can visualise hierarchical data in a concentrated, effective way regardless of their size. More specifically, in the workflow explanation area Michaelis et al have proposed a provenance focused approach using RITE networks to represent workflows[20].

The key advantage of the representation and similarity measures for workflow cases in the CBR system represented as graphs containing temporal information about the workflows allows the visualisation of workflows that provides the basis for providing explanation to support the monitoring of workflows.

A workflow event log audit trace is represented in its simplest form (no other contextual temporal relationships) as a simple graph segment:

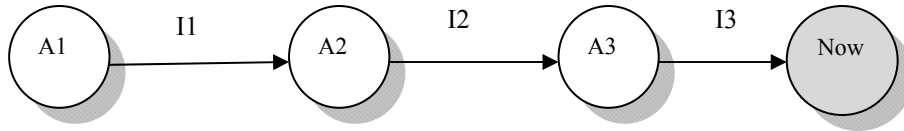


Fig. 3. A Simple workflow Event log segment

which can then be represented as:

(Action1, Actor1, Interval1, Action2, Actor2, Interval2, Action3, Actor3, Interval3)

An example of this would be (intervals are in days):

(CoordUpload, John, 3, ModUpload, Phil, 0, CoordUpload, John, 5)

In the first instance the name of the person involved was ignored, focusing solely on the role involved in the action. The similarity measure between two actions A_1 and A_2 is defined as:

$$\sigma(A_1, A_2) = 1 \text{ if } A_1 = A_2 \text{ and } \sigma(A_1, A_2) = 0 \text{ if } A_1 \neq A_2$$

The similarity measure between two intervals I_1 and I_2 is defined as:

$$\sigma(I_1, I_2) = 1 - |I_1 - I_2| / (|I_1| + |I_2|), \max(|I_1|, |I_2|) > 0, \sigma(0, 0) = 1$$

The Maximum Common Subgraph (MCSG) between cases C and C' is assembled starting right (latest) to left (earliest) calculating similarity measures matching each interval and action in C to the corresponding one in C' , stopping when the similarity between two edges falls under a threshold set at 0.5. For example, given the following two cases:

$C = (\text{CoordUpload}, \text{John}, 3, \text{ModUpload}, \text{Phil}, 0, \text{CoordUpload}, \text{John}, 5)$ and

$C' = (\text{ModUpload}, \text{Phil}, 4, \text{ModUpload}, \text{Phil}, 0, \text{CoordUpload}, \text{Mary}, 3)$

Assembling the MCSG:

1. $\sigma(5, 3) = 1 - 2/8 = 0.75$
2. $\sigma(\text{CoordUpload}, \text{CoordUpload}) = 1$
3. $\sigma(0, 0) = 1$
4. $\sigma(\text{ModUpload}, \text{ModUpload}) = 1$
5. $\sigma(4, 3) = 1 - 1/7 = 0.857$
6. $\sigma(\text{CoordUpload}, \text{ModUpload}) = 0$.. MCSG Matching stops

So, the overall similarity between C and C' from eq. 1 is:

$$S(C, C') = (0.75 + 1 + 1 + 1 + 0.857)^2 / 6^2 = 0.59$$

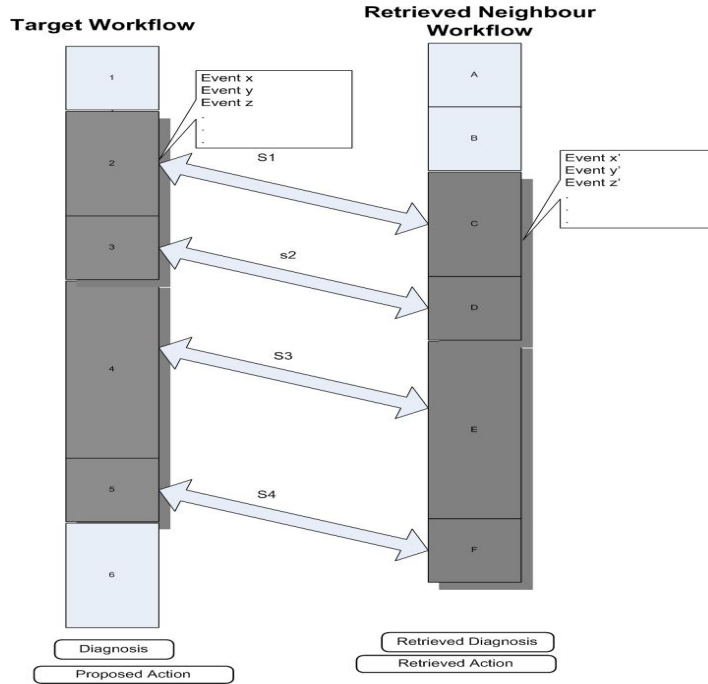


Fig. 4. Visualising the similarity between workflows

When comparing two workflows, the similarity is measured as the normalised sum of similarities between a number of non overlapping MCSG segments in the two workflows. The identification of these workflow segments and their visual communication to the user provides the visual explanation of the CBR monitoring process:

- It explains visually how similar two workflows are.
- It explains visually which parts of the workflow contribute most to this similarity. This allows the user to focus to specific patterns in parts of the workflow that may flag a workflow as problematic.
- It shows visually past remedial actions taken and the results of such actions. This allows a user to understand why the system has made a particular diagnosis and provides an insight into possible remedial options to the current target workflow case problem.

Fig 4 shows the visualisation of the similarity between simple linear workflow event logs. The MCSG is highlighted and there is an ability to drill down to the events and intervals in individual workflow segments. The similarity measures and diagnosis and proposed actions are also shown to provide the user with a deeper understanding of the workflow monitoring process.

4 The Architecture of the Workflow Intelligent Monitoring System

CBR-WIMS is an Intelligent Workflow Monitoring System incorporating a CBR component and an explanation module. The role of the system is to assist the transparent management of workflows in a business process and to orchestrate, choreograph, operate, monitor and adapt the workflows to meet changing business processes and unanticipated operational problems and inconsistencies. Fig. 5 above shows the overall architecture and components of CBR-WIMS. The system allows process managers to create, modify and adapt workflows to suit the changing business needs, and/or to allow for variations related to special business requirements. Furthermore, managers can monitor workflow operation and get diagnostic warnings and context sensitive advice. If a fault or possible problem pattern is detected, this is reported to the workflow operations manager together with the retrieved similar cases and associated recorded experience of any known remedy/course of action. The system provides explanation to users based on the graphical representation of the target workflows and the comparison to the retrieved nearest neighbours.

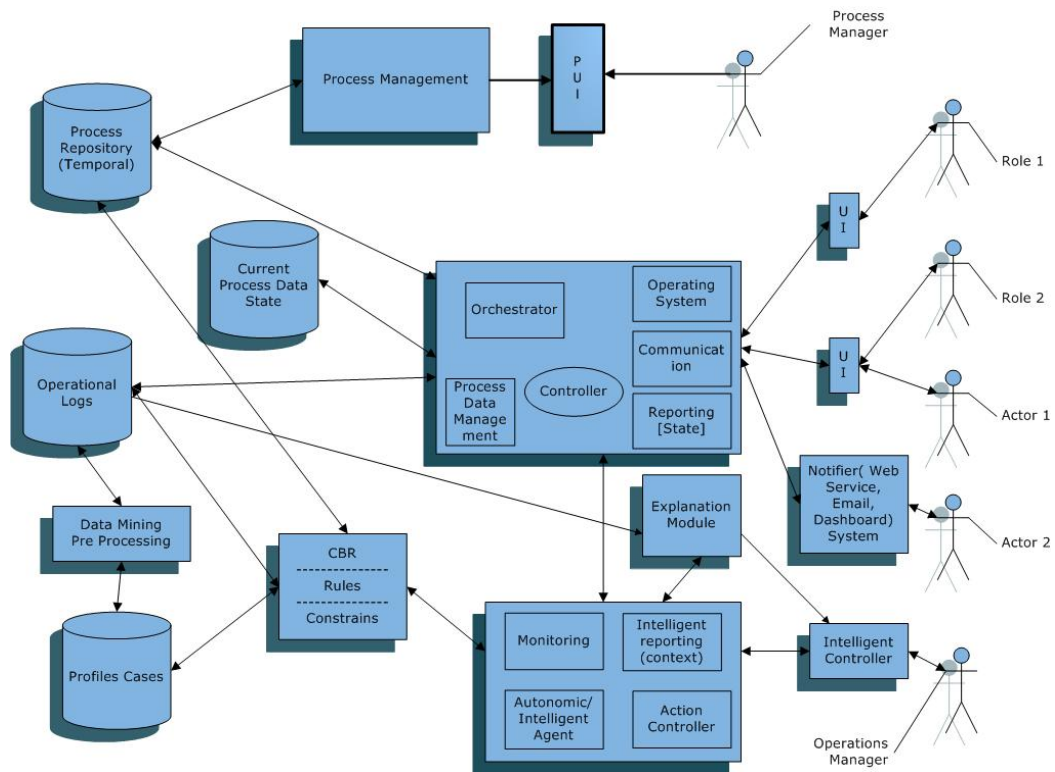


Fig. 5. The Intelligent Workflow Monitoring System Architecture

5 Providing Workflow Monitoring Experiments and Evaluation

In order to evaluate the suitability of the approach proposed in this paper, a number of simple experiments were conducted using the CBR-WIMS system. A small number (20) of exam moderation workflows were monitored and compared to a case base of 130 known workflows which contained expert monitoring classification and advice. The ability of the system to provide suitable monitoring advice on this data set has been evaluated in previous research [12]. The purpose of this experiment was to establish the ability of the system to explain the decisions of the CBR workflow monitoring system.

