LNAI 6880

Ashwin Ram
Nirmalie Wiratunga (Eds.)

Case-Based Reasoning Research and Development

19th International Conference on Case-Based Reasoning, ICCBR 2011
London, UK, September 2011
Proceedings

Springer

Ashwin Ram   Nirmalie Wiratunga (Eds.)

Case-Based Reasoning Research and Development

19th International Conference
on Case-Based Reasoning, ICCBR 2011
London, UK, September 12-15, 2011
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Ashwin Ram
Georgia Institute of Technology
College of Computing, School of Interactive Computing
85 Fifth Street NW, Atlanta, GA 30308, USA
E-mail: ashwin@cc.gatech.edu

Nirmalie Wiratunga
The Robert Gordon University
School of Computing, Computing Technology Research Centre
St. Andrew Street, Aberdeen AB25 1HG, UK
E-mail: n.wiratunga@rgu.ac.uk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-23290-9 e-ISBN 978-3-642-23291-6
DOI 10.1007/978-3-642-23291-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011934745

CR Subject Classification (1998): I.2, H.3, H.4, H.5, J.1, H.2.8

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at ICCBR 2011: 19th International Conference on Case-Based Reasoning (http://www.iccbr.org/iccbr11/) held during September 11–14, 2011 in Greenwich, London. There were 67 submissions. Each submission was reviewed by at least three Program Committee members. The committee decided to accept 32 papers following a highly selective process. Of these papers, 22 were selected for oral presentation and 10 papers for poster presentation. Each submission was identified as fitting either the technical or applied stream and judged using the following criteria: relevance, significance, originality, technical quality and clarity. Scientific research formed the basis of judgement for technical paper significance and the social, economic or other commercial impact formed the basis of judgement for applied paper significance. The program also contained three invited talks which are included in this volume.

The International Conference on Case-Based Reasoning (ICCBR) is the pre-eminent international meeting on case-based reasoning (CBR). Previous ICCBR conferences have been held in Sesimbra, Portugal (1995), Providence, USA (1997), Seeon Monastery, Germany (1999), Vancouver, Canada (2001), Trondheim, Norway (2003), Chicago, USA (2005), Belfast, UK (2007), Seattle, USA (2009) and most recently in Alessandria, Italy (2010).

Day 1 of ICCBR 2011, Industry Day, provided hands-on experiences of using CBR in cutting-edge applications (e.g., a lessons learned system for the UK Health & Safety Laboratory, the History of 12 CBR systems at General Electric, CBR applications at the German Association of Machinery Manufacturers and a service report-based CBR approach to machine diagnosis). A key highlight was the invited talk on the Watson question answering system presented by William Murdock from IBM. This talk discussed how structure mapping (an algorithm originally developed for analogical reasoning) is applied to determine similarity between content in questions and passages. Another highlight was the third Annual Doctoral Consortium (DC) event which involved presentations by 11 research students in collaboration with their respective senior CBR research mentors. Day 2 featured topical workshops on CBR and the computer games industry, CBR and augmentation, different forms of CBR including process-oriented, cognitive and human-centered forms. These were complemented with the popular full-day live computer-cooking contest workshop with a new focus on adaptation and an open-feature challenge. Days 3 and 4 comprised presentations and posters on technical and applied CBR papers, as well as invited talks from two distinguished scholars: Kris Hammond, Northwestern University and Steffen Staab, University of Koblenz-Landau. Kris Hammond explored how the CBR core view of "Reasoning as Remembering" can be transformed into "Reasoning as Search" to enable the integration of Web resources within two CBR systems.

Steffen Staab, in his talk, discussed the role of similarity metrics for reasoning with ontologies and the resulting integration challenges therein.

The presentations and posters covered a wide range of CBR topics of interest both to practitioners and researchers, including CBR methodology covering case representation, similarity, retrieval, and adaptation; provenance and maintenance; recommender systems; multi-agent collaborative systems; data mining; time series analysis; Web applications; knowledge management; legal reasoning; healthcare systems and planning systems.

Many people participated in making ICCBR 2011 a success. Miltos Petridis, University of Brighton, UK, served as Conference Chair, with Nirmalie Wiratunga, Robert Gordon University, UK, and Ashwin Ram, Georgia Institute of Technology, USA, as Program Co-chairs. We would especially like to thank Belen Diaz Agudo, Complutense de Madrid, Spain, and Amelie Cordier, University Claude Bernard, France, for serving as Workshop Coordinators and Ian Watson, University of Auckland, for chairing Industry Day. We wish to thank David W. Aha for organizing the valuable doctoral consortium and seeking conference support funds for this event from the Office of Naval Research Global (ONRG). We are very grateful for the generous support of the sponsors of ICCBR 2011: Attensity for student bursaries in support of the computer cooking contest and to Amelie Cordier for organizing the contest; the University of Greenwich for administration and local area support and the British Computer Society for helping with registration packs.

We thank the Program Committee and all our additional reviewers for their thoughtful and timely participation in the paper selection process. We acknowledge the time and effort put in by the members of the Local Organizing Committee at the University of Greenwich, including all student helpers and in particular Liz Bacon, our Local Chair. Finally, we appreciate the support provided by EasyChair in the management of this conference and thank Springer for its continuing support in publishing the proceedings of ICCBR.

September 2011

Ashwin Ram
Nirmalie Wiratunga

Organization

Program Chairs

Ashwin Ram	Georgia Institute of Technology, USA
Nirmalie Wiratunga	The Robert Gordon University, UK

Conference Chair

Miltos Petridis	University of Brighton, UK

Industry Day Coordinator

Ian Watson	University of Auckland, New Zealand

Workshop Coordinators

Belén Díaz-Agudo	Universidad Complutense de Madrid, Spain
Amélie Cordier	Claude Bernard University, France

Doctoral Consortium Coordinator

David W. Aha	Naval Research Laboratory, USA

Local Organizer

Liz Bacon	University of Greenwich, UK

Program Committee

Agnar Aamodt	Norwegian University of Science and Technology, Norway
Klaus-Dieter Althoff	DFKI / University of Hildesheim, Germany
Josep Lluis Arcos	IIIA-CSIC, Spain
Kevin Ashley	University of Pittsburgh, USA
Ralph Bergmann	University of Trier, Germany
Isabelle Bichindaritz	University of Washington Tacoma, USA
Derek Bridge	University College Cork, Ireland
William Cheetham	General Electric, USA
Susan Craw	The Robert Gordon University, UK
Sarah Jane Delany	Dublin Institute of Technology, Ireland

Additional Reviewers

Table of Contents

Application Papers

Reasoning as Search: Supporting Reasoning with Distributed Memory*

Kristian J. Hammond

Northwestern University,
Department of Electrical Engineering and Computer Science,
2133 Sheridan Road,
Evanston, IL 60208

Abstract. The central idea behind Case-Based Reasoning has always been the notion that reasoning can be supported using memories of past problem solving. One bottleneck in this work has often been the development of case libraries needed to support this reasoning rather than the transformation of the cases themselves. In much of our work, we have taken an approach in which we treat web-recourses as the distributed knowledge engineering that can be integrated into memory- or case-based reasoning systems. We have been working on how we can take the core view of "Reasoning as Remembering" and transform it into "Reasoning as Search". The primary issues in this work are how to map problem-solving or task needs onto the queries required to find initial candidates, filter those candidates for relevance and then manage the exploitation of the results. I will outline how we have done this in two systems we have built recently, News at Seven and Baleen, systems that track the world of social media, news and the Web to support narrative generation.

Keywords: Case-based reasoning, CBR, narrative generation, search, intelligent information systems, dynamic memory, memory-based reasoning.

1 The Problem

One of the most persistent issues in Case-Based Reasoning (CBR) has been the need for robust case libraries that can used to support reasoning. In much the same way that rule-based systems were hampered by the need to build up domain level rule sets, CBR has been throttled by the need for large scale case bases.

There are many ways to address this problem that range from the use of existing libraries that were created for other reasons to the incremental learning of these libraries from completed problem solving exercises or the manual production of the cases needed to support a given system.

In most instances, the libraries and the languages that are used to represent cases within them are usually highly structured. Likewise, the queries that are used to

* Some of this material is based upon work supported by the National Science Foundation under Grants No. III-0917261 and 0535231. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

access them are also usually highly structured and aimed at getting those and only those cases that are genuinely relevant to the problem at hand.

There are exceptions, but in general, there remains an ongoing need for the production of extensive, high quality and robust case-libraries that are both well structured and welled indexed.

Much of the work we have been doing over the past few years has been aimed at trying to find ways in which reasoning can be supported in the absence of these requirements.

2 The Opportunity and Challenge

One approach to this problem is to step back from the dual assumptions that case libraries have to be associated of a single point of view, problem solving approach or even system and that they have to be well organized or structured. That is, shift perspective towards repositories that are the product of the distributed submission of information where there is little, if any, control over the structure of the items submitted. Which is to say, use the Web.

The opportunity here is that there is an ever-growing mass of information online that can be minded if you know exactly what you are looking for and know how to go about doing so. This information is accessible using search tools that have been tuned for the retrieval of the most relevant content given a specific query.

The challenge, of course, is that this "information" is not information at all, but instead unstructured free text that is only informative when processed either by humans who know something about what they are reading or by systems that have similar knowledge. And while there are many both commercial and openly available tools for doing web search, many are tuned for goals and needs that are very different than those of a case-based reasoning system.

The ultimate challenge is to find a way to look at different tasks from the point of view of information retrieval, the trade-offs between precision and recall and support of the kinds of modifications that have to be performed if we are to actually used the ”cases” that are found through search.

In order to deal with this challenge, we decided to focus on content generation as a task in general. This focus provided us with mechanisms for characterizing the goals and needs of our systems in terms of lexically based search along with a framework for doing syntactic and lexical manipulation of cases to fit the generation needs of a system.

3 Filling in the Narrative: News at Seven

News at Seven is a fully automated approach to creating broadcast news. Starting with a set of user preferences, the system finds relevant stories, edits these stories for broadcast presentation, uncovers relevant background visual materials (e.g., videos and still images), and augments the primary text using sources such as blogs and other commentary found on the Web. These resources are then used to drive the construction of new material that is, in turn, used to control a set of animated characters who reside in a virtual "performance world" we have created for them. The final output of the system is an online Flash presentation that uses animated avatars with generated speech and is modeled after traditional nightly news broadcasts. The system is robust, scalable, and highly flexible.

News at Seven can be used to generate a variety of program styles or dynamics, with specific anchor activities and interactions including traditional news but also entertainment stories (in an "Access Hollywood" style) and film reviews (paralleling Ebert and Roper).

It is this last dynamic that I want to focus on for the moment. The interaction that the system generates for this particular dynamic is driven by two elements. The first, and obvious one, is the name of a movie. The second is what people are saying about it in both the aggregate and the specific.

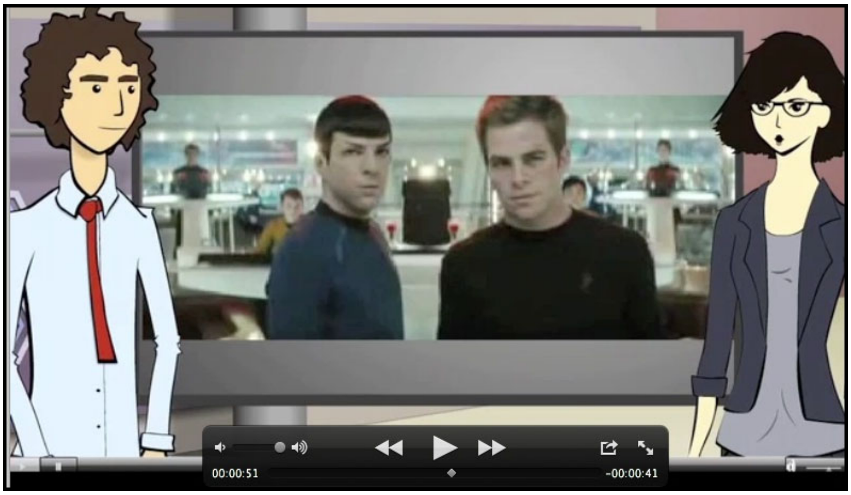


Fig. 1. News at Seven reviews Star Trek

Once the system is handed a movie title, it performs a series of searches aimed at figuring out, in general, what people think of this film. Although it is looking for an overarching opinion, it is not looking for a consensus so much as a characterization. Occasionally, everyone either hates or loves a film but more often, there are divided reactions to it. The goal of the system at this phase is to capture this characterization. To do this, the system builds a set of queries aimed at social media sites and platforms and gathers statements of opinion about the film that it categorizes as being positive, negative or neutral.

While doing this, there are some important elements of the processing that the system must attend to.

First, it has to make sure that the statements it is looking at are actually related to the movie whose title it used to craft its query. To do this, it needs to filter the statements using terms that tend to be used by moviegoers to describe their own actions. This allows the system to ignore a significant portion of the material that the search has pulled back.

Second, it has to do a credible job of determining sentiment with regard to the statements it has pulled down. While noisy, this can be done using standard sentiment terms, identification of negation, and proximity. Fortunately, at this stage, the system is looking for a fairly course grained characterization that it can use to drive the next stage of actual content generation.

The initial wave of retrieved content is used to determine what narrative *arcs* or *angles* are going to drive the generation process. In effect, the system is trying to figure out what kind of story it wants to tell. Is it a story about a blockbuster, about a film that has mixed reviews, a nice film that people like but not a lot of people see, etc? Once this is determined, the arc that dominates determines the dynamic (in terms of both what information is gathered and how it is presented) that will hold.

If the arc that holds is a mixed opinion arc, for example, the system then crafts a series of searches aimed at finding specific statements associated with the different film elements (actors, story, production value) that are strongly positive or negative. These statement are then used, with some minor syntactic modification, to create a back and forth dialog between two anchors. In cases of ambiguity, lack of syntactic clarity, grammatical problems, the statements are simply filtered out in favor of clearer and more useful ones. The result is a strikingly natural dialog that moves easily from point to point (driven by the narrative arc) focused on positive and negatives aspects of the film (provide by the secondary search) pulled together with interstitial material that the system uses to join the individual statements together.

Two relevant points here.

First, the tension between precision and recall that is the standard driver in search is muted here by the two roles that search plays. In the first phase of determining overall sentiment, recall is more important and any lack of precision results in noise the system is already dealing with. In the second phase, recall is completely irrelevant in that the system is actually looking for specific statements that fit its needs rather than a collection of statements.

Second, the two central processes of CBR, goal based retrieval and modification are still in play. The difference is simply that the retrieval is supported by a substrate of lexically based information retrieval rather a more structured approach and the modification is more syntactic in nature, changing the tense and structure of a statement to fit a new set of rhetorical goals.

One final note about News at Seven. One aspect of the system is that it searches for and modifies both video and stills for its use in crafting a user experience. In doing this, it is using the same core technology that it uses to find and modify text selections. In both these instances, it is defining an information goal, crafting a query aimed at finding a solution to that goal and then modifying the results it retrieves to best fit the problem at hand.

4 Serving the Narrative: Baleen

Baleen is part of the next wave of systems doing machine-generated content that are being developed by the company Narrative Science. Like News at Seven, the system is designed to use search to satisfy specific information goals. It however, is aimed at satisfying those goals using news rather than opinion.

Baleen's goals come from a parent system that is generating narratives as a product of data analysis rather than through search. At any given point in time, this parent system may be in the midst of generating a story in which the facts of the matter are clear in the data, but exogenous information that might explain those facts might not be directly available. In those situations, Baleen is called with explanatory goals that direct it to find the specific item that can then be incorporated into the ongoing story.

To make this clear, consider the example of the stock market. Occasionally companies (and their associated stock) undergo interesting changes. 52 week highs or lows, increase in price, increase in volume, etc. The stories associated the these changes can be written through a combination of tracking the current state of the world and then comparing the data pulled with historical information as well as data about other similar companies and how they are doing.

In crafting these stories, however, there is often a gap in the narrative in that the data alone does not explain *why* these changes have taken place. It is in these moments when Baleen is invoked.

Baleen is called with an information goal. In this instance the goal is to explain, for example, a sudden stock drop. Baleen uses existing data to determine the various ways a company can be described (its names and nick names) and the kind of industry the company is within. The latter is used to provide precise information about what actions (and thus terms) are appropriate to use in describing both positive and negative events. These three elements are combined to create an extremely focused query that is then used to access news sources.

The results of this search are then filtered using the same sort mechanism applied by News at Seven to make sure that the individual results are actually on point. The system is very aggressive about removing individual results in that, as with News at Seven, it is really only looking for a single result that can be modified to fit its needs. In order to verify results, however, it does need to look to see if entries cluster together, ignoring those individuals for which no cluster exists.

In the end, a single result is selected and then modified to fit the system's needs.

5 Conclusion

Both these systems are part of an ongoing push to make more and more use of the ambiguous, fluid and growing wealth of both structured and unstructured data that is the Web. Both however, are also cast in the mold of CBR with the notion that the massive knowledge engineering effort that is the Web will eventually create a case library for us that will be able to support most, if not all, of our reasoning needs.

Structure Mapping for Jeopardy! Clues

J. William Murdock

IBM T.J. Watson Research Center,
P.O. Box 704,
Yorktown Heights, NY 10598
murdockj@us.ibm.com

Abstract. The Jeopardy! television quiz show asks natural-language questions and requires natural-language answers. One useful source of information for answering Jeopardy! questions is text from written sources such as encyclopedias or news articles. A text passage may partially or fully indicate that some candidate answer is the correct answer to the question. Recognizing whether it does requires determining the extent to which what the passage is saying about the candidate answer is similar to what the question is saying about the desired answer. This paper describes how structure mapping [1] (an algorithm originally developed for analogical reasoning) is applied to determine similarity between content in questions and passages. That algorithm is one of many used in the Watson question answering system [2]. It contributes a significant amount to Watson's effectiveness.

1 Introduction

Watson is a question answering system built on a set of technologies known as DeepQA [2]. Watson has been customized and configured to compete at Jeopardy!, an American television quiz show. Watson takes in a question and produces a ranked list of answers with confidence scores attached to each of these answers.

One of the stages in the DeepQA question answering pipeline is deep evidence scoring. This stage receives as input a question and a candidate answer in the context of some supporting evidence (typically a passage containing that answer). Questions typically have a focus identified for them (i.e., the term in the question indicating the answer being sought). For example, a deep evidence scorer could be given a question like "*He was the first U.S. President*" and a passage like "*George Washington was the first U.S. President.*" "*He*" in the question will be marked as the focus. If the candidate answer is "*George Washington,*" each of the deep evidence scorers will attempt to determine the extent to which what the passage says about the "*George Washington*" addresses what the question asks about the "*He*". In this example, there is a perfect match, and all of Watson's deep evidence scoring mechanisms will conclude that this passage strongly supports the specified answer. However, other passages may answer the question less directly, or provide evidence for only a portion of what the question is asking for (e.g., that Washington was *a* president).

The examples above do not require any explicit analogy. One could envision passages that say (for example) that (a) Charles de Gaul was a great French general who fought for the liberation of France, (b) that Charles de Gaulle was the first president of the fifth republic of France, and (c) that George Washington was a great American general who fought for the liberation of the U.S.; by analogy, one might suspect that George Washington was the first president of the U.S. DeepQA does not explicitly do reasoning of this sort, but may in future work.

This paper provides a detailed description of one of Watson's deep evidence scoring algorithms: the Logical Form Answer Candidate Scorer (LFACS). LFACS uses a logical form of a question (e.g., a Jeopardy! clue) containing a specified focus term and the logical form of a passage containing a specified candidate. LFACS employs a structure mapping algorithm similar to the one described in [1]. LFACS embodies a variety of specializations of structure mapping that are driven by the nature of its task. For example, LFACS is *pragmatic* in the sense described in [3], because it has a specific inference it is intended to draw: the extent to which the candidate answer in the passage corresponds to the answer in the clue.

2 Role in the Architecture

The DeepQA architecture is described in [2]; this section provides a minimal description of the architecture to explain the context in which LFACS is used. DeepQA begins with **question analysis**, which applies a variety of natural-language processing algorithms to the question text. These algorithms include general purpose text processing such as parsing and semantic relation detection. They also include processing that is specific to analyzing questions, e.g., determining the focus. In Jeopardy! a question focus is often denoted by a pronoun with no anaphor or a common noun with the word "*this*" as a determiner. For example, in the question "*Ambrose Bierce penned this sardonic reference work in 1906*," the focus is "*work*." The focus is defined to be the term in the question that would correspond to the answer in a corresponding assertion.

Question analysis in DeepQA is followed by **primary search and candidate generation**, which finds candidate answers in variety of sources. Some of those sources are natural-language text while others are structured sources such as knowledge bases. Answers are subjected to preliminary scoring (including answer typing, etc.), and those answers that seem poor (i.e., have a confidence score below a fixed threshold, according to a statistical model) are filtered out.

All candidate answers that pass through the filter are then processed by **supporting evidence retrieval**. That component conducts a search for passages that contain the candidate answer and as many other terms from the question as possible. This retrieval step provides a set of potentially relevant passages for each answer, regardless of where it was originally found (text, knowledge bases, etc.). Candidate answers that were found in text will also have one or more passages from the primary search. Passages from both types of search are used as supporting passages.

The supporting passages are analyzed in **deep evidence scoring**, in which a variety of algorithms assess the degree to which the passage provides evidence in support of some candidate answer. LFACS, described below, is one of these deep evidence scoring components.

The **final merging and ranking** step combines equivalent candidate answers (e.g., "*Richard Nixon*" and "*Richard M. Nixon*") and determines the confidence that each answer is correct. It ranks the answers by their confidence scores. The final merging and ranking component uses statistical machine learning; the features used to compute a confidence for each answer come from algorithms throughout the pipeline. LFACS is one source of features used by this component.

3 Syntactic-Semantic Graphs

LFACS reasons over syntactic-semantic graphs of both the question and the passage. In these graphs, nodes are terms in the clue (e.g., a word or a proper name) and edges encode syntactic and/or semantic relations among those terms. The syntactic portions of the graph are derived from an English-Slot Grammar (ESG) parse [4]. The semantic portions of the graph are derived from pattern-based relation detectors. Syntactic relations are useful for identifying similarity when questions and passages have a similar structure (e.g., "He wrote *Utopia*" – "Thomas More wrote *Utopia*"). Semantic relations are useful when passages use different structures with equivalent meaning (e.g., "He wrote *Utopia*" – "Thomas More, author of *Utopia*"). Relation detection is very challenging and the relation detection capabilities in DeepQA, while very precise, have only a moderate level of coverage. LFACS can be effective when content in passages have similar structure or when they have similar semantics that fall within the coverage of our relation detectors. Because syntactic and semantic relations are combined in a single graph, LFACS can combine insights from each. For example, consider the following actual Jeopardy! clue:

It's believed Shakespeare wrote part of a 1595 play about this "Utopia" author.

Some content in the clue is covered by semantic relations such as the one between an author and a work by that author. However, there are other key relationships in this clue such as the one between a play and the person that the play is about. DeepQA does not have recognizers for this relationship, but is able to parse the text. Consider the following (made-up) sample passage:

We saw a 16th century play about Thomas More, who wrote Utopia.

The syntactic-semantic graph for this passage a semantic (*authorOf*) edge between *Thomas More* and *Utopia*; that edge matches the corresponding semantic edge in the graph of the clue. In addition, passage has syntactic edges that correspond to syntactic edges in the clue. *Thomas More* in the passage is the object of the preposition *about*, while the focus of the clue is the object of the preposition *about* in the clue. As a result the matching algorithm (see next section) is able to align the following terms in the clue to terms in the passage using semantic and/or syntactic edges: *1595*, *play*, *about*, *Utopia*, *author*. There are still some important terms in the clue that are not covered by this passage (e.g., *Shakespeare*). Our algorithm assesses the quantity and importance of the terms that it is able to align and asserts a numerical value for how strong it considers the match to be; that numerical value is used by the DeepQA final merger as one of the features that influences the evaluation of answers.

4 Algorithm

LFACS performs a form of structure mapping. The algorithm is similar to the one described in [1], with customization to reflect the nature of the content (extracted NLP results), the fact that LFACS has a single pre-specified inference to draw: Specifically, LFACS is trying to judge whether the passage provides support for a specific, designated candidate answer. Below are the key steps in structure mapping that are defined in [1], with descriptions of how those steps are realized in LFACS:

- Local Match Construction: LFACS matches both edges and nodes. Edges are matched using a formal ontology, e.g., the *authorOf* relation is a subrelation of the *creatorOfWork* relation. Nodes are matched using a variety of resources for determining equivalent terms, e.g., WordNet [5], Wikipedia redirects, and has specialized logic for matching dates, numbers, etc.
- Global Map Construction: Unlike [1], LFACS is only concerned with global matches that align the focus to the specified candidate answer. Thus global map construction begins with the focus and candidate answer and search outward from those nodes through the space of local matches. As in [1], the global match construction process ensures consistency of global maps, requiring that no single node in the question map to multiple nodes in the passage.
- Candidate Inference Construction: LFACS omits this step because the inference to be drawn is implied by its inputs (aligning the focus to the candidate answer).
- Match Evaluation: As in [1], the total score for a match in LFACS is the sum of the match scores for the local match hypotheses included in the maximal consistent global map. Local match scores in LFACS are computed using inverse-document frequency (IDF) from our text corpus. Terms with high IDF scores occur rarely in the corpus so the fact that they align with the clue is less likely to be a coincidence and thus more likely to imply that the answer is correct.

In using this algorithm, we have encountered a wide variety of technical issues that are specific to natural-language. For example, some concepts can be expressed as either a verb or a noun (e.g., *destroy-destruction*). We address those issues through some combination of graph preprocessing (e.g., adding edges to indicate the logical subject of *destruction* during relation detection) and specialized logic that is internal to the local match construction (e.g., allowing the *destroy* to match *destruction*).

Our approach to generating local match hypotheses mostly focuses on determining equivalence (or at least rough equivalence) between nodes. This focus reflects the fact that we are interested in similarity, but not analogy *per se*. If we were to try to address examples like the Charles de Gaul analogy in the introduction of this paper, we would need to relax those restrictions and adjust the confidence in our conclusions accordingly. This may be extremely important in domains where there is less direct evidence involving the candidate answers.

5 Evaluation and Conclusions

Detailed evaluations of deep evidence scoring components will be presented in a future publication. LFACS has statistically significant impact on question answering accuracy when included in either a simple baseline DeepQA question answering system or to the complete Watson question answering system that competed with human grand champions. This impact, while significant, is small: less than half of one percent in the full system; the full system has an enormous number of answer scoring components and there is a great deal of overlap in the signal they provide. Other deep evidence scoring components in DeepQA (e.g., counting term matches, comparing word order) are more aggressive in what they consider to be a match. These aggressive components have the disadvantage that they do not draw on the full richness of the syntactic and semantic structure but the advantage that they can draw evidence from passages that have little structural similarity to the question.

The impact of LFACS when added to the simple baseline was smaller than that of the more aggressive components. However, in the complete system (containing many more features), the impact of LFACS (while small in an absolute sense) is larger than the impact of those components. The effect of ablating *all* of the deep evidence scoring components in the full system is much bigger than the effects of ablating any of them. These results have important implications for developers of question answering (or similar) technology. Simple, aggressive approaches are well-suited to quickly and easily attaining moderate effectiveness. However, as a system becomes more sophisticated, the opportunities for components of that sort to have impact becomes very limitted. In those cases, more algorithms such as LFACS that make effective use of syntatic and/or semantic structure can further enhance the effectiveness of a question answering system. As a result, additional and improved algorithms of this sort that draw on the full richness of our deep syntatic and semantic analysis are an important area for future research.

References

1. Falkenhainer, B., Forbus, K., Gentner, D.: The Structure Mapping Engine: Algorithm and examples. Artificial Intelligence 41, 1–63 (1989)
2. Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J., Schlaefer, N., Welty, C.: Building Watson: An Overview of the DeepQA Project. AI Magazine 31(3), 59–79 (2010)
3. Forbus, K., Oblinger, D.: Making SME greedy and pragmatic. In: Proceedings of the Cognitive Science Society (1990)
4. McCord, M.C.: Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars. In: Studer, R. (ed.) Natural Language and Logic. LNCS, vol. 459, pp. 118–145. Springer, Heidelberg (1990)
5. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)

Ontologies and Similarity

Steffen Staab

Institute for Web Science and Technologies,
Universität Koblenz-Landau, Germany
staab@uni-koblenz.de
http://west.uni-koblenz.de

1 Introduction

Ontologies [9] comprise a definition of concepts describing their commonalities (genus proximum) as well as their differences (differentia specifica). One might think that with the definition of commonalities and differences, the definition of similarities in and for ontologies should follow immediately. Traditionally, however, the contrary is true, because the method background of ontologies, i.e. logics-based representations, and similarity, i.e. geometry-based representations, have been explored in disjoint communities that have mixed only to a limited extent. In this short paper we survey how our own work touches on the intersection between ontologies and similarity. While this cannot be a comprehensive account of the interrelationship between ontologies and similarity, we aim it to be a stepping stone for inspiration and for indicating entry points for future investigations.

2 Similarity

When analyzing the interplay of ontologies and similarities, we have encountered two issues that need to be clarified first:

1. For which entities should similarity be assessed?
   - Objects: In many applications, the eventual target is the assessment of similarity between objects. For instance, in information retrieval one may want to search for the document vector neighboring most closely to a given query vector whereby the assessment of similarity should take the ontology-based semantics of query and document representations into account. Here, the ontology comes as an auxiliary means of influencing the geometric space in a desired manner, e.g. [10].
   - Concepts: Frequently the entities to be compared are the concepts defined in one or several ontologies. For instance, the integration of two information systems may be pursued by aligning the two ontologies that conceptualize the underlying information systems. Correspondencies between concepts from the two ontologies may be explored by taking different types of similarities between concepts into account, e.g. [7].

- Ontologies: In knowledge engineering the task of comparing ontologies is sometimes required in order to answer questions about the match between two ontologies, e.g. if they have been learned by automatic means, e.g. [6]. Not all of the measures used here are similarity measures in the mathematical sense, but similarities are fed into precision/recall-like measure in order to judge the container-containee relationship between two ontologies.

2. What is the objective of this similarity assessment?
   - Numeric Similarity Assessment: The most common type of similarity assessment is based on the mathematical notion of similarity measure that fulfills the following core properties for any given entities e, e_1, e_2:

$$\begin{aligned} sim(e_1, e_2) &\in [0, 1] \\ sim(e, e) &= 1 \\ sim(e_1, e_2) &= sim(e_2, e_1) \end{aligned}$$

   - Preference Ordering: While one may find quite some efforts in the literature for aligning the numerical assessment of similarity with judgements found in user experiments, in many application cases numerical measures are in fact not needed. Application cases like clustering of objects primarily need information about which pair of objects is most similar to each other — regardless of a numerical value. This is particularly relevant when ontological knowledge is so incomplete that deciding about such a preference ordering is still possible, while further measurements cannot be reasonably predicted.

3 Ontological Foundations for Similarity Assessments

Even within computer science, the word "ontology" is used in two senses. In its proper sense [9] an ontology is *a formal specification of a shared conceptualization of a domain of interest*. Thereby, the conceptualization abstracts from the particularities of a particular situation, e.g. a household ontology would typically include the definition that a desk is a table, while it would not contain statements about whether a desk is actually found in a particular household, which rather constitutes a situational and possibly changing aspect:

$$desk \sqsubseteq table$$

The word "formal" refers to the use of a mathematical mechanism for describing an ontology. In practice, the Web Ontology Language OWL, which is derived from the paradigm of description logics [14], is most frequently used as a notation for writing down ontologies, as it is an standardized, expressive language with a clear formal semantics. In description logics, one may distinguish definitional knowledge about concepts and relationships found in the T(erminological)-Box, e.g. every desk being a table, and assertional knowledge about objects, e.g. a specific object being a desk and occurring in a given household, such as found in the following A(ssertional)-Box:

$$Desk(object1).occur(object1, household2).$$

Thus, the Web Ontology Language provides convenient language constructs for specifying an ontology as well as for specifying the facts found in a particular situation. This convenience led to the use of the term "ontology" in a second meaning, namely as refering to a complete knowledge base consisting of an ontology in the proper sense and factual knowledge about a situation.

3.1 Logics-Based Ontology Representations

The full advantage of description logics is derived from the expressiveness of the language allowing for m the possibility to use concepts and relations to define new concepts. The following definition describes that a desk that has a Minibar which only contains FancyDrink*s* is a FancyDesk:

$$FancyDesk \sqsubseteq Desk \sqcap \exists hasPart.(MiniBar \sqcap \forall contains.FancyDrink)$$

This advantage, however, is problematic when, e.g., concept expressions are to be compared. For instance, the following definition specifies that a FavoriteDesk is a Desk with a Compartment having at least some SingleMalt:

$$FavoriteDesk \sqsubseteq Desk \sqcap \exists hasPart.(Compartment \sqcap \exists contains.SingleMalt)$$

When assessing the similarity between FancyDesk and FavoriteDesk, the intricate logic expressions lead to many non-trivial problems that we have analysed in [4]. The core result of this analysis was that existing similarity measures were not adequately reflecting the richness of description logics with some of them being unsound. We then suggested a new measure that uses the richness of description logics in a sound way, however, it is very expensive to implement.

3.2 Lattice-Based Ontology Representations

Historically, researchers did not approach the problems of similarity in ontology-based representations, but they rather relied on simpler representations. Some of them cannot be called ontology representations anymore, because they fail to reflect the richness of interactions between concepts and relationships between concepts.

A very elegant mechanism is formal concept analysis [8]. Here objects (table1 ... seat4) are represented by whether they have a property (hasLegs ... hasCompartment) or not. An example is given in Table 1.

This table of binary decision is analysed revealing what Ganter and Wille call 'formal concepts'. Each formal concept has an extension consisting of a subset of objects and an intension consisting of a subset of properties, e.g.

Table 1. Formal context representing the relations between objects and properties

	hasLegs	offersSeating	hasSurface	hasCompartment
table1	X		X	
desk2	X		X	X
chair3	X	X	X	
seat4	X	X	X	

({chair3,seat4},{hasLegs, offersSeating, hasSurface}) is a formal concept — obviously representing the class of all chairs. Also subclass relationships between formal concepts can be derived, for this particular example there will be a class of desks (containing desk1) and a class of tables (containinng table1 and desk2), with the former being more specific than the latter.

This representation is very interesting during ontology engineering and for data mining (and many extensions towards non-binary table entries and higher-arity relationships exist), but it is not close to common ontology representations. The structures are still very useful as approximations of ontological structures and the binary vectors lend themselves as a basic means for set-based assessments of similarity between objects, properties and formal concepts.

Formal concept analysis has been used to induce a taxonomy from words and their correlations to other words in texts [2]. While we are not aware of direct applications of formal concept analysis starting with an ontology, we see fruitful possibilities for such future use — especially because the given method constitutes a well-researched mathematical method with corresponding well-defined operators.

3.3 Graph-Based Ontology Representations

Graph-based representations of ontologies are the most frequently used means to assess similarities in or between ontologies. The advantage of using graphs is that the definitions are easy to implement and to adjust to specific needs. The disadvantage is that graphs also cannot capture the richness of descriptions found in OWL ontologies.

Most of the work on determining similarity between concepts in ontologies built on the core idea of Resnik [13] that such similarity is defined on the basis of a. a stable taxonomy, b. the counting of taxonomic links between two concepts that are to be compared and c. some normalization to take into account the height of a concept in this taxonomy, i.e. some version of measuring the extension of two concepts.

4 Applying Similarities and Ontologies

A very comprehensive survey of similarity measures in ontologies can be found in the dissertation by d'Amato [3]. Interestingly, these assumptions by Resniks (and 'followers') did not match very well the actual situation found in description logics ontologies. But independently from which representation is chosen to define which exact similarity measure, there is a wide range of applications for which this combination has been used. We simply give here an indicative list without much further explanation (and without claiming completeness): Information Retrieval [10]; Machine Learning [1]; Web Service Discovery [11]; Ontology Alignment [7]; Ontology Learning: Evaluation [12,6]; Ontology Learning: Induction [2]; Indexing of description logics ontologies [5].

5 Conclusion

We have sketched hier the possible space of how different ontology representations (or approximations of ontologies) may be used as a foundation for similarity measures. Depending on which entities are to be compared and what the purpose of comparison is a corresponding similarity measure must be chosen. It is intriguing to note that many of the existing similarity methods do not match with the assumptions underlying description logics ontologies — making existing methods invalid and requiring further exploration of this space of ontologies and similarities.

Acknowledgements. This research has been co-funded by DFG in the project "Multipla".

References

1. Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 58–71. Springer, Heidelberg (2007)
2. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. JAIR — Journal of AI Research 24, 305–339 (2005)
3. d'Amato, C.: Similarity-based Learning Methods for the Semantic Web. PhD thesis, University of Bari (2007)
4. d'Amato, C., Staab, S., Fanizzi, N.: On the influence of description logics ontologies on conceptual similarity. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 48–63. Springer, Heidelberg (2008)
5. D'Amato, C., Staab, S., Fanizzi, N., Esposito, F.: Dl-link: A conceptual clustering algorithm for indexing description logics knowledge bases. International Journal of Semantic Computing 4(4), 453–486 (2011)
6. Dellschaft, K., Staab, S.: On how to perform a gold standard based evaluation of ontology learning. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 228–241. Springer, Heidelberg (2006)
7. Ehrig, M., Staab, S., Sure, Y.: Bootstrapping ontology alignment methods with apfel. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 186–200. Springer, Heidelberg (2005)

8. Ganter, B., Wille, R.: Formal Concept Analysis Mathematical Foundations. Springer, Heidelberg (1999)
9. Guarino, N., Oberle, D., Staab, S.: What is an ontology? In: Handbook on Ontologies, 2nd revised edn. (2009)
10. Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. In: Proceedings of the International Conference on Data Mining, ICDM 2003. IEEE Press, Los Alamitos (2003)
11. Klusch, M., Fries, B., Sycara, K.P.: Owls-mx: A hybrid semantic web service matchmaker for owl-s services. Journal of Web Semantics 7(2), 121–133 (2009)
12. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 251–263. Springer, Heidelberg (2002)
13. Resnik, P.: Using information content to evaluate semantic similarity in a taxanomy. In: Proc. of Int. Joint Conf. for Artificial Intelligence (IJCAI 1995), pp. 448–453 (1995)
14. Staab, S., Studer, R. (eds.): Handbook on Ontologies, 2nd revised edn. (2009)

Retrieval of Semantic Workflows with Knowledge Intensive Similarity Measures

Ralph Bergmann[1] and Yolanda Gil[2]

[1] University of Trier,
Department of Business Information Systems II,
54286 Trier, Germany
bergmann@uni-trier.de
[2] Information Sciences Institute,
University of Southern California,
Marina del Rey, CA 90292, USA
gil@isi.edu

Abstract. We describe a new model for representing semantic workflows as semantically labeled graphs, together with a related model for knowledge intensive similarity measures. The application of this model to scientific and business workflows is discussed. Experimental evaluations show that similarity measures can be modeled that are well aligned with manual similarity assessments. Further, new algorithms for workflow similarity computation based on A* search are described. A new retrieval algorithm is introduced that goes beyond traditional sequential retrieval for graphs, interweaving similarity computation with case selection. Significant reductions on the overall retrieval time are demonstrated without sacrificing the correctness of the computed similarity.

1 Introduction

Today, workflows are an established means for modeling business processes and to automatically control their execution. Recently, workflows are more widely used for many purposes different from business process automation. *Scientific workflows* are executable descriptions of automatable scientific processes such as computational science simulations and data analyses [18]. Further, workflows are used for modeling medical guidelines, to represent project plans, or to describe information gathering strategies. Such new applications of workflows typically deal with a number of new difficulties, particularly due to an increasing number of workflows potentially relevant, an increasing complexity of the individual workflows, and an increased demand for more flexibility (*agile workflows*). To deal with those new challenges in workflow management, reasoning methods for semantically enriched workflow representations have a high potential to support workflow modeling, composition, adaptation, analysis, and optimization. This is what we call *semantic workflow reasoning*. Also case-based reasoning (CBR) has already demonstrated its high potential in semantic workflow applications [19,15,3,14,9,7,17]. Repositories of workflows can be used as case bases, enabling the application of case retrieval and adaptation methods.

In this paper we focus on case retrieval of semantic workflows as one method that contributes to semantic workflow reasoning. Workflow cases clearly belong to the class of highly structured case representations, for which graph-based representations are promising [8,6,14,15,16]. A particular challenge is the *similarity-based retrieval* of semantic workflow descriptions, as workflow similarity enables to retrieve workflows that do not match the query exactly, but that are a good starting point for adaptation. However, previous work on workflow retrieval either ignores the graph structure but uses a simple flat case representation [3], use (parts of) the graph structure, but only uses textual labels instead of semantic descriptions [16,14,11,8,7], or uses semantically annotated graph structures, but a crisp matching approach rather than similarity [9].

In this paper, we present a new general framework for workflow and related similarity modeling. Workflows are represented as semantically annotated graphs, extending previous proposals for graph-based workflow representations [16,8,17]. The well-known local/global principle to modeling similarity measures [4,2] is extended to these graph representations, providing a flexible means for workflow similarity modeling. This framework is domain independent and general in the sense that it covers control-flow oriented workflows such as business processes as well as data-flow oriented workflows such as scientific workflows. As with all graph-based representations, the computational complexity of the similarity computation becomes a critical issue during retrieval. As a second contribution of this paper, we developed and analyzed several approaches for similarity computation and retrieval based on different search algorithms. The third contribution is the application and experimental evaluation of the framework and the retrieval algorithms in two different application areas.

2 Semantic Workflows and Workflow Retrieval

Workflow representations typically reflect the dataflow or control flow structure among the tasks of a process. Today, various workflow representation formats are used, depending on the kind of workflow. Representation approaches for business workflows have a strong focus on the control flow, usually implementing (some of) the workflow patterns proposed by van Aalst. Typical control flow patterns are *sequence*, *and-split*, *and-join*, *xor-split*, *xor-join*, and possibly *loops*. Figure 1a shows an example of a business workflow within a University administration, according the representation used in the CAKE project [3,16] at the University of Trier. On the other hand, scientific workflows have a strong focus on the dataflow, typically restricting the control flow to a partial ordering of the tasks. Such a simple control structure offers several advantages and has been sufficient to support a variety of applications [18]. Each task (or software component) in the scientific workflow can have input datasets and input parameters, as well as output datasets. Dataflow links indicate what data are output by a task and consumed by another task. Figure 1b shows an example of a scientific workflow describing some data mining process according to the representation in the Wings project [10] at the Information Sciences Institute. Semantic workflow

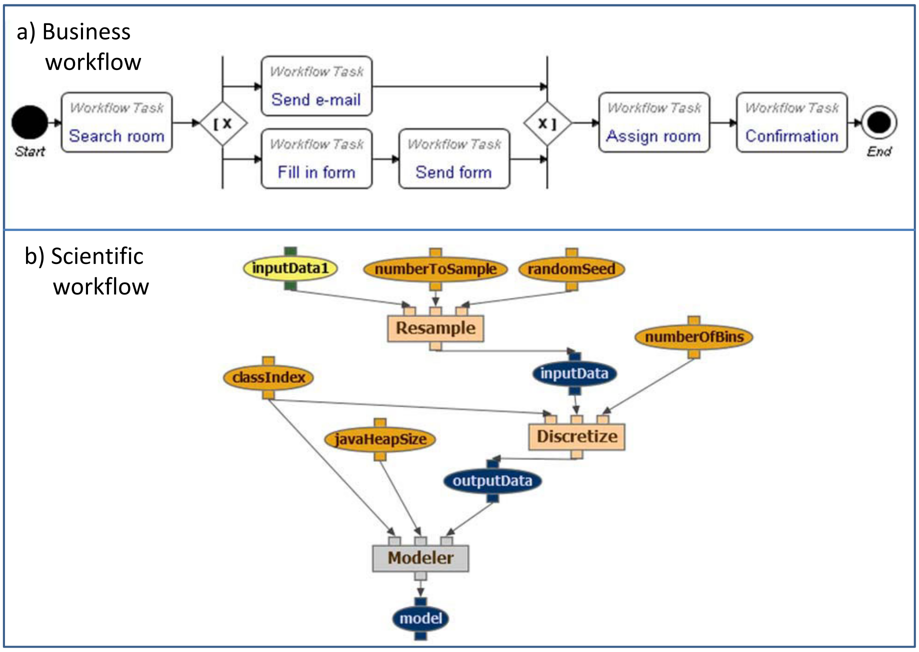


Fig. 1. Example of an administrative business workflow

representations enrich the above described workflow formats by adding meta data and constraints to the individual elements of the workflow. Ontologies are used to formalize the respective workflow domain by organizing the metadata descriptions of the occurring workflow elements. This allows to capture the semantics of tasks, data items, and control flow items.

The workflow representation in the CAKE system [3] uses an object-oriented representation (originally developed for cases) for ontology representation and metadata annotation. Tasks can be organized in a hierarchy of classes, each of which contains certain properties of a task, which can be inherited from the super class. For example, the *Assign room* task includes a role description stating that the assignment must be performed by the responsible administrative department. Another property may state the average duration for task execution. The semantic workflow representations in Wings augment traditional workflow representations with semantic constraints that indicate metadata properties of workflow tasks and datasets [9,10]. Each workflow constituent has a workflow variable associated with it, and the semantic constraints express constraints over those variables. In the case of data variables, the constraints are expressed in terms of metadata properties. For example, an input dataset may be restricted to be a discrete dataset, and if so then its variable would have a constraint stating that it has the property of being discrete. In the case of components, the constraints express how the execution of the software component changes the metadata properties of the input datasets into output datasets. Wings uses workflow reasoning algorithms to enrich workflows by propagating the constraints throughout the workflow structure [9]. This algorithm takes simple workflow graphs created by users and automatically augments them with semantic information by adding and propagating semantic constraints that can be used for subsequent retrieval as we propose in this paper.

For semantic workflow representation some languages that include semantic constraints expressed in OWL have been proposed, such as OWL-S, WSMO, SWSL, and SWSF. These languages include task decomposition structures and complex precondition expressions. Our CAKE and Wings approaches could be used with any of those languages, though only a small subset of the constructs allowed in those languages are used in our systems.

For workflow retrieval, several CBR and related approaches exist today. The CODAW system [15] supports incremental modeling of workflows by a similarity-based reuse of the workflow templates using an HTN planner that employs an inexact, graph-based matching. Leake et al. [14] evaluate the execution paths of past workflows in order to support user extension of workflows that are under construction. Recently, a probabilistic similarity model for workflow execution paths was proposed [1]. The myGrid project[1] focuses on the discovery, reuse and repurposing of bioinformatics workflows [12]. They distinguish between direct *re-use* of an existing workflow as is and *re-purposing* an existing workflow by modifying some aspect of it. Goderis [12] defines requirements and bottlenecks for workflow re-use based on many user interviews and practical experiences.

Previous work investigated structure-less workflow retrieval based on query and workflow representation being plain textual descriptions or sets of (social) tags [12], or abstract workflow features [3]. Queries including structural properties of the workflows are treated using graph matching approaches such as subgraph isomorphism or edit-distance measures [12,16,14,8]. Matching techniques based on semantic types of data has been done for the discovery of individual services or software components [13], but not for the more complex structures that workflows represent. Semantically annotated graph structures are used in workflow retrieval in Wings [9], but retrieval is restricted to crisp matching, not incorporating similarity measures.

3 Graph-Based Framework for Workflow Similarity

3.1 Representation of Semantic Workflows

In line with previous work on graph-based workflow representation [5,8], we represent workflows as semantically labeled directed graphs $W = (N, E, S, T)$ where N is a set of nodes and $E \subseteq N \times N$ is a set of edges. $T : N \cup E \rightarrow \Omega$ associates to each node and each edge a *type* from Ω. $S : N \cup E \rightarrow \Sigma$ associates to each node and each edge a *semantic description* from a semantic meta data

[1] `www.mygrid.org.uk`

language Σ. While Ω is fixed for the used workflow representation, Σ is domain dependent. We do not demand a particular language for Σ, but just assume some language for semantic metadata for which similarity measures can be defined (see Sect. 3.2). This general graph structure can be used to represent different kinds of workflows, such as scientific or business workflows. It is further specialized for the representation of workflows by distinguishing different types of nodes and edges, enumerated in Ω and assigned through T:

Each workflow consists of exactly one **workflow node**. The semantic description associated to a workflow node represents some overall properties of the entire workflow. This can be a workflow classification in some ontology, a set of semantic tags, or other properties related to the quality or execution requirements of the workflow.

Each task in a workflow is represented by a **task node**. Its semantic description typically classifies the task in some task ontology and may provide additional functional properties of the task. It also includes the description of workflow roles (i.e. human agents or services) for the execution of those tasks.

Each data item in a workflow is represented by a **data node**. Its semantic description classifies the data item within some data type ontology and may specify certain data properties (e.g. a data set has missing values) relevant for retrieval. Control-flow centric workflows (Fig. 1a) may have no data nodes.

Each control flow element in a workflow, such as split/join elements for and/xor blocks, are represented as a **control-flow node**. The semantic description of a control flow node specifies which control flow element is represented. Data-centric workflows (e.g. Fig. 1b) may have no control flow nodes.

The workflow node is linked to each of the other nodes by a **part-of edge**. The semantic description of such an edge specifies the role of the associated node within the overall workflow. Data nodes can be linked with the workflow node via edges having one of the following semantic description: *is-input*, *is-output*, *is-intermediate*, *is-parameter*. For example, in Fig. 1b *inputData1* is overall workflow input data and hence linked with *is-input*. *inputData* in Fig. 1b is data produced and further processed within the workflow and is hence linked with *is-intermediate*. Further, the semantic descriptions for part-of edges include *has-task* for task nodes linked to a workflow node and *has-control* for links to control-flow nodes.

The dataflow among tasks is represented using **dataflow edges**. A dataflow edge links data nodes to task nodes and vice versa. The semantic description of such an edge indicates whether the data item was *consumed* as input data or *produced* as output data by the task. All arrows shown in Fig. 1b are turned into dataflow edges in the graph.

The control-flow among tasks is represented using **control-flow edges**. Such an edge connects two task nodes or a task node with a control-flow node. An edge from node $n1$ to $n2$ indicates that node $n2$ may be executed after node $n1$. All arrows shown in Fig. 1a will be turned into control-flow edges in the graph.

Semantic constraints among properties of data nodes are represented using **constraint edges** that connect two data nodes. The semantic description

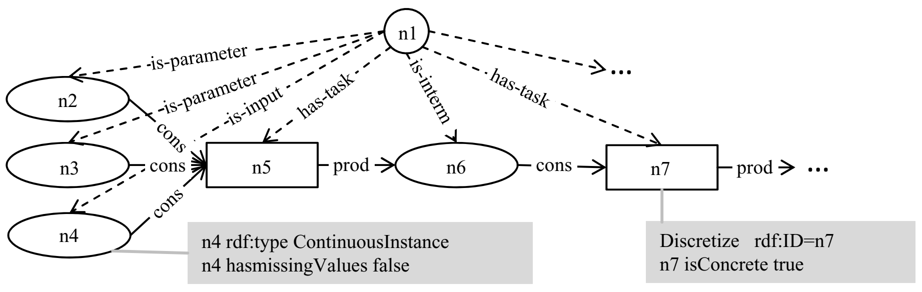


Fig. 2. Fraction of a workflow graph: the Resample and Discretize task are shown

includes the definition of a constraint. For example, a constraint may state that test and training data must come from the same domain.

Figure 2 shows the graph representation of a fraction of the workflow from Fig. 1b. It shows one workflow node (circle), four data nodes (ovals), and two task nodes (boxes) as well as some of the edges. A simplified fraction of the semantic descriptions of a data and a task node are shown in the grey boxes.

In the following, we assume that a repository of workflows is given, which we consider as case-base $CB = \{CW_1, ...CW_n\}$. We aim at retrieving workflows from this case base given a particular query. This query QW is also a workflow in the above described representation. It just specifies some workflow elements together with the semantic description that are considered requirements on the workflows to be retrieved. For example, a query could state that someone is looking for workflows that first discretize some continuous data and then use the discrete data in a decision tree modeler. Hence, the query would contain a workflow node, two data notes: one specifying the continuous data and one specifying the discretized intermediate data. Further, it includes two task nodes, one that specifies a task that can discretize data and one that specifies a task which is from the class decision tree modeler. The respective edges are included in the query as well. The similarity-based retrieval we aim at, could then retrieve for example the workflow shown in Fig. 1b, although it is not a perfect match as the modeler task in this workflow does not specify which concrete modeler to be used. However, it can be potentially specialized to a decision tree modeler. A different workflow from the repository containing a J48 modeler might be a better match with respect to this task.

3.2 Modeling Workflow Similarity

In order to assess the similarity of a case workflow $CW = (N_c, E_c, S_c, T_c)$ wrt. a query $QW = (N_q, E_q, S_q, T_q)$, we need to consider the constituents of the workflow as well as the link structure. This requires similarity models that enable the comparison of workflow elements. We now present a new similarity model, which can be considered an enhancement of the well-known local/global approach for structural CBR [4,2], particularly for object-oriented similarity measures [2]. The local similarity measures assess the similarity between two nodes or two edges of the same type. The global similarity for workflows is obtained by an aggregation function combining the local similarity values within a graph mapping process. The local similarity assessments are based on the semantic descriptions of the nodes and edges. Therefore, we assume a similarity function:

$$sim_{\Sigma} : \Sigma \times \Sigma \rightarrow [0, 1].$$

The detailed formalization of this similarity measure depends on the language Σ and can itself be modeled using the local/global approach aggregating local similarity measures for the individual properties in the semantic description. In our work, we treat the semantic descriptions in an object-oriented fashion and use the well established similarity measures described in [2]. Thereby, class hierarchies of tasks and data, as well as properties like those shown in Fig. 2 can be appropriately considered in the similarity model.

Nodes of equal type have a similarity value assigned, based on the similarity of the semantic descriptions. Based on sim_{Σ}, we define node similarity $sim_N(n_q, n_c)$ for $n_q \in N_q$ and $n_c \in N_c$ as follows:

$$sim_N(n_q, n_c) = \begin{cases} sim_{\Sigma}(S_q(n_q), S_c(n_c)) & \text{if} \quad T_q(n_q) = T_c(n_c) \\ 0 & \text{otherwise} \end{cases}$$

Edge similarity is more sophisticated as it should consider not just the semantic description of the edges being compared, but also the nodes that are linked by the edges. For example, two dataflow edges should only be considered similar if they link similar data objects to similar tasks. To reflect such considerations, we define edge similarity $sim_E(e_q, e_c)$ for $e_q \in E_q$ and $e_c \in E_c$ as follows:

$$sim_E(e_q, e_c) = \begin{cases} F_E \begin{pmatrix} sim_{\Sigma}(S_q(e_q), S_c(e_c)), \\ sim_N(e_q.l, e_c.l), \\ sim_N(e_q.r, e_c.r) \end{pmatrix} & \text{if} \quad T_q(e_q) = T_c(e_c) \\ 0 & \text{otherwise} \end{cases}$$

For an edge e, the expression $e.l$ denotes the left node of the edge and $e.r$ denotes the right node. The function F_E is an *aggregation function* that combines the semantic similarity between the edges and the similarity of the connected nodes to the overall similarity value. In our implementations we define F_E as follows: $F_E(S_e, S_l, S_r) = S_e \cdot 0.5 \cdot (S_l + S_r)$.

The overall workflow similarity between QW and CW is now defined by means of a *legal mapping*, i.e., a type-preserving, partial, injective mapping function $m : N_q \cup E_q \rightarrow N_c \cup E_c$ that satisfies the following five constraints:

$$\begin{array}{lll} T_q(n_q) = T_c(m(n_q)) & T_q(e_q) = T_c(m(e_q)) & \\ m(e_q.l) = m(e_q).l & m(e_q.r) = m(e_q).r & \forall x, y \; m(x) = m(y) \rightarrow x = y \end{array}$$

Please note that in such a legal mapping m a case node or edge can only be the target of one query node or edge, respectively. Since the mapping can be partial, not all nodes and edges of the query must be mapped to the respective case elements. Also an edge can only be mapped if the nodes that the edge connects are also mapped to the respective nodes which are linked by the mapped edge. The similarity between QW and CW wrt. a mapping m is now defined using a second aggregation function F_W for workflows as follows:

$$sim_m(QW, CW) = F_W \begin{pmatrix} (sim_N(n, m(n))|n \in N_q \cap \text{Dom}(m)), \\ (sim_E(e, m(e))|e \in E_q \cap \text{Dom}(m)), \\ |N_q|, |E_q| \end{pmatrix}$$

The aggregation function combines the individual similarity values for mapped nodes and edges to an overall similarity value wrt. the mapping m. Dom(m) denotes the domain of m. The parameters $|N_q|$ and $|E_q|$ enables F_W to consider partial mappings, i.e. nodes and edges not mapped should not contribute to the overall similarity. In our implementation we define F_W as follows:

$$F_W((sn_1, \ldots, sn_i), (se_1, \ldots, se_j), n_N, n_E) = \frac{sn_1 + \cdots + sn_i + se_1 + \cdots + sn_j}{n_N + n_E}$$

Please note that each mapping m can be interpreted as a particular suggestion for the reuse of the case workflow: the mapped case nodes and edges should be considered solution elements for the respective nodes and edges in the query. The similarity value for the mapping assesses the utility of this reuse opportunity. For a case there are usually several mappings that reflect different ways of reusing the case. Hence, the overall workflow similarity $sim(QW, CW)$ is determined by the best mapping (see also [5]), i.e the mapping with the highest similarity:

$$sim(QW, CW) = \max\{sim_m(QW, CW)| \text{ mapping } m\}.$$

To summarize, the presented similarity model for workflows is defined by three *model parameters*, 1st the similarity function sim_Σ for semantic descriptions, 2nd the aggregation function for edges F_E, and 3rd the aggregation function for workflows F_W.

3.3 Experimental Evaluation

We now focus on the question, whether the computed similarity values are appropriate within the context of a CBR application aiming at retrieving workflows for reuse. For this purpose, the similarity measures should be able to approximate the utility of the cases wrt. the problem formulated within the query (see [2], p.94*ff.*) in the context of the concrete application. Moreover, a good and broadly applicable similarity model should enable to flexibly determine appropriate model parameters for many applications. Whether this is the case, can of course only be assessed by analyzing a series of real-world applications. As a first step towards this goal, we developed two initial workflow retrieval applications in two complementary domains: *administrative business workflows* (similar to Fig. 1a) and *scientific data mining workflows* (similar to Fig. 1b). Based on our previous work [17,9] we developed for each domain an ontology for workflows, tasks (and data) as well a case base of 20 workflows. For both domains, the average number of nodes in the cases is 10; the average number of edges is 18.

The aim of the first experiment is to show, whether it is possible to model similarity measures that approximate the utility assessment of a human user. More precisely, the hypothesis is:

H1: *The similarity model enables to define model parameters such that the computed similarity is better aligned with an expert's assessment than a lexical similarity measure.*

For each domain we determined 10 queries that are related to the 20 cases in the case base. For each query, an expert was asked to select the 5-6 best suitable cases in the case base and to determine a partial order among those best cases. For each domain, a similarity model was developed by hand: similarity values within sim_Σ have been adjusted manually and the functions F_E and F_W are those mentioned before in this section. For each query, we computed the similarity of each case in the case base by applying an exhaustive search algorithm that computes all mappings, thereby producing an optimal solution. As baseline for the comparison, we computed the similarity by a lexical similarity measure, in particular a Levenshtein similarity measure on the task and data names. Besides this, the same functions F_E and F_W were used.

For both similarity measures, we compared the ordering of the cases with those of the human expert. As a measure of correctness we used the ranking measure proposed by Cheng et al. [6]: They define the *correctness* and *completness* of an ordering $\sqsupset$ with respect to a reference order $\sqsupset_*$ based on the concordance C of the two orders, i.e., by the number case pairs ordered equivalently by both orders and by the discordance D, i.e. by the number of case pairs for which both orders differ. Correctness and completeness are defined as follows: $Corr = (C - D)/(C + D)$ and $Compl = C/|\sqsupset_*|$.

In our evaluation we determine for each of the 10 query workflows QW the order on the 20 cases through: $CW_1 \sqsupset CW_2$ iff $sim(QW, CW_1) > sim(QW, CW_2)$. The partial reference order $\sqsupset_*$ is given by the human ranking for the same query. The value for correctness is within the interval [-1,1]. If it has a value of 1 then every ordering stated in $\sqsupset$ is correct, if it has a value of -1, every ordering stated in $\sqsupset$ contradicts the reference order. A value of 0 indicates that there is no correlation between both orders. The completeness value is within the interval [0,1]. The higher the value the more orderings are contained in $\sqsupset$ (compared to

Similarity	Administrative business workflows			Scientific data mining workflows		
	Correct	Complete	Retrieval Time	Correct	Complete	Retrieval Time
Semantic	0.708	0.910	24.00 sec	0.840	0.693	8.00 sec
Lexical	0.542	0.911	24.36 sec	0.593	0.751	7.78 sec

Fig. 3. Similarity Measure Evaluation

situations in which $\sqsupset$ does not distinguish the cases due to equal similarity). Figure 3 shows the results of the evaluation for the semantic similarity measure compared to the Levenshtein measure for both domains. The averaged correctness and completeness values over all 10 queries are displayed. The figure clearly shows an improved correctness for the semantic similarity measure in both domains, while the completeness slightly suffers for the data mining workflows. This clearly confirms the hypothesis H1.

Additionally, Fig. 3 shows the average retrieval time over all 10 queries using linear retrieval over the full case base, applying exhaustive search for similarity assessment. These figures clearly show that this approach is computationally unacceptable for practical applications. This motivates our work on a more scalable framework, described in the next section.

4 Similarity Computation and Workflow Retrieval

While similarity computation by exhaustive search guarantees to find the optimal match, it is computationally not feasible. Greedy search is proposed as alternative search algorithm for similarity assessment of workflow graphs [14,8,5] as it enables to quickly find a local optimum. However, the resulting error can hardly be controlled as the local optimum can differ significantly from the global optimum, hence producing too low similarity values. The A* search algorithm promises to be good alternative over the above mentioned approaches, given a well-informed admissible heuristic function can be determined. In the following, we will develop several A* search variants and demonstrate their performance and similarity error in experimental evaluations.

4.1 Similarity Computation by A* Search

The A* algorithm maintains a priority queue of open search nodes. In each step, the first (best) node in the queue is removed. If it represents a solution, A* terminates. Otherwise this node is expanded, i.e. each successor node in the search is determined and inserted into the priority queue. When applied for finding the best match m between a query and a case graph elements, a search node S basically represents the current mapping $S.m$ as well as the not yet mapped nodes $S.N$ and edges $S.E$. During node expansion, the mapping $S.m$ is extended in all possible ways by one additional mapping of a node or edge. The order in which the expanded nodes are inserted into the priority queue is essential for A*. Therefore, each search node S is evaluated by a function $f(S) = g(S)+h(S)$. In the traditional formulation, A* aims at minimizing cost, hence $g(S)$ are the cost already occurred and $h(S)$ is a heuristic estimation function for the remaining cost to the solution. As we apply A* for maximizing the similarity value, the functions must be interpreted differently, i.e. $g(S)$ is the similarity of the current mapping S, while $h(S)$ is an heuristic estimation of the additional similarity that can be achieved through the mapping of the remaining nodes and edges. Nodes are inserted into the priority queue in decreasing order of $f(S)$, i.e. the search node with the highest f-value is expanded first. To achieve an admissible heuristic estimation function, $h(S)$ must be an upper bound of the similarity. In the A* algorithm shown below, the value $f(S)$ is also stored in the search node, noted as $S.f$. The following A* search algorithm (divided into the top level looping algorithm and the separate expansion function) is called with a query workflow QW and a case workflow CW and returns the similarity value.

```
$A^*Search\,(QW = (N_q, E_q, T_q, S_q), CW = (N_c, E_c, T_c, S_c))$
{ $S_0.N = N_q$;   $S_0.E = E_q$;   $S_0.m = \emptyset$;   $S_0.f = 1$;   $Q =< S_0 >$;
  **while** $first(Q).N \neq \emptyset$ **and** $first(Q).E \neq \emptyset$ **do** { Q = $Expand(Q)$ };
  **return**($first(Q).f$); }

$Expand\,(Q)$
{ S = $first(Q)$; Q = $rest(Q)$; $x_q = select(S)$;
  **forall** $x_c \in E_c \cup N_c$ s.th. the mapping $S.m \cup (x_q, x_c)$ is legal **do**
  { $S'.m = S.m \cup (x_q, x_c)$; $S'.N = S.N \setminus \{x_q\}$; $S'.E = S.E \setminus \{x_q\}$;
    $S'.f = sim_{S'.m}(QW, CW) + h(S')$;
    $Q = insert(S', Q)$; }
  **Return** $Q$; }
```

Here, *first*(Q) is the first priority queue node, *rest*(Q) removes the first node and returns the rest and *insert*(S', Q) inserts the new node S' into Q according to the f-value. During insert, the maximum size of the queue can be restricted (maximum queue size) to cut some branches of the search tree to improve performance on the risk of loosing global optimality. The *select* function determines the next node or edge to be expanded. x_q can be either a query node or edge.

We developed two versions of A* with different estimation and select functions. The first version *A* I* uses a naive estimation function which assesses the similarity of each not yet mapped node or edge as 1. Assuming the aggregation function F_W shown in Sect. 3.2, the contribution of each not mapped element to the overall similarity is computed by h_I. The select function first matches all nodes, before the edges are selected. Which element from $S.N$ or $S.E$ is selected is not determined; we choose randomly according to an internal node/edge id.

$$h_I(S) = \frac{|S.N| + |S.E|}{|N_q| + |E_q|}$$

$$select_I(S) = \begin{cases} n_q \in S.N & \text{if } S.N \neq \emptyset \\ e_q \in S.E & otherwise \end{cases}$$

The second version *A* II* uses a better informed admissible heuristic. For each not mapped query node or edge it determines the maximum possible similarity a mapping can achieve independent of the mapping of the other nodes or edges. These values can be computed prior to search and cached. Also it aims at matching edges as soon as possible. As matching edges requires the connecting nodes being matched already, the edge expansion does lead to a low branching factor. It is 1 if between two nodes there is at most one edge per type. Hence the size of the queue does not increase, while the accuracy of $f(S')$ increases.

$$h_{II}(S) = \sum_{x \in S.N \cup S.E} \left(\max_{y \in N_c \cup N_e} \{sim_{E/N}(x,y)\} \right) \cdot \frac{1}{|N_q|+|E_q|}$$

$$select_{II}(S) = \begin{cases} e_q \in S.E & \text{if } e_q.l \notin S.N \text{ and } e_q.r \notin S.N \\ n_q \in S.N & otherwise \end{cases}$$

4.2 Parallelized A* Retrieval

While the described A* algorithms aim at improving performance of a single similarity assessment, linear retrieval with large case bases will still require one run of the search algorithm per case in the case base. To ensure better scalability with growing case bases, we now propose a parallelized A* II variant, called *A*P*. It enables to compute the top k cases from the case base without fully computing the similarity for all cases. Therefore the search process is parallelized for all cases, maintaining one queue Q_i for each case. In every step, the node from the queue with the highest f-value from all queues of not already finished search processes is expanded. Search terminates, when at least k searches have terminated and when the similarity of the k-best case is higher than all f-values of the remaining queues. Since the f-values are upper bounds for the final similarity, it is ensured that none of the remaining cases can ever exceed the similarity of the known k-best case. Hence, completeness of k-best retrieval is guaranteed. The following algorithm returns the list of the k-best cases (and possibly some more) together with its similarity value.

```
A*P Retrieval$(QW = (N_q, E_q, T_q, S_q), CB = \{CW_1, \ldots, CW_n\}, k)$
{ $S_0.N = N_q$;   $S_0.E = N_E$;   $S_0.m = \emptyset$;   $S_0.f = 1$;
  for $i = 1 \ldots n$ do { $Q_i =< S_0 >$ } ;
  res = $\emptyset$;
  repeat
  { $j = \arg\max_{i \notin res}\{first(Q_i).f\}$ ;
    $Q_j = Expand(Q_j)$ ;
    if $first(Q_j).N = \emptyset$ and $first(Q_j).E = \emptyset$ then res = res $\cup\{j\}$ ;
  } until $|res| \geq k$ and $k$-th$(first(Q_i).f | i \in res) \geq \max_{j \notin res}\{first(Q_j).f\}$ ;
  return $\{(first(Q_i).f, i) \mid i \in res\}$ }
```

4.3 Experimental Evaluation

We evaluated the performance of the three A* variants also in relation to exhaustive search and greedy search. The hypotheses to be evaluated are:

H2a: *The average retrieval time of A*I is shorter than exhaustive search.*
H2b: *The average retrieval time of A*II and A*P is shorter than of A*I.*
H2c: *The average similarity error of A*I,II,P are lower than of greedy search.*
H2d: *The average retrieval time of A*P is lower than A*II, if* $k << |CB|$.

Retrieval Method	Qsize	k	Retrieval time [sec]	Similarity Error
Exhaustive			795.269	
Greedy	1	20	0.112	0.221
A* I	10	20	0.279	0.092
	100	20	3.829	0.102
	300	20	21.075	0.091
A* II	1	20	0.254	0.023
	10	20	0.792	0.011
	100	20	15.051	0.005
A* P	1	5	0.226	0.018
	10	5	0.584	0.008
	100	5	9.332	0.004

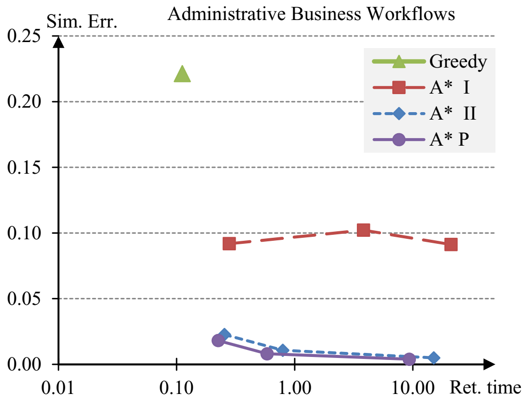


Fig. 4. Retrieval performance: Administrative business workflows

Retrieval Method	Qsize	k	Retrieval time [sec]	Similarity Error
Exhaustive			937.500	
Greedy	1	20	0.028	0.100
A* I	10	20	0.415	0.090
	100	20	5.244	0.069
	300	20	19.937	0.055
A* II	1	20	0.345	0.037
	10	20	0.852	0.020
	100	20	7.093	0.007
A* P	1	5	0.311	0.005
	10	5	0.667	0.007
	100	5	5.663	0.004

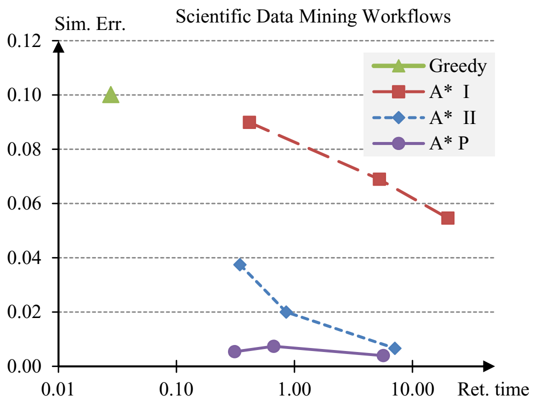


Fig. 5. Retrieval performance: Scientific data mining workflows

We tested the hypotheses for the two workflow domains. The similarity models and the case base of 20 cases from Sect. 3.3 are used. To assess the retrieval performance we used 30 queries for each domain: the 10 queries from Sect. 3.3 plus the 20 cases of the case base itself. We determined the average retrieval time per query as well as the average similarity error of the retrieved cases. As a base line for this we determined for each query and case the optimal similarity value by running the search algorithms over a very long time period.

Figures 4 and 5 show the results for the two domains for the different A* variants with different limits on the queue size. A*P is evaluated for $k = 5$ (out of 20 cases). For the other algorithms the performance does not depend on the number of cases to be retrieved. The graphs plot the similarity error over the retrieval time (logarithmic scale). Increasing the queue size limit clearly leads to an increased retrieval time, but also to a reduced error. The figures clearly indicate that the hypotheses H2a,b and c are confirmed. Concerning H2d, the advantage of A*P over A*II is not very dominant. Therefore, we performed an additional experiment with a case base of 203 cases from the scientific data mining domain used in [9]. Figure 6 shows the average retrieval time for different values of k. It clearly confirms hypothesis H2d.

Qsize	Retrieval Time [sec]		
	A* P, k=5	A* P, k=10	A* II
10	0,386	0,483	3,04
100	0,667	3,98	14,15

Fig. 6. Retrieval performance: Scientific data mining workflows, 203 cases

5 Conclusion and Future Work

We generalize and extend previous approaches for workflow similarity using graph-based representations in several ways. We explicitly cover data centric as well as control-flow centric workflows. We link with semantic representations and introduce knowledge intensive similarity measures according to the local/global principle. Further extensions are desirable to represent hierarchical workflows. The proper treatment of workflow agility requires representing workflow instances (rather then templates as in the current approach) including the execution state. Also, more practical experience with applications of the model is needed, involving semantic models of a larger scale. This also raises the question of methodologies for developing appropriate similarity models. New methods for learning similarity measures are demanded as a tool for this purpose.

The developed similarity assessment and retrieval algorithms show a satisfying performance in terms of computation time and retrieval error. We demonstrated scalability to medium sized case bases, but an extension to case base sizes $\gg 1000$ may require a course-grained pre-selection of cases according to the MAC/FAC idea, as proposed by Leake et al. [14]. A case retrieval net over the semantic descriptions seems a suitable approach for this purpose to be investigated in future research. Also, the extension of A*P towards a multi-threaded variant exploiting the parallelization of multi-core CPUs seems promising. Future work could also explore whether kernel methods for labeled graphs are a suitable alternative approach for similarity assessment of workflows.

Acknowledgement. This work is the outcome of a sabbatical visit of the first author at the Information Sciences Institute in 2010. This research was funded in part under grant IIS-0948429 from the US National Science Foundation. The authors appreciate the fruitful discussion on the topic of the paper by the various groups' members in Trier and Marina del Rey.

References

1. Becker, J., Bergener, P., Breuker, D., Räckers, M.: On measures of behavioral distance between business processes. In: Proceedings of the 10th International Conference on Wirtschaftsinformatik, vol. 2, pp. 665–674 (2011)
2. Bergmann, R.: Experience Management. LNCS (LNAI), vol. 2432. Springer, Heidelberg (2002)
3. Bergmann, R., Freßmann, A., Maximini, K., Maximini, R., Sauer, T.: Case-based support for collaborative business. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 519–533. Springer, Heidelberg (2006)
4. Burkhard, H.D., Richter, M.M.: On the notion of similarity in case based reasoning and fuzzy theory. Soft Computing in Case Based Reasoning (2000)
5. Champin, P.A., Solnon, C.: Measuring the similarity of labeled graphs. CBR Research and Development, 1066–1067 (2003)
6. Cheng, W., Rademaker, M., Baets, B.D., Hüllermeier, E.: Predicting partial orders: ranking with abstention. Machine Learning and Knowledge Discovery in Databases, 215–230 (2010)
7. Chinthaka, E., Ekanayake, J., Leake, D., Plale, B.: CBR based workflow composition assistant. In: World Conference on Services-I, pp. 352–355 (2009)
8. Dijkman, R., Dumas, M., Garcia-Banuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
9. Gil, Y., Kim, J., Florez, G., Ratnakar, V., González-Calero, P.A.: Workflow matching using semantic metadata. In: Proceedings of the 5th International Conference on Knowledge Capture, pp. 121–128 (2009)
10. Gil, Y., Ratnakar, V., Kim, J., González-Calero, P., Groth, P., Moody, J., Deelman, E.: Wings: Intelligent Workflow-Based design of computational experiments. IEEE Intelligent Systems 26(1), 62–72 (2011)
11. Goderis, A., Li, P., Goble, C.: Workflow discovery: the problem, a case study from e-science and a graph-based solution. International Journal of Web Services Research 5(4) (2008)
12. Goderis, A.: Workflow Re-use and Discovery in Bioinformatics. Ph.D. thesis, University of Manchester (2008)
13. Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., Stevens, R.: Deciding semantic matching of stateless services. In: Proceedings of the National Conference on Artificial Intelligence, vol. 21, p. 1319 (2006)
14. Leake, D.B., Kendall-Morwick, J.: Towards Case-Based support for e-Science workflow generation by mining provenance. In: Althoff, K.D., Bergmann, R., Minor, M., Hanft, A. (eds.) Advances in CBR, pp. 269–283 (2008)
15. Madhusudan, T., Zhao, J.L., Marshall, B.: A case-based reasoning framework for workflow model management. Data & Knowledge Engineering 50(1), 87–115 (2004)
16. Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. In: Weber, R., Richter, M.M. (eds.) CBR Research and Development, pp. 224–238 (2007)
17. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards Case-Based adaptation of workflows. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS (LNAI), vol. 6176, pp. 421–435. Springer, Heidelberg (2010)
18. Taylor, I.J., Deelman, E., Gannon, D.B.: Workflows for e-Science. Springer, Heidelberg (2007)
19. Weber, B., Wild, W., Breu, R.: CBRFlow: enabling adaptive workflow management through conversational Case-Based reasoning. In: Funk, P., Gonzlez-Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 434–448. Springer, Heidelberg (2004)

Qualitative vs. Quantitative Plan Diversity in Case-Based Planning

Alexandra Coman and Héctor Muñoz-Avila

Department of Computer Science and Engineering, 19 Memorial Drive West,
Lehigh University, Bethlehem, PA 18015
{alc308,hem4}@lehigh.edu

Abstract. Plan diversity has practical value in multiple planning domains, including travel planning, military planning and game planning. Existing methods for obtaining plan diversity fall under two categories: quantitative and qualitative. Quantitative plan diversity is domain-independent and does not require extensive knowledge-engineering effort, but can fail to reflect plan differences that are truly meaningful to users. Qualitative plan diversity is based on domain-specific characteristics which human experts might use to differentiate between plans, thus being able to produce results of greater practical value. However, the previous approach to qualitative plan diversity assumes the availability of a domain metatheory. We propose a case-based planning method for obtaining qualitative plan diversity through the use of distance metrics which incorporate domain-specific content, without requiring a domain metatheory. To our knowledge, this is the first time qualitative plan diversity is being explored in a case-based planning context.

Keywords: diversity, case-based planning, qualitative diversity, quantitative diversity, diversity metrics.

1 Introduction

Diversity-aware planning consists of generating two or more plans which, while solving the same problem, are dissimilar from one another, thus covering a large portion of the solution space, and providing a good indication of the range of available possibilities.

Plan diversity has practical value in multiple planning domains, including military planning [14] (e.g., offensive versus defensive plans, or defensive plan variants), travel planning [12] and route planning (e.g. using local roads versus using highways). In mixed-initiative planning environments [13], diverse plans can provide the user with genuine alternatives, potentially highlighting useful solutions that may otherwise not be considered. In plan-based intrusion-detection [1], they raise awareness of manifold threats. In game environments, plan diversity could be used to assist the player in exploring multiple different game-play strategies, as well as for modeling non-player characters exhibiting varied behavior, adding to the realistic atmosphere and enjoyment factor of the gaming experience [16].

Depending on the type of plan distance (the measure of the dissimilarity between two plans) on which they are based, previous approaches to diversity-aware planning can be seen as belonging to one of two categories: *qualitative* and *quantitative*.