Workflow managers used WIMS-CBR to establish whether the system provides useful explanation on:

1. The classification of a workflow as “stalled”/”not stalled”
2. The reasons for a “stalled” diagnosis
3. Proposed remedial actions.

Figure 6 shows the typical view of the explanation screen of WIMS-CBR. The user can select any of the nearest neighbours of the target case and investigate the similarity and the retrieved advice. The colour coding of regions shows the degree of similarity. Clicking on particular matching workflow segments within the MCSG allows the user to visualise the areas of overlap which are automatically highlighted in the same colour and compared on the screen. The system allows the users to “drill” down to particular areas of similarity between the workflows and investigate the monitoring advice retrieved by each of the most similar neighbours.

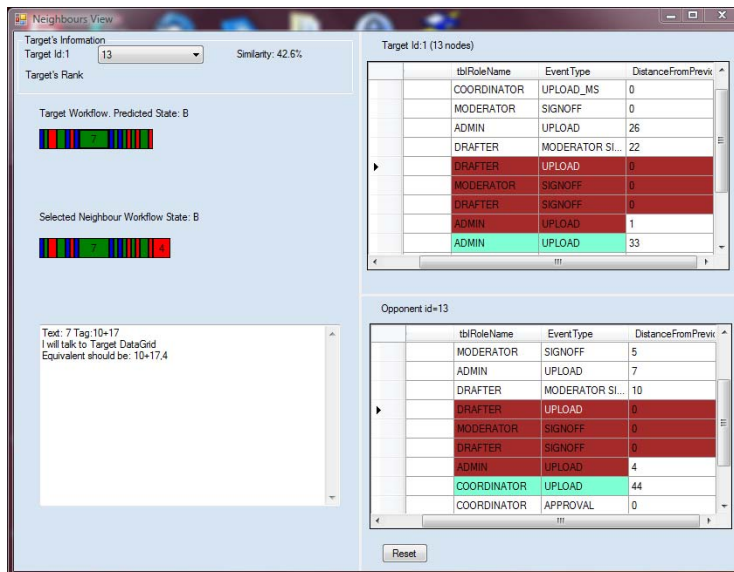


Fig. 6. The Similarity explanation screen in WIMS CBR

The table below (table 1) summarises the results of the simple evaluation of the explanation capabilities of WIMS-CBR. The experts replied to the questions using a scale of 1 (disagree) to 5 (agree strongly). The results were averaged over the 20 target cases.

	WIMS-CBR no explanation	WIMS-CBR with explanation
Correct classification is clear	3.2	4.2
Similarity is obvious to the 3NN	2.8	3.9
Advice clarity	3.3	4.5

Table 1. Evaluation of the explanation and advice

This preliminary evaluation has shown that the explanation module in WIMS-CBR can provide a significantly better insight to workflow managers to allow them to understand better the monitoring warnings and achieve an insight and confidence on the associated advice on remedial action. It is worth noting that even in cases that the CBR system retrieved low quality solution (mainly false positives), the experts reported that the explanation system in WIMS-CBR allowed them to discard the retrieved solution as drilling down to workflow event data showed that the retrieved advice was not relevant to the problem in hand.

6 Conclusions

This paper discussed an approach for the explanation of intelligent diagnosis and monitoring of workflows based on incomplete operation data in the form of temporal log data. This was based on a graph representation of workflows using temporal relationships. Within the CBR process, providing suitable explanations for similarity matching of cases is key to making the CBR approach effective. This is particularly relevant with a complex case representation. The workflow process is orchestrated by a software system using BPEL technologies in service oriented architecture in the CBR-WIMS system. The CBR-WIMS architecture was evaluated in the context of a system that monitors exam moderation workflows. The use of explanation of the matching and similarity measures presented here was showed through a preliminary evaluation using real data from a typical workflow process. The evaluation showed that the approach is capable of providing useful explanations to the CBR workflow monitoring process. Further work will concentrate on further evaluation of the approach, explaining using more complex case representation across multiple workflow logs and providing contextual temporal knowledge, also allowing the adaptation of solutions to retrieved similar workflows using local optimisation criteria.

References

1. Business Process Management Initiative (BPMI): BPMN 1.1: OMG Specification, February, 2008, <http://www.bpmn.org/>, accessed April 2009.

2. OASIS: BPEL, The Web Services Business Process Execution Language Version 2.0. <http://www.oasis-open.org/apps/org/workgroup/wsbpel>, May 2006.
3. Workflow Management Coalition (WfMC): XPD L 2.1 Complete Specification (Updated Oct 10, 2008), <http://www.wfmc.org/xpdl.html>, accessed April 2009.
4. Aamodt, A. and Plaza, E. "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches" *Artificial Intelligence Communications* 7, no. 1 (1994): 39-52
5. Minor, M., Tartakovski, A. and Bergmann, R.: Representation and Structure-Based Similarity Assessment for Agile Workflows, in Weber, R., O. and Richter, M., M.(Eds) *CBR Research and Development, Proceedings of the 7th international conference on Case-Based Reasoning, ICCBR 2007*, Belfast, NI, UK, August 2007, LNAI 4626, pp 224-238, Springer-Verlag, (2007)
6. Dijkman, R.M., Dumas, M., Garcia-Banuelos, L. Graph matching algorithms for business process model similarity search. In U. Dayal, J. Eder (Eds.), *Proc. of the 7th Int. conference on business process management*. (LNCS, Vol. 5701, pp. 48-63). Berlin: Springer. (2009)
7. van der Aalst, W., Alves de Medeiros, A. K., Weijters, A : Process Equivalence: Comparing two Process Models Based on Observed Behavior, In *Proc. Of BPM 2006*, vol 4102 of LNCS, pp 129-144, Springer, (2006)
8. Bunke, H., Messmer, B.T.: Similarity Measures for Structured Representations. In: Wess, S., Richter, M., Althoff, K.-D. (eds) *Topics in Case-Based Reasoning*. LNCS, vol. 837, pp 106-118, Springer, Heidelberg (1994)
9. Mileman, T., Knight, B., Petridis, M., Cowell, D., Ewer, J. (2002): Case-Based Retrieval of 3-D shapes for the design of metal castings in *Journal of Intelligent Manufacturing*, Kluwer. 13(1): 39-45; Feb 2002.
10. Wolf, M., Petridis, M.: Measuring Similarity of Software Designs using Graph Matching for CBR, In *workshop proceedings of AISEW 2008 at ECAI 2008*, Patras, Greece (2008)
11. Kapetanakis, S., Petridis, M., Ma, J., Bacon, L. : Workflow Monitoring and Diagnosis Using Case-Based Reasoning on Incomplete Temporal Log Data, in *Workshop proceedings of the 8th International Conference on Case Based Reasoning*, Seattle, USA, (2009)
12. Kapetanakis, S., Petridis, M., Ma, J., Bacon, L. "CBR-WIMS, an Architecture for the Intelligent Monitoring and Diagnosis of Workflows" in *proceedings of the 14th UK CBR workshop*, Peterhouse, December 2009, (Ed M. Petridis), CMS press, pp. 67-78
13. Ma, J., Knight, B.: A General Temporal Theory, *the Computer Journal*, 37(2), 114-123 (1994).
14. Allen, J., Hayes, P.: Moments and Points in an Interval-based Temporal-based Logic, *Computational Intelligence*, 5, 225-238 (1989).
15. Ma, J., Knight, B.: A Framework for Historical Case-Based Reasoning, in K.D. Ashley and D.G. Bridge (Eds.): *ICCBR 2003*, LNAI 2689, pp. 246-260 (2003)
16. Ma, J., Knight, B., Petridis, M.: Deriving Explanations from Partial Temporal Information, in *workshop proceedings of ExACT-08 at ECAI 2008*, Patras, Greece (2008)
17. H. Kioke and K. Ohno, "SnortView: Visualization System of Snort Logs," in *CCS Workshop on Visualization and Data Mining for Computer Security*. Washington, DC, USA.: ACM Press, October 2004, pp.143 – 147.
18. A.Makanju et al.(2008) , "LogView: Visualizing Event Log Clusters",in *Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust*, pp99-108.
19. B. Schneiderman, "Tree Visualisation With Tree-maps: a 2-D space filling approach." *ACM Transactions on Graphics*, vol. II, no. 1, pp. 92-99, January 1992.
20. Michaelis, J. R., Ding, L and McGuinness, D. L. "Towards the Explanation of Workflows" in *workshop proceedings of ExACT-09 at IJCAI 2009*, Pasadena, USA (2009)
21. Weber, B., Wild, W. and Brey, R. "CBRFlow: Enabling Adaptive Workflow Management Through Conversational Case-Based Reasoning", in *Proceedings of ECCBR04, Advances in Case-Based Reasoning*, LNCS, Vol. 3155, 434-448, Springer (2004)

Learning to Explain with ABML

Martin Možina, Matej Guid, Jana Krivec,
Aleksander Sadikov, and Ivan Bratko

Faculty of Computer and Information Science,
University of Ljubljana, Slovenia,
`martin.mozina@fri.uni-lj.si`

Abstract. The main advantage of machine learning algorithms that learn simple symbolic models is in their capability to trivially provide justifications for their decisions. However, there is no guarantee that these justifications will be understood by experts and other users. Induced models are often strange to the domain experts as they understand the problem in a different way. We suggest the use of argument based machine learning (ABML) to deal with this problem. This approach combines machine learning with explanations provided by domain experts. An ABML method is required to learn a model that correctly predicts learning examples and is consistent with the provided explanations. The present paper describes an application of ABML to learning a complex chess concept of an attack on the castled king. The explanation power of the learned model for this concept is especially important, as it will be used in a chess tutoring application.