Quantitative plan distance is domain-independent and has the advantage of not requiring domain knowledge aside from the usual domain transition model (e.g., a collection of STRIPS operators). The most common example of a quantitative distance metric is an action-set metric that counts the number of actions two plans do not have in common. This approach is, however, inflexible, as well as likely to produce misleading results: two plans identified as distant using a quantitative, action-set metric could, in essence, be similar (e.g. in combat-based games, two plans may have very little overlap in terms of the actions they execute, while being both implementations of a defensive strategy).

Qualitative plan distance, on the other hand, is based on domain-specific knowledge, thus having the potential to reflect subtler semantic differences that a human expert might take into account when comparing two plans (e.g. even if consisting of otherwise identical actions, a plan involving first-class air-travel will, from the point of view of a budget-conscious customer, be radically different from its economy-class counterpart). In contrast to the more mechanical quantitative approach, qualitative plan comparison should "see" plans much like human users would: as endeavors characterized by cost, risk, degrees of preference, etc. On the downside, the method for achieving qualitative plan diversity proposed previously in generative planning [12] is knowledge-intensive, requiring, in addition to the domain transition model, an extended domain theory ("metatheory").

In our previous work [3], we took the first steps in exploring plan diversity in case-based planning [5,17,20], and did so using a quantitative approach. We now propose a case-based planning method for obtaining qualitative plan diversity without the need for a domain metatheory. This is achieved through the use of distance metrics which, themselves, incorporate the minimal domain-specific content that is required for the purposes of obtaining diversity.

We aim to show that, when run in a real environment, qualitatively-diverse plan sets can produce more varied results than quantitatively-diverse plan sets. To our knowledge, this is the first time qualitative plan diversity is explored in a case-based planning context.

In Section 2, we describe qualitative and quantitative plan distance in more detail. Then, in Section 3, we exemplify and compare quantitative and qualitative plan diversity in a planning domain. In Section 4, we present a case-based retrieval algorithm that is amenable to both quantitative and qualitative distance metrics. Section 5 is dedicated to the comparative experimental evaluation of the diversity of plans obtained using quantitative and qualitative distance metrics. Section 6 provides an overview of related work, followed by final remarks in Section 7.

2 Qualitative and Quantitative Plan Distance

Adapting the case diversity definition formulated by Smyth and McClave [19], we can define the diversity *Div(Π)* of a set of plans *Π* as the average dissimilarity between pairs of plans in the set:

$$Div(\Pi) = \sum_{\pi,\pi' \in \Pi} \frac{Dist(\pi,\pi')}{\frac{|\Pi|\times(|\Pi|-1)}{2}} \tag{1}$$

where the *plan distance Dist*: Π×Π →[0,1] is a measure of the dissimilarity between two plans. This formula is a direct adaptation, for the purposes of plan comparison, of the case diversity formula proposed by Smyth and McClave [19],[1] as *Dist* can be considered to be the complement of a similarity measure *Sim*: [2]

$$Dist(\pi, \pi') = 1 - Sim(\pi, \pi') \tag{2}$$

It should immediately be pointed out that the problem of comparing plans is nontrivial: each plan may have an arbitrary number of actions, each with any number of parameters. Furthermore, the plan space for a given problem is potentially infinite (e.g., in a travel domain, a plan can be arbitrarily lengthened by repeatedly going back and forth between two locations). It follows that the notion of completeness [11], as defined for analysis tasks, such as recommender systems, cannot be applied to case-based planning (the set of possible solutions to planning problems is not limited to the contents of the case base, but includes all adapted plans which could possibly be obtained from those cases; and there are infinitely many such plans).

The types of plan distance used in generative planning fall under two categories, which we will be referring to as *quantitative* and *qualitative*.

Quantitative plan distance is based on plan elements (such as actions) derivable from the domain transition model,[3] which are not interpreted in any domain-specific way. It follows that any two distinct plan elements are considered equally distant from one another (e.g. in a cooking domain, the action of adding lemon juice to a dish is considered equally distant from the action of adding vinegar and the action of adding sugar). Quantitative plan comparison, therefore, generally consists of counting the plan elements which plans have, or do not have, in common. An example of a quantitative distance metric (a normalized version of the metric used in [6]) is:

$$Dist_{Quant}(\pi_1, \pi_2) = 1 - \frac{common(\pi,\pi')}{\max(|\pi|,|\pi'|)} \tag{3}$$

where *common*(π,π') is the number of actions that plans π and π' have in common and |π| is the number of actions in plan π.

Qualitative plan distance is based on interpretation, using domain knowledge, of the components of plans (e.g. in a cooking domain: lemon juice and vinegar are both sour, but sugar is sweet; in a travel domain: a first-class plane ticket is expensive, while an economy one is affordable). As multiple bases for qualitative distance can be defined for the same domain, it is possible to vary the set of features along which one would like to see diversity (e.g. in a travel domain, variation of ticket cost, but not

[1] The same formula is used in a generative planning context by Myers and Lee [12], under the name of "dispersion".

[2] We are expressing diversity in terms of distance metrics. However, all formulas for distance metrics can be rewritten in terms of the complementary similarity metric. We maintain this interchangeability by always using normalized versions of the metrics, so that their values fall in the [0,1] interval.

[3] The minimal domain theory required in planning.

means of transportation). This has a practical advantage over quantitative diversity. However, with the greater potential benefits of qualitatively diverse plan generation, comes the greater complexity of obtaining it. Unlike quantitative metrics, which are domain-independent, qualitative metrics require domain-specific knowledge to be encoded and utilized. Previously, this was achieved by Myers and Lee [12] in an HTN planning context. Their approach, however, involves considerable knowledge engineering effort: for the purposes of diverse plan generation, Myers and Lee require a "metatheory" providing additional domain information, thus allowing plans to be compared in terms of high-level features, such as the objects which fulfill various "roles" in plans and the domain-specific characteristics of various types of actions (e.g. the speed of travel by a given means of transportation).

We propose a method of obtaining qualitative diversity which requires neither an HTN planning context, nor a domain metatheory. Instead, it is based solely on the domain transition model, a case base of plans, and *qualitative distance metrics* which incorporate all the domain-specific, qualitative content that is required for the purposes of creating diversity.

Our approach is motivated by the observation that obtaining plan diversity does not require a comprehensive qualitative model of the domain. It is sufficient to "equip" the diversity metric with minimal knowledge regarding the selected features it should base its differentiation between plans on. It immediately follows that multiple qualitative metrics can be defined for any domain, each metric reflecting the minimal useful information necessary for a particular diverse-plan retrieval task. These metrics can then be used separately or compounded as needed, offering much greater power and flexibility in generating diverse plan sets that are truly useful in practical situations.

3 Qualitative and Quantitative Plan Diversity in a Real-Time Strategy Game Context

To exemplify possible uses of quantitative and qualitative plan diversity, we assume a real-time strategy game context, which is characterized by many of the complexities of real domains of practical interest: it is dynamic (the world state evolves while the agent deliberates), non-deterministic (no specific action outcome can be guaranteed), partially observable, and adversarial (agents in each team seek to maximize their performance metric by minimizing the opponents' performance) [15].

Assume the following game configuration: the types of available units are peasants, soldiers, archers and mages. Units vary in terms of attack capabilities (e.g. soldiers are close-combat units, archers and mages long-range attack units) and robustness (e.g. peasants are very weak). The game score is computed by adding points for enemy kills and subtracting points for loss of friendly units. The amount added/subtracted on the destruction of a unit depends on the type of unit in question. The actions that can be taken by units are: *move* (the unit attempts to move to a specified location on the map), *patrol* (the unit moves back and forth between its current location and a specified location on the map) and *attack* (the unit attacks any enemies at a given location). The action signature is <action name (parameter1, parameter2)>, where *parameter1* specifies the unit which will undertake the action and *parameter2* specifies the target location of the action (e.g. action *Move(soldier1, loc1)* instructs unit *soldier1* to move to the map location *loc1*). There are two teams, one controlled using our plans, the other controlled by the built-in enemy AI.

Consider the set of 3 plans in Fig. 1, and assume that we have already retrieved Plan 1 and are now trying to find a second plan, out of the two remaining ones, that is maximally distant from Plan 1, making the resulting pair of retrieved plans maximally diverse (using the diversity-aware retrieval algorithm described in Section 4).

Plan1: Move (soldier1, loc1), Move (soldier2, loc2), Move (mage1, loc3), Attack (soldier3, loc4)
Plan 2: Attack (soldier2, loc4)
Plan 3: Move (soldier1, loc1), Move (soldier2, loc2), Move (mage1, loc3), Attack (archer1, loc4)

Fig. 1. Sample plans for a real-time strategy domain. The action parameters specify the unit which will be undertaking the action and the map location at which the action will take place.

First, let us consider quantitative diversity (in our previous work [3], we demonstrated quantitative plan diversity in a real-time strategy game domain). To do so, we use the quantitative metric $Dist_{Quant}$ (Formula 3). As a result, the plan that is chosen is Plan 2: it shares no actions with Plan 1 (the *attack* actions in the two plans use distinct soldier units), therefore the distance between them is 1 (the maximum possible distance).

However, an informed analysis, using domain-specific information, of the individual actions yields significant information: an *attack* action indicates an offensive approach to the game; a more neutral *move* action could be interpreted in various ways: moving to a location on one's own side of the map may be considered a defensive action, while attempting to move towards the enemy side is likely offensive, indicating the intention to engage in battle. Therefore, Plans 1 and 2 may not be meaningfully different at all. They both culminate in an *attack* action at the same map location, using units, which, while distinct, are of the same type (soldiers). The three other actions that differentiate Plan 1 from Plan 2 may not be of great consequence at all, if the locations the units are moving to are on the friendly side of the map and not very far from their initial locations.

Let us now consider, instead, a qualitative distance metric which considers two plans maximally diverse if they attack using a different *type* of unit, and identical if they use units of the same type to attack, even if the units are distinct (we will be using a more elaborated variant of this metric in our experiments).

This method assesses Plan 2 as being maximally similar to Plan 1: they use units of the same type to attack, and the other actions in Plan 1 are ignored for the purposes of comparison, as they were not specified in the metric definition (this is an example of a qualitative metric including only the minimal amount of domain information that is relevant to the task at hand, thus reducing the knowledge engineering effort, and improving retrieval performance). As a result, the qualitative method picks the maximally distant Plan 3, which attacks using an archer, a unit very different from a soldier: it is long-range, weaker in close combat, and its loss incurs a different score penalty than the loss of a soldier. This makes the selected plans significantly different relatively to the rules of the game.

4 Plan-Diversity-Aware Retrieval Algorithm

To demonstrate plan-diversity-aware case retrieval, we use a variant of the Greedy Selection[4] algorithm proposed by Smyth and McClave [19] (Fig. 2). The algorithm retrieves a set of *k* diverse cases. First, it automatically adds to the retrieved set the case that is maximally similar to the new problem. Then, for *k-1* steps, it retrieves the case that maximizes an evaluation metric taking into account both the similarity to the new problem and the relative diversity to the set of solutions selected so far. The key difference between the original Greedy Selection method (used for analysis tasks) and our variant (used for planning, which is a synthesis task) stems from the fact that plan-diversity-aware retrieval needs to take the solution plan into account, in addition to the problem. During retrieval, the problem is considered for similarity purposes, while the solution is considered for diversity purposes.

```
define PlanDiversityGreedySelection(n,C,k)
begin
   R := {}
   For i := 1 to k
      Sort C by SimPlDiv(n,c,R) for each c in C
      R := R + First(C)
      C := C – First(C)
   EndFor
return R
end
```

Fig. 2. The Plan Diversity Greedy Selection algorithm: a case-based planning variant of Greedy Selection [19]

We assume a transformational-analogy adaptation method, in which the contents of a case are a problem (consisting of an initial and/or final state) and a solution, consisting of a plan. The new problem is defined in terms of initial and/or final state.

In Fig. 2, *n* is the new problem, *C* the case-base, and *k* the number of cases we aim to retrieve. In our variant of the algorithm, the quality based on which retrieval occurs is:

[4] We chose to use general Greedy Selection, rather than its variant Bounded Greedy Selection [19], which improves performance for large case bases, as retrieval from our particular case base is manageable with the general algorithm. Alternatively, we can assume our case base to consist of only the top *bk* most similar cases, making our algorithm a variant of Bounded Greedy Selection.

$$SimPlDiv(n, c, R) = \alpha Sim(n, c) + (1 - \alpha) RelPlDiv(c.\pi, R.\Pi) \quad (4)$$

where *Sim* is the case similarity measure used for traditional similarity-based retrieval (most generally, similarity of initial and/or final states), α a parameter used for varying the complementary weights assigned to the similarity and diversity retrieval criteria, *c.π* the solution plan of case *c*, *R.Π* the set of solution plans in the set of cases *R*, and *RelPlDiv(π,Π)*, the diversity of a plan *π* relative to a set of plans *Π* (adapted from the *RelDiversity* formula proposed by Smyth and McClave [19]):

$$RelPlDiv(\pi, \Pi) = \frac{\sum_{\pi' \in \Pi} Dist(\pi, \pi')}{|\Pi|} \quad (5)$$

Dist can be any distance metric, either quantitative or qualitative.

5 Experimental Evaluation

Our experimental environment is real-time strategy game Wargus, which has previously been used in case-based planning work [3,15].

5.1 Experimental Setup

Game Configuration. We run two-player Wargus games on two 32x32 tile maps (Fig. 3), with our team's plans executed against the built-in Wargus enemy AI. The types of units and available actions are as described in Section 3. Each plan represents an individual battle (in which one of our armies engages the enemy), rather than a complete, prolonged game. This restriction was necessary so as not to allow excessive implicit game-play diversity, which might render meaningless the difference in variance between results produced using different metrics. The two maps on which we test our plans are topologically different: the first has one gap in the forest separating the two armies, while the second has two gaps, located at different coordinates than the gap in the first map. This difference is meaningful for the following reason: on the second map, units will sometimes make different choices as to which gap to use to pass to the other side: sometimes, all units will use the same gap, at other times, they will split up, sometimes they will even "hesitate", marching towards one gap, then returning to the other one. This ensures considerably different game behavior between the two maps.

Case-based Planning System. In our case-based planning system, we use the following convention: the cases are interpreted as battle-plan blueprints, so that every unit in a case is an abstracted representation of an entire army of units of that type (e.g. a soldier stands for an army of soldiers). New problems consist of actual game configurations, specifying number of armies of each type, as well as number of units in each army.

The *case base* consists of 100 distinct cases, each composed of an initial state (the problem) and a plan (the solution). The *initial state* is represented in terms of numbers of armies of each type. Each of these armies is represented by one unit in the plan.

The *plans* were generated using the FastForward [7] generative planner, modified so as to generate multiple plans for the same problem. All plans contain an attack action by one unit (which represents the entire attacking army in the adapted plan). No goal state is specified: the general goal is to obtain the highest possible score, and there is never one single final state through which this is achieved.

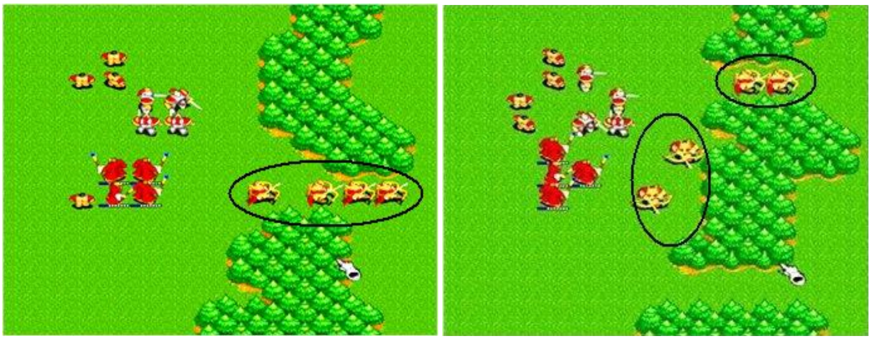


Fig. 3. The two topologically-different game maps, with archer armies highlighted. Note how the archer army in the second map has split up into two divisions, each using a different gap to pass.

The *new problems* consist of initial game states, indicating the number of armies of each type (soldier, archer, mage, peasant), as well as the number of units in each of the armies. All units in an army are of the same type. There are 5 new problems, with varying numbers of armies of each type, as well as number of units per army.

The *adaptation* algorithm is consistent with the idea of a retrieved plan serving as blueprint. As each unit in the retrieved plan represents an army, each army *A* in the new problem will be matched to a unit *U* (of the same type as the units in *A*) in the retrieved plan. All units in *A* will then perform all actions performed by *U* in the retrieved plan. The matching will occur in order of the numbering of units in the retrieved plan, with one exception: if unit *U* is the attacking unit in the retrieved plan, *U* will be the first to be assigned to an army of its type in the new problem, assuming such an army exists. This will always be the case with our problems: they all contain at least one army of each type, in order to be able to take at least partial advantage of any retrieved plan.

For *case retrieval*, we use the PlanDiversityGreedySelection retrieval algorithm (Fig. 2), where *k=4*, *α*=0.5, and *Sim* is a similarity metric Sim_{InitSt}, based on the initial states of the compared cases:

$$Sim_{InitSt}(\mathrm{c}_1.\mathrm{IS}, \mathrm{c}_2.\mathrm{IS}) = \frac{\sum_{i=1}^{n} \frac{\min\,(numArmiesType_i(\mathrm{c}_1.\mathrm{IS}), numArmiesType_i(\mathrm{c}_2.\mathrm{IS}))}{\max\,(numArmiesType_i(\mathrm{c}_1.\mathrm{IS}), numArmiesType_i(\mathrm{c}_2.\mathrm{IS}))}}{n} \qquad (6)$$

In Formula 6, *n* is the number of types of units (in our experimental setup, *n*=4) and *numArmiesType*$_i$*(c.IS)* is the number of armies of units of type *i* in the initial state of case *c*.

As the distance metric *Dist*, we use the quantitative metric $Dist_{Quant}$ (Formula 3), as well as a game-specific qualitative metric, which we call $Dist_{Wargus}$:

$$Dist_{Wargus}(\pi_1, \pi_2) = \begin{cases} 0, if\ attackUnitsType(\pi_1) = attackUnitsType(\pi_2) \\ d, 0 < d \leq 1, if\ attackUnitsType(\pi_1) \neq attackUnitsType(\pi_2) \end{cases} \tag{7}$$

NEW PROBLEM	**ADAPTED PLAN**
Initial State **1 soldier army** (4 units) **1 peasant army** (4 units) **2 mage armies** (4 units each) **2 archer armies** (4 units each)	*move (archer1, 05_05)* *move (archer2, 05_05)* *move (archer3, 05_05)* *move (archer4, 05_05)* move (peasant1, 03_02) move (peasant2, 03_02) move (peasant3, 03_02) move (peasant4, 03_02) *move (mage1, 04_07)*
RETRIEVED CASE ***Initial State*** **1 soldier army** **1 peasant army** **1 mage army** **1 archer army** ***Plan*** move (archer1, 05_05) move (peasant1, 03_02) move (mage1, 04_07) move (archer1, 24_07) patrol (soldier1, 01_04) move (soldier1, 05_04) attack (archer1, 24_07)	*move (mage2, 04_07)* *move (mage3, 04_07)* *move (mage4, 04_07)* move (archer1, 24_07) move (archer2, 24_07) move (archer3, 24_07) move (archer4, 24_07) *patrol (soldier1, 01_04)* *patrol (soldier2, 01_04)* *patrol (soldier3, 01_04)* *patrol (soldier4, 01_04)* move (soldier1, 05_04) move (soldier2, 05_04) move (soldier3, 05_04) move (soldier4, 05_04) *attack (archer1, 24_07)* *attack (archer2, 24_07)* *attack (archer3, 24_07)* *attack (archer4, 24_07)*

Fig. 4. Sample case, new problem, and the corresponding adapted plan (units in the adapted plan are not annotated with the army they belong to because, in this example, there is only one army of each type). The second action parameter indicates the coordinates of the location at which the action should take place.

In Formula 7, *attackUnitsType(π)* is the type of units in the attacking army of plan π, and *d* is the degree of difference between two types of units, as defined based on game-specific knowledge, and indicated in Table 1 below.

Table 1. Domain-specific degrees of distance between types of Wargus units

Unit type 1	**Unit type 2**	***d***
Peasant	Soldier	0.50
Peasant	Archer	0.75
Peasant	Mage	0.90
Soldier	Peasant	0.50
Soldier	Archer	0.50
Soldier	Mage	0.75
Archer	Peasant	0.75
Archer	Soldier	0.50
Archer	Mage	0.50
Mage	Peasant	0.90
Mage	Soldier	0.75
Mage	Archer	0.50

5.2 Experimental Evaluation

Evaluation Method. To evaluate the diversity of game-play sessions which are based on the sets of generated plans, we observe the variation of two game-specific *evaluation metrics*. The primary metric is Wargus *score* (computed as in Section 3); the secondary metric is *time* (the duration, in game cycles, of game-play sessions).

Our *hypothesis* is that plans obtained using retrieval based on the qualitative plan-diversity metric $Dist_{Wargus}$ will produce greater game-play variation (reflected in the evaluation metrics), than plans obtained using the action-set quantitative distance metric $Dist_{Quant}$. We expect that, when run in the game, adaptations of plans retrieved using the qualitative distance metric will produce significantly more variation (as measured using standard deviation and assessed using the F-test) of Wargus scores[5] than adaptations of quantitatively-diverse sets of plans. We expect to see a similar behavior with regard to time, but with less confidence, as we have observed that game duration tends to vary more between runs of the same plan, on the same map.[6]

[5] Note how we have chosen one of the countless possible domain-specific, qualitative distance metrics in accordance with our purpose: that of obtaining easily quantifiable diversity. Had our objective been different, we might have opted for a distance metric producing some form of diverse game behavior which is not so clearly reflected in score variation.

[6] Had we chosen time as the primary metric, we might have retrieved plans which use diverse route waypoints, encouraging the variation of game duration more clearly than that of score.

Results. In Fig. 5, each point in each chart represents the standard deviation of score or time (as indicated) for one plan set of 4 plans, where each plan is run in the game 5 times. The two data sets in each chart correspond to results obtained using the quantitative distance metric $Dist_{Quant}$ and the qualitative metric $Dist_{Wargus}$ in retrieval. There are 5 plan sets for each of the 5 new problems, on each of the 2 maps (50 plan sets in all).

As can be seen in the charts, for *score*, the standard deviation of $Dist_{Wargus}$ results per plan set is consistently higher than that of $Dist_{Quant}$ results.[7] Being highly diverse, the $Dist_{Wargus}$ score sets always include the highest recorded score per problem/map combination (while $Dist_{Quant}$ sets do not).

The F-test score results indicate that the difference between the variances of the $Dist_{Wargus}$ and $Dist_{Quant}$ score data sets is statistically significant, at the 95% confidence level, for all problems, on both maps, with the $Dist_{Wargus}$ data set displaying the greater variance.

For the secondary metric of *time*, the standard deviation of $Dist_{Wargus}$ results is greater than that of $Dist_{Quant}$ results on all but 2 of the 25 plan sets on the first map, and all but 3 out of the 25 plan sets on the second map.

The F-test indicates that the $Dist_{Wargus}$ data sets display greater variance, and the variance difference is statistically significant, at the 95% confidence level, on 4 of the 5 problems on each map. On the second map, the difference is statistically significant (with greater variance for the $Dist_{Wargus}$ data set), at the 90% confidence level, on the remaining problem. For the remaining problem on the first map, the variance is slightly greater for the $Dist_{Quant}$ data set, but the difference is not statistically significant.

To sum up, $Dist_{Wargus}$ results are significantly more diverse than $Dist_{Quant}$ results on all problems for the score metric, and on the majority of problems for the time metric. This is consistent with our expectations.

In terms of plan quality, we have noticed that plans retrieved using $Dist_{Wargus}$ (and, consequently, the adapted plans based on them) tend to be shorter, on average, than plans retrieved using $Dist_{Quant}$ (the reason for this should be obvious from the way the two metrics are computed, with $Dist_{Quant}$ easily increasable by lengthening any of the compared plans, as long as the added actions are not encountered in the other plan). Plan length relates to the time it takes to execute the strategy outlined in the plan. It follows that shorter plans may, in this context, be preferable to longer ones. This suggests that well-chosen qualitative distance metrics can also help ensure that retrieved plans are of good quality.

[7] The question might be raised whether plan sets producing highly diverse scores, from high to low (rather than all of the plans playing the game expertly) are ever of practical value: a simple example is the modeling of AI enemies, which, to make the game environment realistic (as well as not discouragingly difficult) should vary in intelligence and ability. Also, in partially unknown environments (e.g. the map remains the same, but the enemy force may vary over consecutive plans), we may benefit from experimenting with multiple diverse plans, even if some of them behaved poorly in a slightly different game configuration.

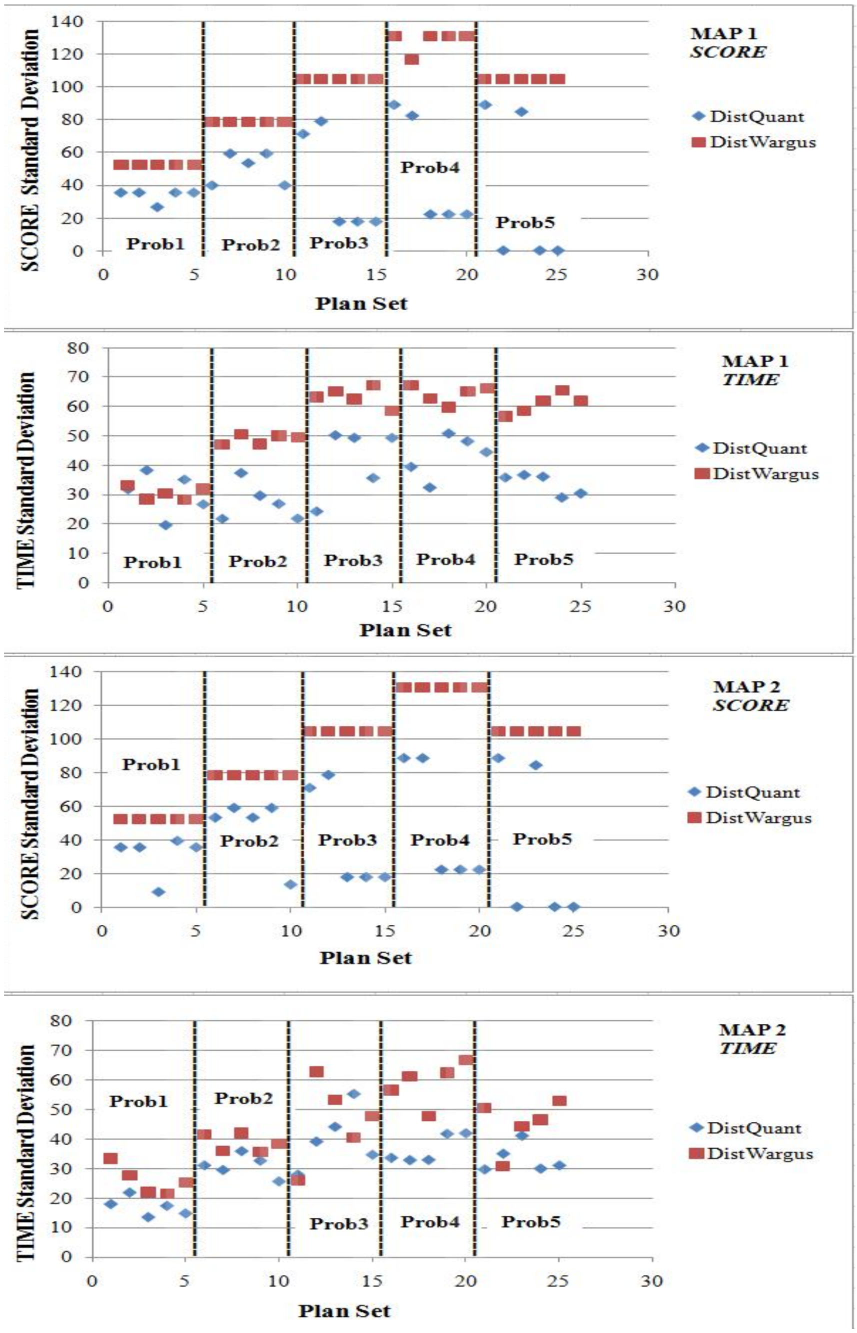


Fig. 5. Standard deviation of game scores and time (game duration)

6 Related Work

Case diversity was explored extensively in case-based recommender systems [2,8,9,10,18,19].

In case-based planning, we began to explore diversity in our previous work [3]. However, the focus there was on comparing plan diversity with state diversity, and we only demonstrated quantitative plan distance. While we also tested the diversity of plans by running them in the Wargus game, the game configurations were less sophisticated (with fewer unit types and simpler plans), as was our case-based planning system.

In generative planning (which involves generating plans from scratch, rather than through case retrieval and adaptation), quantitative plan diversity has been explored by Srivastava *et al.* [21]. A method for qualitative-diversity-aware plan generation has been proposed by Myers and Lee [12], in HTN planning. Their knowledge-intensive approach does not use distance metrics at plan generation time. Instead, it directs the generative planner towards regions of the search space which are identified as representing qualitatively different plan attributes, using a domain metatheory (an extended description of the planning domain in terms of high-level attributes, supplementing the standard domain model). In [4], we explore quantitative and qualitative plan diversity in generative planning.

To our knowledge, apart from our previous work [3,4], no other work on plan diversity (generative or case-based) assesses plan diversity by running plans in their environment, and observing behavior and results thus obtained. Instead, this is achieved by analyzing the plans themselves [12,21].

Myers [14] explores qualitative plan comparison (identifying similarities and differences between plans) through the use of a domain metatheory. The approach assumes an HTN planning paradigm, and defines plan distance purely on the basis of high-level characteristics specified in the metatheory. It does not deal with diverse plan generation, but with the computation of distance between already available plans. The related problem of plan stability is explored by Fox *et al.* [6]. Plan stability aims at reducing the difference between an original plan and a repaired plan. The similarity metric they use for this purpose is quantitative. Storyline diversity in a gaming environment, for the purpose of enhancing the player's experience, is explored by Paul *et al.* [16].

7 Conclusions and Future Work

Our work brings two main contributions to case-based plan diversity research. First, to our knowledge, we approach qualitative diversity in case-base planning for the first time. Second, we obtain qualitative plan diversity through the use of a qualitative plan distance metric at case retrieval time.

In a game domain, we show how qualitative plan diversity can, by reflecting characteristics specific to the domain in question, produce more meaningful plan variation than quantitative diversity. In addition, we evaluate the diversity of generated plans by running them in the environment and observing their behavior, as opposed to examining the structure of the plans themselves.

In future work, we aim to explore qualitative plan diversity in various real domains of practical interest, once again testing the diversity of plans by running them in the environments. We are also interested in exploring diversity in online planning, which should be particularly interesting in game domains, such as the one used herein.

We also plan to analyze the trade-off between plan diversity and plan quality; and to explore whether qualitative distance metrics can be used to help ensure that the sets of retrieved plans are not only diverse, but also composed of individual plans of good quality.

Acknowledgements. We thank the anonymous reviewers for their comments.

References

1. Boddy, M., Gohde, J., Haigh, T., Harp, S.: Course of Action Generation for Cyber Security Using Classical Planning. In: Biundo, S., et al. (eds.) Proc. of ICAPS 2005, pp. 12–21. AAAI Press, Menlo Park (2005)
2. Bridge, D., Kelly, J.P.: Ways of Computing Diverse Collaborative Recommendations. In: Wade, V.P., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 41–50. Springer, Heidelberg (2006)
3. Coman, A., Muñoz-Avila, H.: Case-based Plan Diversity. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 66–80. Springer, Heidelberg (2010)
4. Coman, A., Muñoz-Avila, H.: Generating Diverse Plans Using Quantitative and Qualitative Plan Distance Metrics. In: Proc. of AAAI 2011 (to appear, 2011)
5. Cox, M.T., Muñoz-Avila, H., Bergmann, R.: Case-based Planning. The Knowledge Engineering Review, 1–4 (2005)
6. Fox, M., Gerevini, A., Long, D., Serina, I.: Plan Stability: Replanning Versus Plan Repair. In: Long, D., et al. (eds.) Proc. of ICAPS 2006, pp. 212–221. AAAI Press, Menlo Park (2006)
7. Hoffmann, J., Nebel, B.: The FF Planning System: Fast Plan Generation Through Heuristic Search. Journal of Artificial Intelligence Research 14, 253–302 (2001)
8. McGinty, L., Smyth, B.: On the Role of Diversity in Conversational Recommender Systems. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 276–290. Springer, Heidelberg (2003)
9. McSherry, D.: Increasing Recommendation Diversity Without Loss of Similarity. In: Proc. of the Sixth UK Workshop on Case-based Reasoning, Cambridge, UK, pp. 23–31 (2001)
10. McSherry, D.: Diversity-Conscious Retrieval. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 219–233. Springer, Heidelberg (2002)
11. McSherry, D.: Completeness Criteria for Retrieval in Recommender Systems. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 9–29. Springer, Heidelberg (2006)
12. Myers, K.L., Lee, T.J.: Generating Qualitatively Different Plans through Metatheoretic Biases. In: Proc. of AAAI 1999, pp. 570–576. AAAI Press, Menlo Park (1999)
13. Myers, K.L., Tyson, W.M., Wolverton, M.J., Jarvis, P.A., Lee, T.J., des-Jardins, M.: PASSAT: A User-centric Planning Framework. In: Proc. of the 3rd Intl. NASA Workshop on Planning and Scheduling for Space, Houston, TX (2002)
14. Myers, K.L.: Metatheoretic Plan Summarization and Comparison. In: Long, D., et al. (eds.) ICAPS 2006, pp. 182–192. AAAI Press, Menlo Park (2006)

15. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: On-line Case-Based Planning. Computational Intelligence Journal 26(1), 84–119 (2010)
16. Paul, R., Charles, D., McNeill, M., McSherry, D.: MIST: An Interactive Storytelling System with Variable Character Behavior. In: Aylett, R., Lim, M.Y., Louchart, S., Petta, P., Riedl, M. (eds.) ICIDS 2010. LNCS, vol. 6432, pp. 4–15. Springer, Heidelberg (2010)
17. Serina, I.: Kernel Functions for Case-based Planning. Artificial Intelligence 174(16-17), 1369–1406 (2010)
18. Shimazu, H.: ExpertClerk: Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. In: Nebel, B. (ed.) Proc. of IJCAI 2001, pp. 1443–1448. Morgan Kaufmann, Seattle (2001)
19. Smyth, B., McClave, P.: Similarity vs. Diversity. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 347–361. Springer, Heidelberg (2001)
20. Spalazzi, L.: A Survey on Case-Based Planning. Artificial Intelligence Review 16, 3–36 (2001)
21. Srivastava, B., Kambhampati, S.T., Nguyen, M., Do, B., Gerevini, A.: Domain Independent Approaches for Finding Diverse Plans. In: Veloso, M.M. (ed.) Proc. of IJCAI 2007, pp. 2016–2022. AAAI Press, Menlo Park (2007)

On Dataset Complexity for Case Base Maintenance

Lisa Cummins and Derek Bridge

Department of Computer Science,
University College Cork,
Ireland
{l.cummins,d.bridge}@cs.ucc.ie

Abstract. We present what is, to the best of our knowledge, the first analysis that uses dataset complexity measures to evaluate case base editing algorithms. We select three different complexity measures and use them to evaluate eight case base editing algorithms. While we might expect the complexity of a case base to decrease, or stay the same, and the classification accuracy to increase, or stay the same, after maintenance, we find many counter-examples. In particular, we find that the RENN noise reduction algorithm may be over-simplifying class boundaries.

1 Introduction

Case base editing is a form of *case base maintenance* in which algorithms use heuristics to select cases to delete from case bases. In the context of case-based classification, *noise reduction algorithms* seek to delete noisy cases, with the goal of increasing classification accuracy; *redundancy reduction algorithms* seek to delete redundant cases, with the goal of increasing retrieval efficiency while, as much as possible, not harming classification accuracy.[1]

Suppose that we are comparing case base maintenance algorithms. How do we measure their efficacy? Obviously, we want to measure the number of cases that the algorithms delete from the case base: other things being equal, the more that are deleted the better. But we also need to see the effect on the resulting classifier. Often researchers report the change in the classification accuracy (or error) measured before and after maintenance on a separate held-out test set. But two problems arise.

The first problem, as is apparent from the previous paragraph, is that case base maintenance is a multi-objective optimization problem. We cannot, in general, maximize both the number of cases deleted and the accuracy of the resulting case base. We can see this in an extreme form by imagining an algorithm that proposes deletion of all but one case: deletion has been maximized but a case-based classifier will now incorrectly predict the class of all target problems whose true class is different from the class of the case that remains in the case base.

[1] This paper is based upon work partially supported by the Science Foundation Ireland under Grant Number 05/RFP/CMS0019.

In our previous work, we proposed two strategies for investigating the trade-off between these two objectives [3]. One was to compute the Pareto front, i.e. those algorithms not dominated by any other algorithms. The other was to combine the percentage of cases deleted and the percentage accuracy of the case base after maintenance into a single number. We proposed to use their harmonic mean.

But evaluating maintenance algorithms by the accuracy of the resulting case base has a second, little-recognized problem. Many of the maintenance algorithms use a classifier to decide which cases to delete (e.g. [1,5]). Assuming that the maintenance algorithm uses the same (or nearly the same) classifier as we use to measure accuracy pre- and post-maintenance, we have no independent measure of what the maintenance algorithm is achieving, and we run the risk that our maintenance algorithm is directly (or nearly directly) optimizing that which we measure. It would be useful to have independent measures that could be used to evaluate maintenance algorithms. In this paper, we investigate using *dataset complexity measures*.

Section 2 describes the maintenance algorithms that we will be evaluating. Section 3 reviews dataset complexity measures, and selects the three that we will use in this paper. Section 4 explains our experimental methodology. Then Sections 5, 6 and 7 present the results for noise reduction algorithms, redundancy reduction algorithms and composites that combine the two, respectively. Section 8 describes some related work, and Section 9 draws conclusions and presents ideas for future work. This, to the best of our knowledge, is the first paper to evaluate case base editing algorithms using dataset complexity measures. Our conclusions are therefore tentative ones. But we find reason to doubt the efficacy of the RENN noise reduction algorithm.

2 Case Base Editing Algorithms

The most common case-base maintenance algorithms are listed in Table 1.

As the Table shows, there are two types of case-base maintenance algorithm: those that delete noisy (or harmful) cases, and those that delete redundant cases. In practice, case-base maintenance algorithms are often composites, comprising a noise reduction phase followed by a redundancy reduction phase. For example, Brighton & Mellish's Iterative Case Filtering (ICF) algorithm comprises a noise-filtering phase using RENN followed by their own bespoke redundancy reduction phase [1]. So that we can separately evaluate the individual components of these composites, we have extracted the redundancy reduction phase of the composite algorithms, naming them in the table, and treating them as separate algorithms. For example, ICFR is our designation for the redundancy reduction phase of ICF, and we refer to ICF by the designation RENN$\rightarrow$ICFR to emphasize its composite nature.

It is well-known that the algorithms in Table 1 perform differently on different datasets (see, e.g., [1]): one size does not fit all in case base maintenance. This follows naturally from the fact that each of the atomic maintenance algorithms heuristically targets different types of cases to remove:

Table 1. Atomic case-base maintenance algorithms, and classic composites

Name used in this paper	Name in the literature	Description
Atomic noise reduction algorithms		
RENN	RENN	Repeated Edited Nearest Neighbour [17]
BBNR	BBNR	Blame-Based Noise Reduction [5]
Atomic redundancy reduction algorithms		
ICFR	—	Redundancy reduction phase of ICF [1]
RCR	—	Redundancy reduction phase of RC [13]
CRR	CRR	Conservative Redundancy Reduction [5]
Classic composite algorithms		
RENN→ICFR	ICF	Brighton & Mellish's Iterative Case Filtering [1]
RENN→RCR	RC	McKenna & Smyth's algorithm [13]
BBNR→CRR	CBE	Delany & Cunningham's Case-Base Editing algorithm [5]

- RENN regards a case as noisy if it has a different class to the majority of its k nearest neighbours [17].
- BNNR, by contrast, regards a case as noisy if it causes other cases to be mis-classified [5].
- ICFR and CRR both regard 'interior' cases, i.e. ones within clusters of same-class cases, as redundant ones, and both aim to retain cases on the boundaries between classes because these cases are important for classification accuracy.
  - ICFR removes cases that are solved by more cases than they themselves solve [1].
  - CRR removes cases that solve other cases [5].
- RCR, however, aims to retain a case if it is surrounded by many cases of the same class, while treating as redundant, and deleting, those that surround it [13].

In this paper, we want to further explore the effect of these algorithms by looking at changes they bring in the values of dataset complexity measures.

3 Measures for Maintenance

Ho & Basu survey dataset complexity measures in [9,10]. Subsequently, Orriols-Puig et al. have made available DCoL, the Data Complexity Library, which is an open-source C++ implementation of 13 measures based on those in the Ho & Basu survey [14].[2] However, we have found problems with the definitions of several of these measures. We present revised definitions in [2]. These are the measures whose labels contain a prime, e.g. F_2'. In [2], we also describe 4 additional measures: one is from the Case-Based Reasoning (CBR) literature [7]; we define the other 3 from ideas presented in the CBR literature [12,16]. Table 2 summarizes all 17 measures. Ho & Basu placed each complexity measure into one of three categories [9,10], and we have also shown these in the Table.

[2] `http://dcol.sourceforge.net/`

Table 2. The dataset complexity measures considered in this paper. (For formal definitions, see [2])

Measure	Description
Measures of overlap of attribute values	
F_1	Maximum Fisher's Discriminant Ratio
F'_2	Volume of Overlap Region
F'_3	Maximum Attribute Efficiency
F'_4	Collective Attribute Efficiency
Measures of separability of classes	
N'_1	Fraction of Instances on a Boundary
N_2	Ratio of Average Intra/Inter Class Distance
N_3	Error Rate of a 1NN Classifier
L_1	Minimized Sum of Error Distance of a Linear Classifier
L_2	Training Error of a Linear Classifier
C_1	Complexity Profile
C_2	Similarity-Weighted Complexity Profile
N'_5	Separability Emphasis Measure
Measures of geometry, topology and density of manifolds	
L_3	Nonlinearity of a Linear Classifier
N_4	Nonlinearity of a 1NN Classifier
T'_1	Fraction of Maximum Covering Spheres
T_2	Number of Instances per Attribute
T_3	Dataset Competence

For the purposes of this paper, we select just three of the complexity measures, one from each of the categories. We try to select ones that will reveal the changes that the maintenance algorithms bring about, and ones that are computed in a manner that is, as much as possible, independent of the way the maintenance algorithms work and the way we measure changes in accuracy.

Measures of overlap of attribute values. The measures in this category characterize the extent to which the attributes in a dataset discriminate between the class labels. An attribute that takes one value v in a case that is labeled by class c^+ but takes another value $v', v' \neq v$, in another case that is labeled by class c^- is less discriminatory than an attribute whose values align perfectly with the class labels.

In [2], we did not find these measures to be predictive of classifier accuracy. But we might expect them to be much more revealing of the effects of maintenance, especially of noise reduction. An incorrectly-labeled case may decrease the discriminatory power of one or more attributes.

We choose to use F'_3, which is our modified version of Ho & Basu's maximum attribute efficiency. The efficiency of an attribute is defined as the proportion of cases in the case base which have values for that attribute that do not fall into the overlap for that attribute. We define the overlap for an attribute as the set of its values that appear in differently labeled cases; in other words, if a's value is v in case x whose class is c^+, and a's value is v in another case x' whose class is c^-, then v is in a's overlap.

We do not use F_1 because we cannot apply it to datasets that include symbolic values or multi-class classification (where there are more than two class labels); and we avoid F_2' because, in the experiments that we describe in section 4, we found that most of its values on most datasets, both pre- and post-maintenance, tend to be zero. F_4' would be just as good as F_3': there is not much to choose between them here.

Measures of separability of classes. The measures in this category estimate the length and linearity of class boundaries. We found all of these measures to be predictive of classifier accuracy [2].

We exclude L_1 and L_2 because they cannot handle multi-class classification problems and symbolic-valued attributes. We exclude N_5' because it is simply the product of N_1' and N_2 and so produces values intermediate between the two.

C_1 and C_2 are new measures that we introduced in [2], where we found them to be the two most predictive of classifier accuracy. We based both of them on Massie et al.'s case complexity measure [12]. But, when we computed these measures pre- and post-maintenance, we found, overwhelmingly, that they disagreed with the other measures about how complexity had changed, and so we exclude them here.

This leaves N_1', N_2, N_3. We exclude N_3: it is computed using a 1NN classifier, and therefore may not be independent enough of the maintenance algorithms. The other two measures are reasonably independent of the maintenance algorithms. N_1' creates a minimum spanning tree (MST) and counts the proportion of cases in the MST that are connected to cases of a different class. In our version, we repeatedly shuffle and recompute the MST, and average the results, which allows for the fact that MSTs are not unique. N_2 is the ratio between the sum of the distances from each case to its nearest like neighbour and the sum of the distances from each case to its nearest unlike neighbour. We will choose N_1' since we found it to be more predictive of classifier accuracy than N_2 [2].

3.1 Measures of Geometry, Topology and Density of Manifolds

In this category, it is easy to choose T_1'. We exclude L_3 and N_4 because they cannot be computed on datasets with symbolic-valued attributes. T_2 is simply the number of instances in the dataset divided by the number of attributes: since the former will decrease (or, at least, not increase) and the latter will not change after maintenance, this measure is giving little insight beyond what we learn already from recording the percentage of cases deleted, so we exclude it. As for T_3, this is one of the new measures that we introduced in [2]; it is based on competence groups and their coverage [16]. We did not find this new measure to be predictive of classifier accuracy, and we have not found it to be useful here either: for all case bases and all maintenance algorithms, it shows complexity increasing, rather than decreasing, after maintenance. The notions of competence groups and their coverage were never designed for use as a complexity measure, and it is therefore no criticism of the competence idea that we have not found it to be useful either here or in [2]. It is possible that taking more account of the number of competence groups, rather than their coverage, would yield a more useful measure. Investigating this is left to future work.

For each case x in the case base, T_1' computes its adherence subset. This subset contains x itself and all its nearest neighbours up to, but excluding, the nearest unlike neighbour. T_1' is then the number of adherence subsets, ignoring those that are contained within, or equal to, another. While this measure was not very predictive of classifier accuracy [2], it behaves as we would expect for maintenance, mostly showing a decrease in complexity after maintenance.

In summary, then, we will proceed to use F_3', N_1' and T_1' in the rest of this paper.

4 Experimental Methodology

We took 25 datasets: 19 from the UCI repository [8]; plus the Breathalyser dataset [6]; and five email datasets [5]. For evaluation, we performed repeated holdout on each of the datasets. Each dataset was divided randomly into three splits: a 60% training set, a 20% test set, and a final 20% which was required for evaluation of other systems in our research (not reported in this paper) and hence was discarded here. We created 10 different splits of the data and we report all results as averages over the 10 splits.

We ran each of the maintenance algorithms in Table 1 on the training set and recorded the percentage of cases deleted. We also recorded the accuracy (percentage of cases correctly classified) on the held-out test set both before and after maintenance. Additionally, we computed the values of each of the 17 complexity measures on the case bases before and after maintenance. We normalized the values of the measure to the $[0, 100]$ range in order to make comparisons easier. We also had the problem that in some measures low values mean low complexity but in others high values mean low complexity. In this paper, when computing changes in complexity, we normalize further to arrange matters so that a positive change always means a lowering of complexity.

Although making a positive change mean a decrease in complexity is perhaps counter-intuitive, we do this for the benefit of graphs later in the paper. It means that moving upwards and rightwards in the graphs is a good thing, corresponding to decreases in complexity and increases in classification accuracy respectively. We show this in Figure 1. The upper-right quadrant shows that the maintenance algorithm has unambiguously improved accuracy and complexity. The top-left quadrant signifies lower complexity but also lower accuracy. This might occur if the maintenance algorithm over-simplifies class boundaries. The bottom-left quadrant signifies higher complexity and lower accuracy. There are any number of reasons that this might happen but one is the deletion of correctly-labeled cases near incorrectly-labeled ones, so that the incorrectly-labeled case will be retrieved for a larger number of target problems. The bottom-right quadrant is the situation of lower accuracy and higher complexity. Fortunately, our graphs

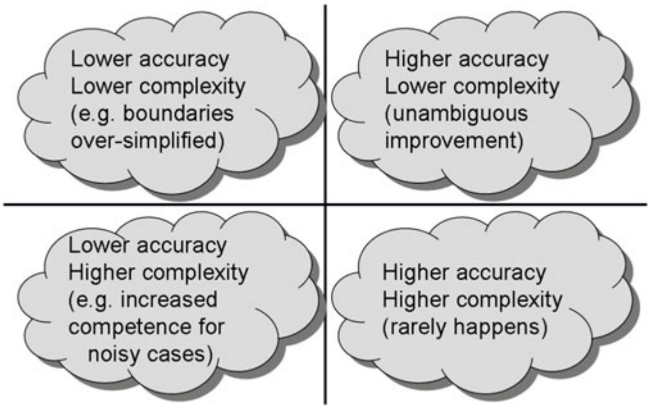


Fig. 1. Interpretation of graphs in this paper

Table 3. Percentage of cases deleted, change in accuracy, and change in normalized complexity measures: mean (and standard deviation) over 25 datasets. (In this paper, a positive change in a complexity measure implies lower complexity.)

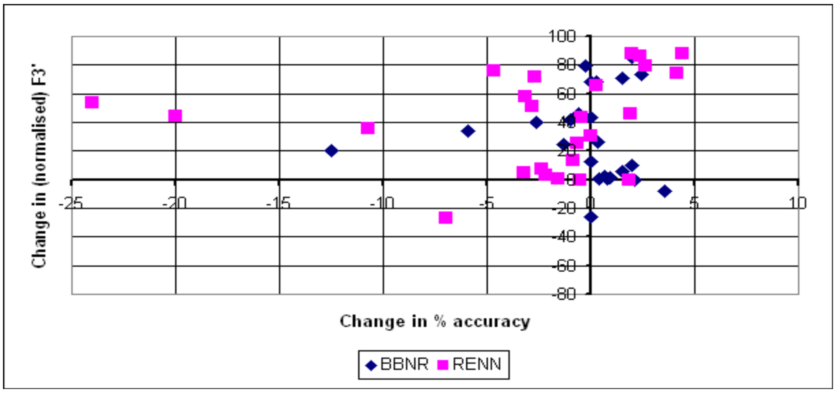
	% of cases deleted	Change in % accuracy	Change in F_3'	Change in N_1'	Change in T_1'
BBNR	23.08 (15.06)	-0.24 (3.15)	31 (31)	6 (5)	4 (5)
RENN	23.17 (18.67)	-2.70 (6.74)	41 (34)	22 (18)	23 (19)
CRR	35.94 (7.86)	-2.00 (2.75)	24 (32)	-14 (7)	-9 (6)
ICFR	61.85 (22.15)	-4.97 (2.75)	26 (34)	-37 (14)	-26 (15)
RCR	77.59 (9.57)	-2.93 (2.43)	32 (34)	-27 (9)	-14 (18)
BBNR→CRR	59.38 (12.58)	-1.54 (3.37)	44 (35)	15 (17)	16 (18)
RENN→ICFR	78.41 (16.12)	-5.90 (6.64)	48 (33)	-5 (26)	10 (22)
RENN→RCR	88.71 (2.11)	-4.25 (6.85)	49 (34)	1 (20)	8 (24)

show this to be very rare. Of course, as we said earlier, maintenance is a multi-objective optimization problem — losses in accuracy may be a price worth paying if the case base becomes more compact, and so it is not necessarily a 'bad thing' for a dataset and algorithm to be in a quadrant other than the top-right one.

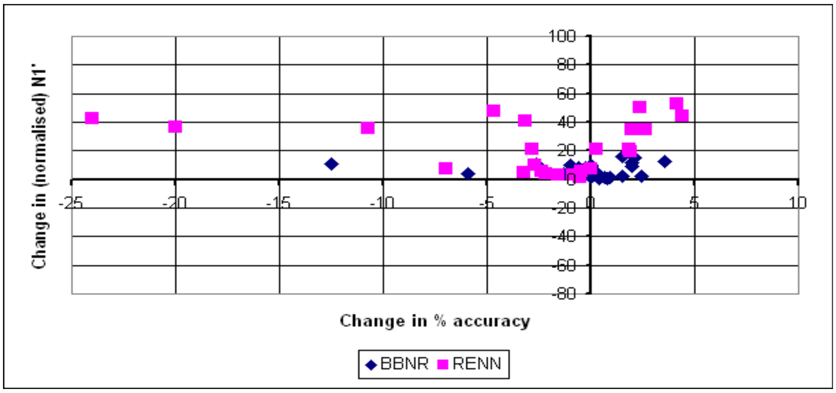
We summarize the results in Table 3, which shows averages over the 25 datasets, and reports the changes in the 3 measures that we selected in section 3.

5 Atomic Noise Reduction Algorithms

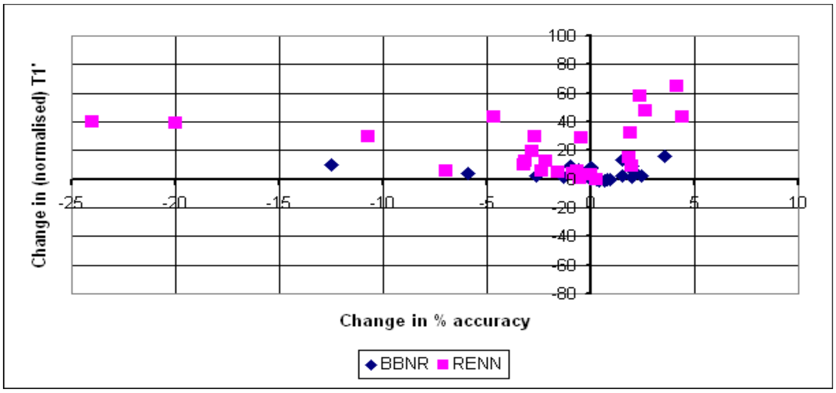
Figure 2 shows results for the atomic noise reduction algorithms, BBNR and RENN. The x-axis is the same in all three graphs in Figure 2. It is the change in accuracy on the held-out test set: positive values mean that accuracy is higher after maintenance than it was before maintenance. On the y-axis, we plot the



(a) F_3': Maximum Attribute Efficiency



(b) N_1': Fraction of Instances on a Boundary



(c) T_1': Fraction of Maximum Covering Spheres

Fig. 2. Changes in accuracy and complexity: atomic noise reduction algorithms

change in the normalized values of the complexity measure: F_3' in Figure 2a, N_1' in Figure 2b, and T_1' in Figure 2c. We repeat the point made earlier that, perhaps counter-intuitively, positive values here mean that the complexity has gone down after maintenance. Hence, the upper-right quadrant of every graph is where we might hope to be: increased accuracy and lower complexity. There are 50 points on each graph, representing what happens to each of the 25 datasets in the case of BBNR (diamonds) and in the case of RENN (squares).

We might expect that a good noise reduction algorithm would either improve accuracy or, if there was no noise, would leave it unchanged. But we see that the algorithms sometimes improve accuracy and sometimes worsen it. BBNR seems to do better than RENN: in 18 of the 25 datasets, BBNR improves accuracy or leaves it unchanged; in only 9 of the 25 does RENN do the same. In the case of RENN, some quite severe falls in accuracy occur (20-25%). We might think that the explanation would lie in the number of cases deleted. But, in fact, for most datasets, the two algorithms delete quite similar numbers of cases. The difference in the percentages deleted is less than or equal to 5% for 20 datasets. The mean percentage deleted over the 25 datasets by BBNR is 23.08%, where for RENN it is 23.17%. In only 7 of the datasets does RENN delete more, although in three of these the difference in the percentage deleted is more than 10%.

Both algorithms lower complexity, across all 3 complexity measures on nearly all datasets. Let's consider F_3' first (Figure 2a). In an extreme example, if two cases have exactly the same attribute values but are labeled by different classes, then one of the cases is noisy (unless there are other attributes in the domain that would distinguish the two cases but which are not part of the dataset). We would hope that a noise reduction algorithm would delete at least one of these cases and ones like them and this should be reflected in an improvement to F_3'. Figure 2a shows that both algorithms do seem to be successful in doing this, and to the same degree. The average change in F_3' in the case of BBNR is 31 with standard deviation (s.d.) also 31, and in the case of RENN it is 41 with s.d. 34.

A noise reduction algorithm might be expected also to 'clean up' the boundary between classes: removal of a noisy case should simplify the boundaries. This is what N_1' is supposed to measure, and Figure 2b confirms that both algorithms succeed in improving this measure of complexity. RENN does this more aggressively than BBNR. Its improvements to N_1' are often greater: it improves N_1' by 22 on average (s.d. 18) whereas BBNR improves it by 6 (s.d. 5). But this may explain why RENN loses accuracy and therefore places many datasets in the upper-left quadrant: it may be over-simplifying class boundaries.

Finally, by removing noisy cases, a noise reduction algorithm should produce a case base in which clusters of like cases are fewer in number but larger in size. This is what T_1' is supposed to measure, and Figure 2c shows again that this is what is happening: the changes are positive (meaning lower complexity), or close to zero, across all 25 datasets. Again RENN may be too aggressive: its mean is 23 (s.d. 19) compared to BBNR's mean of 4 (s.d. 5).

We conclude that, generally speaking, RENN's heuristic for identifying harmful cases is not as successful as BBNR's. It aggressively deletes cases, thereby over-simplifying the case base, achieving larger improvements in complexity but at the expense of classifier accuracy.

6 Atomic Redundancy Reduction Algorithms

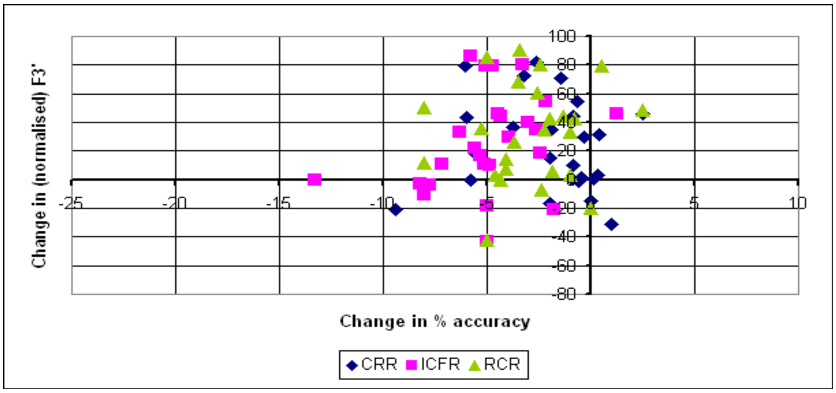
Figure 3 shows the performance of the three atomic redundancy reduction algorithms (CRR, ICFR and RCR). These algorithms are normally preceded by a noise reduction phase. It is important to be clear that here we are running these algorithms without previously doing any noise reduction. The algorithms may therefore not run as intended, because they usually expect to run on case bases with little or no noise, and the results will reflect this.

The standout observation across the three graphs in Figure 3 is that most points are to the left of the origin on the x-axis, showing that they mostly worsen classification accuracy. Of course, as explained above, this may be because the algorithms are not being applied to the kinds of case bases that they expect, i.e. ones that have undergone noise reduction. Another observation is that CRR (the diamond) tends to give smaller decreases in accuracy (mean -2%, s.d. 2.75) than RCR (the triangle, mean -2.93%, s.d. 2.43), which in turn gives smaller decreases than ICFR (the square, mean -4.97%, s.d. 2.75). This is only partly explained by looking at the percentage of cases deleted. CRR is conservative: it deletes the least and therefore does least damage to accuracy. Indeed it deletes 35.94% cases on average (s.d. 7.86). Counter-intuitively, though, RCR deletes the most (mean 77.59%, s.d. 9.57) with less damage to accuracy than ICFR, which deletes fewer cases on average (61.85%, s.d. 22.15).

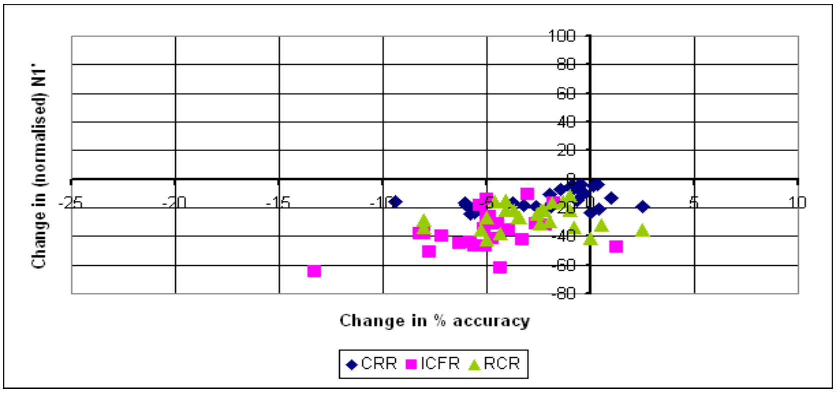
According to F_3', complexity becomes lower (Figure 3a), but in the case of both N_1' and T_1', complexity increases (Figures 3b and 3c). In the case of N_1' and T_1', most points are in the lower-left quadrants. One hypothesis is that noisy cases, which have not previously been removed by a noise reduction phase, remain untouched by the redundancy reduction algorithms, and that correctly-labeled cases are removed. Without these correctly-labeled cases, the noisy cases grow in competence. They will be among the neighbours of a wider range of target problems, leading to reduced classifier accuracy. And, with fewer correctly-labeled cases relative to incorrectly-labeled ones, boundaries are more complex and there are more clusters, and denser clusters, of noisy cases.

7 Composite Algorithms

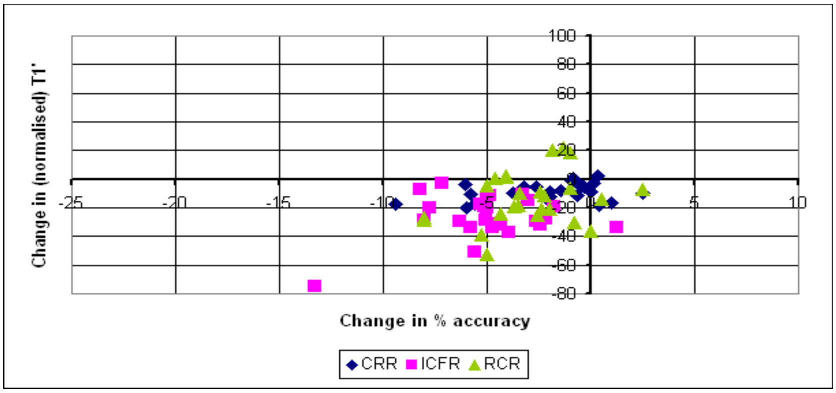
Figure 4 shows the effects of the composite algorithms, designated here by BBNR→CRR (diamond), RENN→ICFR (square) and RENN→RCR (triangle). We see a much greater spread in the effects on classifier accuracy: sometimes it improves; more often, it does not. BBNR→CRR does the least harm and has the narrowest spread (mean -1.54%, s.d. 3.37) compared to RENN→RCR (mean -4.25%, s.d. 6.85) and RENN→ICFR (mean -5.9%, s.d. 6.64), but it is also the ones that deletes the least — 59.8% compared to 78.41% and 88.71%.



(a) F'_3: Maximum Attribute Efficiency

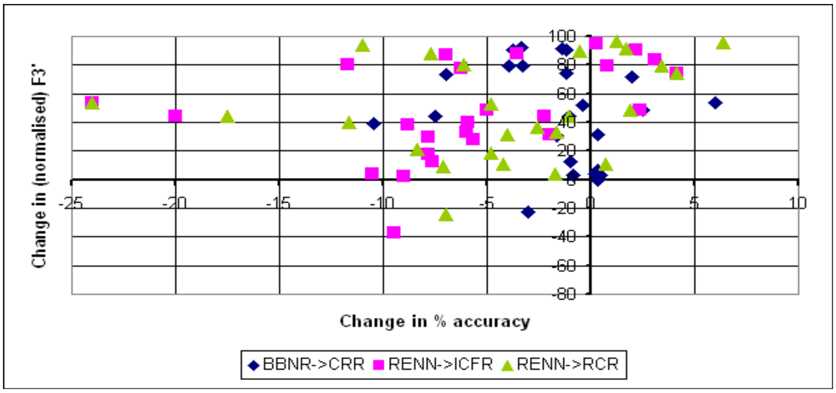


(b) N'_1: Fraction of Instances on a Boundary

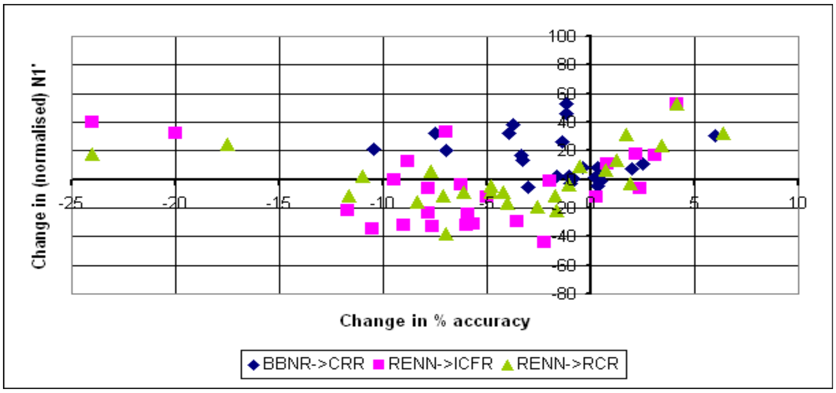


(c) T'_1: Fraction of Maximum Covering Spheres

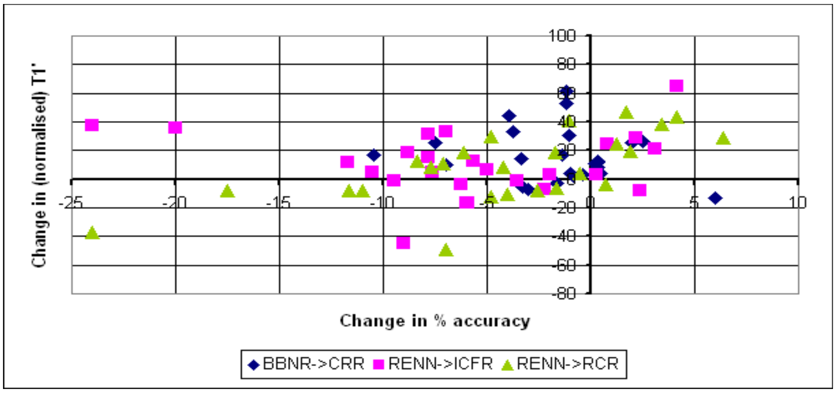
Fig. 3. Changes in accuracy and complexity: atomic redundancy reduction algorithms



(a) F_3': Maximum Attribute Efficiency



(b) N_1': Fraction of Instances on a Boundary



(c) T_1': Fraction of Maximum Covering Spheres

Fig. 4. Changes in accuracy and complexity: composite algorithms

Of course, these composites run redundancy reduction after noise reduction, and we find that the redundancy reduction is having a substantial effect. On its own, BBNR deletes 23.08% on average, but BBNR→CRR deletes 59.38% (36.3 more). RENN deletes 23.17% on average, but RENN→RCR deletes 88.71% (65.54 more) and RENN→ICFR deletes 78.41% (55.24 more).

The BBNR→CRR composite seems to have a clear advantage over its constituents, BBNR and CRR. BBNR→CRR deletes on average about the same amount as the sum of the amount deleted by BBNR alone and CRR alone. But the decrease in mean accuracy is less than the sum of the decreases caused by BBNR alone and CRR alone. In the cases of RENN→ICFR and RENN→RCR, the percentages of cases deleted by the composites exceed the amounts deleted by their constituents but, unless these very high levels of deletion are desirable, it might be better to use their redundancy constituents alone, since these still delete quite a lot of cases but with a smaller decrease in mean accuracy.

BBNR→CRR almost always lowers complexity: F_3' improves by 44 (s.d. 35), N_1' by 15 (17) and T_1' by 16 (18). And it achieves these results with the least damage to accuracy. Results are more mixed for the other two composites. They both make improvements to F_3' of nearly 50 (s.d. just over 30) and to T_1' of about 10 (s.d. just over 20). But according to N_1', RENN→ICFR makes complexity worse (-5 with s.d. 26) and RENN→RCR improves it but only by 1 (s.d. 20). Although BBNR→CRR and RENN→ICFR are more similar in their redundancy reduction phases (both aim to delete 'interior' cases), RENN→ICFR and RENN→RCR have the same initial noise reduction phase, and this seems to make their behaviour more similar in these results. Again, unless RENN's high levels of deletion are especially desirable, it may be better to avoid it in favour of BBNR→CRR, or just ICFR or RCR on their own.

In previous work, we briefly looked at the idea of composites in which ICFR and RCR were preceded by BBNR, rather than by RENN [3], and our new results here suggest that these should be investigated in more depth.

8 Related Work

There has been a number of studies that investigate the relationship between the dataset complexity measures and classifier accuracy, including [9,10,11]. But, we are presenting here the first ever study of complexity measures and case base editing algorithms.

The closest paper to our own is by Pranckeviciene et al. [15], which investigates a different form of maintenance, namely *dimensionality reduction*. They apply two dimensionality reduction techniques, Forward Feature Selection (FFS) and Linear Programming Support Vector Machine (LPSVM), to 5 high-dimension, sparse, bio-medical datasets. They choose 3 complexity measures to analyze the datasets before and after maintenance (using Ho & Basu's original definitions, not our modified ones): N_1, N_2, and T_1. The complexity results are somewhat mixed: complexity is sometimes reduced and sometimes not. The 3 measures that are used in [15] are selected to give insight into separability of classes, while being independent of classifiers. It might be interesting to choose a wider range of measures that provide different insights into the data.

9 Conclusions and Future Work

In this paper, we have reviewed a number of the most significant case base editing algorithms, along with a set of dataset complexity measures. We chose a small set of complexity measures, trying to ensure that they were different from each other (by choosing them from different categories of measures) and would reveal different insights into the operation of the maintenance algorithms, while at the same time preferring ones that were independent of classifiers (since some of the maintenance algorithms use classifiers in their operation). We then set up an experiment in which 8 maintenance algorithms were run on case bases created from 25 datasets. We reported the percentage of cases deleted, the change in accuracy, and the changes in the 3 complexity measures that we had selected.

This, to the best of our knowledge, is the first paper to evaluate case base editing algorithms using dataset complexity measures. Conclusions, at this stage, are only tentative, pending further research. In particular, we would like to use a wider range of datasets, e.g. image processing or bioinformatics datasets. Additionally, experiments with artificial datasets could be useful. They would allow us to control the degree of noise or possibly even the complexity.

In this paper, we have found reason to believe that the RENN noise reduction algorithm may be too aggressive. It often deletes a lot of cases but sometimes at the price of quite large decreases in classifier accuracy. At the same time, complexity often increases. This suggests that RENN may be over-simplifying class boundaries. Composite algorithms that use RENN as their initial phase are affected in a similar fashion. Indeed, the problems are compounded, whether the subsequent redundancy reduction phase targets interior cases (as with ICFR) or boundary cases (RCR).

The implications are that RENN needs much closer investigation. If it is important to delete a lot of cases, it may be better to use the ICFR or RCR redundancy reduction algorithms alone: fewer cases will be deleted, but the damage to accuracy may be lower.

We have found that BBNR may be doing a better job at identifying noise than RENN. It deletes as much on its own as RENN does, with less damage to accuracy. As suggested in [3], there is reason to investigate novel composites, in which BBNR precedes ICFR or RCR.

Finally, we think the ideas in [4] may be relevant to future work. In [4], Delany identifies 8 types of case profile, depending on the non-emptiness or otherwise of four sets associated with each case (the reachability set, the coverage set, the liability set and the dissimilarity set). Her paper looks empirically at what happens when cases with certain profiles are deleted. It would be interesting to see what effect deletion of cases with different profiles has on dataset complexity.

References

1. Brighton, H., Mellish, C.: On the consistency of information filters for lazy learning algorithms. In: Rauch, J., Żytkow, J.M. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 283–288. Springer, Heidelberg (1999)
2. Cummins, L.: Combining and Choosing Case Base Maintenance Algorithms. PhD thesis, Department of Computer Science, University College Cork, Ireland (forthcoming, 2011)
3. Cummins, L., Bridge, D.: Maintenance by a committee of experts: The MACE approach to case-base maintenance. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 120–134. Springer, Heidelberg (2009)
4. Delany, S.J.: The good, the bad and the incorrectly classified: Profiling cases for case-base editing. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 135–149. Springer, Heidelberg (2009)
5. Delany, S.J., Cunningham, P.: An analysis of case-based editing in a spam filtering system. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 128–141. Springer, Heidelberg (2004)
6. Doyle, D., Cunningham, P., Bridge, D., Rahman, Y.: Explanation oriented retrieval. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 157–168. Springer, Heidelberg (2004)
7. Fornells, A., Recio-García, J.A., Díaz-Agudo, B., Golobardes, E., Fornells, E.: Integration of a methodology for cluster-based retreval in jcolibri. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 418–433. Springer, Heidelberg (2009)
8. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
9. Ho, T.K., Basu, M.: Measuring the complexity of classification problems. In: Procs. of the 15th Intl. Conference on Pattern Recognition, pp. 43–47 (2000)
10. Ho, T.K., Basu, M.: Complexity measures of supervised classification problems. IEEE Trans. on Pattern Analysis and Machine Intelligence 24(3), 289–300 (2002)
11. Macià, N., Bernadó-Mansilla, E., Orriols-Puig, A.: On the dimensions of data complexity through synthetic data sets. In: Procs. of the 11th Intl. Conference of the Catalan Association for Artificial Intelligence, pp. 244–252 (2008)
12. Massie, S., Craw, S., Wiratunga, N.: Complexity profiling for informed case-base editing. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 325–339. Springer, Heidelberg (2006)
13. McKenna, E., Smyth, B.: Competence-guided case-base editing techniques. In: Blanzieri, E., Portinale, L. (eds.) EWCBR 2000. LNCS (LNAI), vol. 1898, pp. 186–197. Springer, Heidelberg (2000)
14. Orriols-Puig, A., Macià, N., Bernadó-Mansilla, E., Ho, T.K.: Documentation for the data complexity library in C++. Technical Report GRSI Report No. 2009001, Universitat Ramon Llull (2009)
15. Pranckeviciene, E., Ho, T.K., Somorjai, R.: Class separability in spaces reduced by feature selection. In: Procs. of the 18th Intl. Conference on Pattern Recognition, pp. 254–257 (2006)
16. Smyth, B., McKenna, E.: Modelling the competence of case-bases. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 208–220. Springer, Heidelberg (1998)
17. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Trans. on Systems, Man, and Cybernetics 6(6), 448–452 (1976)

Improving Case Retrieval by Enrichment of the Domain Ontology

Valmi Dufour-Lussier[1,2], Jean Lieber[1,3],
Emmanuel Nauer[1,4], and Yannick Toussaint[1,5]

[1] LORIA – UMR 7503. B.P. 239, F-54506,
Vandœuvre-lès-Nancy CEDEX, France
[2] Université Nancy 2
[3] Université Henri Poincaré, Nancy 1
[4] Université Paul Verlaine, Metz
[5] INRIA – Nancy Grand-Est
`firstname.lastname@loria.fr`

Abstract. One way of processing case retrieval in a case-based reasoning (CBR) system is using an ontology in order to generalise the target problem in a progressive way, then adapting the source cases corresponding to the generalised target problem. This paper shows how enriching this ontology improves the retrieval and final results of the CBR system. An existing ontology is enriched by automatically adding new classes that will refine the initial organisation of classes. The new classes come from a data mining process using formal concept analysis. Additional data about ontology classes are collected specially for this data mining process. The formal concepts generated by the process are introduced into the ontology as new classes. The new ontology, which is better structured, enables a more fine-grained generalisation of the target problem than the initial ontology. These principles are tested out within TAAABLE,[1] a CBR system that searches cooking recipes satisfying constraints given by a user, or adapts recipes by substituting certain ingredients for others. The ingredient ontology of TAAABLE has been enriched thanks to ingredient properties extracted from recipe texts.

Keywords: ontology refinement, formal concept analysis, progressive retrieval.

1 Introduction

This paper studies the effect of the domain ontology on a case-based reasoning (CBR) system and the improvement of result quality using a more fine-grained ontology.

TAAABLE [2] is a system that has been originally designed as a candidate of the *Computer Cooking Contest*. It is also used as a brain teaser for research in knowledge-based systems, including CBR and ontology engineering. Like many CBR systems, TAAABLE uses an ontology to retrieve the source cases that are the most similar to a target case. TAAABLE retrieves and creates cooking recipes by adaptation. According to constraints given by the user, such as inclusion or rejection of ingredients, the type or the origin of the dish, the compatibility with some diets (vegetarian, nut-free, etc.), the system looks up, in the recipe base (which is a case base), whether some recipes satisfy

[1] `http://taaable.fr`

these constraints. Recipes, if they exist, are returned to the user; otherwise the system is able to retrieve similar recipes (i.e. recipes that match the target query partially) and adapts these recipes, creating new ones. Searching similar recipes is guided by several ontologies, i.e. hierarchies of classes (ingredient hierarchy, dish type hierarchy and dish origin hierarchy), in order to relax constraints by generalising the user query. Relaxation is iterative and progressive. The goal is to find the most specific generalisation of the target case (the one with the minimal cost) for which recipes exist in the case base. Adaptation consists of substituting some ingredients of the source cases by the ones required by the user.

Many CBR systems use class or index hierarchies (taxonomies, ontologies, object-based hierarchies [5], etc.) for retrieval, especially to compute similarity between cases. Therefore, the ontology structure impacts retrieval, raising the issue of how to refine this ontology for improving this process. This paper proposes a novel approach to automatise this refinement.

The main idea for realising this goal is as follows. In a given ontology, the siblings of a class C may be either close to C of far from it, depending on the level of details of this part of the ontology. For instance, if an ingredient ontology is very detailed for berries (i.e. with a deep hierarchy below the class `Berry`) and very coarse for vegetables (i.e. each vegetable class is a direct subclass of the class `Vegetable`) then the similarity between blackberries and blueberries would counter-intuitively be much lower than the similarity between pumpkins and parsnips. Generalisation of the query is the basis for case retrieval and it is computed using a similarity measure defined on ontologies. Introducing new classes into the ontology provides a more fine-grained generalisation. Going up in a more structured hierarchy will return less and more similar cases than going up in a less structured hierarchy.

A data mining process based on formal concept analysis (FCA) extracts new classes and enriches the initial ontology by structuring it better. Additional sources are collected and used to introduce new properties about classes of the initial ontology. This work helps avoid adaptation failures, i.e. creating a recipe that is not *cookable* because some actions that were applied to the substituted ingredient are not applicable to the substitution ingredient: for instance, when substituting *mascarpone* (a creamy cheese) for *mozzarella* (a semi-firm cheese), the *slice* action will not be applicable anymore.

The paper is organised as follows: Section 2 presents TAABLE inference engine principles, Section 3 describes related work about existing approaches for improving retrieval in CBR and ontology refinement using FCA, Section 4 details the ontology refinement process, and Section 5 shows on an example the effect of the refined ontology in the TAABLE system.

2 Reasoning Principles in TAABLE

TAABLE retrieves cases using query generalisation, then adapts them by substitution. This section gives a simplified description of the TAABLE system. For more details about the TAABLE inference engine, see e.g. [2].