Key words: machine learning, argument based machine learning, learning to explain, chess

1 Introduction

The common weakness of state-of-the-art machine learning algorithms achieving the best results in terms of accuracy (e.g. ensemble methods or SVMs) is their inability to explain their predictions. Domain experts and users alike perceive these methods and their predictions as black boxes. However, machine learning is often used to get a better understanding of the relation between inputs and outputs, especially so if machine learning is used as a knowledge acquisition tool [3]. In such cases, it is usually preferable to use methods like decision trees, rule-based models, or linear models that produce hypotheses that are mostly easily understood and interpreted.

Yet, even easily understood models are not always enough. The induced correlations can often seem illogical or simply strange to the domain experts as they would explain the same case using different terms. Pazzani [7] showed experimentally that people will grasp a new concept more easily if the concept is consistent with their knowledge. More elaborate studies of understanding new concepts were produced by the cognitive learning community [10]. They showed that when we learn about new presented materials, we always start off with our

prior knowledge and try to merge them. If the new concepts are inconsistent with our prior knowledge, the new knowledge will likely be distorted or even rejected.

A recently developed approach called Argument Based Machine Learning (ABML) [6] is an extension of classical machine learning (ML) that allows the use of experts' explanations provided in the form of arguments for particular learning examples. An ABML method is required to learn a model that correctly predicts learning examples and is consistent with the provided explanations. Therefore, the explanations direct the search towards models that are more comprehensible in the light of experts' background knowledge. The benefit is twofold: (a) new knowledge is assimilated more easily, and (b) the accuracy of the model is usually increased.

We are interested in building systems for automated commentary and intelligent tutoring. The main characteristic of such systems is that their knowledge has to facilitate good explanation. In the present paper we present part of our work targeted at creating a program for automated commentary of chess games. The current chess software is terribly lacking in this respect; the chess-playing programs have long ago surpassed the human world-champion level of play, yet, they are completely unsuitable for providing commentary [11]. This is to be expected as they were designed for the former, and their language, so to speak, is not intended for the latter. The situation is in many ways reminiscent of the difference between SVMs and decision trees or rules as described in the first paragraph.

The paper is organized as follows. We first briefly introduce the ABML paradigm, and then we present its application for learning a difficult concept that will be featured in automated commentary. Afterwards, the results of both machine learning and ABML are presented and compared to one another, giving special consideration to the ability of both paradigms to give explanations for their decisions. Last section contains conclusions and specifies some necessary future work.

2 Argument Based Machine Learning

Argument Based Machine Learning (ABML) [6] is machine learning extended with some concepts from argumentation. In this section, we give a short summary of ABML taken from [6].

Argumentation is a branch of artificial intelligence that analyzes reasoning where arguments for and against a certain claim are produced and evaluated [8]. A typical example of such reasoning is a law dispute at court, where plaintiff and defendant give arguments for their opposing claims, and at the end of the process the party with better arguments wins the case.

Arguments are used in ABML to enhance learning examples. Each argument is attached to a single learning example only, while one example can have several arguments. There are two types of arguments; positive arguments are used to explain (or argue) why a certain learning example is in the class as given, and

negative arguments are used to explain why it should not be in the class as given. We used only positive arguments in this work, as negatives were not required. Examples with attached arguments are called *argued examples*.

Arguments are usually provided by domain experts, who find it natural to articulate their knowledge in this manner. While it is generally accepted that giving domain knowledge usually poses a problem, in ABML they need to focus on one specific case only at a time and provide knowledge that seems relevant for this case and does not have to be valid for the whole domain. The idea can be easily illustrated with the task of commenting chess games. It would be hard to talk about chess moves in general to decide precisely when they are good or bad. However, if an expert is asked to comment on a particular move in a given position, he or she will be able to offer an explanation and provide relevant elements of this position. Naturally, in a new position the same argument could be incorrect.

An ABML method is required to induce a theory that uses given arguments to explain the examples. Thus, arguments constrain the combinatorial search among possible hypotheses, and also direct the search towards hypotheses that are more comprehensible in the light of expert's background knowledge. If an ABML method is used on normal examples only (without arguments), then it should act the same as a normal machine learning method.

In this paper, we will use a modified version of the ABCN2 [6] method (the modification is described in the following section), which is an argument based extension of the well known method CN2 [2]. ABCN2 learns a set of unordered probabilistic rules from argued examples. In ABCN2, the theory (a set of rules) is said to explain the examples using given arguments, when there exists at least one rule for each argued example that contains at least one positive argument in the condition part. This definition is a bit simplified, since it omits the use of negative arguments, as they are not relevant for this paper¹.

2.1 Interactions between expert and ABML

In ABML, experts are asked to provide their prior knowledge in the form of arguments for the learning examples rather than the general domain knowledge. However, asking experts to give arguments to the whole learning set is not likely to be feasible, because it would require too much time and effort. The following loop describes the skeleton of the procedure that picks out critical examples - examples that ABML can not explain without some help:

1. Learn a hypothesis with ABML using given data.
2. Find the most critical example and present it to the expert. If a critical example can not be found, stop the procedure.
3. Expert explains the example or changes the class of the example; the explanation is encoded in arguments and attached to the learning example.

¹ Due to space limitations, we will only roughly describe some of the mechanisms of ABML (see [6] or/and its website www.ailab.si/martin/abml for precise details).

4. Return to step 1.

To finalise the procedure we need to contemplate the following two questions:

- How do we select “critical” examples?
- How can we achieve to get all necessary information for the chosen example?

Identifying critical examples The main property of critical examples is that the current hypothesis can not explain them well, or, in other words, it fails to predict their class. Since ABCN2 gives probabilistic class prediction, we define the most critical example as the example with the highest probabilistic error. The probabilistic error can be measured in several ways. We use a k -fold cross-validation repeated n times (e.g. $n = 4, k = 10$), so that each example is tested n times. The most critical example is thus the one with highest average probabilistic error.

Are expert’s arguments good or should they be improved? Here we describe in details the third (3) step of the above algorithm, where the expert is asked to explain the critical example. First, the expert considers the class of the example whether it is incorrectly labelled. In such a case, the class value of the example is changed. Otherwise, the expert explains the example with arguments. Using expert’s arguments, ABML will sometimes be able to explain the critical example, while sometimes this will still not be entirely possible. In such cases, we need additional information from expert. The whole procedure for one-step knowledge acquisition is described with the next 5 steps:

Step 1: Explaining critical example. In this step, the expert is asked the following question: “Why is this example in the class as given?” The answer can be either “I don’t know” (the expert is unable to explain the example) or a set of arguments $A_1, \dots, A_k$ all confirming the example’s class value can be given. If the system gets the answer “don’t know”, it will stop this procedure and try to find another critical example.

Step 2: Adding arguments to example. Arguments A_i are given in natural language and need to be translated into domain description language (attributes). Each argument supports its claim with a number of reasons. When a reason is simply an attribute value of the example, then the argument is simply added to the example. On the other hand, if reasons mention other concepts, not currently present in the domain, these concepts need to be included in the domain as new attributes before the argument can be added to the example.

Step 3: Discovering counter examples. Counter examples are used to spot if arguments suffice to successfully explain the critical example or not. If ABML fails to explain the example, then the counter examples will show where the problem is. Here, ABML is first used to induce a hypothesis H_1 using previous learning data only and H_2 using learning data together with new arguments. A counter example is defined as: it has a different class value

from the critical example, its probabilistic error increases in H_2 with respect to H_1 , and H_2 mentions arguments (given to the critical example) while explaining the counter example.

Step 4: Improving arguments. The expert needs to revise the initial arguments with respect to the counter example. This step is similar to steps 1 and 2 with one essential difference; the expert is now asked "Why is critical example in one class and why counter example in the other?" The answer is added to the initial argument.

Step 5: Return to step 3 if counter example found.

3 Case Study: Experimental Setup

Our goal was to formalize the concept of *an attack on the castled king* for the purposes of generating automatic commentary. That is, the computer should be able to reliably judge whether and why a particular side in a chess game aims at attacking the opponent's king that has already castled.

The concept of attack on the castled king can be considered as a positional concept of high complexity. There is some general agreement in the chess literature and among chess players about the intuition behind this concept. However, small details may prevail in an assessment whether particular side has chances to generate pressure against the opponent's king, and many factors can potentially influence such an assessment (see Fig. 1). Moreover, chess masters usually also rely on *dynamic* factors in chess positions when estimating chances of attacking successfully, while our concept should be *static* in its nature for its ultimate purpose of annotating chess games (see [4] for details). Our formalized model, however, does not aim at correctly estimating chances of success when attacking, but merely to estimate whether the attack is possible at all.

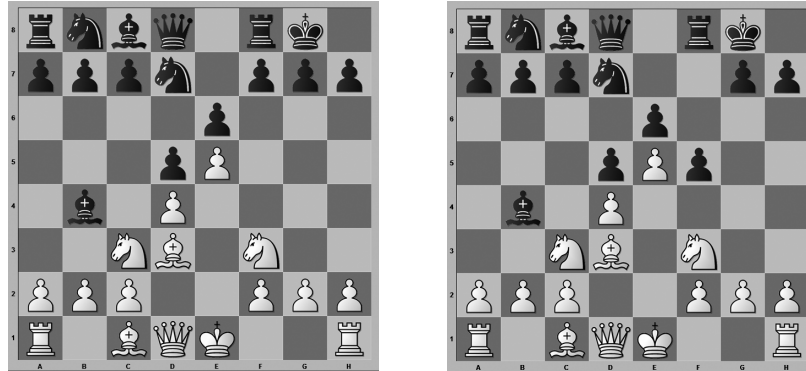


Fig. 1. In the position in the left-hand diagram White-to-move gains a strong attack by applying the classic bishop sacrifice: 1.Bd3xh7+ Kg8xh7 2.Nf3-g5+. However, by only the f7 pawn being on f5 square as shown in the right-hand diagram, White does not have any pressure against the opponent's king at all.

3.1 Data set description

The positions in the data set were gathered from the Chess Informant database (see <http://www.sahovski.com>). Our chess experts (woman grandmaster Jana Krivec and FIDE master Matej Guid) selected 577 middlegame positions from real chess games where the black player has castled on the king side and white still has a queen on the board. They classified each position either as *attack* (white has chances for an attack) or *no attack* (white cannot or is not planning to attack the black king). Of the 577 positions, 350 were labelled as *attack* and 228 as *no attack*. We then randomly selected 277 positions for learning and 278 for testing; stratification was used to preserve class distributions in both subsets.

Initially, the chess experts carefully selected twelve attributes they considered useful for identifying an attack on the king in a chess position. Each of these attributes corresponds to a well-known chess position feature that is regularly considered by chess players. Some examples of such attributes are: *pawn_attack* (number of white pawns on files f,g, and h that progressed to fourth row or higher), *pawn_defense* (number of black pawns protecting the black king), *close_combat_pieces* (number of white pieces in the area around the black king), etc. As will become evident later, this set of descriptive attributes was significantly enlarged during the interaction between the expert and the computer.

3.2 Problem definition

Our goal is to learn a set of rules for classifying a position with respect to whether the white player is attacking the black king or not. Rules will be evaluated according to the following criteria:

Classification accuracy: the usual measure in machine learning for evaluating goodness of a learned model.

Precision and recall: besides classification accuracy, we aim at high precision, since in our tutoring application, the cost of misclassifying an example without attack as *attack* is much higher as misclassifying an example with attack as *no attack*.

Explanation power of rules: it is not the accuracy of classification that is most important to meet the requirements stated in the introduction to our application: it is the quality, in terms of interpretability, of the rules generated that is crucial, as we aim at providing good commentary.

Since our main intention is to use these rules to explain why there is an attack in a position, we decided to learn only rules for the class *attack*. However, to avoid low precision with such rules, we modified the ABCN2 algorithm to induce only rules with pure class distributions, namely, a rule can cover only positions with an attack and should not cover any positions without an attack. Furthermore, the procedure for selecting critical examples in the ABML refinement loop had to be changed to return only positions with an attack. If a negative position (without attack) became critical — which is possible, as we use internal cross-validation — the critical example became a positive position covered by the rule that misclassified the negative example.

4 Case study: experiments and results

The first set of rules was generated from examples without any arguments. ABCN2 altogether induced 12 rules achieving 84% classification accuracy, 87% precision, and 86% recall. The high accuracy, precision and recall imply that the initial set of attributes was relatively well-defined. Afterwards, we began with the ABML refinement loop. In the remainder of this section we will present a single iteration of the loop to illustrate the interaction between the experts and the algorithm and then, at the end, provide results of the final model.

Figure 2 shows one of the critical examples, automatically selected by our algorithm.

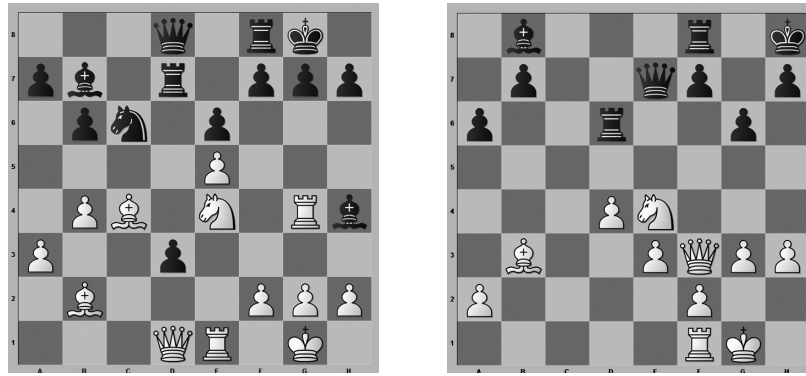


Fig. 2. Computer: “Why is it possible for White to attack the opponent’s king?” (left) But why is it NOT possible for White to attack the opponent’s king here? (right)

Current rules at the time failed to classify the example on the left-hand diagram as *attack*, as it was previously judged by the experts. The following question was given to the experts: “Why is it possible for White to attack the opponent’s king?” The experts used their domain knowledge to give the following arguments: “White rook is placed on a half-open file in front of the castled position of the black king and there are several white pieces joining the attack.” Based on the experts’ explanation, the following argument was attached to this critical position:

- “*open_file* is true” (white queen or white rook are placed on an open or half-open file *f*, *g* or *h*),
- “*attacking_pieces* is high” (number of white pieces that attack one of the following squares: *f6*, *g6*, *h6*, *f7*, *g7*, *h7*, *g8* or *h8*).

Note that both attributes were already included into the domain at the time. The following rule covering the critical example was produced:

```
IF open_file is true AND attacking_pieces > 1
  AND defends_around_king <=10
THEN ATTACK = true;
```

The method automatically found additional restrictions to improve the experts' explanation. The attribute *defends_around_king* expresses the number of squares in the square f5-h5-h7-f7 defended by black pieces including black pawns.

Afterwards, the algorithm found a counter-example to this argument shown on the right-hand diagram in Figure 2. Now the question to the experts was: *But why is it NOT possible for White to attack the opponent's king here?* The experts were asked to compare this position to the position in the critical example. They gave the following answer: *"White queen indeed is placed on a half-open file in front of the castled position of the black king, but here it just does not seem so strong. In general, having a white queen or a white rook placed on an open or half-open file g or h is stronger than on an open or half-open file f."*

Previously, the attribute *open_file* indicated an open file f, g, or h with a strong piece on it. During this step, it was changed so that only open or half-open files g or h counted.

Fig. 3. Computer: *"Why is it NOT possible for White to attack the opponent's king in this position, comparing to the one on the left-hand diagram in Fig. 2?"* Again, the experts have been asked to give a description based on their general knowledge in the presented domain. Their answer was: *"In this position, more black pieces participate in the defence of the king."*



Then, the position shown in Fig. 3 became the new counter-example. The experts again explained the difference between the critical and the new counter-example: *"In this position, more black pieces participate in the defence of the king."* A new attribute, *defends_by_pieces_around_king*, was therefore programmed and included into the domain. It is the number of squares in the square f5-h5-h7-f7 defended by black pieces, not counting black pawns. Based on the experts explanation, the previous argument was now extended by *"defends_by_pieces_around_king* is low." With the new attribute and the extended argument the following rule was obtained:

```
IF open_file is true AND attacking_pieces > 1
  AND defends_by_pieces_around_king <=13
THEN ATTACK = true;
```

The algorithm could not find any counter-examples to the above argument, therefore the ABML refinement loop proceeded to the next iteration.

In total, the ABML process consisted of 38 iterations. In ten of those iterations, experts decided to change the class of the example and in others arguments

were provided. Arguments altogether suggested 14 new attributes that increased the number of attributes to 26. The final model contained 16 rules and the improvement of the model's quality is obvious: the classification accuracy increased to 91%, precision to 93% and recall to 92%.

A possible criticism of this experiment could be the high number of iterations between the expert and the algorithm. According to our previous experience with ABCN2, this number was always considerably smaller. At the moment, we are unable to explain it; finding an explanation for this phenomenon remains for the future work.

5 Comparison of Explanation Power

The previous section showed the difference in discrimination power between the stand-alone machine learning and ABML. While the increased accuracy of ABML is certainly useful, it is the reasons for this or that decision that are likely to be of even greater importance from the viewpoint of creating intelligent commentary. The present section thus attempts to evaluate the difference in explanation power between the two paradigms. The evaluation mostly relies on the opinions of chess experts, consequently being somewhat subjective.

Let us start by appraising the rules themselves. Stand-alone ML produced 12 rules. Three of those rules included at least one of the two terms that the experts marked as counter-intuitive if not downright absurd. These terms are: (1) the attacking side has *less* than three pieces on the offensive, and (2) the defending side has *more* than two pawns shielding the king. Even without expert opinion it is clear that it should be just the other way around: the more pieces the attacking side has, and the lesser pawn shield the defending side has, the higher chances of an attack. However, it is not uncommon for ML to produce such seemingly nonsensical explanations as an artefact of the data. In our case, it is likely the consequence that no learning positions where one side would have many attacking pieces and the other no pawn shield were included. Such positions are rarely, if ever, encountered in professional play as the weaker side would already resign by then.

On the other hand, ABML produced 16 rules, and none of them included any illogical terms as deemed by the experts.