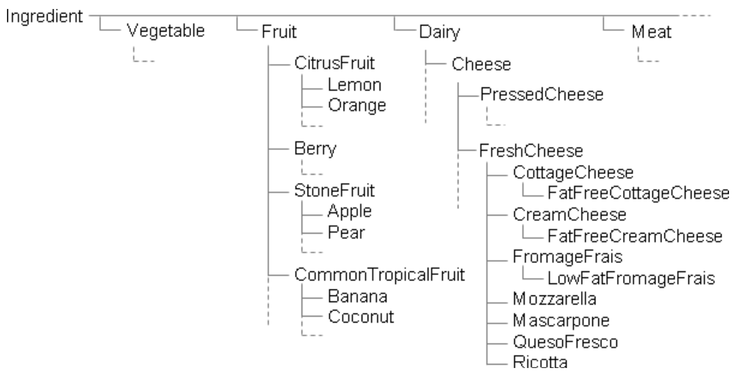


Fig. 1. A part of the ingredient hierarchy, including the details of the `FreshCheese` class

Case retrieval

Domain ontology. An ontology $\mathcal{O}$ defines the main classes and relations relevant to cooking, and guides retrieval. $\mathcal{O}$ is a set of atomic classes organised into several hierarchies (ingredient, dish type, dish origin). Given two classes `B` and `A` of this ontology, `A` is subsumed by `B`, denoted by "$\mathtt{B} \sqsupseteq \mathtt{A}$", iff the set of instances of `A` is included in the set of instances of `B`. For instance, "`Riccota` (referring to the Italian fresh cheese) is subsumed by `FreshCheese`" means that all instances of `Riccota` are instances of `FreshCheese`. For the sake of simplicity, only the ingredient hierarchy is considered in the remainder of this paper. A part of this hierarchy is given in Fig. 1.

Source cases. For the inference engine, recipes are represented by a conjunction of the ingredients they call for, and is encoded in the same language as the one of the queries. Thus, a recipe with ricotta, tomato, olive oil, and salt, is represented by:

$$R = \exists\mathtt{ing.Ricotta} \sqcap \exists\mathtt{ing.Tomato} \sqcap \exists\mathtt{ing.OliveOil} \sqcap \exists\mathtt{ing.Salt} \quad (1)$$

Target query. In the TAABLE system, a query Q is a conjunction of terms $\exists\mathtt{ing}.I$, where I is a class in the ingredient hierarchy, meaning that the recipe should contain the ingredient I. For example, searching a recipe with tomato and fresh cheese corresponds to $Q = \exists\mathtt{ing.Tomato} \sqcap \exists\mathtt{ing.FreshCheese}$. If `C` and `D` are two conjunctions (for example, two queries), $\mathtt{C} = \bigsqcap_i \exists\mathtt{ing.A}_i$ is subsumed by $\mathtt{D} = \bigsqcap_j \exists\mathtt{ing.B}_j$ if, for each j, there exists at least one i such that $\mathtt{A}_i$ is subsumed by $\mathtt{B}_j$ in $\mathcal{O}$.

Given a recipe R and a query Q, R solves Q if the set of recipes represented by R is included in the set of recipes represented by Q. Thus, the recipe represented by (1) solves the query $Q = \exists\mathtt{ing.FreshCheese} \sqcap \exists\mathtt{ing.Tomato}$. This can be computed simply by testing if R is more specific than Q. It is important to note that R solves Q does not mean that it is a "good" solution, even if the hypothesis that recipes of the case base are "good" is made (this is a working hypothesis: as no information about the quality of the recipes in the case base is available, it is simplest to consider the recipes as being good or, at least, equally good).

Algorithm. The algorithm consists of an iterated generalisation of the target query minimising cost, until at least one recipe of the case base solves the modified query. The query that has been modified at iteration t is noted $\Gamma_t(Q)$, and Γ_0 is the identity. The generalisation cost of Γ_t, denoted by $\mathtt{cost}(\Gamma_t)$, grows with t (see below for the `cost` function definition). Each generalisation Γ_t is written as a composition of substitutions $\mathtt{A} \rightsquigarrow \mathtt{B}$ such that $(\mathtt{A},\mathtt{B})$ is a specialisation link between `A` and `B` in the hierarchy representing the ontology. Furthermore, this composition must be applied to Q: if $\Gamma = (\mathtt{A}_p \rightsquigarrow \mathtt{B}_p) \circ \ldots \circ (\mathtt{A}_2 \rightsquigarrow \mathtt{B}_2) \circ (\mathtt{A}_1 \rightsquigarrow \mathtt{B}_1)$ then $\mathtt{A}_1$ is an element of the conjunction Q, $\mathtt{A}_2$ is an element of the conjunction $(\mathtt{A}_1 \rightsquigarrow \mathtt{B}_1)(Q)$, etc. The algorithm running through the generalisation space of Q is an A* algorithm which takes as parameter the cost function. It stops when at least one recipe R such that R solves $\Gamma_t(Q)$ is found.

Cost function. Let Γ be a composition of substitutions. Function `cost` is additive for the composition: $\mathtt{cost}(\sigma_2 \circ \sigma_1) = \mathtt{cost}(\sigma_1) + \mathtt{cost}(\sigma_2)$. For a basic substitution $\sigma = \mathtt{A} \rightsquigarrow \mathtt{B}$ (where `A` is subsumed by `B`), the cost is computed as follows:

$$\mathtt{cost}(\mathtt{A} \rightsquigarrow \mathtt{B}) = \mu(\mathtt{B}) - \mu(\mathtt{A}) \quad \text{with} \quad \mu(\mathtt{X}) = \frac{\text{number of recipes with } \mathtt{X}}{\text{total number of recipes}}$$

Therefore, $0 \leq \mathtt{cost}(\mathtt{A} \rightsquigarrow \mathtt{B}) \leq 1$.

Function `cost` has the following property: if Γ has a cost computed with respect to a given ontology and a new node is added into this ontology (without changing the recipe base), then the cost of Γ will not change. For instance, if $\Gamma = \mathtt{Ricotta} \rightsquigarrow \mathtt{FreshCheese}$ and the `ItalianFreshCheese` class is "added" into the ontology "between" `Ricotta` and `FreshCheese`, the cost of Γ remains unchanged. So, there is an essential difference between this cost function and one based on the number of edges separating two nodes (number that grows by adding an intermediate node). This property ensures stability of the cost regardless of the introduction of intermediate classes.

Adaptation. Let Q be a query, Γ a generalisation function of this query, and R a recipe solving $\Gamma(Q)$. Let $\mathtt{T}_i$, $\mathtt{B}_j$ and $\mathtt{S}_k$ be classes such as $Q = \bigsqcap_i \exists\mathtt{ing}.\mathtt{T}_i$ (`T` as target), $\Gamma(Q) = \bigsqcap_j \exists\mathtt{ing}.\mathtt{B}_j$ and $R = \bigsqcap_k \exists\mathtt{ing}.\mathtt{S}_k$ (`S` as source). $\Gamma(Q)$ is more general than both R and Q, thus for all j, there exists k and i such that $\mathtt{S}_k$ and $\mathtt{T}_i$ are subsumed by $\mathtt{B}_j$. Then the adaptation consists of substituting $\mathtt{T}_i$ for $\mathtt{S}_k$ for all $\mathtt{B}_j$, such that $\mathtt{B}_j \neq \mathtt{T}_i$. If several $\mathtt{T}_i$ or $\mathtt{S}_k$ correspond to one $\mathtt{B}_j$, several adaptations are possible and are returned to the user. These substitutions, along with the preparation instructions, give the modifications that must be applied to the recipe in order to address the query.

Example. Let $Q = \exists\mathtt{ing}.\mathtt{Tomato} \sqcap \exists\mathtt{ing}.\mathtt{Mascarpone} \sqcap \exists\mathtt{ing}.\mathtt{Oil}$. Among the generalisations of this query, say that $\Gamma = \mathtt{Mascarpone} \rightsquigarrow \mathtt{FreshCheese}$ is the substitution with the minimal cost (apart from identity). As `Ricotta` is subsumed by `FreshCheese` in the ontology, TAAABLE retrieves the recipe R given by (1). The adaptation will consist in substituting $\mathtt{S}_k = \mathtt{Ricotta}$ by $\mathtt{T}_i = \mathtt{Mascarpone}$ (generalisation-specialisation through $\mathtt{B}_j = \mathtt{FreshCheese}$).

However, a non-realistic adaptation can be returned, in which some actions have to be applied to objects on which they cannot be applied to. We will address this problem after introducing our approach to ontology refinement.

Table 1. A binary context with 6 objects (cheeses from the `FreshCheese` category) described by 5 properties (the `FreshCheese` category and 4 cooking actions which can be applied to these cheeses)

	`FreshCheese`	`sliceAble`	`beatAble`	`cutAble`	`mashAble`
`Mascarpone`	×		×		
`Ricotta`	×		×	×	×
`Mozzarella`	×	×		×	
`FromageFrais`	×		×		
`CottageCheese`	×		×		×
`QuesoFresco`	×		×	×	×

3 Formal Concept Analysis for Ontology Refinement

The goal of the refinement process is to add intermediate classes into the initial hierarchy of the system. These additional classes will enable for a better distinction of the similarity between classes immediately subsumed initially by a same class. Fig. 2 illustrates how B, C, and D which are indistinguishably similar on the left-hand side are distinguished better after introducing an intermediate class (in this example, C and D more similar to each other than to B).

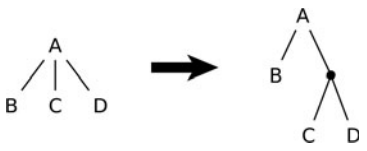
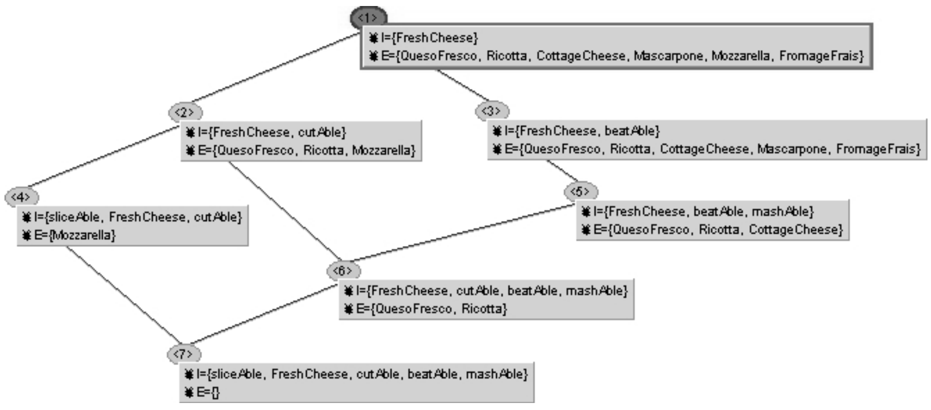


Fig. 2. Refining the ontology by introducing a new class

3.1 Formal Concept Analysis

FCA is a classification method based on lattice theory. A *formal context* is a triple $\mathcal{K} = (G, M, I)$, where G is a set of individuals (called *objects*), M a set of properties (called *attributes*) and I the relation on $G \times M$ stating that an object is described by a property [9]. Table 1 shows an example context: G is a set of 6 objects (which are fresh cheese subclasses), M is the set of 5 properties applicable to those cheeses. Among theses properties, one (`FreshCheese`) comes from the initial hierarchy. `FreshCheese` is a class which is more general than the 6 cheese classes that have to be classified (in other words, `FreshCheese` denotes both the class of fresh cheese in the ontology and the property of being a fresh cheese). The 4 other properties are cooking actions that can be applied: `sliceAble` (can be sliced), `beatAble` (can be beaten), `cutAble` (can be cut), and `mashAble` (can be mashed).

The approach for merging properties coming from several contexts, for a given set of objects, is called *context apposition*. Formally, if $\mathcal{K}_1 = (G, M_1, I_1)$ and $\mathcal{K}_2 = (G, M_2, I_2)$ are two binary contexts about the same set of objects G, a binary context, written $\mathcal{K}_1|\mathcal{K}_2$ merging $\mathcal{K}_1$ and $\mathcal{K}_2$ can be built by: $\mathcal{K}_1|\mathcal{K}_2 = (G, M_1 \cup M_2, I_1 \cup I_2)$ [4].



<1>
I={FreshCheese}
E={QuesoFresco, Ricotta, CottageCheese, Mascarpone, Mozzarella, FromageFrais}
<2>
I={FreshCheese, cutAble}
E={QuesoFresco, Ricotta, Mozzarella}
<3>
I={FreshCheese, beatAble}
E={QuesoFresco, Ricotta, CottageCheese, Mascarpone, FromageFrais}
<4>
I={sliceAble, FreshCheese, cutAble}
E={Mozzarella}
<5>
I={FreshCheese, beatAble, mashAble}
E={QuesoFresco, Ricotta, CottageCheese}
<6>
I={FreshCheese, cutAble, beatAble, mashAble}
E={QuesoFresco, Ricotta}
<7>
I={sliceAble, FreshCheese, cutAble, beatAble, mashAble}
E={}

Fig. 3. The lattice corresponding to the binary context given in Table 1

A *formal concept* is a pair (P, O) where P is a set of properties (the *intent* of the formal concept), O is a set of objects (the *extent* of the formal concept), such that (1) P is the set of all properties shared by objects in O and (2) O is the set of all objects sharing properties P. For example, $(\{\texttt{FreshCheese}, \texttt{beatAble}, \texttt{mashAble}\}, \{\texttt{QuesoFresco}, \texttt{Ricotta}, \texttt{CottageCheese}\})$ is a formal concept (node number 5 in Fig. 3).

The set $\mathcal{C}_{\mathcal{K}}$ of all formal concepts of the context $\mathcal{K} = (G, M, I)$ is partially ordered by extent inclusion, also called the *specialisation* (denoted by $\leq_{\mathcal{K}}$), between concepts. $\mathcal{L} = \langle \mathcal{C}_{\mathcal{K}}, \leq_{\mathcal{K}} \rangle$ is a complete lattice, called the *concept lattice*. The lattice $\mathcal{L}$ can be drawn as a Hasse diagram where nodes are concepts, and edges are specialisation links. Fig. 3 illustrates the lattice corresponding to the binary context given in Table 1. The top concept (with number 1, in the figure) contains all the objects. In our example, its intent is `FreshCheese`, a property shared by all the objects. By contrast, the bottom concept is defined by the set of all properties. In our example, its extent is empty as none of the objects is described by all the properties.

A number of algorithms have been proposed for the construction of concept lattices (see [9]). For our application, we used CORON, a software platform implementing a rich set of algorithmic methods for symbolic data mining, including concept lattice construction algorithms [11]. Fig. 3 was generated by GALICIA,[2] another FCA tool, which provides visualisation functionalities.

3.2 Ontology Building Using FCA

Many works propose using FCA to build ontologies. Cimiano [6] proposes an approach to build a concept hierarchy from texts in the tourism domain. The objects (hotels, cars, bicycles, tours, etc.) are characterised by the actions that can be applied to them (can be reserved, can be driven, can be rented, etc.). In order to reduce the lattice size, a numerical classification is used as a first step of the binary context building.

[2] http://www.iro.umontreal.ca/~galicia

Stumme *et al.* [10] propose an approach to merging two ontologies. The ontology classes are identified in the texts and a binary context is built for each of the two ontologies, in which classes are characterised by the documents in which they appear. Context apposition is used for aggregating the classes of the two ontologies according to their presence in the texts.

Bendaoud *et al.* [4] show how FCA can be used as a unified framework for ontology building and refinement. They point out that the information that may be extracted from the texts depends on the textual resources itself. Thus, from a scientific paper, objects can be extracted and properties can be linked to objects as it is done in [6]. But an ontology organises classes of a given domain into a hierarchy. This information about the specific/generic organisation is seldom present in scientific papers. In [4], these two types of information are managed separately and are then combined with context apposition to build a single lattice.

We use this type of approach to build the ingredient ontology in the cooking domain. Recipe texts are used in order to characterise the ingredients according to how they are cooked, and the initial ontology is used in order to get the specific/generic relations.

4 Ontology Refinement Process

FCA has been successfully used in CBR: in [1] for building cases from texts (objects correspond to texts and properties to relevant terms), in [7] for organising a case base for the purpose of retrieval. In the present work, FCA is used for ontology refinement.

The goal of the refinement process is to add intermediate classes into the initial hierarchy of the system. To do that, the classes of the initial ontology are characterised with additional properties. These properties enable the creation of a binary context that will be exploited for building a concept lattice with FCA. The formal concepts of the lattice are inserted into the initial hierarchy in order to enrich its organisation. The following four steps describe the ontology building and refinement process. We detail those steps with examples about an ingredient hierarchy refinement.

4.1 Ingredient Hierarchy Overview

The initial ingredient hierarchy has been semi-automatically created from common sense classifications (e.g. a *citrus fruit* is a *fruit*). All ingredients used in the recipe base have been introduced. The ingredient hierarchy currently contains 1500 classes. An excerpt is shown in Fig. 1. The ontology (and the recipes of the case base) are managed in a semantic wiki [3]. The *high* level of the hierarchy contains general classes: `Vegetable`, `Fruit` , `Dairy`, `Meat`, etc. The *low* level of the hierarchy contains classes that are ingredients actually used in the recipes.

From the sole hierarchy, the *similarity* between two *sibling ingredients* (i.e. immediate subclasses of a single class) cannot be ranked. For example, mascarpone is as close to mozzarella as it is to ricotta. Consequently, when a query is generalised for case retrieval, if `Mascarpone` is generalised to `FreshCheese`, the set of remaining cases will contain recipes using mozzarella as well as recipes using mascarpone. This is why we want to refine the hierarchy organisation for gathering initial classes that are similar (i.e. classes that share at least one property). Generalising a class C into a parent concept will ensure that sibling classes of C own at least the same properties as C. The generalisation will be more fine-grained.

4.2 Local Refinement

The action of refining a limited part of the ontology is called *local refinement*. For instance, we could refine locally the subpart about `Nut` (walnut, pecan nut, etc.), the subpart about `Berry` (raspberry, blueberry, etc.), the subpart about `FreshCheese`, etc. This approach has been preferred to global refinement for several reasons:

– Focusing on a part of the hierarchy (i.e. a subset of classes) is a way to limit the size of the binary context by reducing the number of objects and the number of associated properties. A reduced context will produce a lattice with a smaller size, which can be viewed and interpreted more easily.
– We do not want to build classes gathering ingredients that do not belong in the same part of the hierarchy. For instance, we do not want to group kiwis and onions merely because it is possible to apply a given action, such as *to peel*, to both.
– For each part of the hierarchy, the properties taken into account for the binary context may be different: *to toast* makes sense for nuts but not for liquids.

Therefore we ought to focus on the refinement of *(bottom)* parts of the hierarchy, while maintaining the *top* structure. The local refinements will be plugged into the initial ontology (see hereafter).

Formally, R refers to the class which is the root of the part of the hierarchy that has to be refined. The context $\mathcal{K}_R = (G_R, M_R, I_R)$ is the binary context composed of:

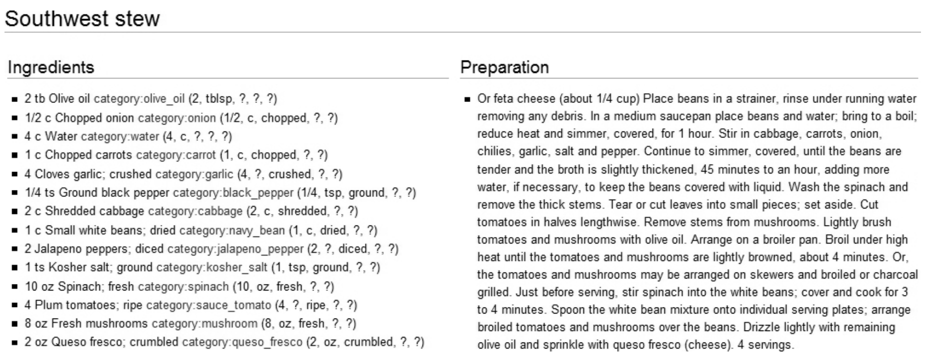
– a set of objects: in this work, objects used for FCA are classes of the ingredient hierarchy, e.g. `Orange`, more specific than R: $G_R = \{x \mid R \sqsupseteq x\}$,
– a set of properties (cooking actions) M_R selected to characterise and discriminate the objects of G_R,
– the relation I_R such that I_R relates $i \in G_R$ to $a \in M_R$ if the action a can be applied to the ingredient i.

The next two sections illustrate the refinement process for $R =$ `FreshCheese`.

4.3 Building the Binary Context for the Refinement

The ontology refinement aims at modifying an existing ontology. Properties of the initial ontology must be retained, and new properties introduced to organise the classes (ingredients) better against each other. Aggregating ricotta with mascarpone and distinguishing these two fresh cheeses from mozzarella is an expected outcome. Two binary contexts must be merged:

– A binary context capturing the initial ontology structure by associating to each ingredient class their superclasses, e.g. the object `Ricotta` has the properties `Ricotta` and `FreshCheese`.



Southwest stew

Ingredients

- 2 tb Olive oil category:olive_oil (2, tblsp, ?, ?, ?)
- 1/2 c Chopped onion category:onion (1/2, c, chopped, ?, ?)
- 4 c Water category:water (4, c, ?, ?, ?)
- 1 c Chopped carrots category:carrot (1, c, chopped, ?, ?)
- 4 Cloves garlic; crushed category:garlic (4, ?, crushed, ?, ?)
- 1/4 ts Ground black pepper category:black_pepper (1/4, tsp, ground, ?, ?)
- 2 c Shredded cabbage category:cabbage (2, c, shredded, ?, ?)
- 1 c Small white beans; dried category:navy_bean (1, c, dried, ?, ?)
- 2 Jalapeno peppers; diced category:jalapeno_pepper (2, ?, diced, ?, ?)
- 1 ts Kosher salt; ground category:kosher_salt (1, tsp, ground, ?, ?)
- 10 oz Spinach; fresh category:spinach (10, oz, fresh, ?, ?)
- 4 Plum tomatoes; ripe category:sauce_tomato (4, ?, ripe, ?, ?)
- 8 oz Fresh mushrooms category:mushroom (8, oz, fresh, ?, ?)
- 2 oz Queso fresco; crumbled category:queso_fresco (2, oz, crumbled, ?, ?)

Preparation

- Or feta cheese (about 1/4 cup) Place beans in a strainer, rinse under running water removing any debris. In a medium saucepan place beans and water; bring to a boil; reduce heat and simmer, covered, for 1 hour. Stir in cabbage, carrots, onion, chilies, garlic, salt and pepper. Continue to simmer, covered, until the beans are tender and the broth is slightly thickened, 45 minutes to an hour, adding more water, if necessary, to keep the beans covered with liquid. Wash the spinach and remove the thick stems. Tear or cut leaves into small pieces; set aside. Cut tomatoes in halves lengthwise. Remove stems from mushrooms. Lightly brush tomatoes and mushrooms with olive oil. Arrange on a broiler pan. Broil under high heat until the tomatoes and mushrooms are lightly browned, about 4 minutes. Or, the tomatoes and mushrooms may be arranged on skewers and broiled or charcoal grilled. Just before serving, stir spinach into the white beans; cover and cook for 3 to 4 minutes. Spoon the white bean mixture onto individual serving plates; arrange broiled tomatoes and mushrooms over the beans. Drizzle lightly with remaining olive oil and sprinkle with queso fresco (cheese). 4 servings.

Fig. 4. Example of a recipe. The recipe contains a textual list of ingredients and preparation steps. Each ingredient line is parsed in order to extract structured data: quantity, unit, ingredient and additional pieces of information such as an action already applied or some precision about the ingredient (e.g. *fresh*).

– A binary context describing ingredients through the culinary actions that can be performed on them, e.g. ricotta and mascarpone can be beaten but mozzarella cannot, though mozzarella can be sliced, which is not the case for creamy cheeses.

Ingredient properties are extracted from 73795 recipes taken from Recipe Source.[3] Ingredient properties are extracted both from the ingredient list, which includes some cooking actions, and from the textual preparation, from which the actions applied to the ingredients are extracted using NLP tools [8]. The latter approach uses both classical morpho-syntactic analysis methods and a discourse representation specifically developed for procedural texts (instruction texts, including recipes). The discourse representation is well adapted for solving some complex anaphoric phenomena appearing in those types of texts. An example of a recipe used for property extraction is given in Fig. 4.

Extracting the properties of the ingredient list is done through an automatic annotation process, because the recipes are initially in a textual form. This annotation process was designed in order to obtain a formal representation of the recipes that can be handled by the CBR engine. The annotation process uses a terminological base for indexing the recipes by classes of the ingredient hierarchy. The quantities, the units, and the ingredient modifiers such as *chopped*, *diced*, *crumbled*, etc. (cf. Fig. 4) are also extracted. These modifiers are mostly cooking actions applied to the ingredients. Pairs (ingredient, modifier) and pairs (ingredient,applied action) extracted from the textual preparation are used for building the binary context. Some examples can be seen in the annotated recipe example given in Fig. 4. The set of properties M_R of $\mathcal{K}_R$ contains the cooking actions required for refining the ingredients of G_R. For example, let $R =$ $\texttt{FreshCheese}$ and $M_R = \{\texttt{mashAble}, \texttt{meltAble}, \texttt{beatAble}, \texttt{mixAble}, \texttt{sliceAble}, \texttt{cutAble}, \texttt{crumbleAble}\}$. Then, table 2 presents the binary context $\mathcal{K}_{\texttt{FreshCheese}}$.

[3] http://www.recipesource.com/

Table 2. A binary context for cheeses from the `FreshCheese` category, described by generic/specific properties and cooking actions that can be applied to the cheeses

	FreshCheese	CottageCheese	FatFreeCottageCheese	CreamCheese	FatFreeCreamCheese	FromageFrais	LowFatFromageFrais	Mozzarella	Mascarpone	QuesoFresco	Ricotta	mashAble	meltAble	beatAble	mixAble	sliceAble	crumbleAble	cutAble
CottageCheese	×	×										×	×					
FatFreeCottageCheese	×	×	×									×	×					
CreamCheese	×			×								×	×	×	×			
FatFreeCreamCheese	×			×	×							×	×	×	×			
FromageFrais	×					×							×	×	×			
LowFatFromageFrais	×					×	×						×	×	×			
Mozzarella	×							×								×		×
Mascarpone	×								×				×	×	×			
QuesoFresco	×									×							×	×
Ricotta	×										×		×	×	×			

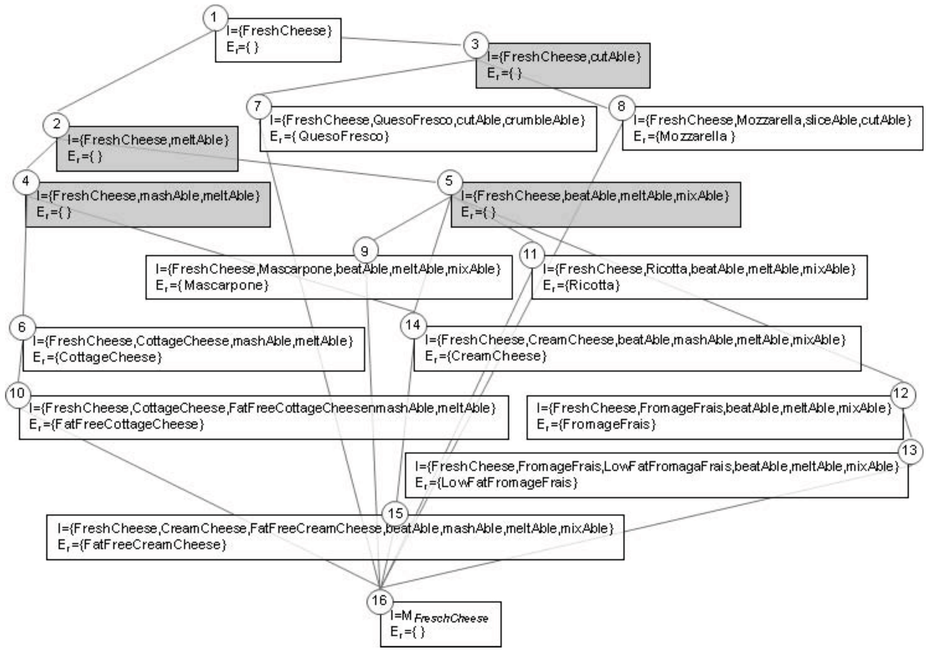

1 I={FreshCheese} E_r={ }
3 I={FreshCheese,cutAble} E_r={ }
7 I={FreshCheese,QuesoFresco,cutAble,crumbleAble} E_r={ QuesoFresco}
8 I={FreshCheese,Mozzarella,sliceAble,cutAble} E_r={Mozzarella }
2 I={FreshCheese,meltAble} E_r={ }
4 I={FreshCheese,mashAble,meltAble} E_r={ }
5 I={FreshCheese,beatAble,meltAble,mixAble} E_r={ }
9 I={FreshCheese,Mascarpone,beatAble,meltAble,mixAble} E_r={ Mascarpone}
11 I={FreshCheese,Ricotta,beatAble,meltAble,mixAble} E_r={Ricotta}
6 I={FreshCheese,CottageCheese,mashAble,meltAble} E_r={CottageCheese}
14 I={FreshCheese,CreamCheese,beatAble,mashAble,meltAble,mixAble} E_r={CreamCheese}
10 I={FreshCheese,CottageCheese,FatFreeCottageCheesenmashAble,meltAble} E_r={FatFreeCottageCheese}
12 I={FreshCheese,FromageFrais,beatAble,meltAble,mixAble} E_r={FromageFrais}
13 I={FreshCheese,FromageFrais,LowFatFromagaFrais,beatAble,meltAble,mixAble} E_r={LowFatFromageFrais}
15 I={FreshCheese,CreamCheese,FatFreeCreamCheese,beatAble,mashAble,meltAble,mixAble} E_r={FatFreeCreamCheese}
16 I=M_FreschCheese E_r={ }

Fig. 5. Concept lattice restructuring the fresh cheeses

4.4 Merging the Local Lattice and the Initial Hierarchy

Given a class R of the initial hierarchy and the lattice $\mathcal{L}_R$ built from the context $\mathcal{K}_R$, the initial ontology is refined by adding ontology classes corresponding to some of the concepts of $\mathcal{L}_R$. The choice of these concepts in $\mathcal{L}_R$ is based on the notion of *reduced extent* of a concept C of $\mathcal{L}_R$. Following the terminology of object-oriented languages, the reduced extent $E_r(C)$ of C is the set of *direct* instances of C. In other words, an object x belongs to the reduced extent of C if it belongs to the extent of C but does not belong to the extent of any direct descendant D of C in the lattice:

$$E_r(C) = E(C) \setminus \bigcup\{E(D) \mid D\text{: direct descendant of } C \text{ in the lattice}\}$$

It can be proven that $x \in E_r(C)$ iff the properties of C are exactly the ones of $I(C)$, the intent of C (whereas an $x \in E(C)$ may have additional properties, outside of $I(C)$). Moreover, for a context $\mathcal{K}_R$ built as explained in the previous section, for each concept C of $\mathcal{L}_R$, either $E_r(C) = \emptyset$ or $E_r(C)$ contains exactly one element.

Fig. 5 is the lattice $\mathcal{L}_{\texttt{FreshCheese}}$ (with the reduced extents instead of the extents). The grey concepts C are the ones that are chosen in order to be added to the initial ontology. The concept with an empty extent (concept 16 in Fig. 5) covers no instance and thus is not added to the ontology. Some concepts correspond to classes that are already in the ontology and need not be added. This is the case for the root concept, (in the example, `FreshCheese`, concept 1), and for the concepts C having exactly one element in their reduced extent: $E_r(C) = \{P\}$ (concepts 6 to 15). Such a FCA concept corresponds to the class P of the initial ontology. For instance, if C is the class P of the ontology, $E_r(C) = \{\texttt{Mozzarella}\}$ and the concept C corresponds to the ontology class `Mozzarella`.

The remaining FCA concepts are structuring concepts that are added to the ontology (concepts 2 to 5, in grey in the Fig. 5). Such a node C has an empty reduced extent, thus its extent equals the union of the extents of its direct descendants in $\mathcal{L}_R$. Its intent factorises properties of its subconcepts. Adding such a concept to the ontology, means reifying a new ontology class with a label obtained by concatenation of the intent properties. For example, the concept 5 leads to the class in the ontology whose label is `FreshCheese_beatAble_meltable_mixAble`. This class refers to the fresh cheeses that can be beaten, melted, and mixed. The part of the refined ontology regarding fresh cheeses is given on the right hand-side of Fig. 6.

5 Effect of the Ontology Refinement on the CBR Process: An Example

This section shows how the ontology refinement affects the CBR system. An example is given of TAAABLE's performance *before* and *after* the refinement in order to show how retrieval is affected. The two TAAABLE systems use the same case base: the recipe book provided by the *Computer Cooking Contest,* containing 1488 recipes.

In order to evaluate the effect of the refinement, two TAAABLE systems were instantiated: TAAABLE$_0$ using the initial ontology and TAAABLE$_1$ using the refined ontology. The two systems were given the same query to answer. Fig. 7 shows the answer of

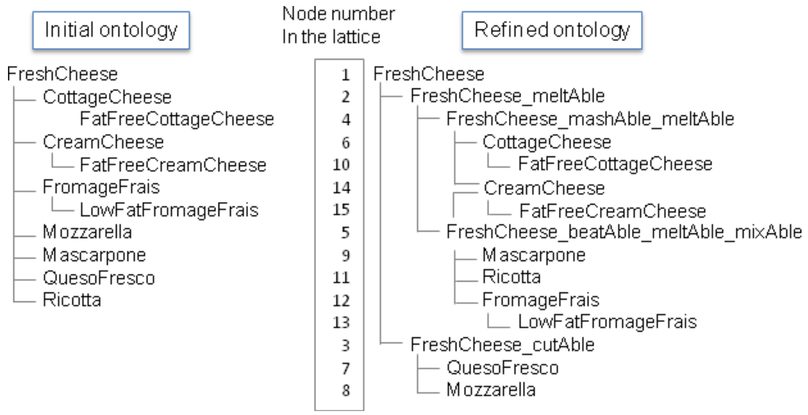
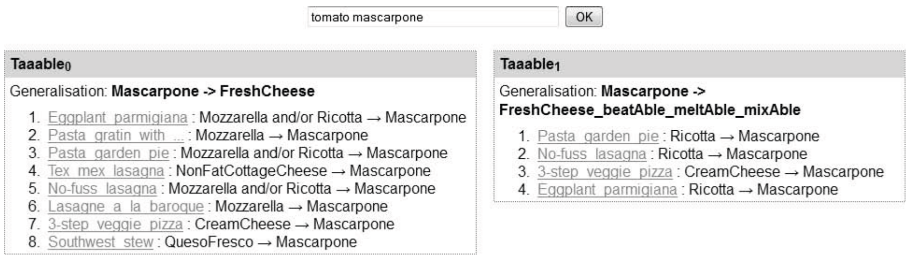


Fig. 6. Initial (left) and refined (right) ontology from the `FreshCheese` class down


tomato mascarpone OK

Taaable$_0$

Generalisation: **Mascarpone -> FreshCheese**

1. Eggplant parmigiana : Mozzarella and/or Ricotta → Mascarpone
2. Pasta gratin with ... : Mozzarella → Mascarpone
3. Pasta garden pie : Mozzarella and/or Ricotta → Mascarpone
4. Tex mex lasagna : NonFatCottageCheese → Mascarpone
5. No-fuss lasagna : Mozzarella and/or Ricotta → Mascarpone
6. Lasagne a la baroque : Mozzarella → Mascarpone
7. 3-step veggie pizza : CreamCheese → Mascarpone
8. Southwest stew : QuesoFresco → Mascarpone

Taaable$_1$

Generalisation: **Mascarpone -> FreshCheese_beatAble_meltAble_mixAble**

1. Pasta garden pie : Ricotta → Mascarpone
2. No-fuss lasagna : Ricotta → Mascarpone
3. 3-step veggie pizza : CreamCheese → Mascarpone
4. Eggplant parmigiana : Ricotta → Mascarpone

Fig. 7. TAAABLE$_0$ and TAAABLE$_1$'s answers to the query $\exists\texttt{ing.Tomato} \sqcap \exists\texttt{ing.Mascarpone}$

the two systems to a query asking for a recipe with `tomato` and `mascarpone`. Since no recipe contains both those ingredient in the case base, the two systems search for similar cases.

For TAAABLE$_0$, the retrieval stops after the generalisation `Mascarpone` $\rightsquigarrow$ `FreshCheese`. At this step, 8 recipes satisfying the query $\exists\texttt{ing.Tomato} \sqcap \exists\texttt{ing.FreshCheese}$ are found and adapted by substituting `Mascarpone` for `FreshCheese`. Some of the recipes contain several `FreshCheeses`. In this case, the system proposes to substitute one or more ingredients (this is the case for the recipes numbered 1, 3 and 5). For TAAABLE$_1$, the retrieval is more fine-grained: the generalisation stops on the `FreshCheese_beatAble_meltable_mixAble` class, which is more specific than `FreshCheese`. This class represents the fresh cheeses that can be beaten, melted, and mixed, actions that are applicable to mascarpone.

The analysis of the results presented in Fig. 7 shows that:

– TAAABLE$_1$ returns only 4 answers, while TAAABLE$_0$ returns 8 answers (including the 4 answers of TAAABLE$_1$ with different substitutions). The generalisation is less sharp in the case of the refined ontology than with the initial one.

Table 3. List of actions applied to the substituted ingredients of recipes returned by TAABLE$_0$ with tomato and mascarpone. A cross in the failure column indicates adaptation failure because an action cannot be applied upon the substitution ingredient.

	Substituted ingredient	Applied actions	Failure
Eggplant parmigiana	Ricotta	add	
	Mozzarella	cut	×
Pasta gratin with etc.	Mozzarella	grate	×
Pasta garden pie	Mozzarella	shred, sprinkle	×
	Ricotta	stir	
Tex mex lasagna	NonFatCottageCheese	mix	
No-fuss lasagna	Mozzarella	shred	×
	Ricotta	mix	
Lasagne a la baroque	Mozzarella	slice, add	×
3-step veggie pizza	CreamCheese	sprinkle	
Southwest stew	QuesoFresco	crumble	×

– The substitutions proposed by TAABLE$_0$ are more variable than the ones proposed by TAABLE$_1$. With TAABLE$_1$, mascarpone is substituted for ricotta or cream cheese, while TAABLE$_0$ additionally proposes to substitute mascarpone for mozzarella, non-fat cottage cheese or *queso fresco*. And the larger the set of substituting ingredients, the more likely the adaptation is to fail, because there is less similarity between the ingredients.

Examining the proposed substitutions recipe by recipe allows to evaluate the adaptation success or failure. If the adaptation fails, the reason for failure is identified. Table 3 gives, for each recipes of TAABLE$_0$, the actions that were applied to the substituted ingredient. If the action cannot be applied to the substituting ingredient — in our case, the mascarpone — the adaptation fails. These failures are marked with a cross in the *Failure* column. There are 3 clear cases of failure, for the following recipes: *Pasta gratin with etc.* in which the mascarpone should be grated; *Lasagne a la baroque* in which the mascarpone should be sliced; and *Southwest stew* in which the mascarpone should be crumbled. There are also three *half*-failures: answers in which TAABLE$_0$ proposes to choose the ingredient that has to be substituted. So, adapting the *Pasta garden* recipe may be considered as a success if the mascarpone replaces the ricotta, but as a failure if the mascarpone replaces the mozzarella and that it must be shredded. It is the same for the *No-fuss lasagna*, where substituting ricotta succeeds while substituting mozzarella fails, and for the *Eggplant parmigiana*, where substituting ricotta succeeds while substituting mozzarella fails. 6 of the 11 substitutions fail with TAABLE$_0$ on this example.

Table 4 gives, for each recipe of TAABLE$_1$, the actions that must be applied to the substituted ingredient, a failure being marked with a cross. The refined ontology improves the final result, as none of the failed adaptations proposed by TAABLE$_0$ are proposed by TAABLE$_1$: there is no case of either clear or half-failure. However, one recipe of TAABLE$_0$ *(Tex-mex lasagna)* for which the adaptation succeeds is not proposed by

Table 4. List of actions applied to the substituted ingredients of recipes returned by TAAABLE$_1$ with tomato and mascarpone. A cross in the failure column indicates adaptation failure because an action cannot be applied upon the substitution ingredient. For TAAABLE$_1$, there is in fact no such failure

	Substituted ingredient	Applied actions	Failure
Pasta garden pie	Ricotta	mix	
No-fuss lasagna	Ricotta	mix	
3-step veggie pizza	CreamCheese	sprinkle	
Eggplant parmigiana	Ricotta	add	

TAAABLE$_1$ because of the increased distance between `NonFatCottageCheese` and `Mascarpone` in TAAABLE$_1$'s ontology that results from the stop condition of the algorithm of TAAABLE running through the generalisation space. This problem can be solved by asking, in the system interface, to keep searching similar recipes beyond the current generalisation, at the risk of reintroducing failed adaptations.

6 Conclusion

This paper shows how a domain ontology refinement immediately impacts the retrieval process of a CBR system and how it improves the final result of TAAABLE. Our approach enables a more progressive retrieval of source cases and, within the framework of TAAABLE, eliminates some adaptation failures. Beyond the results of our approach, illustrated by a concrete example, this work raises some issues that must be addressed.

A first issue is the supervision of the set of properties taken into account for the refinement. In this work, this set has been limited manually because there are potentially more than 250 cooking actions. Selecting a priori some properties (cooking actions) that discriminate the ingredients limits unwanted aggregations. However, as the use context of an ingredient may change, it is not impossible, that given a set of actions A_1, an ingredient i be closer to an ingredient i_1 and, given another set of actions $A_2 \neq A_1$, i be closer to $i_2 \neq i_1$. So, for one context, ricotta could be closer to mascarpone and, in another context, closer to cottage cheese.

Another issue is case indexation. Each (recipe) case is actually indexed by its ingredients, regardless of how they are used. Improving the case representation by indexing them by the ingredients jointly with their transformations is another way to eliminate some of the adaptation failures of this type.

Finally, another point planned in order to improve the adaptation in TAAABLE is to manage action sequences and guarantee that the actions are applicable after the adaptation, by using pre-conditions based on domain knowledge: ingredient consistency (*solid*, *liquid*, *soft*...), ingredient sensory properties (*sweet*, *salted*...), etc. This knowledge makes it possible to check whether, after the adaptation, the recipe is *cookable*.

Acknowledgements. The authors wish to thank the reviewers for their helpful comments and suggestions for future work.

References

1. Asiimwe, S., Craw, S., Wiratunga, N., Taylor, B.: Automatically Acquiring Structured Case Representations: The SMART Way. In: Applications and Innovations in Intelligent Systems XV. Springer, Heidelberg (2007)
2. Badra, F., Bendaoud, R., Bentebitel, R., Champin, P.-A., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J., Meilender, T., Mille, A., Nauer, E., Napoli, A., Toussaint, Y.: Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In: ECCBR Workshops, Workshop of the First Computer Cooking Contest, pp. 219–228 (2008)
3. Badra, F., Cojan, J., Cordier, A., Lieber, J., Meilender, T., Mille, A., Molli, P., Nauer, E., Napoli, A., Skaf-Molli, H., Toussaint, Y.: Knowledge Acquisition and Discovery for the Textual Case-Based Cooking system WikiTaaable. In: Workshops of the 8th International Conference on Case-Based Reasoning (2009)
4. Bendaoud, R., Napoli, A., Toussaint, Y.: Formal Concept Analysis: A unified framework for building and refining ontologies. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 156–171. Springer, Heidelberg (2008)
5. Bergmann, R., Stahl, A.: Similarity measures for object-oriented case representations. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 25–36. Springer, Heidelberg (1998)
6. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. In: Journal of Artificial Intelligence Research (JAIR 2005), vol. 24, pp. 305–339. AAAI Press, Menlo Park (2005)
7. Díaz-Agudo, B., González-Calero, P.A.: Classification Based Retrieval Using Formal Concept Analysis. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 173–188. Springer, Heidelberg (2001)
8. Dufour-Lussier, V., Lieber, J., Nauer, E., Toussaint, Y.: Text adaptation using formal concept analysis. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 96–110. Springer, Heidelberg (2010)
9. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
10. Stumme, G., Maedche, A.: FCA-MERGE: Bottom-Up Merging of Ontologies. In: 17th International Joint Conferences on Artificial Intelligence (IJCAI 2001), pp. 225–234. Morgan Kaufmann Publishers, Inc., San Francisco (2001)
11. Szathmary, L., Napoli, A.: CORON: A Framework for Levelwise Itemset Mining Algorithms. In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 110–113. Springer, Heidelberg (2005)

Preference-Based CBR: First Steps toward a Methodological Framework

Eyke Hüllermeier and Patrice Schlegel

Department of Mathematics and Computer Science,
Marburg University, Germany
{eyke,pschlegel}@mathematik.uni-marburg.de

Abstract. Building on recent research on preference handling in artificial intelligence and related fields, our general goal is to develop a coherent and universally applicable methodological framework for CBR on the basis of formal concepts and methods for knowledge representation and reasoning with preferences. A preference-based approach to CBR appears to be appealing for several reasons, notably because case-based experiences naturally lend themselves to representations in terms of preference relations, even when not dealing with preference information in a literal sense. Moreover, the flexibility and expressiveness of a preference-based formalism well accommodate the uncertain and approximate nature of case-based problem solving. In this paper, we make a first step toward a preference-based formalization of CBR. Apart from providing a general outline of the framework as a whole, we specifically address the step of case-based inference. The latter consists of inferring preferences for candidate solutions in the context of a new problem, given such preferences in similar situations. Our case-based approach to predicting preference models is concretely realized for a scenario in which solutions are represented in the form of subsets of a reference set. First experimental results are presented to demonstrate the effectiveness of this approach.

1 Introduction

Despite its great practical success, work on the theoretical foundations of CBR is still under way, and a coherent and universally applicable methodological framework is yet missing. The *CBR cycle* proposed by Aamodt and Plaza [1] is a commonly accepted process model, which nicely illustrates the main aspects of the case-based problem solving paradigm. Likewise, the metaphor of *knowledge containers*, introduced by Richter [2], provides a general framework for the structuring of knowledge in CBR. However, both are high-levels models and still rather far from the conceptual realization and implementation of a case-based problem solver. On the other extreme, many CBR systems have been designed for solving concrete problems. These, however, are mostly tailored for a specific purpose and not easily applicable to a wider range of problems.

In-between these two extremes, there is arguably space for developing CBR *methodologies* [3]. On the one hand, a CBR methodology should be sufficiently general and abstract, so as to allow for the development of generic algorithms, for analyzing formal properties, proving theorems, etc. On the other hand, it should also be sufficiently concrete, so as to support the development of specific applications. To make this idea more tangible, consider as an analogy the formalism of graphical models, by now an established methodology for the design of probabilistic expert systems [4]. This class of models disposes of a formal theory and generic algorithms, but also tools for supporting the design of models for concrete applications.

In [5], a general *constraint-based framework* of CBR has been developed. Roughly speaking, the core assumption of CBR, suggesting that similar problems have similar solutions, is formally interpreted as a constraint: given a new query problem to be solved in conjunction with previous experience in the form of a solution to a similar problem, it restricts the set of candidate solutions of the query. Starting with the classical view of a constraint as a subset of feasible elements of a relation, more flexible variants of this approach have later been developed on the basis of different frameworks of approximate reasoning and reasoning under uncertainty. Taking such models as a point of departure, case-based reasoning can be realized, respectively, as constraint-based, probabilistic, or fuzzy logic-based reasoning in a formally sound way.

In this paper, we make a first step toward an alternative methodological framework for case-based reasoning on the basis of formal concepts and methods for knowledge representation and problem solving with *preferences*. The topic of preferences has recently attracted considerable attention in artificial intelligence (AI) research and plays an increasingly important role in several AI-related fields, including, e.g., agents, constraint satisfaction, decision theory, planning, machine learning, and argumentation [6,7,8]. Preference-based methods are especially appealing from an AI perspective, notably as they allow one to specify desires in a declarative way, to combine qualitative and quantitative modes of reasoning and to deal with inconsistencies and exceptions in a quite flexible manner. Indeed, a preference can be considered as a relaxed constraint, which, if necessary, can be violated to some degree.

The important advantage of an increased flexibility of a preference-based problem solving paradigm is nicely explained in [9]: “Early work in AI focused on the notion of a goal—an explicit target that must be achieved—and this paradigm is still dominant in AI problem solving. But as application domains become more complex and realistic, it is apparent that the dichotomic notion of a goal, while adequate for certain puzzles, is too crude in general. The problem is that in many contemporary application domains ... the user has little knowledge about the set of possible solutions or feasible items, and what she typically seeks is the best that’s out there. But since the user does not know what is the best achievable plan or the best available document or product, she typically cannot characterize it or its properties specifically. As a result, she will end up either asking for an unachievable goal, getting no solution in response, or asking for too little, obtaining a solution that can be substantially improved.”

Our claim is that the above insights do not only apply to AI in general but to CBR in particular. In fact, as will be argued in more detail below, case-based experience can be modeled in terms of preference information in a quite convenient way and, moreover, case-based inference can be realized quite elegantly in the form of preference processing. As pointed out in the above quotation, a key advantage in comparison to a constraint-based approach, as developed in [5], is an increased flexibility and expressiveness, which appears to be especially advantageous for CBR. To some extent, these points were already put forward in [10], albeit within a much more narrow scope.

The remainder of the paper is organized as follows: In the next section, the main ideas of our approach to preference-based CBR are outlined in an informal way. A formal model of a core part of preference-based CBR, namely the inference step responsible for predicting a "contextualized" preference relation on the solution space, is then introduced in Section 3. Section 4 is devoted to an experimental study, in which we seek to demonstrate the general feasibility of case-based learning on the basis of preference information. The paper ends with some concluding remarks and an outlook on future work in Section 5.

2 Main Ideas and Basic Setting

Even though several generalizations have been proposed in recent years, experience in CBR is most commonly represented in the form of problem/solution tuples $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{X} \times \mathbf{Y}$, where $\boldsymbol{x}$ is an element from a problem space $\mathbf{X}$, and $\boldsymbol{y}$ an element from a solution space $\mathbf{Y}$. Despite its apparent simplicity, this representation is quite expressive, especially since $\mathbf{X}$ and $\mathbf{Y}$ can be arbitrarily complex spaces. Yet, upon closer examination, it also exhibits some disadvantages and principle limitations, both from a knowledge acquisition and reuse point of view.

- *Existence of correct solutions*: First, the representation assumes the existence of a "correct" solution for each problem, and implicitly even its uniqueness. In many application domains, this assumption is not tenable. Take the cooking domain as an example: If the problem is to prepare a vegetarian pasta meal for two persons, there is definitely not a single "correct" recipe. Instead, there will be many possible alternatives, maybe more or less preferred by the user.
- *Verification of optimality*: Even if the existence of a single correct solution for each problem could be assured, it will generally be impossible to verify the optimality of the solution that has been produced by a CBR system. Consequently, a solution $\boldsymbol{y}$ may only be a suboptimal solution to a problem $\boldsymbol{x}$, so that storing and later on reusing the case $(\boldsymbol{x}, \boldsymbol{y})$ can be misleading. This problem is less critical, though does not dissolve, if only "acceptable" instead of optimal solutions are required.
- *Loss of information*: Storing only a single solution $\boldsymbol{y}$ for a problem $\boldsymbol{x}$, even if it can be guaranteed to be optimal, may come along with a potential loss of information. In fact, during a problem solving episode, one typically tries or at least compares several candidate solutions. Retaining a single one then

captures the information that it was the best candidate. The potentially useful piece of experience that, for example, a second candidate $\boldsymbol{y}'$, despite being worse than $\boldsymbol{y}$, was still better than a third alternative $\boldsymbol{y}''$ is lost, however. On the other hand, if a problem solving process was canceled before a provably optimal solution could be found, there is in principle no case to be stored in the case base. Again, however, retaining the piece of knowledge that a candidate $\boldsymbol{y}$ was not acceptable, or that a candidate $\boldsymbol{y}'$, even if suboptimal, was at least better than $\boldsymbol{y}''$, could be useful.

– *Limited guidance*: From a reuse point of view, a retrieved case $(\boldsymbol{x}, \boldsymbol{y})$ only suggests a single solution, namely $\boldsymbol{y}$, for a query problem $\boldsymbol{x}_0$. Thus, it does not imply a possible course of action in the case where the suggestion fails: If $\boldsymbol{y}$ is not a good point of departure, for example since it cannot be adapted to solve $\boldsymbol{x}_0$, there is no concrete recommendation on how to continue.

2.1 Preference-Based Knowledge Representation

To avoid these problems, we propose a *preference-based* approach to representing and processing experiences in CBR. The basic idea is to replace experiences of the form "solution $\boldsymbol{y}$ (optimally) solves problem $\boldsymbol{x}$" by information of the form "$\boldsymbol{y}$ is better (more preferred) than $\boldsymbol{y}'$ as a solution for $\boldsymbol{x}$". More specifically, the basic "chunk of information" we consider is symbolized in the form $\boldsymbol{y} \succeq_{\boldsymbol{x}} \boldsymbol{y}'$ and suggests that, for the problem $\boldsymbol{x}$, the solution $\boldsymbol{y}$ is at least as good as the solution $\boldsymbol{y}'$. This type of knowledge representation obviously overcomes the problems discussed above. As soon as two candidate solutions $\boldsymbol{y}$ and $\boldsymbol{y}'$ have been tried as solutions for a problem $\boldsymbol{x}$, these two alternatives can be compared and, correspondingly, a strict preference in favor of one of them or an indifference can be expressed (recall that, from a weak preference relation $\succeq$, a strict preference $\succ$ and an indifference $\sim$ are derived as follows: $\boldsymbol{y} \succ \boldsymbol{y}'$ iff $\boldsymbol{y} \succeq \boldsymbol{y}'$ and $\boldsymbol{y}' \not\succeq \boldsymbol{y}$, and $\boldsymbol{y} \sim \boldsymbol{y}'$ iff $\boldsymbol{y} \succeq \boldsymbol{y}'$ and $\boldsymbol{y}' \succeq \boldsymbol{y}$). To this end, it is by no means required that one of these solutions is optimal. It is worth mentioning, however, that knowledge about the optimality of a solution $\boldsymbol{y}^*$, if available, can be handled, too, as it simply means that $\boldsymbol{y}^* \succ \boldsymbol{y}$ for all $\boldsymbol{y} \neq \boldsymbol{y}^*$. In this sense, the conventional CBR setting can be considered as a special case of preference-based CBR.

The above idea of a preference-based approach to knowledge representation in CBR also suggests a natural extension of the case retrieval and inference steps, that is, the recommendation of solutions for a new query problem: Instead of just proposing a single solution, it would be desirable to predict a *ranking* of several (or even all) candidate solutions, ordered by their (estimated) degree of preference:

$$\boldsymbol{y}_1 \succeq_{\boldsymbol{x}} \boldsymbol{y}_2 \succeq_{\boldsymbol{x}} \boldsymbol{y}_3 \succeq_{\boldsymbol{x}} \cdots \succeq_{\boldsymbol{x}} \boldsymbol{y}_n \tag{1}$$

This is indeed comparable to an information retrieval scenario, such as web search, where normally not only a single solution is shown to the user, but instead a complete list of potential matches. Thus, the last problem mentioned above, namely the lack of guidance in the case of a failure, can be overcome.

As a side remark, we note that a kind of ranking of solutions can in principle also be obtained in the classical approach to CBR, namely by ordering the

Table 1. Exemplary preferences of different persons regarding four coffee drinks.

Anne:	Cappuccino	$\succ$	Espresso	$\succ$	Latte	$\succ$	Americano
Lisa:	Cappuccino	$\succ$	Latte	$\succ$	Espresso	$\succ$	Americano
Peter:	Americano	$\succ$	Latte	$\succ$	Espresso	$\succ$	Cappuccino
Paul:	Latte	$\succ$	Americano	$\succ$	Espresso	$\succ$	Cappuccino

solutions associated with the k cases in the case base which are most similar to the query problem. However, apart from lacking a formal foundation, there is a very important difference to our approach. In fact, our fundamental assumption is that, with each problem, one can (at least theoretically) associate a preference order over the set of potential solutions, instead of just a single (correct) solution. Needless to say, this order will normally not coincide with the ordering obtained by sorting the presumably best solutions (top-choices) for similar problems.

As an illustration, consider the simple example in Table 1, where four persons are listed in decreasing order of their similarity to the query person, say, Mary. For each person, the preferences regarding four types of coffee drinks are given in terms of a total order. To predict Mary's preferences on the four alternatives, one could, for example, simply adopt the preference relation of the most similar person, which is Anne, or aggregate the preference relations of all persons (e.g., by a simple Borda count, which yields Latte $\succ$ Cappuccino $\succ$ Espresso $\sim$ Americano). In any case, the result will be different from the order obtained by sorting the top-choices of the four people (Cappuccino, Cappuccino, Americano, Latte). In fact, the example also shows that this approach will normally not even lead to a proper ranking: While some alternatives may never occur, since they are never ranked first (like Espresso), others may occur multiple times (Cappuccino appears on the first two positions). In any case, the example makes clear that sorting solutions in the classical CBR setting does in general not yield a proper preference relation (ranking) on the set of all candidate solutions.

2.2 Important Issues in Preference-Based CBR

The approach outlined so far is overly simplistic and needs to be refined in several respects. Important problems to be addressed include the following:

– How to represent, organize and maintain case-based experiences, given in the form of preferences referring to a specific context, in an efficient and effective way?
– How to select and access the experiences which are most relevant in a new problem solving situation?
– How to combine these experiences and exploit them to infer a solution or, more generally, a preference order on a set of candidate solutions, for the problem at hand?

Regarding the notion of a "problem", we like to mention that the problem description in CBR can be quite general. In particular, in addition to properties of the actual problem itself, it may contain further information, for example about a user, so that the term "context" would perhaps be even more appropriate. From a preference point of view, this is very important, since different users, e.g., members of a web community, may have different preferences. For instance, it makes a great difference whether a culinary preference is expressed by a vegetarian or by a non-vegetarian. In general, we assume the problem to be specified by a finite number of attributes. The domain of an attribute can simply be an unordered or totally ordered set (e.g., categorical or numeric attribute), but can also have a hierarchical structure. In the cooking domain, for example, attribute values are often organized in the form of taxonomies (allowing for the specification of values at different levels of abstraction).

2.3 Structure of the Solution Space

A solution in CBR can be as simple as a single value (like in CBR for classification and regression tasks [11,12]) but may also appear as a complex object assembled from a number of basic components. Needless to say, the concrete structure of the solution space will be important from a methodological point of view, because different types of problem solving will call for different methods. The following (non-exhaustive) list of problem types and related solution spaces may be envisioned in increasing order of complexity.

- A solution space $\mathbf{Y}$ is specified by the conventional attribute-value representation, i.e., the description of a solution in terms of a fixed number of attribute values (numerical, categorical, etc.). This type of representation is natural, commonly used, and even relevant for structured representations, which can often be mapped to flat feature vectors in a reasonable way. For example, if the nutritional value of a dish is of main interest, a recipe can be mapped to an amount of protein, vitamins, etc.
- Spaces of the form $\mathbf{Y} = 2^C$, where C is a finite set of labels. Thus, a solution is a subset of C. Despite its simplicity, this specific structure is quite general and indeed relevant for many applications. For example, in its most basic form, a cooking recipe is simply represented by a set of ingredients, that is, by a subset of the set C of all potential ingredients (perhaps on different levels of abstraction as specified by an underlying taxonomy).
- A combination of the two previous scenarios is of the form $\mathbf{Y} = 2^D$, where D is a set of objects characterized in terms of an attribute-value representation. For example, instead of just knowing the name of an ingredient, it is now possible to capture some of its properties. Thus, each solution is a subset of objects, where each object is in turn a feature vector. This type of representation of alternatives has recently been advocated in [13], especially from a preference handling point of view.
- Another generalization of the first scenario is a solution space $\mathbf{Y}$ based on a representation in terms of a feature vector with set-valued attributes, i.e., in which each attribute can assume several values of an underlying domain simultaneously instead of only a single one. For example, a recipe could

be described, amongst others, by an attribute `SideDish` with underlying domain {`potatoes`, `rice`, ...}. Most recipes will include exactly one side dish, but dishes with no side dish or more than one do of course exist.

- Solution spaces $\mathbf{Y}$ in the form of a specific class of *graphs*, another generic data structure that can be used for modeling purposes in a rather flexible way. In particular, in addition to the components themselves, it allows one to capture relationships between them. This type of solution space has recently been considered in CBR in connection with workflows [14].

It is worth mentioning that the "construction" of a solution in the first four scenarios can in principle be reduced to solving a fixed number of "prediction" problems, namely assigning a value to each attribute (in the case of multi-valued attributes, prediction comes down to solving a multi-label classification problem [15]). However, it is also important to recognize that these problems are not independent of each other, due to interactions and interdependencies between the attributes, so this simplistic approach is likely to produce suboptimal results.

Finally, let use note that the actual output space on which a preference-based CBR system is operating, at least implicitly, is not given by a solution space $\mathbf{Y}$ itself, but instead by $\mathfrak{P}(\mathbf{Y})$, namely the class of all preference structures on $\mathbf{Y}$. These spaces may become extremely complex, and will therefore not be dealt with in an explicit way.

3 Case-Based Inference

Leaving questions of problem representation, case base organization, case retrieval, etc. (essentially raised by the first two items in Section 2.2) aside, our focus in this paper is on case-based inference (the third item). Given a new query problem $\boldsymbol{x}_0$, standard case-based inference starts by retrieving a subset of (presumably) most relevant cases from the case base, and then proceeds by combining, in one way or another, the solutions of these problems into a candidate solution for $\boldsymbol{x}_0$. The type of aggregation procedure which is applied to this end strongly depends on the structure and representation of solutions, and on the type of preference relation defined on the solution space. In this regard, recall the main scenarios (types of solution spaces) distinguished above.

It is important to recall that problems are not associated with single solutions but rather with preferences over solutions, that is, with elements from $\mathfrak{P}(\mathbf{Y})$. Consequently, we have to consider the problem of combining the preferences associated with the nearest neighbors of the query $\boldsymbol{x}_0$ into a preference relation on candidate solutions. Ideally, such a relation is given in the form of a total order (1), though depending on the completeness of the information at hand, this will of course not always be possible. Roughly, the problem can be stated as follows: Given a set of preferences on candidate solutions coming from a set of relevant problems (associated with corresponding similarity degrees), find a global preference relation on these candidates which is as consistent with these preferences as possible.

3.1 Case-Based Inference as Probability Estimation

To solve this problem in a theoretically sound way, we approach it as a statistical *estimation problem.* Recall that $\mathfrak{P}(\mathbf{Y})$ denotes the set of preference structures on the solution space $\mathbf{Y}$, for example the set of all total orders (rankings) of the solutions $\boldsymbol{y} \in \mathbf{Y}$. Since the true preference model $\mathbf{R}_{\boldsymbol{x}_0} \in \mathfrak{P}(\mathbf{Y})$ associated with the query $\boldsymbol{x}_0$ is not known, we consider it as a random variable Z with distribution $\mathbf{P}(\cdot \,|\, \boldsymbol{x}_0)$, where $\mathbf{P}(\cdot \,|\, \boldsymbol{x}_0)$ is a distribution $\mathbf{P}_\theta(\cdot)$ parametrized by $\theta = \theta(\boldsymbol{x}_0) \in \Theta$. Thus, $\mathbf{P}_\theta(\mathbf{R}_{\boldsymbol{x}_0})$ is the probability that $Z = \mathbf{R}_{\boldsymbol{x}_0}$. The problem, then, is to estimate this distribution or, equivalently, θ on the basis of the information available. This information consists of the preferences $\boldsymbol{y} \succ_{\boldsymbol{x}} \boldsymbol{y}'$ between solutions observed for the neighbors $\boldsymbol{x}$ of $\boldsymbol{x}_0$; let $\mathcal{D}$ denote the complete set of observed preferences collected from the nearest neighbors of $\boldsymbol{x}_0$.

The basic assumption underlying nearest neighbor estimation is that the conditional probability distribution of the output given the input is (approximately) locally constant, that is, $\mathbf{P}(\cdot \,|\, \boldsymbol{x}_0) \approx \mathbf{P}(\cdot \,|\, \boldsymbol{x})$ for $\boldsymbol{x}$ close to $\boldsymbol{x}_0$. This assumption justifies considering the preferences $\mathcal{D}$ observed for the neighbors of $\boldsymbol{x}_0$ as a representative sample of $\mathbf{P}_\theta(\cdot)$ and, hence, estimating θ via maximum likelihood (ML) by

$$\theta^{ML} = \arg\max_{\theta \in \Theta} \mathbf{P}_\theta(\mathcal{D}) \ . \tag{2}$$

An important prerequisite for putting this approach into practice is a suitable data generating process, i.e., a process generating preferences in a stochastic way. Moreover, efficient (and probably approximate) inference procedures are needed to estimate the parameters of this process.

3.2 A Discrete Choice Model

Our data generating process is based on the idea of a discrete choice model as used in choice and decision theory [16]. More specifically, we assume that the (absolute) preference for a solution $\boldsymbol{y} \in \mathbf{Y}$ depends on its distance $\Delta(\boldsymbol{y}, \boldsymbol{y}^*) \geq 0$ to an "ideal" solution $\boldsymbol{y}^*$. The distance measure Δ depends on the application and is supposed to model background knowledge regarding the suitability of solutions. Roughly speaking, $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$ can be seen as a "degree of suboptimality" of $\boldsymbol{y}$: The larger $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$, the less suitable is $\boldsymbol{y}$ as a substitute of the solution $\boldsymbol{y}^*$. For example, one may think of Δ as a kind of edit distance measuring the cost of an adaptation, i.e., the cost of transforming $\boldsymbol{y}$ into $\boldsymbol{y}^*$.

Suppose the latent *utility* of $\boldsymbol{y}$ is of the form

$$U(\boldsymbol{y}) = -\beta \, \Delta(\boldsymbol{y}, \boldsymbol{y}^*) + \epsilon \ ,$$

with $\beta \geq 0$ and an error term ϵ having an extreme value distribution. In conjunction with the assumption of independence between the error terms of different solutions, this leads to the *logit* model of discrete choice:

$$\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{y}') = \frac{1}{1 + \exp\left(-\beta(\Delta(\boldsymbol{y}', \boldsymbol{y}^*) - \Delta(\boldsymbol{y}, \boldsymbol{y}^*)\right)} \tag{3}$$

Thus, the probability of observing the (revealed) preference $\boldsymbol{y} \succ \boldsymbol{y}'$ depends on the degree of suboptimality of $\boldsymbol{y}$ and $\boldsymbol{y}'$, namely their respective distances to the ideal solution, $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$ and $\Delta(\boldsymbol{y}', \boldsymbol{y}^*)$: The larger the difference $\Delta(\boldsymbol{y}', \boldsymbol{y}^*) - \Delta(\boldsymbol{y}, \boldsymbol{y}^*)$, i.e., the less optimal $\boldsymbol{y}'$ in comparison to $\boldsymbol{y}$, the larger the probability to observe $\boldsymbol{y} \succ \boldsymbol{y}'$; if $\Delta(\boldsymbol{y}', \boldsymbol{y}^*) = \Delta(\boldsymbol{y}, \boldsymbol{y}^*)$, then $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{y}') = 1/2$.

The coefficient β can be seen as a measure of precision of the agent's decisions. For large β, the probability (3) converges toward 0 if $\Delta(\boldsymbol{y}', \boldsymbol{y}^*) < \Delta(\boldsymbol{y}, \boldsymbol{y}^*)$ and toward 1 if $\Delta(\boldsymbol{y}', \boldsymbol{y}^*) > \Delta(\boldsymbol{y}, \boldsymbol{y}^*)$; this corresponds to a deterministic (error-free) decision. The other extreme case, namely $\beta = 0$, models an agent making decisions completely at random.

3.3 Maximum Likelihood Estimation

The probabilistic model outlined above is specified by two parameters: the ideal solution $\boldsymbol{y}^*$ and the (true) precision parameter β^*. Consider the problem of estimating these parameters, i.e., the parameter vector $\theta^* = (\boldsymbol{y}^*, \beta^*)$, from a given set $\mathcal{D} = \{\boldsymbol{y}^{(i)} \succ \boldsymbol{z}^{(i)}\}_{i=1}^N$ of observed preferences. In our case, $\mathcal{D}$ is given by the set of preferences collected from the query's nearest neighbors.

To solve this problem, we refer to the maximum likelihood (ML) estimation principle. Assuming independence of the preferences, the likelihood of $\theta = (\boldsymbol{y}, \beta)$ is given by

$$L(\theta) = L(\boldsymbol{y}, \beta) = \mathbf{P}(\mathcal{D} \,|\, \theta) = \prod_{i=1}^{N} \frac{1}{1 + \exp\left(-\beta(\Delta(\boldsymbol{z}^{(i)}, \boldsymbol{y}) - \Delta(\boldsymbol{y}^{(i)}, \boldsymbol{y})\right)} . \qquad (4)$$

Numerically, it is more convenient to deal with the log-likelihood

$$\ell(\theta) = \ell(\boldsymbol{y}, \beta) = -\sum_{i=1}^{N} \log\Big(1 + \exp\Big(-\beta(\Delta(\boldsymbol{z}^{(i)}, \boldsymbol{y}) - \Delta(\boldsymbol{y}^{(i)}, \boldsymbol{y})\Big)\Big) . \qquad (5)$$

The maximum likelihood estimation (MLE) $\theta_{ML} = (\boldsymbol{y}^{ML}, \beta^{ML})$ of θ^* is given by the maximizer of (5) (and hence the maximizer of (4)). Noting that, if $\mathbf{Y}$ is a discrete solution space, $\boldsymbol{y}$ is a discrete parameter while β is a continuous one, we tackle the corresponding optimization problem in two steps.

For a fixed $\boldsymbol{y}$, (5) becomes a one-dimensional function of β. The optimum of this function cannot be found analytically, however, since it is differentiable, an optimal β can easily be found by means of standard numerical optimization techniques like the Newton method. In other words, an optimal β, which we denote $\beta^{ML}(\boldsymbol{y})$, can easily be determined as a function of $\boldsymbol{y}$.

Assuming a discrete solution space $\mathbf{Y}$, the optimization of $\boldsymbol{y}$ can be implemented by means of any general purpose search method. The perhaps simplest approach is hill climbing: Starting with an initial solution $\boldsymbol{y}$, one determines

$$\boldsymbol{y}' = \arg \max_{\boldsymbol{y} \in \mathcal{N}(\boldsymbol{y})} \ell(\boldsymbol{y}, \beta_{ML}(\boldsymbol{y})) \ , \qquad (6)$$

where $\mathcal{N}(\boldsymbol{y}) \subset \mathbf{Y}$ is the neighborhood of $\boldsymbol{y}$. The current solution $\boldsymbol{y}$ is then replaced by $\boldsymbol{y}'$, and this process is continued until $\boldsymbol{y} = \boldsymbol{y}'$. Upon termination of this process, the MLE θ^{ML} is given by the currently optimal solution $(\boldsymbol{y}, \beta^{ML}(\boldsymbol{y}))$.

Since hill climbing is prone to local optima, it is important to find a good initial solution. To this end, we make use of the concept of a *generalized median.* Recall that the generalized median of a set of elements $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\} \subset \mathbf{Y}$ is given by

$$\boldsymbol{y}_{med} = \arg\min_{\boldsymbol{y} \in \mathbf{Y}} \sum_{i=1}^{N} \Delta(\boldsymbol{y}, \boldsymbol{y}^{(i)}) \ . \tag{7}$$

To verbalize, the generalized median is the element that minimizes the sum of distances to the $\boldsymbol{y}^{(i)}$. Now, suppose a set of preferences $\{\boldsymbol{y}^{(i)} \succ \boldsymbol{z}^{(i)}\}_{i=1}^{N}$ (instead of a set of single elements) to be given. As an extension of (7), we propose to look for

$$\boldsymbol{y}_{med} = \arg\min_{\boldsymbol{y} \in \mathbf{Y}} \left(\sum_{i=1}^{N} \Delta(\boldsymbol{y}, \boldsymbol{y}^{(i)}) - \sum_{i=1}^{N} \Delta(\boldsymbol{y}, \boldsymbol{z}^{(i)}) \right) \ , \tag{8}$$

that is, for a solution which is as close as possible to the preferred solutions $\boldsymbol{y}^{(i)}$ and, at the same time, as distant as possible from the non-preferred solutions $\boldsymbol{z}^{(i)}$. This solution is then taken as a starting point of the hill climbing procedure.

Note that a MLE $(\boldsymbol{y}^{ML}, \beta^{ML})$ induces a ranking (with ties) of the complete solution space $\mathbf{Y}$: For all $\boldsymbol{y}, \boldsymbol{z} \in \mathbf{Y}$,

$$\boldsymbol{y} \succeq \boldsymbol{z} \quad \text{iff} \quad \Delta(\boldsymbol{y}, \boldsymbol{y}^{ML}) \leq \Delta(\boldsymbol{z}, \boldsymbol{y}^{ML}) \ . \tag{9}$$

Also note that the MLE $\boldsymbol{y}^{ML}$ is the unique top-element of this ranking (at least provided that $\Delta(\boldsymbol{y}, \boldsymbol{y}') = 0$ implies $\boldsymbol{y} = \boldsymbol{y}'$).

3.4 The Subset Solution Space

As a concrete example, that we shall return to in Section 4 below, consider the special case of the "subset solution space", that is, the case where $\mathbf{Y} = 2^C$, with C being a finite set of labels or items (note that the subset-relation defines a complete lattice structure on $\mathbf{Y}$). Alternatively, exploiting a one-to-one correspondence between subsets (of a finite reference set) and binary vectors (of fixed length), we let $\mathbf{Y} = \{0,1\}^M$, where $M = |C|$. Thus, solutions are vectors $\boldsymbol{y} = (y_1, \ldots, y_M) \in \{0,1\}^M$, with $y_i = 1$ if the i-th item is contained in the corresponding set and $y_i = 0$ otherwise.

There are several commonly used distance measures for subsets (binary vectors), including the Hamming distance

$$\Delta_H(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{M} \sum_{i=1}^{M} |y_i - y_i'|$$

and the Jaccard distance

$$\Delta_J(\boldsymbol{y}, \boldsymbol{y}') = \frac{\sum_{i=1}^{M} \min(y_i, y_i')}{\sum_{i=1}^{M} \max(y_i, y_i')} \ .$$

Given a measure of this kind, a reasonable definition of the neighborhood in (6) is $\mathcal{N}(\boldsymbol{y}) = \{\boldsymbol{y}' \in \mathbf{Y} \,|\, \Delta(\boldsymbol{y}, \boldsymbol{y}') = 1\}$, i.e., $\boldsymbol{y}'$ is a neighbor of $\boldsymbol{y}$ if these two vectors differ by exactly one entry. For the Hamming loss, the determination of the initial solution (8) becomes especially simple. In fact, it is easily verified that a solution $\boldsymbol{y}^{med}$ is given by

$$y_j^{med} = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} \mathbb{I}(y_j^{(i)} = 0) + \mathbb{I}(z_j^{(i)} = 1) < N \\ 0 & \text{otherwise} \end{cases} ,$$

where $\mathbb{I}$ is the indicator function, i.e., $\mathbb{I}(P) = 1$ if the predicate P is true and $= 0$ if P is false.

4 Experiments

We conducted a number of experiments which are meant to provide a first validation of our approach. The main goal of these experiments is to show that, in principle, preference-based CBR is feasible. More specifically, we seek to show that a CBR agent can indeed learn from *indirect* feedback in the form of preferences. Moreover, we compare this kind of learning with the standard approach in which *direct* supervision is available.

4.1 Problem Solving Scenario

In agreement with the motivation underlying preference-based CBR, we consider a scenario in which the supervision is limited in the sense that, despite the possibility to *compare* candidate solutions, it is difficult or even impossible to determine an optimal or correct solution. Again, one may think of applications like cooking as an example: Two recipes can be compared, e.g., by cooking the meals and then testing which of them has a better taste, but there is usually no way to figure out the optimal solution. As explained earlier, the standard problem/solution representation becomes arguable in this setting.

As an alternative, we implement the following CBR procedure simulating a problem solving agent, which, roughly speaking, has the ability to compare candidate solutions and to "guess" new solutions, but not to verify the optimality of a solution. The agent proceeds from an initial case base, in which a case is a problem $\boldsymbol{x}$ together with a set of p pairwise preferences of the form $\boldsymbol{y} \succ_{\boldsymbol{x}} \boldsymbol{z}$. The agent then solves a sequence of problems in turn, and each problem solving episode consists of the following steps:

(i) Retrieval: Given a new query problem $\boldsymbol{x}_0$, the agent retrieves the preferences associated with the k nearest neighbors of $\boldsymbol{x}_0$ in the current case base. Thus, the agent gathers a set $\mathcal{D}$ of preferences, consisting of $k \cdot p$ comparisons in total.

(ii) Prediction: Based on this set of preferences, the agent derives a new solution for $\boldsymbol{x}_0$. More specifically, using our discrete choice model, it derives the ranking (9) and takes the top-ranked element $\boldsymbol{y}^{ML}$ as a prediction.

(iii) Evaluation: To measure the performance of the agent's solution, this solution is compared with the truly optimal solution $\boldsymbol{y}_0$ (which is not known to the agent). More specifically, we define the performance in terms of the Hamming distance $\Delta_H(\boldsymbol{y}_0, \boldsymbol{y}^{ML})$.

(iv) Indirect supervision: The agent is given feedback in the form of comparisons of its solution with p alternative solutions $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(p)}$. Thus, a set of p pairwise preferences of the form $\boldsymbol{y}^{ML} \succ \boldsymbol{y}^{(i)}$ or $\boldsymbol{y}^{(i)} \succ \boldsymbol{y}^{ML}$ is produced. These pairwise preferences are stored together with $\boldsymbol{x}_0$ as a new case in the case base.

The alternative solutions $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(p)}$ in step (iv) may originate from different sources. For example, the agent itself may try different solutions. To this end, it may sample suboptimal alternatives from the ranking (9), e.g., using sampling methods like tournament selection [17]. One may also imagine that the agent participates in a competition, in which its solution $\boldsymbol{y}^{ML}$ is compared with the solutions of other participants. In our experiments, we simply generated the $\boldsymbol{y}^{(i)}$ at random (i.e., by sampling from a uniform distribution on $\mathbf{Y}$). Each comparison ($\boldsymbol{y}^{ML}$ vs. $\boldsymbol{y}^{(i)}$) is implemented by means of our discrete choice model (3); recall that this model allows for erroneous comparisons, and that the corresponding level of noise is determined by the precision parameter β.

4.2 Problem Solving Domain

As an application domain, we consider the problem of *multi-label classification* (MLC), an extension of the standard classification problem that has received increasing attention in machine learning in recent years [18]. There are mainly two reasons for this choice. First, the output to be predicted in MLC is a subset of labels relevant for the query instance and, therefore, exactly matches the assumptions of the subset solution space. Second, there are benchmark data set available for the MLC problem.[1]

As a concrete example, we consider the *emotions* data that was created from a selection of songs from 233 musical albums [19]. From each song, a sequence of 30 seconds after the initial 30 seconds was extracted. The resulting sound clips were stored and converted into wave files of 22050 Hz sampling rate, 16-bit per sample and mono. From each wave file, 72 numerical features have been extracted, falling into two categories: rhythmic and timbre. Then, in the emotion labeling process, 6 main emotional clusters are retained corresponding to the Tellegen-Watson-Clark model of mood: amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely and angry-aggressive. The task is to predict the subset of labels that apply to each individual song.