The emphasis we give to such illogical terms is not unwarranted. While such terms can often be ignored when pure discriminative power is the (main) issue, they are very harmful when the model is used to explain the data, and even more so when used for tutoring. A teacher using illogical argumentation (even of a correct decision) is never a good idea.

How harmful are the three rules using illogical terms in our case? ML rules *correctly* decided that one side aims at attacking the opposing king in 131 cases, however, the harmful argumentation was present in no less than 112 of these 131 cases. Imagine your teacher using the wrong argumentation in 85% of the cases.

The difference in the number of rules between the two paradigms is also worth noting. Basically, each rule represents a specific pattern of arguing in favor of

one side aiming to attack the opponent's king. Having more rules thus means having a wider range of concepts with which the system can explain the target notion. Such increased richness of the explanation language is usually a desirable property when building a commenting/tutoring system.

Figures 4 and 5 illustrate the difference in explanation power between the rules learned by the two paradigms. The second figure presents a position in which the ML rules do not detect the attack, while the first figure shows position where both ML and ABML correctly detected the attack, yet the explanation they offered differs significantly. Reason that seemed illogical to experts is written in *italic*. The chess experts examined over 30 explanations and unanimously preferred the richer and correct explanation given by the ABML.

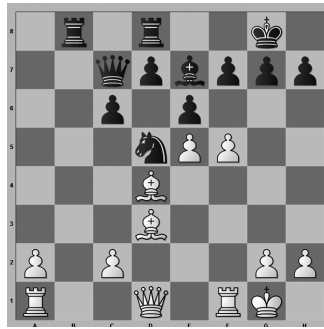


Fig. 4. **ML rules:** is attack, since $\text{pawn_attack} > 0$ (number of pawns on files f,g,h and on rows 4,5,6), $\text{pawn_defense} > 2$ (number of black pawns surrounding the black king), and $\text{pieces_on_king_side} > 0$ (number of white pieces without pawns on files f,g,h). **ABML rules:** is attack, since $\text{strong_piece_attack} > 0$ (number of strong pieces that can get to an open or semi open file on files f,g,h in 1 move), $\text{defends_pieces_around_king} < 6$ (number of squares defended by pieces around black king), and $\text{pawns_fifth_row} > 0$ (number of white pawns on files f,g,h on fifth row).

6 Related Work

The learning algorithm our work is based on, ABCN2, combines expert explanations with a rule learning algorithm, which can be seen as a special type of integrating machine learning and knowledge acquisition. It was already shown that a combination of a domain expert and machine learning yields the best results [14]. In the field of inductive logic programming (ILP), several approaches were proposed and they all can integrate prior expert knowledge with machine learning (e.g. HYPER and FOIL). Additionally, ABML refinement loop is also an interactive algorithm; it queries experts in turns for more knowledge. In literature, several similar approaches were already proposed, e.g. [1, 9]. However, these systems require experts to directly revise the learned models or to provide new attributes that would enable further learning, whereas in our system, the experts are asked to explain *particular* learning examples. We prefer the latter

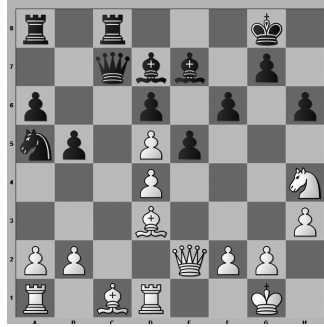


Fig. 5. ML rules: no attack. **ABML rules:** is attack, since `attacking_pieces > 2` (number of pieces attacking area around black king), `defends_pieces_around_king < 6`, `knight_go_on_safe_spot > 0` (number of squares around king where knight can be safely positioned), and `close_defending_pieces < 2` (number of black pieces standing close to the black king)

principle, since empirical observations have shown that humans are better at providing specific explanations than providing generic knowledge pertaining to the problem.

According to its name, explanation-based generalization (EBG) [5] could be the closest relative to ABML. However, this technique heavily relies on the provided general prior knowledge (not explanations). First, EBG uses this prior knowledge to automatically construct explanations of individual learning examples, and the “learned” model is then a logical generalization of these explanations. Note that EBG is not inductive learning; it assumes perfect and complete knowledge of the domain, which makes it inapplicable in domains where complete knowledge is unavailable.

An interesting extension of EBG is the DISCIPLE system [13, 12] for learning rules in problem solving domains. This system uses the EBG technique if perfect domain theory is available. However, in the case of imperfect theory, the system will ask the expert for explanation of examples and use these explanations to construct general rules. Knowledge elicitation through explanations in DISCIPLE is similar to that of ABML, however the methods differ in the selection of critical examples and how expert’s explanations are used in learning.

7 Conclusion

We argued a case for using a new paradigm, ABML, instead of classical machine learning for the purpose of learning high-level concepts and their explanations. One potential use of such concepts and explanations is for generating automatic commentary and/or tutoring. The paper showed how the interaction between the experts and ABML can lead to improved argumentation power of the learned models that go along with the increased accuracy of such models.

The concept chosen for the case study in the paper is a very complex one, especially so, because its notion is a bit vague. However, we believe the concept

was successfully grasped by the final model, albeit requiring more work than initially thought. The amount of work required seems disproportionate in comparison with other, similar concepts that we successfully learned in the past. It is possible, though, that the notion of an attack in chess is just vague enough for the experts to be a very difficult problem to tackle. An analysis of this is an obvious avenue for the future work.

Acknowledgments. This work was partly funded by ARRS, Slovenian Research Agency.

References

1. Wray Buntine and David Stirling. Interactive induction. *Machine intelligence*, 12:121–137, 1991.
2. Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In *Machine Learning - Proceeding of the Fifth European Conference (EWSL-91)*, pages 151–163, Berlin, 1991.
3. Edward A. Feigenbaum. Some challenges and grand challenges for computational intelligence. *Source Journal of the ACM*, 50(1):32–40, 2003.
4. Matej Guid, Martin Možina, Jana Krivec, Aleksander Sadikov, and Ivan Bratko. Learning positional features for annotating chess games: A case study. In *Lecture Notes in Computer Science*, volume 5131, pages 192–204, 2008.
5. Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.
6. Martin Možina, Jure Žabkar, and Ivan Bratko. Argument based machine learning. *Artificial Intelligence*, 171(10/15):922–937, 2007.
7. Michael J. Pazzani. Influence of prior knowledge on concept acquisition: Experimental and computational results. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 17:416–432, 1991.
8. Henry Prakken and Gerard Vreeswijk. *Handbook of Philosophical Logic, second edition*, volume 4, chapter Logics for Defeasible Argumentation, pages 218–319. Kluwer Academic Publishers, Dordrecht etc, 2002.
9. Debbie Richards. Ripple down rules: a technique for acquiring knowledge. pages 207–226, 2003.
10. Jeremy Roschelle. Learning in interactive environments: Prior knowledge and new experience. In J. H. Falk and L. D. Dierking, editors, *Public institutions for personal learning: Establishing a research agenda*, pages 37–51, 1995.
11. Aleksander Sadikov, Martin Možina, Matej Guid, Jana Krivec, and Ivan Bratko. Automated chess tutor. In *Proceedings of the 5th International Conference on Computers and Games*, 2006.
12. G. Tecuci, M. Boicu, C. Boicu, D. Marcu, B. Stanescu, and M. Barbulescu. The disciple-rkf learning and reasoning agent. *Computational Intelligence*, 21(4):462–479, 2005.
13. G. Tecuci and Y. Kodratoff. Apprenticeship learning in imperfect domain theories. *Machine Learning: An Artificial Intelligence Approach*, 3:514–551, 1990.
14. Geoffrey I. Webb, Jason Wells, and Zijian Zheng. An experimental evaluation of integrating machine learning with knowledge acquisition. *Mach. Learn.*, 35(1):5–23, 1999.

Explanations in Dialogue Systems through Uncertain RDF Knowledge Bases

Daniel Sonntag and Martin Theobald

German Research Center for Artificial Intelligence DFKI GmbH
and Max Planck Institute for Informatics (MPII), Stuhlsatzenhausweg,
66123 Saarbruecken, Germany

Abstract. We implemented a generic dialogue shell that can be configured for and applied to domain-specific dialogue applications. The dialogue system works robustly for a new domain when the application backend can automatically infer previously unknown knowledge (facts) and provide explanations for the inference steps involved. For this purpose, we employ URDF, a query engine for uncertain and potentially inconsistent RDF knowledge bases. URDF supports rule-based, first-order predicate logic as used in OWL-Lite and OWL-DL, with simple and effective top-down reasoning capabilities. This mechanism also generates explanation graphs. These graphs can then be displayed in the GUI of the dialogue shell and help the user understand the underlying reasoning processes. We believe that proper explanations are a main factor for increasing the level of user trust in end-to-end human-computer interaction systems.

1 Introduction

Multimodal interaction with large and dynamic data repositories is an important topic for the next generation of human-computer interaction systems. Over the last several years, we have focused on the idea that Semantic Web [1] data structures provide new opportunities for *semantically-enabled user interfaces*. The explicit representation of the *meaning* of data allows us to (1) transcend traditional keyboard and mouse interaction metaphors, and (2) provide representation structures for more complex, collaborative interaction scenarios with more complex result presentations. Over the last years, we have adhered strictly to the developed rule “No presentation without representation,” in order to log the state of the dialogue system, the displayed elements, and the user queries explicitly.