4.3 Results

We implemented the scenario described in Section 4.1 with different values for the parameters p (number of pairwise comparisons per case) and β (precision of

[1] `http://mlkd.csd.auth.gr/multilabel.html`

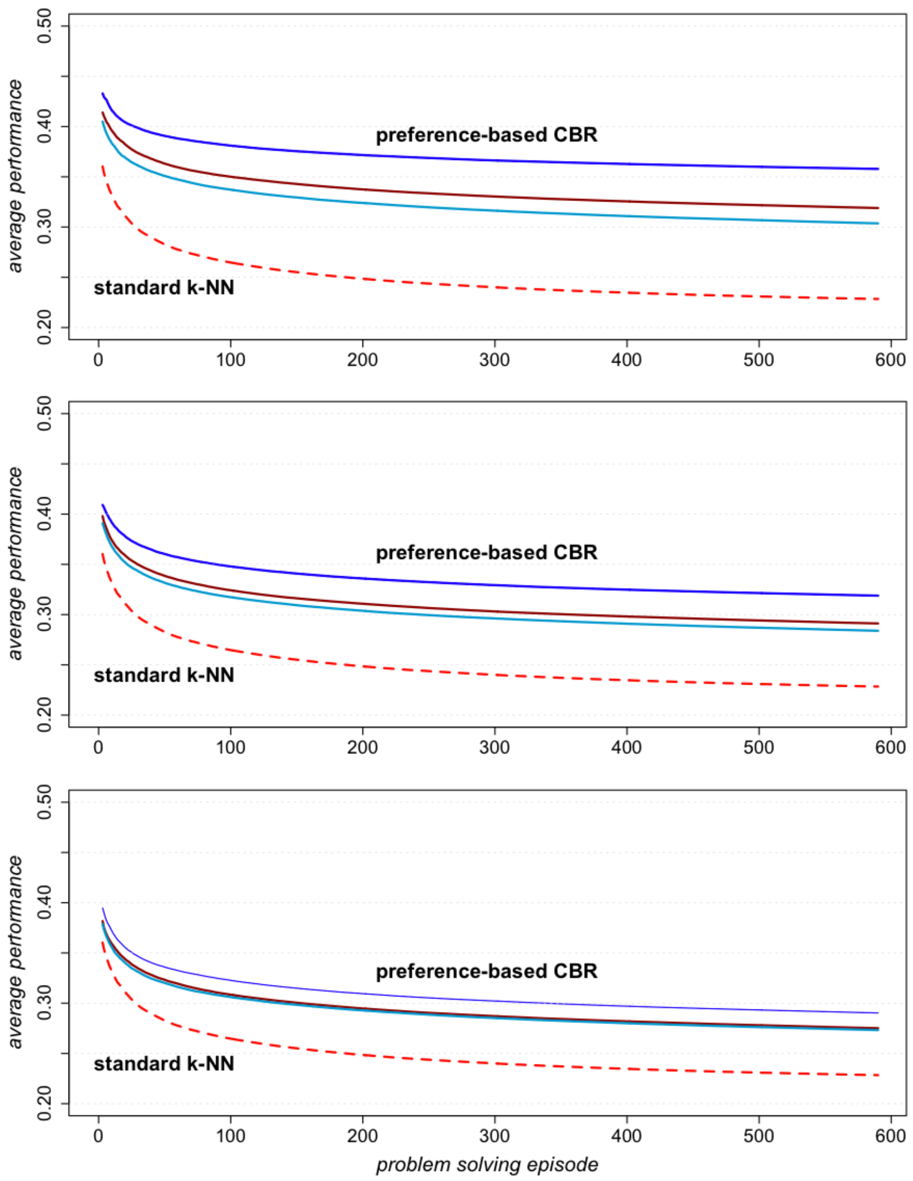


Fig. 1. Performance curves for the emotions data: Standard 3-NN as a baseline (dashed curve) and preference-based CBR (solid) with 7 (upper), 15 (middle) and 30 (lower) preferences per case and precision values of $\beta = 5$, 10 and 20, respectively

the comparisons); the number of retrieved cases was fixed to $k = 3$. As a baseline, we compared with "standard" CBR, in which the supervision is *direct*: Instead of the indirect supervision in step (iv) of our problem solving scenario, the true solution $\boldsymbol{y}_0$ is shown to the agent (and stored in the case base). Given a new query problem $\boldsymbol{x}_0$, the agent retrieves the solutions of its k nearest neighbors and derives the generalized median (7) as a prediction. As a distance measure on the problem space $\mathbf{X}$, we always used the simple Euclidean distance.

In Fig. 1, the performance is shown for the emotions data in terms of a curve $t \mapsto P(t)$, where $P(t)$ is the average performance in the first t problem solving episodes (i.e., the average distance $\Delta_H(\boldsymbol{y}_0, \boldsymbol{y}^{ML})$). Since this curve depends on the order in which problems are encountered, it was "smoothed" by averaging over several random permutations of a data set. The initial case base always comprised the first k cases, for which pairwise comparisons between randomly generated solutions (or, in the case of the baseline, the true solutions) were added.

As can be seen, standard k-NN performs slightly stronger than preference-based CBR. This, of course, was to be expected: While k-NN is fully supervised, having access to the true solutions of the cases stored in the case base, preference-based CBR is only guided through indirect hints in the form of pairwise comparisons (which are perhaps even noisy). Seen from this point of view, it still performs rather well, and the difference between the methods becomes smaller with an increasing number of preferences per case. Moreover, the learning effects due to an extension of the case base are quite comparable. Similar results, which are omitted here due to space restrictions, were obtained for other MLC data sets.

5 Concluding Remarks

Our project agenda envisions a methodological framework of preference-based CBR, which disposes of a sound theoretical basis and, at the same time, accommodates a wide spectrum of potential applications. Ideally, a user can easily "parametrize" this framework, e.g., by choosing the type of output space and the distance measure defined on this space, whereas the methods themselves are completely generic and essentially independent of the concrete application at hand. In this regard, as already remarked in the introduction, we are to some extent guided by AI methodologies like probabilistic graphical models and constraint satisfaction.

Needless to say, this paper is only a first step toward this goal. First, by focusing on the case-based inference part, we only considered one aspect of CBR, albeit an important one. Apart from this, there are of course a number of further issues that need to be addressed, such as case-base organization and maintenance. Second, as already noticed earlier, the methods used for implementing preference-based CBR strongly depend on the type and structure of the solution space. Although the formal approach outlined in Section 3 is quite general, we later on focused on the subset solution space. Thus, similar methods have to be developed and validated for other types of solution spaces, too.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
2. Richter, M.M.: The knowledge contained in similarity measrues (1995) (invited talk at ICCBR 1995), https://kluedo.ub.uni-kl.de/files/117/no_series_120.pdf
3. Watson, I.: Case-based reasoning is a methodology not a technology. In: Mile, R., Moulton, M., Bramer, M. (eds.) Research and Development in Expert Systems XV, London, pp. 213–223 (1998)
4. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge (2009)
5. Hüllermeier, E.: Case-Based Approximate Reasoning. Theory and Decision Library, Series B: Mathematical and Statistical Methods, vol. 44, 370 pages. Springer, Heidelberg (2007)
6. Doyle, J.: Prospects for preferences. Comput. Intell. 20(2), 111–136 (2004)
7. Goldsmith, J., Junker, U.: Special Issue on preference handling for Artificial Intelligence 29(4) (2008)
8. Domshlak, C., Hüllermeier, E., Kaci, S., Prade, H.: Preferences in AI: An overview. Artificial Intelligence (to appear)
9. Brafman, R.I., Domshlak, C.: Preference handling–an introductory tutorial. AI Magazine 30(1) (2009)
10. Brinker, K., Hüllermeier, E.: Label ranking in case-based reasoning. In: Richter, M.M., Weber, R.O. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 77–91. Springer, Heidelberg (2007)
11. Kibler, D., Aha, D.W., Albert, M.K.: Instance-based prediction of real-valued attributes. Computational Intelligence 5, 51–57 (1989)
12. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning 6(1), 37–66 (1991)
13. Binshtok, M., Brafman, R.I., Domshlak, C., Shiomony, S.E.: Generic preferences over subsets of structured objects. Journal of Artificial Intelligence Research 34, 133–164 (2009)
14. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards case-based adaptation of workflows. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 421–435. Springer, Heidelberg (2010)
15. Cheng, W., Hüllermeier, E.E.: Combining instance-based learning and logistic regression for multilabel classification. Machine Learning 76(2-3), 211–225 (2009)
16. Peterson, M.: An Introduction to Decision Theory. Cambridge Univ. Press, Cambridge (2009)
17. Butz, M., Sastry, K., Goldberg, D.E.: Tournament selection: Stable fitness pressure in XCS. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1857–1869. Springer, Heidelberg (2003)
18. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. Int. J. of Data Warehousing and Mining 3(3), 1–13 (2007)
19. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: Proc. Int. Conf. Music Information Retrieval (2008)

How Many Cases Do You Need? Assessing and Predicting Case-Base Coverage

David Leake and Mark Wilson

School of Informatics and Computing,
Indiana University, Bloomington, IN, USA
{leake,mw54}@cs.indiana.edu

Abstract. Case acquisition is the primary learning method for case-based reasoning (CBR), and the ability of a CBR system's case-base to cover the problems it encounters is a crucial factor in its performance. Consequently, the ability to assess the current level of case-base coverage and to predict the incremental benefit of adding cases could play an important role in guiding the case acquisition process. This paper demonstrates that such tasks require different strategies from those of existing competence models, whose aim is to guide selection of competent cases from a known pool of cases. This paper presents initial steps on developing methods for predicting how unseen future cases will affect a system's case-base. It begins by discussing case coverage as defined in prior research, especially in methods based on the representativeness hypothesis. It then compares alternative methods for assessing case-base coverage, including a new Monte-Carlo method for prediction early in case-base growth. It evaluates the performance of these approaches for three tasks: estimating competence, predicting the incremental benefit of acquiring new cases, and predicting the total number of cases required to achieve maximal coverage.

1 Introduction

The case library is a fundamental knowledge container for case-base reasoning (CBR) systems. CBR system development often includes a case acquisition process to capture set of "seed cases," as an initial case-base, after which additional cases are gained during problem-solving. Currently, little quantitative guidance is available to help system-builders to predict the likely benefit of acquiring another seed case and the number of cases which will be required to maximize coverage. The ability to predict the incremental benefit of case acquisition could help to determine whether the effort to acquire new cases is worthwhile; predictions of the case-base size required to achieve a desired performance level—and of whether it is practical to achieve that performance level solely by adding cases—could both help to determine the suitability of CBR for a given task and help system designers decide whether to focus effort on case acquisition or on improving other knowledge containers, such as case adaptation knowledge, to reduce the number of cases that will be needed.

Assessing and predicting the effects of case acquisition is closely connected to estimating *case-base competence*, the ability of a system's case-base to support the solution of potential target problems. Methods to predict case-base competence effects if unseen cases are added to the case-base can in turn be used to estimate incremental competence gains from adding a future case to the case-base. If methods can be developed to extrapolate these predictions to estimate the competence effects of adding larger numbers of cases, those estimates may be used for addressing questions such as the maximum competence the system is likely to achieve.

Case-base competence has received substantial attention in CBR research, from the perspective of guiding generation of compact and competent case-bases. This work has centered on choosing which cases in an existing set of cases to delete (for competence-based deletion, e.g., [4,6]) or to add (for competence-based addition, e.g., [7]). This paper begins with background on previous treatments of competence, based on the representativeness assumption that the existing case-base can be used as a proxy for future problems [6]. Such treatments have proven effective for their intended purpose of guiding competence-based deletion from a known case-base with satisfactory coverage. However, representativeness is not assured for the partial case-bases arising during early phases of case acquisition.

To estimate coverage characteristics, the paper presents a set of empirical methods for future coverage prediction. These include a new approach which uses Monte Carlo integration—assessing coverage for a random sampling of the problem space—to predict coverage without requiring the representativeness assumption. This approach does not require knowledge of the correct solutions for the sampled problems, but can be adjusted to reflect additional knowledge about expected problem distributions and weighted to reflect additional cost/benefit information.

The paper presents an evaluation of its competence estimation methods as bases for three tasks: (1) predicting the incremental benefit of case acquisition, (2) estimating a tight upper bound on the competence a system will achieve, based on the competence effects of adding initial cases, and (3) predicting the number of cases required for the case base to approximate maximal competence. The paper evaluates performance for a range of case-bases. The results support that representativeness-based methods for competence assessment may not be well-suited to early competence estimation, but that both the leave-one-out and Monte Carlo methods can provide useful information, and that the Monte Carlo method has advantages over the leave-one out method for this task.

2 Competence and Representativeness

Smyth and McKenna's seminal work on representativeness-based coverage [6] defines competence as "the range of target problems that a given system can solve." If "competence" is considered to reflect problem-solving accuracy over the entire problem space P, it may be defined as the fraction of the problems

$p_i \in P$ which the reasoner will solve correctly, i.e., for which the case or cases retrieved from the case-base will be adapted to produce a correct solution:

$$Competence(CB) = \frac{\sum_{p_i \in P} Correct(Adapt(Retrieve(CB,\ p_i),\ p_i))}{|P|}$$

A challenge for determining competence for a real problem distribution is that the actual set of problems to be encountered in the future may be impossible to know *a priori*. In the context of determining the contribution of particular cases to the competence of a given case base, Smyth and McKenna address the problem of unavailable target cases by basing competence calculations on the *representativeness assumption*: "The case base is a representative sample of the target problem space" [6]. As Smyth and McKenna observe, in the scenario they consider it is reasonable to expect the representativeness hypothesis to hold: If the case-base were not representative of future problems, CBR would be inappropriate for the task. Accordingly, their proposed competence metric explicitly excludes mention of target problems, instead considering only how the system's cases contribute to solving other cases in its existing case-base. The coverage of a single case c with respect to a case-base C is defined as the set of cases for whose problems it would be retrieved and to which it can be successfully adapted [5]:

$$coverage(c \in C) = \{c' \in C : c' \in RetrievalSpace(c) \cap AdaptationSpace(c)\}$$

The representativeness-based approach has been used as a basis for estimating both global competence [5] and relative coverage [6], a criterion for determining which cases are most important to retain in the case-base. Representativeness approaches have been shown to be effective for guiding competence-preserving case deletion from a set of cases with satisfactory competence. However, during initial case acquisition, before satisfactory coverage is achieved, there is no guarantee that the cases seen will be representative.

Some prior work has studied how to identify additional cases needed for finite case-bases [3]. However, there has been little attention to the problem of predicting the number of additional cases which a CBR system may need to achieve maximal competence. It might appear that prior approaches for assessing case-base competence could also be used to predict the number of cases needed. However, Massie, Craw, and Wiratunga [2] have shown that metrics developed to assess competence are not necessarily good indicators of the accuracy achievable with a given set of cases.

3 Empirical Competence Estimation Strategies

We consider four empirical approaches for estimating competence: leave-one-out testing using a limited number of test cases, leave-one-out testing using all cases currently in the case-base, competence group estimation, and a Monte Carlo method.

3.1 Leave One Out Testing

Leave-one-out testing is a simple approach for estimating accuracy. For tasks for which solvability is Boolean—either a case is solved or it is not—competence depends solely on the percent of target problems solved correctly. For other types of problems, leave-one-out testing can be applied in conjunction with other types of criteria for estimating competence, such as determining the percent of target problems whose solution is within a given threshold of the correct solution, or even simply considering average solution accuracy. Other criteria could use different weightings for different problems (e.g., based on the risks associated with particular types of errors).

We will consider two variants on leave-one-out testing. The first uses all the cases in the current case-base. The second increases efficiency by conducting testing using a smaller subset of the case-base.

3.2 Competence Group Estimation

The representativeness-based competence metric we consider follows Smyth and McKenna's coverage metric [6], based on the notion of competence groups. Each case c has an associated *coverage* within the case-base, consisting of those cases which are retrieved for c by the reasoner's retrieval component and which can be adapted to solve c by the reasoner's adaptation component. A *competence group* is defined as a set of cases such that all cases share coverage with some other case in the group, and no case outside the group shares coverage with any case in the group. The *density* of a case c within a group G is defined as:

$$\text{Density}(c, G) = \frac{\sum_{c' \in G - \{c\}} \text{Similarity}(c, c')}{|G| - 1}$$

The *group density* is defined as the sum of the individual member cases' densities, divided by the cardinality of the group. *Group coverage* is taken as:

$$\text{GroupCoverage}(G) = 1 + (|G| \cdot (1 - \text{GroupDensity}(G)))$$

Case-base coverage is taken as the sum of group coverages over all competence groups in the case-base.

3.3 Monte Carlo Coverage Estimation

Leave-one-out testing and representativeness-based approaches use existing cases in the case-base as proxies for future problems. When the available cases may not be representative, as when few cases have been acquired, this assumption is less appropriate. To predict coverage of cases which may not be reflected by the existing case-base, we propose a Monte Carlo technique. This Monte Carlo method uniformly samples the problem space, and tests whether the sampled points are expected to be solvable. This process does not require actually generating solutions for the sampled points, provided that a criterion exists for determining

Table 1. General Monte Carlo sampling algorithm

```
1. Generate problem set with desired distribution
2. Initialize the total cost to 0
3. For each problem p:
   3a. Find the closest case to p in the case-base
   3b. If that case does not cover p, add the cost of not covering p to the total cost.
```

Table 2. Monte Carlo coverage estimation algorithm for our experiments

```
Apply General Monte Carlo sampling algorithm with:
   Uniform problem distribution for n samples
   Cost = 0 if problem covered by nearest case; else 1.
Return (Monte Carlo result)/n
```

solvability. For example, a sampled case could be considered solvable if it were sufficiently similar to an existing case in the case-base, based on the system similarity metric. Because this method does not require access to any cases beyond those already in the case-base, the number of points it can test is limited only by available processing time. This is contrast to the leave one out approaches, which are limited by the number of cases in the case-base.

The previously-described Monte Carlo process can be enriched in two ways to better reflect pragmatic constraints. We describe these for generality, but leave their exploration for future research:

- **Biased sampling:** When the problems encountered are not uniformly distributed, and if information about the distribution is available, the Monte Carlo sample selection process can be biased to reflect that distribution. The most informative results about the system's ability to cover problems in practice would follow from sampling frequently from regions of the problem space in which future problems are likely to occur and less frequently in problem areas that are unlikely to arise.
- **Problem-specific costs:** Rather than simply considering points as "covered" or not, a cost function could be used to reflect factors such as finer-grained accuracy loss or the costs of failure to cover particular cases, based on knowledge of the importance of those cases, as illustrated in Table 1.

In the following, we apply the general Monte Carlo algorithm using a problem generator which randomly selects problems with a uniform distribution throughout the problem space. Cases which fall within the coverage of an existing case, as determined by a similarity threshold, are recorded as solved. The percent solved is then be used to approximate the coverage of the case-base. Table 2 sketches our general Monte Carlo Coverage algorithm and its application here.

4 Extrapolating from Competence Graphs to Estimate Needed Cases

As illustrated in the following experiments, the observed coverage growth for our sample case-bases followed a standard pattern, reaching an asymptotic value. If the details of this standard pattern can be predicted for a given case-base, such predictions can be applied to in turn predict the number of cases needed to approximate this maximal performance level.

We hypothesized that the general form of competence as a function of case-base size (x) can be approximated by the following function:

$$f(x) = c \cdot (1 - (x + b)^{-p})$$

The shape of the function is displayed by the fitted curves shown in Fig. 2 (all curves shown except for the Empirical Accuracy graph, which represents raw data points).

This function captures the "elbow" or corner point of diminishing returns that is typical of these experiments. Early in case acquisition, insufficient data will be available to make reasonable long-term predictions. However, we hypothesized that prediction algorithms can detect when the "elbow" of the curve is reached, and at that point prediction can begin.

5 Experiments

5.1 Overview and Design

We conducted experiments to compare the performance of representativeness, leave-one-out, and Monte-Carlo integration as a basis for the following tasks:

1. Estimation of system competence
2. Prediction of the marginal competence benefit from acquiring a new case
3. Prediction of the number of cases needed to for the system to achieve accuracy within ϵ of its maximum accuracy level

Our experiments use four classification data sets, drawn from the UCI Machine Learning Repository [1]: Ad (classification of Internet images as ads), Mini-BOONE (classification of particles), Adult (classification of income level), and Car (classification of car models as acceptable to consumers). For each data set, the same naive similarity metric was used, Euclidean distance on problem features normalized by their ranges. Data sets with categorical features were given simple hand-designed numeric distances between categories for those features.

The experiments used 10-fold cross-validation, with each data set split randomly into $f = 10$ folds. Tunable experiment parameters included the range of case-base size to test and the size increments to use, the number of points for the Monte Carlo procedure to sample (for our experiments, 50 points were sampled), and the number of nearest cases to use when generating a CBR solution for test problems (3-NN for our experiments).

Estimation of Competence. The experiments test competence estimation for a variety of case-base sizes. For each case-base size, the cases for the case-base are drawn sequentially from an initial random ordering of the current fold. At each case-base size step, five values are computed:

1. Empirical accuracy: The percentage of problems from the $(f-1)$ test folds which are solved correctly by the case-base
2. A leave-one-out estimate of the case-base's accuracy, using all cases in the current case-base
3. A leave-one-out estimate of the case-base's accuracy, limited to the same number of samples as the Monte-Carlo estimate
4. The representativeness coverage value
5. The Monte Carlo estimate of the case-base's coverage

The comparative results of 3 and 4 are interesting in that they enable comparing the effectiveness of leave-one-out and Monte Carlo methods when each has access to the same amount of information.

Prediction of marginal coverage benefit of next case addition As the basis for prediction of the marginal coverage benefit, the system attempts to fit the values produced by each estimation technique to the curve described in Section 4. To fit the data to the curve, each set of values estimating the competence of the case-base is linearized according to the inverse of the previously stated curve, *i.e.*, by:

$$f^{-1}(y) = (1 - \frac{y}{c})^{-1/p} - b$$

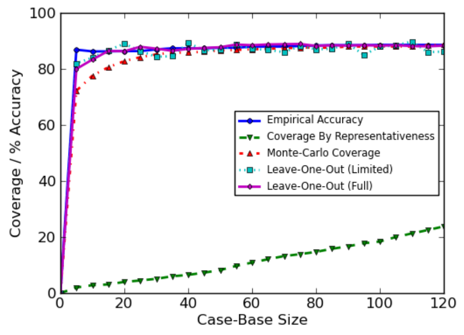
Where $\mathbf{y}$ are the estimate values. A least-squares fitting is used to fit the parameters c, b, and p to the known case-base sizes $\mathbf{x}$.

The curve fitted to the first s points is used to predict the gain in accuracy that will result by expanding the case-base to size $s+i$.

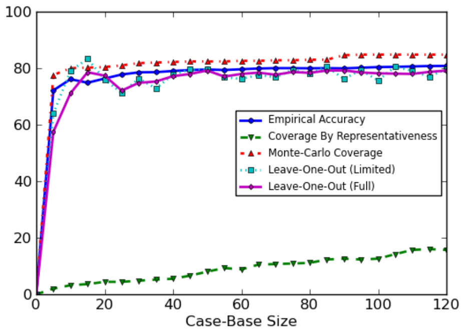
Prediction of the Number of Cases Required for Maximal Coverage. The fitted curve can also be used to predict the number of cases that will be required for the reasoner to reach within ϵ of a target accuracy value a. In our experiments, we use the curve fitted to the estimate values up to s to predict the case-base size at which the reasoner will reach an accuracy of at least $a - \epsilon$.

For our experiments, we set a to the empirical accuracy value obtained with the largest case-base size sampled and set ϵ to 5% of a. In practice, a developer could select any desired target accuracy value less than the maximum.

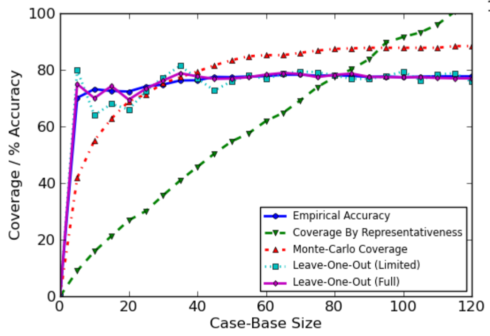
Predictions are only generated after the "elbow" of the curve has been crossed. The number of cases needed to reach a is re-predicted after each acquisition step, as with the marginal-benefit task, allowing the estimation methods to refine the prediction of the needed number of cases after every acquisition. We expect the accuracy of all prediction methods to improve (on average) with each case acquired.



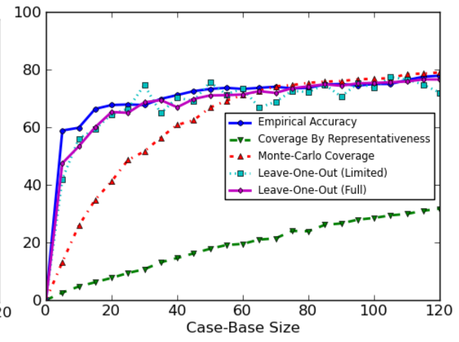
(a) Ads



(b) MiniBOONE



(c) Adult



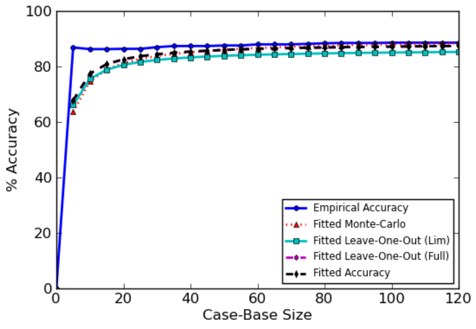
(d) Car

Fig. 1. Estimates of coverage, accuracy, and empirical accuracy

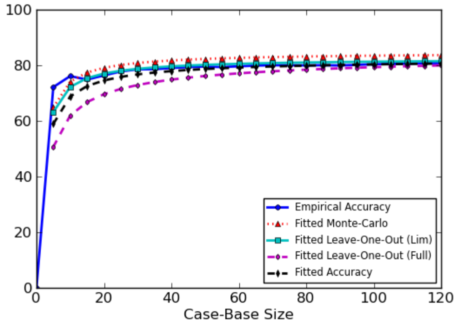
By comparing predicted values to the empirical results – i.e., the gain in empirical accuracy by expanding the case-base to size $s+i$ and the size at which the case-base's empirical accuracy crossed $a-\epsilon$, we determined the accuracy of these predictions for each estimation technique at every case-base size evaluated.

5.2 Results

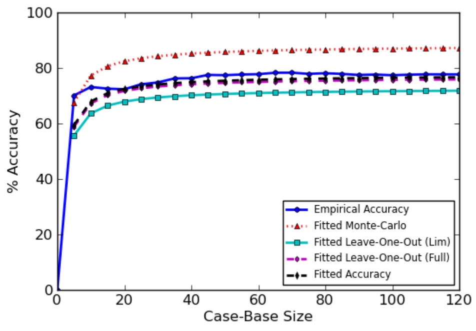
Estimation of Competence. Fig. 1 shows the estimation results and empirical accuracy. We note that the representativeness function is not intended to produce a result in percent accuracy, so it is only meaningful to compare its shape to the shapes of the curves for the other methods. We observe that for both Ads and MiniBOONE, maximal accuracy is approached with a small number of cases. Leave-one-out and Monte Carlo methods both track actual accuracy closely after an initial lag and level off quickly. Results with Adult and Car show



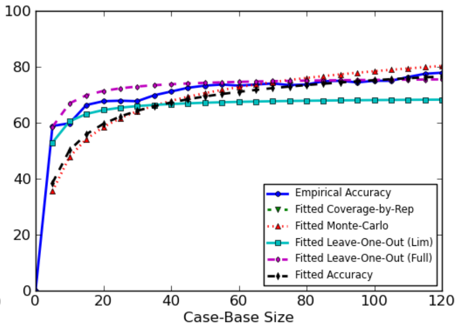
(a) Ads



(b) MiniBOONE



(c) Adult



(d) Car

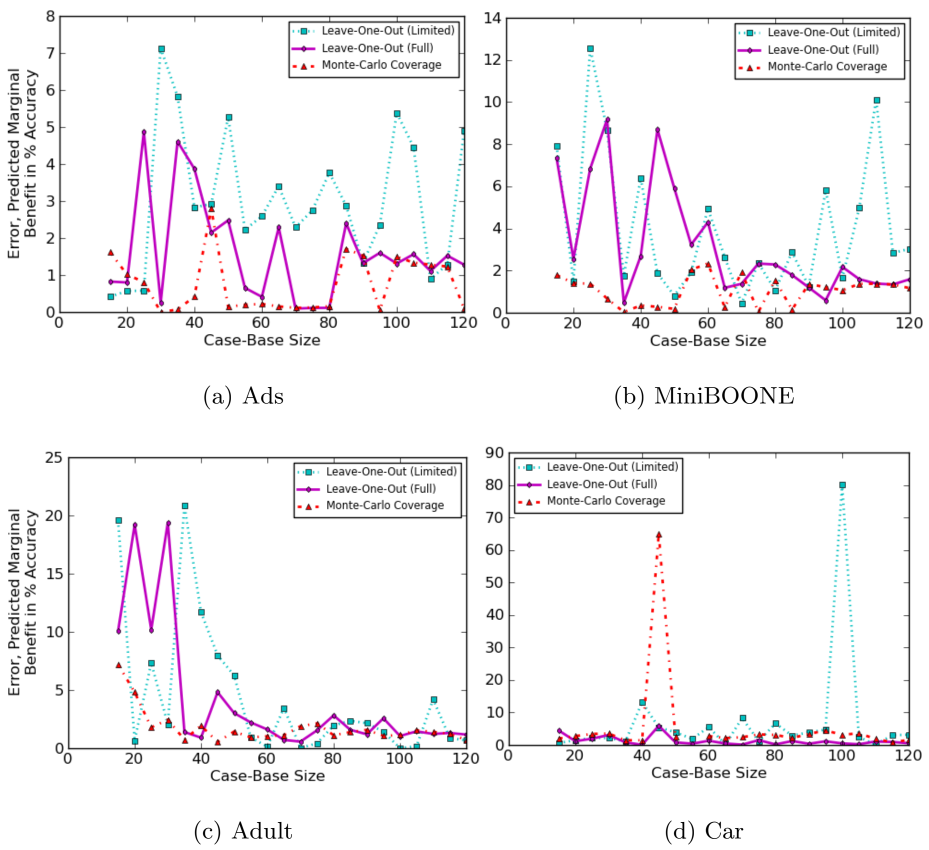
Fig. 2. Empirical accuracy and curves fitted to the estimates of accuracy/coverage

more differentiation between the methods, with full leave-one-out providing the best performance, followed by limited leave-one-out and then Monte Carlo.

The general behavior of the representativeness coverage estimates contrasts with that of other methods, producing linear or nearly linear growth as a function of case-base size. This is observation on representativeness estimates is consistent with results by Massie et al. [2].

Curve Fitting to Accuracy Estimates. Because the general form of the representativeness graph does not match the curve of the other methods, we consider only the results for the other methods. Fig. 2 shows the results of fitting the curve to the other methods, which all fit the general pattern, with some variation compared to empirical accuracy.

Prediction of Marginal Benefit of Acquisition. Fig. 3 shows the absolute error in predicted marginal benefit of case acquisition, for each case-base size

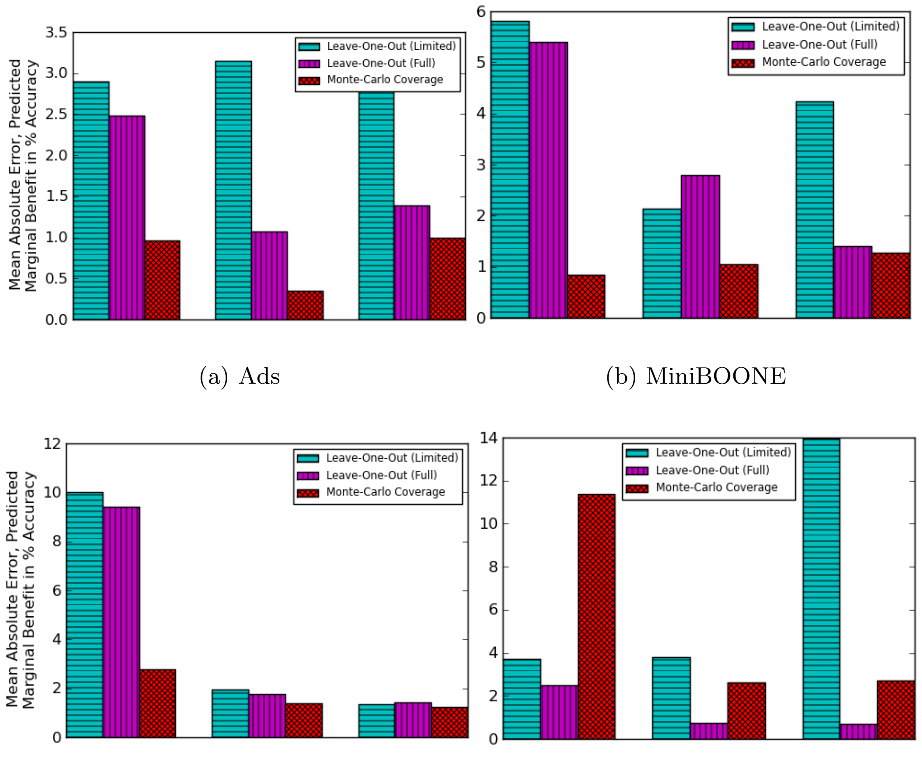


(a) Ads (b) MiniBOONE

(c) Adult (d) Car

Fig. 3. Error in predicted marginal benefit of case acquisition, by case-base size

for which predictions are available. (For some case-base sizes, curve fitting was occasionally unsuccessful for some estimate methods. Missing values in the graph reflect failed curve-fitting, and the estimate technique incurs no error penalty for these missed predictions.) Fig. 4 graphs the means of the available error values for three different regions of case acquisition – early, middle, and late case-base growth. To compute these values, the entire experiment was split evenly into three stages and averages computed for each stage, to illustrate the accuracy of different estimate techniques at each stage. These values illustrate that the Monte-Carlo integration method generally compares favorably with leave-one-out for predicting marginal benefit of new case acquisitions. In the Car data set, the Monte-Carlo technique bests the leave-one-out technique in two out of three stages when the leave-one-out is limited to the same number of samples as Monte-Carlo, but not when leave-one-out is allowed the full range of the case-base. However, see below for a discussion of the time required to execute each test.



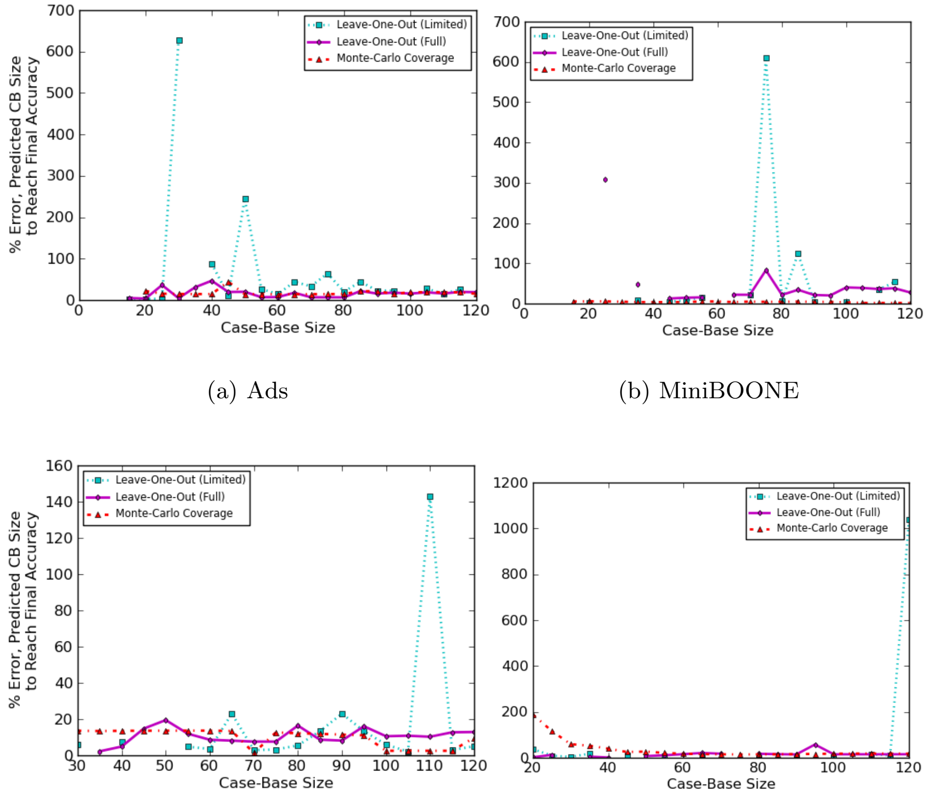
(a) Ads (b) MiniBOONE

(c) Adult (d) Car

Fig. 4. Mean error in predicted marginal benefit of case acquisition, for early, middle, and late stages of case-base growth

Prediction of Number of Cases Needed to Achieve Maximal Accuracy. The absolute error in predicting the case-base size required to reach at least within ϵ of the final experimental accuracy is shown in Fig. 5. These values are presented as percentages of the final case-base size in the experiment. The mean absolute error in these predictions is shown for each data set in Fig. 6. The error in the Monte-Carlo technique is higher here, but it is often possible to produce a prediction with the Monte-Carlo method when such a prediction is impossible with the leave-one-out techniques because a curve could not be fitted.

After 120 cases, the respective errors for limited leave-one-out, full leave-one-out, and Monte Carlo, for Ads are no prediction possible, 19%, and 15%; for MiniBOONE are no prediction possible, 27%, and 2%; for Adult are 5%, 13%, and 10%; and for Car are 1038%, 15%, and 18%.

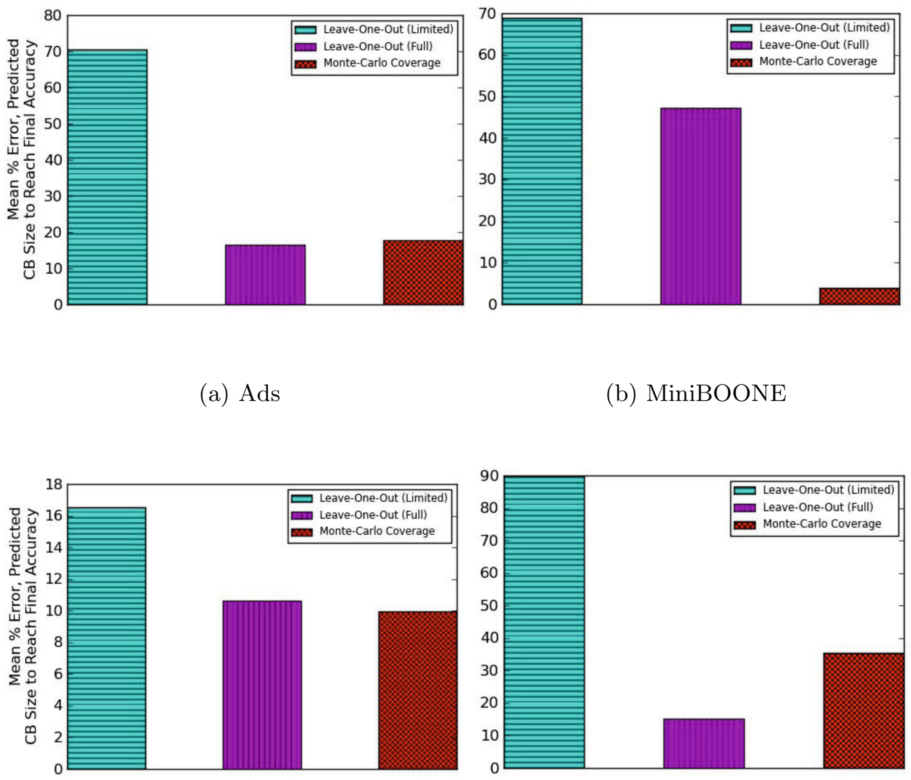


(a) Ads (b) MiniBOONE

(c) Adult (d) Car

Fig. 5. Error in predicted case-base size to reach within ϵ of final experimental accuracy, by case-base size

Note on Computation Time. The time elapsed to compute the estimates with each technique is shown in Fig. 7. The Monte-Carlo coverage method required less time than the representativeness coverage technique or the leave-one-out estimate using the full case-base (although leave-one-out can be faster for very small case-bases, its time grows more quickly and rapidly overtakes the Monte-Carlo technique). When leave-one-out testing is limited to the same number of samples as the Monte-Carlo technique, their elapsed time is comparable; however, as shown by the previous results, the accuracy of the leave-one-out technique is generally compromised by doing so.