On the database side, a new area of information retrieval (IR) has begun with the advent of structured databases in Semantic Web RDF structures and respective query languages. The dominant query language for these RDF¹ repositories is the W3C recommendation SPARQL². For the next generation of human-computer interaction systems, explanation-based inference during data retrieval and uncertain knowledge plays a major role. In order to implement these properties, we use an efficient reasoning framework for graph-based, non-schematic RDF knowledge bases and SPARQL-like

¹ See <http://www.w3.org/TR/rdf-primer/> and <http://www.w3.org/TR/rdf-schema/>.

² <http://www.w3.org/TR/rdf-sparql-query>

queries, Uncertain RDF (URDF). URDF augments first-order reasoning by combining soft rules, with Datalog-style recursive implications, and hard rules, in the shape of mutually exclusive sets of facts. It incorporates the common possible world semantics with independent base facts as it is prevalent in current probabilistic database approaches. And it supports semantically more expressive, probabilistic first-order representations, like, for example, Markov Logic Networks.

In this paper, we discuss a prototype system which provides a dialogue-based interaction with such a probabilistic advanced database while following the URDF model. URDF allow us to infer and present uncertain knowledge from Semantic Web databases in the multimodal dialogue context. More precisely, we gain graph-based knowledge for dialogue-based explanations with confidences for inferred knowledge. While incorporating the URDF functionality into a dialogue system, we provide a first prototype implementation of an explanation-aware multimodal dialogue system (as implemented in the Comet music retrieval system [2] and in the dialogue system for the medical domain [3], but without URDF and explanation knowledge). Such dialogue systems can answer complex questions and provide additional multimedia material such as graphs or videos.

Generally speaking, dialogue-based question answering (QA) allows a user to pose questions in natural speech, followed by answers presented in a concise textual form with multimedia material [4]. For example, “Who is the German Chancellor?” The short answer is “Angela Merkel” accompanied by a picture. The user, however, should not only be presented the factual answers to such questions, but also some explanations about the actual QA process. As [5] show, proper explanations are one main factor that influences the level of user trust in complex (adaptive) artificial intelligence systems. Deriving and using explanations in dialogue-based QA is a unique opportunity for enhancing trust especially in uncertain, inferred answers in human-computer interaction systems.

The paper is structured as follows: Section 2 discusses related work and Section 3 explains the dialogue system framework. The URDF framework is presented in Section 4, followed by an example dialogue (Section 5) and our conclusions (Section 6).

2 Related Work

Prominent examples of dialogue platforms include OOA [6], TRIPS [7], and Galaxy Communicator [8]; these infrastructures mainly address the interconnection of heterogeneous software components. A comprehensive overview of ontology-based dialogue processing and the systematic realisation of these properties can be found in [9], pp.71-131. Many systems are available that translate natural language input into structured ontological representations (e.g., AquaLog [10]), port the language to specific domains, e.g., ORAKEL [11], or use reformulated semantic structures NLION [12]. AquaLog, for example, presents a solution for a rapid customisation of the system for a particular ontology; with ORAKEL a system engineer can adapt the natural language understanding (NLU) component in several cycles thereby customising the interface to a certain knowledge domain; and NLION uses shallow natural language processing techniques (i.e., spell checking, stemming, and compound detection) to realise a single seman-

tic concept or an ontology property. All of them support the translation to SPARQL queries in principal. However, all of them deal with written keywords or simple semantic relations, e.g., *X isDefinedAs Y*. They do not focus on the much more complex explanation-based answering process while using a dialogue system.³

As introduced, the dominant query language for RDF repositories is the W3C recommendation SPARQL. Similar RDF-based query languages are worth mentioning, such as RDQL⁴ or SERQL⁵. These languages are based on the notion of RDF triple patterns, which can be connected via several query operators such as “union” or “filter”. In previous implementations of dialogue system backends without URDF, we used SPARQL queries because they are the de facto standard. We also used the resources in the Linked Data framework (see [13]). The RDF triple structure itself, which is used in Linked Data, represents enough structure to be called a database index, which maps a wildcard triple pattern onto the matching concrete data triples. Although the Linked Data sources are updated frequently, they can be considered rather static, i.e., OWL-style reasoning about these sources is normally not provided. Moreover, the SPARQL 2 specification (which most of these endpoints implement) provides support for operators such as “group by” or aggregate functions (e.g., COUNT, MIN, MAX, SUM). In the context of unstructured natural language input, SPARQL also provides convenient operator extension, i.e., the “filter” operator, to specify free text searches and even regular expressions based on operations for regular expressions. Examples of how these operators can be used in the context of integrating Linked Data for semantic dialogue and backend access can be found in [14].

3 Dialogue System Framework

In earlier projects [15, 16] we integrated different sub-components to multimodal interaction systems. In the context of the new THESEUS programme⁶, we then implemented a situation-aware dialogue shell for semantic access to image media, their annotations, and additional textual material. We use a distributed, ontology-based, dialogue system architecture, where every major component for speech understanding, dialogue management, or speech synthesis can be run on a different host, increasing the scalability of the overall system.⁷ A shared representation and a common knowledge base ease the dataflow within the system and avoid costly and error-prone transformation processes (c.f. “No presentation without representation”). More precisely, an ontology-based representation of a user query can be used to create a query that can be posed to a (U)RDF endpoint.

³ In addition, these systems directly transfer the input to the desired SPARQL queries without dealing with the complex influences of message passing in dialogue frameworks or input fusion.

⁴ <http://www.w3.org/Submission/RDQL/>

⁵ <http://www.openrdf.org/doc/sesame/users/ch06.html>

⁶ <http://www.theseus-programm.de>

⁷ The dialogue system architecture is based on a generic framework for implementing multimodal dialogue systems (ODP platform, available at <http://www.semvox.de/>).

The dialogue system acts as the middleware between the clients and the backend services (i.e., the RDF repositories with an online API) that hide complexity from the user by presenting aggregated ontological data. Figure 1 shows the dialogue system architecture. The client provides means to connect to the dialogue system via the *event bus*, to notify it of occurred events, to record and play back audio streams, and to render the received display data obtained from the *dialogue system/dialogue manager*. The generated (U)RDF queries are then processed by the backend system (Remote RDF Repository) in order to retrieve the requested entities.

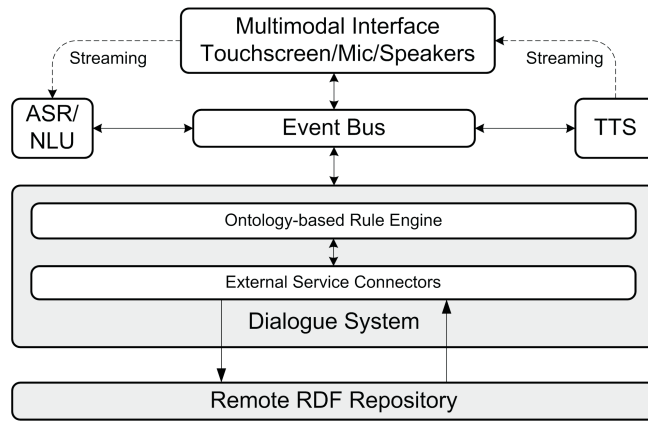


Fig. 1: Dialogue System Architecture

A central building block for component development and an integral part of the dialogue middleware is the included application programming interface for the efficient representation of ontology-based data using extended Typed Feature Structures (eTFS). As described in [17], the eTFS API is tightly integrated into a production rule system which enables a declarative specification of the processing logic in terms of production rules. While processing the user input in the dialogue system, the output of the fusion step⁸ is transferred to the backend system. Figure 2 provides a high-level view and rough sketch of the basic processing chain within the typical QA process. In this work, the backend access has been extended by addressing URDF. We use an Apache Tomcat server⁹ for this purpose. The presentation has to be adapted to the result of the URDF process, i.e., graph-based explanations.

⁸ A modality fusion component keeps track of the ongoing discourse, completes different types of anaphora, and merges input from different modalities. We use a production rule system, FADE, which is part of the ODP distribution.

⁹ <http://tomcat.apache.org/>

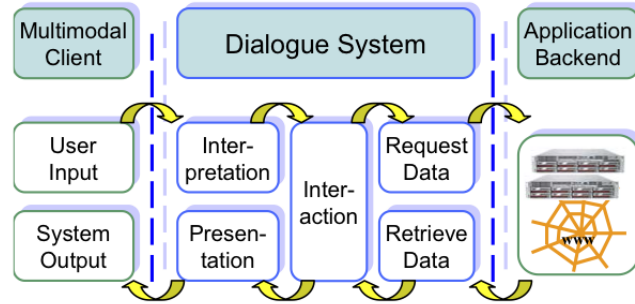


Fig. 2: Basic building blocks and core workflow of multimodal dialogue processing

4 URDF Framework

The URDF project, which is currently under development at the Max Planck Institute for Informatics, aims to enhance SPARQL-style query processing over RDF knowledge bases with simple and effective, top-down reasoning capabilities. Specifically, URDF supports rule-based, first-order reasoning concepts known from OWL-Lite and OWL-DL, thus capturing a decidable subset of first-order predicate logic. Moreover, since information extraction on the Web is often an iterative and inherently noisy process, URDF explicitly targets the resolution of *inconsistencies* between the underlying RDF facts and the inference rules. URDF also augments first-order reasoning by combining *soft rules*, which may be violated by some instances (facts) in the knowledge base, and *hard rules*, which may not be violated by any instance and can therefore be employed as hard consistency constraints, e.g., for capturing functional properties in OWL-Lite. Key to our approach for reasoning over uncertain data and resolving inconsistencies directly at query time is a novel and efficient approximation algorithm for a generalised version of the Weighted MAX-SAT problem, which allows URDF to dynamically cope with noisy data and/or evolving knowledge bases and changing domain constraints (rules). A further key feature of URDF is the ability to capture the resolution steps employed to infer answers by the reasoner in the form of an acyclic derivation graph over grounded rules (aka data “provenance” [18] or “lineage” [19]). This directed acyclic graph (DAG, cf. answer graphs) structure connects both base and derived facts via the rules that were used for grounding, which can be used to *explain* the answers given to a query and represent this explanation in graphical form to the user.

4.1 Representation Model and Expressiveness

URDF considers a *knowledge base* $\mathcal{KB} = \{\mathcal{F}, \mathcal{C}, \mathcal{S}\}$ as a triple consisting of RDF base facts $\mathcal{F}$, soft clauses $\mathcal{C}$, and hard (i.e., strict) rules $\mathcal{S}$. An *RDF graph* is a directed, labeled multi-graph, in which nodes are entities (such as individuals and literals), and labeled edges represent relationships between the entities. For example, an RDF graph can have an edge between the entity *Ullman* and the entity *Stanford*. This edge would be

labeled with the relation name *worksAt*. More formally, an RDF graph over a finite set of relations Rel and a finite set of entities $Ent \supseteq Rel$ is a set of triplets (or facts) $\mathcal{F} \subset (Rel \times Ent \times Ent)$. RDF allows two entities to be connected by multiple relations (e.g., two people can be colleagues and friends at the same time). Thus, facts express binary relationships between entities. For readability, we will write a fact (x, r, y) in common prefix notation as $r(x, y)$.

As opposed to OWL, RDF cannot directly express relationships between facts (i.e., “facts over facts”). Relationships with higher arity can however be represented by introducing an *event entity*, i.e., a new entity that stands in binary relationships with all arguments of the n -ary fact. Alternatively, n -ary relationships can be represented using *reification* [20]. In RDF graphs, there is a distinction between individual entities (such as *Albert Einstein*) and class entities (such as the class *physicist*). Individuals are linked by the *type* relationship to their class. For example, *Albert Einstein* is linked to the class *physicist* by a statement (*AlbertEinstein*, *type*, *physicist*). The classes themselves form a hierarchy. More general classes (such as *scientist*) include more specific classes (such as *physicist*). This hierarchy is expressed in RDF by edges with the *subclassOf* relationship: (*artist*, *subclassOf*, *singer*).

Soft Rules We consider first-order logic rules over RDF facts. A *grounded soft rule* over a set $\mathcal{F}$ of RDF facts is a set $C \subseteq \mathcal{F}$ of facts, where each atomic fact $f \in C$ is marked as either positive or negative and thus becomes a *literal*. For example, a grounded rule could be:

$$\{\neg \text{worksAt}(\text{Ullman}, \text{Stanford}), \text{livesIn}(\text{Ullman}, \text{Stanford})\}_{[0.4]}$$

Each soft rule is assigned a non-negative, real-valued weight. A higher weight indicates that matching the rules is of higher importance than matching a rule with a lower weight. To simplify talking about grounded rules of the same shape, we introduce *non-grounded rules*. A non-grounded rule C' is a grounded rule C over a set of facts in $\mathcal{F}$, where one or more entities are replaced by variables. A non-grounded rule C' over $\mathcal{F}$ implicitly stands for all grounded rules C that can be obtained by substituting the variables in C' by entities. Thus, the following rule subsumes the aforementioned grounded rule:

$$\{\neg \text{worksAt}(\text{Ullman}, x), \text{livesIn}(\text{Ullman}, x)\}_{[0.4]}$$

When grounded, the weight of the ungrounded rule is propagated to all its groundings. We use non-grounded rules solely to increase readability. We allow only *Horn rules*, i.e., rules where at most one literal is positive. Horn rules with exactly one positive literal can equivalently be rewritten as implications, in which all literals are positive. When written as implication, the *body* of a rule is a conjunction and the *head* consists of a single literal. In a first-order representation, only simple literals with no nested predicates or function calls are allowed in the rules. We can, however, extend the expressiveness of our reasoner (and yet remain in first-order) by allowing also rules with simple *arithmetic predicates*, which are “closed” within the rule, i.e., they can be evaluated (and thus be grounded) on-the-fly from the given variable bindings when the rule is processed. A grounded soft rule corresponds to a disjunction of literals, a so-called *clause*.

Hard Rules Hard rules are a distinct set of rules which define *mutually exclusive* sets of facts. Similarly to soft rules, hard rules can be expressed both in grounded and non-grounded form. A *grounded hard rule* is a set of facts $S \subseteq \mathcal{F}$ (also called a *competitor set*) that enforces the following constraint: a possible world $p : \mathcal{F} \rightarrow \{true, false\}$ can assign *true* to at most one fact $f \in S$. For example, the following hard rule

$$\{ \text{bornIn}(\text{Angela_Merkel}, \text{Hamburg}) , \\ \text{bornIn}(\text{Angela_Merkel}, \text{München}) , \\ \text{bornIn}(\text{Angela_Merkel}, \text{Stuttgart}) \} \blacksquare$$

specifies that *Angela Merkel* could be born in at most one out of the above cities. Similarly to soft rules, we introduce *non-grounded hard rules*, where constants may be replaced by variables. For example, $\text{bornIn}(\text{Angela_Merkel}, x)$ may be used to mark all the possible birth places of *Angela Merkel* in the knowledge base as mutually exclusive. Hard rules may not be violated and thus have no weights assigned (hence they are marked by a $\blacksquare$). For expressing these mutual-exclusion constraints, the hard rules encode special Horn clauses with only negated literals. Equivalently, they can be rewritten as a number of conjunctions over binary Horn clauses with pair-wisely negated literals.

4.2 Reasoning Framework

The URDF reasoning framework combines classic first-order reasoning with a generalised Weighted MAX-SAT solver over both soft and hard rules. Query processing with URDF consists of two phases: 1) lookups of basic query patterns against the knowledge base, which involves both direct lookups of base facts in the knowledge base, but also recursively grounding rules and inferring new facts; and 2) resolving potential inconsistencies by a second reasoning step in the form of a Weighted MAX-SAT solver, which yields the final truth assignments to candidate answers obtained from the previous grounding step. That is, given a query in the form of a set of non-grounded atoms, we aim to find an assignment of truth values to the grounded query atoms (and all other grounded facts that are relevant for answering the query), such that the sum of the weights over the satisfied soft rules is maximised, without violating any of the hard constraints.

SLD Resolution and Dependency Graph Construction In the absence of any rules, URDF conforms to a standard (conjunctive) SPARQL engine, with the returned facts consisting only of grounded query atoms over base facts $\mathcal{F}$. URDF, however, allows for the formulation of recursive rules (i.e., with the same predicate occurring in the head as well as in the body of a rule), as well as mutually recursive sets of rules (i.e., with one rule producing grounded facts as input to another rule). Rather than reasoning about all facts in the knowledge base (which would be infeasible at query time), URDF investigates efficient top-down resolution algorithms for grounding rules against the knowledge base. Instead, we compute the so-called *dependency graph*, which consists only of facts in the knowledge base which are relevant for answering the query (including lineage pointers to grounded rules for the derived facts). Dependency graph construction is performed via SLD resolution [21], which is similar to the resolution strategy used in

Prolog and Datalog. Furthermore, SLD resolution over soft rules is also extended by a separate grounding phase for the hard rules (see [22] for algorithmic details).

Resolving Inconsistencies After dependency graph construction, URDF constructs a propositional Boolean formula in conjunctive normal form (CNF) from all the lineage pointers to the grounded soft and hard rules, as well as the base facts in $\mathcal{F}$ used for grounding the rules. Since all rules are readily available in Horn clause form, the CNF can efficiently be constructed as a conjunction of all rules which are embedded in the dependency graph after the grounding phase. URDF employs an efficient approximation algorithm for a variant of the well-known Weighted MAX-SAT problem, which is specifically tailored to our setting, i.e., by considering a generalisation that is able to capture the presence of both soft rules (which may be violated in the MAX-SAT solution) and hard rules (which may not be violated by the solution). The MAX-SAT solver finally assigns *true* to only a subset of answer facts which are free of inconsistencies.

5 Example Dialogue

The following dialogue illustrates a user’s practical interest in using a dialogue interface on top of a semantic URDF search engine for answering natural language questions. The dialogue concentrates around the questions about the URDF contents, i.e., factoid questions about celebrities, and the multimodal presentation of answer content. We use a big touchscreen installation for the presentation of the speech-based user requests (similar to our installation in [2]). On the touchscreen (cf. Figure 3), we display so-called semantic interface elements (SIEs). A SIE is a window on the GUI which displays aggregated multimedia results. For example, the Video SIE (Figure 3, left) displays videos from a YouTube API¹⁰.

- 1 U: “Where is Angela Merkel born?”
- 2 S: Shows corresponding result in a SIE.
- 3 U: “What do Angela Merkel and Al Gore have in common?”
- 4 S: Shows corresponding relation graph.
- 5 U: “Where does he live?”
- 6 S: Shows corresponding relation graph.

Synthesises a summary of the graph’s interpretation.

In the example dialogue, turn (1) results in a structured result display (Figure 4, “Hamburg” is also synthesised) according to factoid QA paradigm. The answer can be looked up directly in the URDF database (only hard rules apply in case of inconsistent knowledge). Turn (3) results in the display of a relation graph (similar to the graph in Figure 3, right). The last user turn (5) “Where does he live?” is of particular interest from both a linguistic and database standpoint. We interpret the utterance as a deictic one, where the determination of the (celebrity) referent is dependent on the context in which it is said. Here, of course, the context is the referent “Al Gore”, stored in the discourse context. In addition, the result is a complex explanation graph (Figure 5) derived from soft rules.

¹⁰ <http://code.google.com/apis/youtube/overview.html>

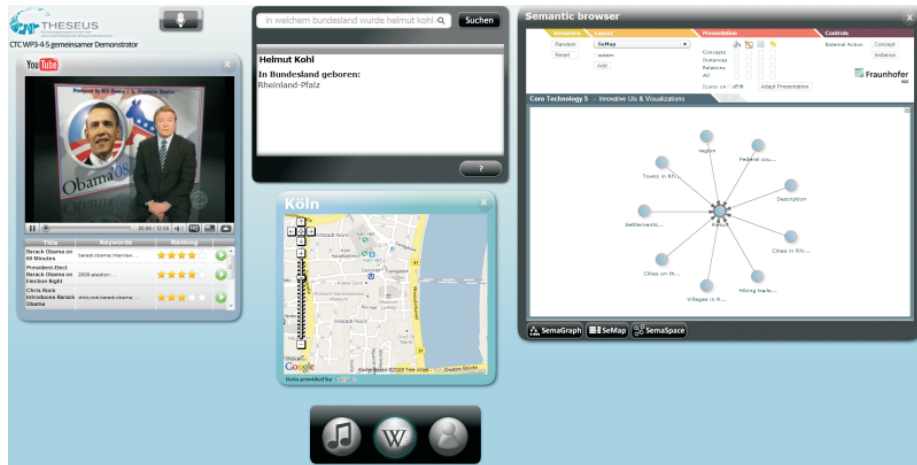


Fig. 3: Touchscreen surface with several semantic interface elements (SIEs)

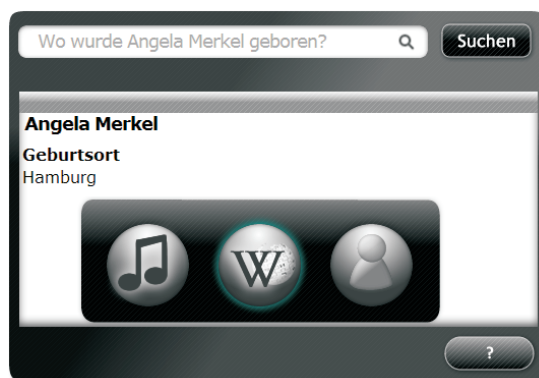


Fig. 4: Factoid answer SIE for the question "Where is Angela Merkel born?"

6 Conclusion

We have discussed explanations in dialogue systems through Uncertain RDF knowledge bases and presented URDF, a query engine for uncertain and potentially inconsistent RDF knowledge bases. This new backend can be integrated into a speech-based dialogue system to answer questions about a specific domain. Whereas factoid questions can be answered by state-of-the-art backend repositories via SPARQL queries, URDF provides the unique opportunity to also reason about uncertain knowledge and provide explanation graphs in the context of multimodal QA.

In our multimodal application scenario, a more complex and tighter integration of the provided result graphs has to be investigated. By extending the functional dialogue shell modules for a more complex dialogue behaviour on the result structures, we should be able to not only display the result graphs, but to paraphrase the result contents in natural language form as well. For example, the result graph could be presented in conjunction with the speech synthesis “I think he lives in Washington, D.C., because his wife, Tipper Gore, also lives there.” This would, however, heavily exceed our current natural language generation capabilities, but pave the way toward speech-based explanations, a main factor for increasing the level of user trust in end-to-end human-computer interaction systems. Automatically inferred knowledge by URDF provides a new data stream and explanations for future, artificial intelligence based interaction systems.

Acknowledgments

This research has been supported in part by the THESEUS programme funded by the German Federal Ministry of Economics and Technology (01MQ07016). Thanks go out to Robert Nesselrath, Yajing Zang, Matthieu Deru, Simon Bergweiler, Gerhard Sonnenberg, Norbert Reithinger, Gerd Herzog, Alassane Ndiaye, Tilman Becker, Norbert Pflieger, Alexander Pfalzgraf, Jan Schehl, Jochen Steigner, and Colette Weihrauch for the implementation and evaluation of the dialogue infrastructure.

References

1. Fensel, D., Hendler, J.A., Lieberman, H., Wahlster, W., eds.: Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. In Fensel, D., Hendler, J.A., Lieberman, H., Wahlster, W., eds.: Spinning the Semantic Web, MIT Press (2003)
2. Sonntag, D., Deru, M., Bergweiler, S.: Design and Implementation of Combined Mobile and Touchscreen-Based Multimodal Web 3.0 Interfaces. In: Proceedings of the International Conference on Artificial Intelligence (ICAI). (2009)
3. Sonntag, D., Möller, M.: Unifying semantic annotation and querying in biomedical image repositories. In: Proceedings of International Conference on Knowledge Management and Information Sharing (KMIS). (2009)
4. Sonntag, D., Engel, R., Herzog, G., Pfalzgraf, A., Pflieger, N., Romanelli, M., Reithinger, N. In: SmartWeb Handheld—Multimodal Interaction with Ontological Knowledge Bases and Semantic Web Services. Volume 4451 of Lecture Notes in Computer Science. Springer (2007) 272–295

5. Glass, A., McGuinness, D.L., Wolverson, M.: Toward establishing trust in adaptive agents. In: *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, New York, NY, USA, ACM (2008) 227–236
6. Martin, D., Cheyer, A., Moran, D.: The Open Agent Architecture: a framework for building distributed software systems. *Applied Artificial Intelligence* **13**(1/2) (1999) 91–128
7. Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., Stent, A.: An Architecture for a Generic Dialogue Shell. *Natural Language Engineering* **6**(3) (2000) 1–16
8. Seneff, S., Lau, R., Polifroni, J.: Organization, Communication, and Control in the Galaxy-II Conversational System. In: *Proceedings of Eurospeech'99*, Budapest, Hungary (1999) 1271–1274
9. Sonntag, D.: *Ontologies and Adaptivity in Dialogue for Question Answering*. AKA and IOS Press, Heidelberg (2010)
10. Lopez, V., Pasin, M., Motta, E.: Aqualog: An ontology-portable question answering system for the semantic web. In: *Proceedings of ESWC*. (2005) 546–562
11. Cimiano, P., Haase, P., Heizmann, J.: Porting natural language interfaces between domains: an experimental user study with the orakel system. In: *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, New York, NY, USA, ACM (2007) 180–189
12. Ramachandran, V.A., Krishnamurthi, I.: Nlion: Natural language interface for querying ontologies. In: *COMPUTE '09: Proceedings of the 2nd Bangalore Annual Compute Conference on 2nd Bangalore Annual Compute Conference*, New York, NY, USA, ACM (2009) 1–4
13. Bizer, C.: The emerging web of linked data. *IEEE Intelligent Systems* **24**(5) (2009) 87–92
14. Sonntag, D., Kiesel, M.: Linked data integration for semantic dialogue and backend access. In: *AAAI Spring Symposium on Linked Data Meets Artificial Intelligence*. (2010)
15. Wahlster, W.: SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In Krah, R., Günther, D., eds.: *Proceedings of the Human Computer Interaction Status Conference 2003*, Berlin, Germany, DLR (2003) 47–62
16. Reithinger, N., Fedeler, D., Kumar, A., Lauer, C., Pecourt, E., Romary, L.: MIAMM - A Multimodal Dialogue System Using Haptics. In van Kuppevelt, J., Dybkjaer, L., Bernsen, N.O., eds.: *Advances in Natural Multimodal Dialogue Systems*. Springer (2005)
17. Schehl, J., Pfalzgraf, A., Pfleger, N., Steigner, J.: The BabbleTunes System. Talk to Your iPod! In: *Proceedings of the 10th International Conference on Multimodal Interfaces (ICMI)*. (2008)
18. Buneman, P., Tan, W.C.: Provenance in databases. In: *SIGMOD Conference*, New York, NY, USA (2007) 1171–1173
19. Benjelloun, O., Das Sarma, A., Halevy, A.Y., Theobald, M., Widom, J.: Databases with uncertainty and lineage. *The VLDB Journal* **17**(1) (2008) 243–264
20. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO - A Large Ontology from Wikipedia and WordNet. *Elsevier Journal of Web Semantics* **6**(3) (2008) 203–217
21. Apt, K.R., van Emden, M.H.: Contributions to the theory of logic programming. *J. ACM* **29**(3) (1982) 841–862
22. Theobald, M., Sozio, M., Suchanek, F., Nakashole, N.: URDF: Efficient reasoning in uncertain RDF knowledge bases with soft and hard rules. Technical Report MPII20105-002, Max Planck Institute Informatics (MPI-INF), Saarbruecken, Germany (2010)

Author Index

Atzmueller, Martin	1
Bacon, Liz	25
Bratko, Ivan	37
Dengel, Andreas	13
Forcher, Björn	13
Guid, Matej	37
Kapetanakis, Stelios	25
Krivec, Jana	37
Ma, Jixin	25
Mozina, Martin	37
Petridis, Miltos	25
Roth-Berghofer, Thomas	1, 13
Sadikov, Aleksander	37
Sintek, Michael	13
Sonntag, Daniel	49
Theobald, Martin	49