(a) Ads (b) MiniBOONE

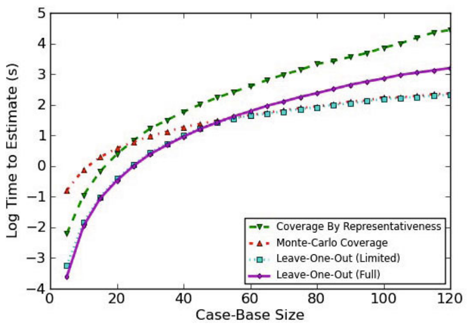
(c) Adult (d) Car

Fig. 6. Mean error in predicted case-base size to reach within ϵ final experimental accuracy

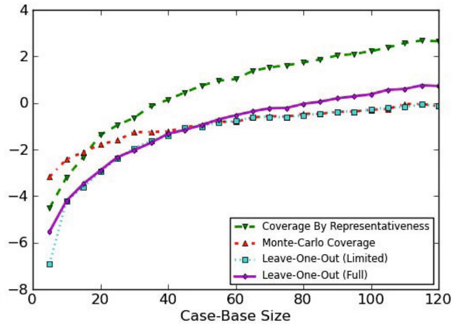
6 Future Work

The previous sections introduce the problem of predicting case-base coverage, illustrate some central points, and present experiments testing initial methods. A number of questions remain. One is how best to handle problem streams with non-uniform distributions, if those distributions are not known *a priori*. Another interesting future area is how to develop automated methods for selecting values such as similarity thresholds for deciding whether to treat a case as covered.

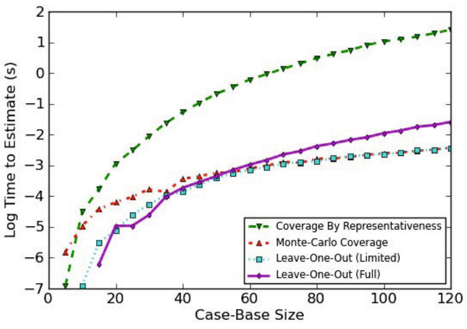
The ability to predict the benefits of case acquisition also raises questions for the tradeoff between increased case adaptation knowledge and increased case knowledge and how to provide guidance for CBR system developers deciding how to divide their effort between augmenting these two knowledge containers.



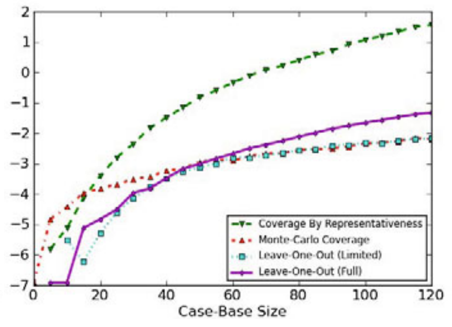
(a) Ads



(b) MiniBOONE



(c) Adult



(d) Car

Fig. 7. Log time (in seconds) to compute accuracy and competence estimates

7 Conclusion

As the acquisition of seed cases is an important part of the development of CBR systems, the ability to predict the benefit of such acquisitions could play a valuable role in guiding case acquisition decisions. Likewise, knowledge of the benefit trends for case acquisition can aid in predicting the number of cases which will be needed to achieve a desired level of accuracy and in predicting limits on the accuracy attainable, aiding predictions of the practicality and effort required to build a CBR system.

This paper explores methods for predicting coverage growth, including a Monte Carlo simulation method to enable predictions early in the case acquisition process, and presents tests illustrating the methods potential. This work provides a first step towards answering the question of how to predict the number of cases it will be necessary to acquire for a CBR system.

Acknowledgments. This work is based on work supported by the National Science Foundation under Grant No. OCI-0721674 and by a grant from the Data to Insight Center of Indiana University.

References

1. Blake, C., Merz, C.: UCI repository of machine learning databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Massie, S., Craw, S., Wiratunga, N.: Complexity-guided case discovery for case based reasoning. In: AAAI 2005: Proceedings of the 20th National Conference on Artificial Intelligence, pp. 216–221. AAAI Press, Menlo Park (2005)
3. McSherry, D.: Automating case selection in the construction of a case library. Knowledge-Based Systems 13(2-3), 133–140 (2000)
4. Smyth, B., Keane, M.: Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, pp. 377–382. Morgan Kaufmann, San Mateo (1995)
5. Smyth, B., McKenna, E.: Modelling the competence of case-bases. In: Cunningham, P., Smyth, B., Keane, M. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 208–220. Springer, Heidelberg (1998)
6. Smyth, B., McKenna, E.: Building compact competent case-bases. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 329–342. Springer, Heidelberg (1999)
7. Zhu, J., Yang, Q.: Remembering to add: Competence-preserving case-addition policies for case base maintenance. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp. 234–241. Morgan Kaufmann, San Francisco (1999)

A Case-Based Approach to Open-Ended Collective Agreement with Rational Ignorance

Sergio Manzano[1,2], Santiago Ontañón[1], and Enric Plaza[1]

[1] IIIA-CSIC, Artificial Intelligence Research Institute (Spanish Scientific Research Council), 08193 Bellaterra, Catalonia, Spain
[2] Universitat Autonòma Barcelona, Bellaterra, 08193 Catalonia, Spain
{sergio,santi,enric}@iiia.csic.es

Abstract. In this paper we focus on how to use CBR for making collective decisions in groups of agents. Moreover, we show that using CBR allows us to dispense with standard but unrealistic assumptions taken in these kind of tasks. Typically, social choice studies voting methods but assumes complete knowledge over all possible alternatives. We present a more general scenario called *open-ended deliberative agreement with rational ignorance (ODARI)*, and show how can CBR be used to deal with rational ignorance. We will apply this approach to the *Banquet Agreement* scenario, where two agents deliberate and jointly agree on a two course meal. Rational ignorance makes sense in this scenario, since it would be unreasonable for the agents to know all the alternatives. Unknown alternatives, as well as a strategy to increase chances of reaching an agreement, are problems addressed using case-based methods.

1 Introduction

Case-based reasoning (CBR) has been applied to a wide number of of real life tasks, and one feature that stands out in comparison to other AI techniques is its resilience and robustness in the presence of incomplete knowledge. This capability of dealing with incomplete knowledge, by analyzing and exploiting the implicit knowledge in a case base, is a core idea in CBR, and a main factor in being able to perform well in real-life applications in which standard oversimplifying assumptions (like having complete knowledge) can not be held.

In this paper we focus on how to use CBR for making collective decisions in groups of agents. Moreover, we show that using CBR allows us to dispense with standard but unrealistic assumptions taken in these kind of tasks. Social choice is the theoretical study of methods for aggregating individual preferences (or utilities) into collective decisions —and how to evaluate adequacy of these collective outcomes (welfare). Typically, social choice studies voting methods and their properties as a means to aggregate individual preferences into a collective outcome. However, current social choice approaches make strong assumptions that are tantamount to require complete knowledge to the individual agents participating in a collective decision (see Section 2).

We claim that these assumptions are too strong for Artificial Intelligence applications, specially in multiagent systems approaches, and that we should be focusing on group decisions where individuals have partial knowledge of the decision domain. Specifically, we propose that CBR offers a practical and natural way to allow individual agents with partial knowledge to achieve collective decisions that are reasonable and satisficing[1].

Let us consider a group decision scenario in a context related to the ICCBR Computer Cooking Competition. While the main challenges of the Computer Cooking Competition are retrieving and adapting a recipe given some query, we propose to focus on the task we call *Banquet Agreement scenario*[2]. Let us consider a group of 2 or more individuals that have to decide on the specific dishes to be served in a banquet: these individuals may be the Chairs of a conference deciding on the Conference Banquet, or the members of two families for a marriage banquet, or a couple inviting a large group of friends to dinner.

This scenario allows us to consider the implications of assuming complete or incomplete knowledge. Classical social choice assumes complete knowledge: every individual *knows* the utility value of every alternative —usually this is phrased as every individual having a utility function over the set of alternatives. However, in the cooking domain, the set of alternatives are the different dishes or recipes, for which there are thousands. Clearly, assuming that an individual has a utility value for *every and all recipes* (complete knowledge) is unfeasible. Moreover, in a group decision, the set of known alternatives (i.e. those for which the utility value is known) to each individual may differ, and the set of alternatives known to *all individuals* in the group may be small or empty.

Thus, in any realistic scenario, specially in group decision, assuming complete knowledge is too restrictive and we have to deal with incomplete knowledge. This means, in the Banquet Agreement scenario, that each individual may know a specific subset of recipes (and thus have utility value for them), but not the rest. In fact, some models of economics recognize this possibility, which is called *rational ignorance*. The notion of rational ignorance means that, in those situations where the cost of acquiring information is greater than the benefits to be derived from the information, it is rational to be ignorant. Thus, in the cooking domain, it is rational to be ignorant — otherwise the time and cost involved in finding all possible recipes, tasting them, and acquiring an individual utility value would be a (maybe pleasurable but) daunting task.

Our approach is based on considering the set of alternatives under discussion to be open ended —instead of being fixed beforehand by an external entity, the alternatives are introduced in the discussion by the individual agents. This

[1] The word *satisfice* was proposed by Herbert Simon in 1956 as an alternative desideratum for AI tasks, in contrast with the more classical word *maximize*; Simon championed the notion of *bounded rationality* as a more realistic approach to rationality that takes into account different kinds of cognitive limitations.

[2] We will consider group decisions as a form of group agreement. In our approach, negotiation, mediation, social choice, etc. are different kinds of processes whose aim is to reach agreements between 2 or more agents. In what follows, we will speak of group decision or agreement interchangeably.

process by which agents try to reach an agreement (and in which the set of alternatives is expanded) will be called *deliberative agreement*[3]. For instance, in the Banquet Agreement scenario one agent may know about one risotto recipe (e.g. mushroom risotto), but once another agent proposes as a candidate agreement another risotto recipe (e.g. fresh asparagus risotto), the set of what we call *public alternatives* is increased. Then, the first agent can deal with one more alternative, and although rational ignorance implies that the agent has no utility value for an unknown alternative, case-based reasoning can be used to estimate utility values of unknown public alternatives by such an agent. Therefore, CBR can be used to deal with rational ignorance (incomplete information) based on what is known by each agent. This capability allows us to deal with more realistic scenarios with an open-ended set of alternatives.

Moreover, we will show that CBR can also be used during deliberative agreement process to reach faster a group agreement. A CBR agent can make a case-based model of another agent using the proposals offered by that agent. In the Banquet Agreement scenario, for instance, if one agent proposes the alternative fresh asparagus risotto, another agent may safely infer that the agent would also like similar recipes, and use CBR to try to propose a preferred recipe (e.g. risotto with "fava" beans) that is also similar to recipes proposed by another agent. Although our approach is valid for n agents, in most of the paper we will focus on the 2 agent scenario for sake of expository clarity

In this paper, we will present an approach to group agreement in multiagent systems in which CBR is used in situations where rational ignorance applies. Case-based reasoning will be shown to deal with incomplete knowledge of individuals and, moreover, support satisficing behavior in combinatorial domains in multi-issue group agreement. The next section introduces the classical notions of single-issue and multi-issue collective decisions, while the rest of the paper presents our approach to open-ended group agreement in Section 3, and the CBR strategy to deal with group agreement and rational ignorance in Section 5. An experimental evaluation in the cooking domain is shown in Section 7, and the paper comes to an end with sections on related work and conclusions.

2 Background

Social choice theory [3,6,1] is a mathematical theory of collective decision making, which is concerned about how groups actually do make decisions, focusing on methods to aggregate individual preferences into a collective decision or choice. A social choice problem consists of a number of individuals that have preferences over a set of alternatives $D^i = x^1, \ldots, x^k$ on an issue X^i. A preference aggregation method, typically a voting method, aggregates the preferences over alternatives, ranks those alternatives in a global ordering, and determines the

[3] While classical social choice focuses on voting methods, current research in deliberative democracy (and deliberation in multimember bodies in general) show that in real life situations it is useful having a first stage of deliberation before the stage of voting on a collective outcome [4].

winning alternative for the issue (the collective decision). Voting methods include the majority rule, approval voting, Borda count, the Condorcet method, etc. Social choice theory studies the properties satisfied or not by different aggregation methods. However, they share some basic assumptions: agents are supposed to have perfect knowledge: they know all possible alternatives $D^i = x^1 \dots x^k$ on the issue at hand X^i and know which ones are preferred over the others.

When the decision is more complex, such decision is modeled as a set of issues, each one with a number of alternatives. Thus, multi-issue social choice consists of a set of issues $X = \{X^1, \dots, X^m\}$ and for each issue X^k there might be a set of alternatives x to choose from, in a domain D^k. The number of possible decisions is now much greater: all possible combinations in $\Omega = D^1 \times \dots \times D^m$. Assuming the issues are independent (i.e. preferences are separable) is a common simplification that allows issue-by-issue voting to achieve a group multi-issue decision. However, the individuals' preferences are not necessarily separable, since the issues may be interdependent, i.e. that an individual's preference for one issue, may depend on the alternative taken for another issue. For instance, in the Banquet Agreement scenario, a group has to decide on a two-course meal. Clearly, an individual's preference on the main course may depend on the alternative taken as a starter, e.g. having "arròs rossejat" as main course rules out having a rice salad as a starter, since both alternatives' main ingredient is rice.

Thus, in multi-issue social choice with dependences between issues, the issue-by-issue voting method may lead to suboptimal results. For instance, in the Banquet Agreement scenario, a subgroup constituting a majority of individuals may select "rice salad" as a starter while another sub group constitute a majority selecting "arròs rossejat" as main course even when no individual votes for having both rice salad and "arròs rossejat" together. The approach we will take is that multi-issue social choice problem is composed of the possible combinations in $D^1 \times \dots \times D^m$ and a set of constraints C over them in such a way that we define a set of valid combinations $\Omega \subset D^1 \times \dots \times D^m$.

A different but related approach to collective decision making is of that of *deliberation* in the study of deliberative democracy [2]. Deliberative democracy contends that collective decision processes should not just aggregate individual preferences but help shaping those individual preferences. Therefore, deliberative democracy encourages the individuals to deliberate about which alternative is to be preferred for an issue, in an open dialogue with one another, before voting. During the public discussion, the individuals may change their preferences since they can acquire new alternatives for the issues at hand, and other relevant information, such as the concerns of the other individuals. Thus, votes and preferences should emerge from processes of deliberation, since then individuals are able to make a more informed decision.

Given that group decisions are not necessarily limited to a process of choosing among given alternatives, but also a process of generating new alternatives (brainstorming), an argument in favor of discussing publicly before voting, is that the limitations due to bounded rationality might be alleviated, since deliberation can provide more creative outcomes [5]. In our approach, a process of deliberation is required for allowing the individual agents to introduce new alternatives to an issue —deliberation makes possible to take collective decisions within a context of rational ignorance (incomplete knowledge).

From the point of view of CBR, this approach continues the work on multiagent case-based reasoning that previously focused on classification tasks [9,7,8]. In this paper, multiagent CBR is used not to solve a problem (find a correct outcome) but reach an agreement (find a decision with high group welfare). Moreover, the CBR approach (together with deliberation democracy approach) are used to develop a more realistic framework for group choice that embrace openness and incompleteness of knowledge.

3 Open-Ended Deliberative Agreement

This section introduces the open deliberative agreement with rational ignorance (ODARI) framework, in which a group of agents deliberates on a set of issues and their alternatives in order to reach an agreement about the alternatives (rational ignorance). Although ODARI is defined for a group on n agents, it is easier to explain the case where there are 2 agents deliberating on an agreement, and we will use a set of 2 agents $\mathcal{A} = \{A_1, A_2\}$ in our exposition here. How CBR is used to deal in the ODARI framework is explained later in Section 5.

The ODARI framework is a model of group decisions in multiagent systems with the following properties:

Open-Endedness. The first feature in ODARI is that it is a multi-issue group decision problem $X = \{X^1, \ldots, X^m\}$ where each issue X^k is open-ended, i.e. the set of alternatives D^k is not fixed and known by all individual agents (rational ignorance).

Deliberation. A second feature is that new alternatives to issues can be introduced during a process of deliberation to reach an agreement.

Interrelated Issues. A third feature is that there is a set of constraints C that determine a subset of valid combinations $\Omega \subset D^1 \times \ldots \times D^m$

Time-sensitiveness. Finally, since the space of valid combinations Ω can be very large, we assume there is a finite time limit that precludes the exploration of all valid combinations in Ω during deliberative agreement process.

We contend that these four features makes the ODARI framework closer to realistic scenarios of group decisions in multiagent systems.

An issue is open-ended when the set of alternatives it may take increases monotonically over time. An agent A_i has a limited experience in each issue X^k, and knows a subset of all alternatives that may exist in the world, which we will denote as D_i^k. The *agreement space* of an agent A_i based on its initial knowledge of the world is $\Omega_i \subset D_i^1 \times \ldots \times D_i^m$, the set of combinations of the individually known alternatives that satisfy the constraints in C.

Thus, ODARI assumes that an alternative $x \in D_i^k$ may be a known alternative to an agent, but unknown to another, and every agent has a utility value (or preference) for each known alternative to him. In our CBR approach, discussed later in Section 5, this experience-based knowledge will be determined by the individual case base of each agent. Moreover, on the Banquet Agreement scenario, the case base is composed of the cooking recipes known to an agent.

The preferences of an agent A_i for each issue X^k will be expressed by a utility function: $U_i^k : D_i^k \rightarrow [0,1]$. However, A_i has no utility value for unknown alternatives (those not in D_i^k). The domain of alternatives for an issue X^k given a group of agents $A_1, \ldots, A_n$ is $D^k = \bigcup_{i=1 \ldots n} D_i^k$, where every agent knows a subset D_i^k. The space of possible combinations is $D^1 \times \ldots \times D^m$, but given a set of constraints C, the space of possible agreements is the subset $\Omega \subset D^1 \times \ldots \times D^m$ of combinations satisfying C. Individual agents have no immediate access to this larger space, since they are working only in a subspace $\Omega_i \subset \Omega$.

During the deliberation process the agents may come to know new alternatives for the issues at hand as they are included in proposals made by other agents. If the agents use this new alternatives the space of possible agreements that may be proposed increases accordingly. Moreover, since this alternatives are presented in a public space (all participating agents have access to all information flow during deliberation), all agents become aware of the new alternatives.

Let l Π^k be the alternatives for the issue X^k made public at some moment in time during deliberation; the agreement space of public alternatives is then $\Omega^P = \Pi^1 \times \ldots \times \Pi^m$. In this situation, Ω^P is the "common knowledge" of the agent group, but the agents still need some way to integrate the unknown alternatives in their preference structure, i.e. the set of "new" alternatives $N_i^k = \Pi^k - D_i^k$ for every issue X^k. Section 5 explains how CBR is used to deal with these new alternatives. Notice, however, that an agent using public alternatives can now generate a larger set of proposals: for each issue X^k the agent can choose from a larger set of alternatives, namely $D_i^k \cup N_i^k$. Therefore, the set of agreements that can be proposed by an agent is now $\Omega_i^P \subset D_i^1 \cup N_i^1 \times \ldots \times D_i^m \cup N_i^m$ (the set of combinations satisfying the set of constraints C).

4 Deliberation Process

The deliberation process among agents is an interaction protocol on which agents propose and accept (or reject) possible agreements of the form $(x^1, \ldots, x^m)$, i.e. assigning one alternative to each of the m issues involved in the deliberative agreement task. For instance, in the Banquet Agreement scenario, with two main courses, an example proposal may be (Xató, arròs-rossejat), where the dishes specify their ingredients: Xató is a endive salad with cot, olives, etc. and a hot sauce, and arròs-rossejat is a fishermen's fried noodles dish with aioli. A *valid* proposal is a combination of alternatives that satisfies the set of constraints C.

In this section we will present the interaction protocol DAP2 for 2 agents participating in the deliberation; extending DAP2 to n agents is possible but the 2 agent scenario is easier to understand. DAP2 allows a group of two agents $\mathcal{A} = \{A_1, A_2\}$ making proposals until one agent accepts a proposal made by the other agent (which becomes the agreement); each proposal is made in a new round of the protocol. A maximum number of rounds M establishes a deadline for reaching an agreement.

During deliberation the agents interchange the following types of messages:

– *propose*(A_i, ω, t): where an agent A_i proposes a valid combination of alternatives ω at the round t; moreover ω has to be a new combination (i.e. ω has never been proposed before).
– *accept*(A_i, ω, t): where an agent A_i accepts the valid combination of alternative ω at the round t; the proposal ω has been proposed previously by the other agent A_j but need not be the last proposal of A_j.

The DAP2 protocol starts at the round $t = 0$ with the token randomly assigned to an agent:

1. The agent A_i who has the token can act in different ways:
   – A_i accepts a previous proposal ω of the other agent A_j sending it message *accept*(A_i, ω, t) and then the protocol terminates with agreement ω.
   – A_i makes a new proposal ω sending *propose*(A_i, ω, t) to the agent A_j; if A_i is unable to find a new proposal an *abstain*(A_i, t) is sent. The token passes to the other agent A_j, and the protocol moves to the step 2.
2. If the deadline M is reached or none of the agents made a proposal in the previous two rounds, the protocol terminates without an agreement. Otherwise a a new round $t + 1$ starts and the protocol moves to the step 1.

For using the DAP2 protocol, agents just need a decision policy that allows them to decide how to act in the protocol (when to accept, and when to make new proposals). We present such policies in the next section.

5 CBR Agents

This section presents how case-based reasoning can address, in a natural way, the challenges associated with a more realistic scenario for collective decision (essentially knowledge incompleteness) in the context of deliberative agreement, where the deliberation process allows agents to acquire new and unknown alternatives. Let use define some auxiliary notions before presenting CBR-based decision policies to be used with the DAP2 protocol.

Open Minded Strategy. We will define two different strategies, open-minded and narrow-minded reasoning strategies. An *open-minded* agent will consider acceptable agreements containing alternatives that are new and unknown for that agent. The *narrow-minded* reasoning strategy, on the other hand, will not consider acceptable any agreement containing unknown alternatives. In the Banquet Agreement scenario, for instance, an agent may like paella but does not know arròs rossejat: the narrow-minded would not accept any agreement with arròs rossejat (even when it is very similar to paella), while the open-minded agent will consider the similarity and may accept agreements with arròs rossejat. Clearly, the open-minded strategy allows a larger space of possible agreements, while the narrow-minded strategy constrains the space of possible agreements to those containing alternatives known to all agents. For instance, between two agents, a narrow-minded strategy has an agreement space $\Omega_1 \cap \Omega_2$, while an open-minded strategy has an agreement space $\Omega_1^P \cup \Omega_2^P$.

Furthermore, an open-minded agent can also use new unknown alternatives when proposing an agreement. For instance, an agent that proposed (green-salad, paella) has later received a proposal containing arròs rossejat; since this means that the other agent likes arròs rossejat and it is similar to paella, the agent can now make a new proposal with more chances to succeed: (green-salad, arròs rossejat). Thus, as we will see in the experimental evaluation section, open-mindedness helps in reaching an agreement faster (while maintaining high levels of satisfaction) by allowing to propose agreements closer to the preferences of the other agent. Narrow-mindedness, on the other hand, risks running out of time without finding a common agreement.

Multi-issue Deliberation. Generating and evaluating proposals of agreement is much more complex in multi-issue group agreement than in single-issue group agreement. Moreover, constraints over the combination of alternatives make infeasible estimating the utility of alternatives in isolation; thus, utility will be measured by a function over possible agreements $U_i : \Omega \rightarrow [0, 1]$. Therefore the CBR agents have to reason about valid combinations of alternatives, but the combinatorial nature of this process together with time-sensitivity makes impossible an approach based on maximization: an approach based on satisficing is needed, where an agent A_i accepts an agreement ω if it is satisfactory to A_i. Later in Section 6 we will formalize this idea with the notion of the aspiration level of an agent, such that when a proposed agreement's surpasses the aspiration level the agent accepts that agreement.

In general, the agents should be able to evaluate any agreement in Ω; thus, for each issue X^k, an agent requires a way of evaluating the utility degree of every alternative in D^k. In the classical approach studied in social choice, since the alternatives are just a set of identifiers, without an intrinsic structure, all knowledge resides in the utility function. However, in ODARI, there are unknown alternatives for any agent A_i —i.e. A_i does not have utility degree for some of alternatives that an issue may take. For this reason, we assume that each CBR agent A_i has a similarity measure over the alternatives of an issue $s_i^k : D^k \times D^k \rightarrow [0, 1]$. This new assumption involves access to some characterization of the alternatives, and the similarity measure works upon that characterization. Consequently, we assume (1) that the alternatives have some characterization or description in some language, and (2) that a similarity measure can be defined on the space of descriptions of alternatives. Clearly, the similarity among alternatives is domain specific; Section 7 presents the similarity we have used in the Banquet Agreement scenario.

Issue Case Base. In our approach, a CBR agent A_i has a case base C_i^k for each issue X^k, where a case $c \in C_i^k$ is a pair $c = \langle x, u \rangle$ such that x is a known alternative to A_i for an issue $X^k$and u is the utility degree of x to A_i, i.e. $c.u = U_i^k(x)$.

Next, using these issue case bases, we will be able to define a function $\overline{U}_i^k$ that estimates the utility of unknown alternatives for each issue. Using $\overline{U}_i^k$ an agent A_i is able to evaluate each possible agreement in Ω, and thus, is able both to accept a proposal, even if it has unknown alternatives, and to use unknown alternatives in generating proposals.

Utility of an Unknown Alternative. The similarity measure allows us to estimate the utility of an unknown alternative x^r of an issue X^r, by using a k-nearest neighbor method which is calculated as the weighted sum of utilities of the k most similar cases in C_i^r:

$$\overline{U}_i^r(x^r) = \frac{\sum_{c \in \mathbb{K}} s_i^r(c.x, x^r) \times c.u}{\sum_{c \in \mathbb{K}} s_i^r(c.x, x^r)}$$

where $\mathbb{K}$ is the set of k most similar cases.

Utility of an Alternative. An agent A_i either knows the utility of an alternative or can estimate it for unknown alternatives.

$$\mathbf{U}_i^k(x) = \begin{cases} U_i^k(x) \text{ if} \, x \in D_i^k \\ \overline{U}_i^k(x) \text{ otherwise} \end{cases}$$

Utility of an agreement. The utility of an agreement ω (a valid combination of alternatives for m issues) is $U_i(\omega) = \frac{1}{m} \sum_{1 \le k \le m} \mathbf{U}_i^k(x^k)$.

Modeling Agent Preferences. An agent does not have any a priori information about the other agent's preferences. During the deliberation process, an agent does not know, for any ω in the agreement space that has never been proposed, whether ω could be satisfactory to the other agent. However, an agent A_i may exploit the information that emerges during the deliberation, in order to acquire clues about the other agents' preferences. Specifically, when an agent A_i has the token, A_i is aware that the proposals made by A_j up to that point are satisfactory to A_j, because as soon as A_j has proposed an agreement, he commits to accept that agreement. Additionally, A_i is aware that no agreement it has proposed up to that point has been satisfactory to A_j, otherwise A_j would have accepted one of them.

For this reason, when an agent A_i makes a proposal that is similar to proposals made by the agent A_j in previous rounds, A_i increases the likelihood of it being accepted by A_j, since the proposals made by A_j are satisfactory to A_j.

Moreover, the agent A_i may exploit the information about the proposals it made that were rejected by A_j. Here, the intuition is that those proposals give some information to A_i of what kind of proposal are *not* satisfactory to A_j. This way, the more similar a possible agreement to the proposals made by A_i, the more likelihood it will not be satisfactory to A_j.

Thus, an agent A_i may use the similarity between proposals to estimate the proposals' likelihood of being satisfactory (or not) to another agent A_j, based in the proposals previously made by both agents. In this sense, the set of proposals have been made during a deliberation process are treated as a "case base" that models A_j's preferences.

Proposal Case Base. Every agent A_i in ODARI has a proposal case base C_i^P such that, for each proposal ω made by any agent a in the deliberation (including itself), there is a case $\langle \omega, a \rangle \in C_i^P$.

Proposal Similarity. An agent A_i has a similarity function $S_i : \Omega \times \Omega \rightarrow [0,1]$, which expresses the similarity between two proposals (two valid combination of alternatives):

$$S_i(\{x_1^1, \ldots, x_1^m\}, \{x_2^1, \ldots, x_2^m\}) = \frac{1}{m} \sum_{1 \leq k \leq m} s_i^k(x_1^k, x_2^k)$$

Let $M_i = \{\omega \in C_i^P | c.a = A_i\}$ be the set of proposals made by A_i to the agent A_j. Notice that proposals in M_i have not been accepted by A_j, since those proposals are not satisfactory to A_j; thus, other proposals similar to M_i are likely to be unsatisfactory to A_j. Let $R_i^j = \{\omega \in C_i^P | c.a = A_j\}$ be the proposals received by A_i from A_j, i.e. the set of proposals that are known to be satisfactory to A_j. Therefore, proposals similar to those in R_i^j are more likely to be satisfactory A_j. Thus, the likelihood of a proposal to be accepted by A_j increases if it is similar to proposals in R_i^j and decreases if it is similar to proposal in M_i.

Proposal Acceptance Likelihood. Following these two heuristic criteria, based in the proposals made by the agents, we will define a function E_i allowing A_i to estimate the likelihood of a new agreement proposal ω' being accepted by another agent as follows:

$$E_i(A_j, \omega') = \frac{1}{2}\Big(\max_{\omega \in R_i^j} S_i(\omega', \omega)\Big) + \frac{1}{2}\Big(\min_{\omega \in M_i} (1 - S_i(\omega', \omega))\Big)$$

6 Proposal Generation and Aceptance

A CBR agent A_i, in the ODARI framework, will propose agreements taking into account (1) the proposals' utility degree to A_i (possibly estimated with a similarity) and (2) the likelihood of the proposals being accepted by the other agent A_j. If A_i follows the open-minded strategy, the agreement proposal has to be selected from the space of possible agreements Ω_i^P. Since a proposal cannot be repeated, however, the space of possible new proposals is $\Omega_i' = \Omega_i^P - P$, where P is the set of agreements already proposed. We define first how the agreement to be proposed is selected by an agent.

Proposal Selection Heuristic. The candidate agreements Ω_i' will be evaluated by a heuristic $H_i : \Omega_i' \rightarrow [0,1]$ that combines the utility for the proposing agent and the likelihood to be accepted by the other agent, as follows:

$H_i(\omega) = (1 - \alpha) \times U_i(\omega) + \alpha \times E_i(A_j, \omega)$.

where $\alpha \in [0,1]$ is the weight given to the acceptance likelihood estimated for the other agent. The selection of the agreement to be proposed is simply $\mathbf{H}_i(\Omega_i') = argmax_{\omega \in \Omega_i'} H_i(\omega)$ (i.e. the agreement with highest H_i value).

Now we turn to the issue of an agent deciding to accept or not a proposed agreement, based on the notion of aspiration level. The main idea is twofold: (1)

an agent A_i accepts an agreement ω when its utility $U_i(\omega)$ is above its aspiration level ∂_i, and (2) the aspiration level ∂_i decreases during the deliberation process.

Aspiration Level. At any moment in time, an agent A_i has proposed M_i agreements. Since all agreements in M_i are satisfactory to A_i, let us take the one with minimum utility $\omega_i^* = argmin_{\omega \in M_i} U_i(\omega)$; therefore $\partial_i = U_i(\omega_i^*)$ — i.e. that the aspiration level is $U_i(\omega_i^*)$, since agent A_i is already proposing an agreement with that utility degree. Any agreement proposal ω that agent A_i receives whose utility for A_i is equal or greater than the aspiration level should be accepted by A_i (since A_i is already proposing agreement with that degree of utility). Notice that ∂_i will decrease monotonically with time, since the set M_i increases with new proposed agreements and no proposed agreement can be withdrawn according to the protocol DAP2.

Decision Policy. Whenever an agent A_i owns the token in protocol DAP2, the decision policy of an agent A_i decides either to accept an agreement proposed by A_j or to propose a new agreement. Now, agent A_i has an aspiration level ∂_i and a new agreement to propose, namely $\mathbf{H}_i(\Omega_i')$. Let $\omega^k = argmax_{\omega \in R_i(j)} U_i(\omega)$ be the best proposal received by A_i from A_j, then the decision policy is:

1. if $U_i(\omega^k) \geq min(\partial_i, U(\mathbf{H}_i(\Omega_i')))$ then Accept ω_k
2. otherwise Propose $\mathbf{H}_i(\Omega_i')$.

i.e. an agent A_i will accept the best proposal received if it has an utility better or equal than the aspiration level or the utility of the next agreement that A_i intends to propose next; otherwise A_i proposes that agreement.

7 Banquet Agreement Scenario

In this section, we experimentally evaluate our approach on the two-issues Banquet Agreement scenario, based on the recipes database of the 2010 Computer Cooking Contest (CCC). Specifically, these experiments involve two agents A_1 and A_2, that will engage in the deliberative process to reach an agreement on a two-course meal, i.e. agreeing on a specific recipe for the starter (issue one) and for the main course (issue two). The agreement must satisfy the following constraint: no main ingredient may be used in both recipes[4]. The experimental database R consists of 600 recipes (with 380 different ingredients) from the CCC, randomly split into two sets: R^1 and R^2 of starters and main courses.

The domain-specific similarity functions among alternatives of an issue is here a similarity s_1 over recipes in R^1 and s_2 over recipes in R^2. Both s_1 and s_2 are based on the Jaccard similarity

$$s(x_1, x_2) = \frac{|Ing(x_1) \cap Ing(x_2)\ |}{|Ing(x_1) \cup Ing(x_2)\ |}$$

where $Ing(x)$ is the set of main ingredients in recipe x.

[4] Main ingredients like rice or potato cannot be repeated, but secondary ingredients like oil or salt can be used in both course's recipes.

The agents' preferences are built by the experiment designer in the following way. Each agent has a profiler-creation function that randomly assigns a utility value in $[0, 1]$ to each ingredient at each run of the experiment. From the utility of ingredients the utility of a recipe x is computed as the normalized sum of the utility of the ingredients in x. The recipes known to a agent constitute a case base where there is a case $\langle x, u \rangle$ for each known recipe x and its utility u. Notice that (1) the agent does not know the utility of ingredients, only the global utility of recipes, so it is unable to ascertain the utility of unknown recipes from their ingredients, and (2) people usually have clear utilities for courses rather than ingredients (although some ingredients may be a no-no), using the hidden profiler-creation function is just a convenient way to generate a large number of profiles for experimentation.

Moreover, random profiles constitute a rather worst case scenario, where any commonality of tastes among two artificial agents might be much lower than other scenarios where participants may share some tastes. Finally, notice that an agent only knows the utility for the recipes in its case base and is ignorant of other recipes contained in other case bases (except for the recipes both agents know, which will be a smaller subset in the experiments).

7.1 Experiments

In our experiments, we have set the maximum number of rounds for DAP2 to $M = 150$, unless specified otherwise, and we have used $k = 5$ for the k-nearest neighbor method $\overline{U}_i(x)$ to estimate the utility degree of unknown alternatives. Given the maximum number of M rounds, if this maximum is reached the protocol terminates without agreement. The experiments, unless specified otherwise, assign 400 recipes to each agent, 200 of which are shared by both agents.

In order to evaluate the quality of the agreements reached by the agents, we need to define a function assessing the degree of "goodness" of these agreements for the group. These functions are called *social welfare functions*, and there are several that are defined in the literature, depending on the criteria of what does it mean for a agreement to be good for the group. The utilitarian welfare $W_U(\omega) = \frac{1}{2}(U_1(\omega)+U_2(\omega))$ measures the overall utility as the average of individual utilities; this takes into account the total but not the inequality of utilities —i.e. a welfare of $W_U(\omega) = 0.5$ may be achieved with individual utilities 0.5 and 0.5 or with 0.9 and 0.1. The egalitarian welfare of an agreement $W_E(\omega) = min_{A_i \in \mathcal{A}} U_i(\omega)$, takes into account the level of inequality by defining welfare as the minimum utility of the two agents. Since this welfare is very strict, we will use a combination of both called the group welfare $W_G(\omega) = \frac{1}{2}(W_U(\omega)+W_E(\omega))$. In the experiments, group welfare is computed using the ingredient-based utility (hidden to the agents).

The experiments are made for different values of $\alpha \in [0, 1]$ —recall that α is the weight used in the Proposal Selection Heuristic. Thus, when $\alpha = 0$ the agent behaves as an egoist, since all proposed agreements take only into account its individual utility. The higher the α the less egoist is an agent, since it will propose agreements that have less utility for itself but are more similar to the

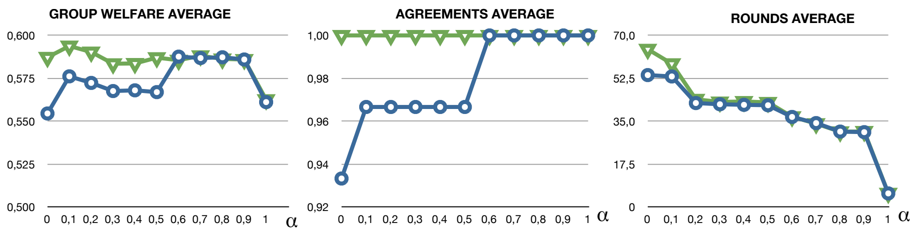


Fig. 1. Group welfare average (left), percentage of times reaching an agreement (center), and number of proposals exchanged (right) for different α values when there is a time limit of 150 rounds (◯) or no limit (▽)

proposals of the other agent. Consequently, the higher the α the faster the aspiration level of the agent decreases during the deliberation process.

Figure 1 shows the open-minded agent relationship with time-sensitiveness. In this experiment, the abscissae are different values of α (for one agent) while the other agent has a random value $\alpha \in [0, 1]$ at each run. The ordinates plot the averages of group welfare, the percent of times an agreement is reached, and the number of rounds needed to reach an agreement. If there is no time limit, the deliberation can spend a lot of time examining a large number of proposals. However, when there is a time limit, being egoist (having values of α close to 0) is a bad option (as shown in Fig. 1): (1) when α increases then the percentage of times in which an agreement is reached also increases (Fig. 1 center), which explains (2) when α is closer to zero group welfare is lower (Fig. 1 left). Finally, the number of exchanged proposals needed to reach an agreement decreases when α increases (Fig. 1 right). Moreover, this last plot shows that the E_i function is useful in estimating the proposals that the other agent may consider satisfactory. For comparison, the centralized method[5] gives a welfare average of $W_G = 0,6735$.

The effect of egoism ($\alpha = 0$) vs. benevolence ($\alpha = 1$) is shown in Fig. 2 for open-minded and narrow-minded agents with a limit of 150 rounds. In this experiment, as well as the following ones, both agents have the same α shown in the abscissa, and both agents are either open-minded or narrow-minded. First, we see in Fig. 2 that open-minded agents achieve agreements with higher welfare values (left), more agreements (center) and with less rounds of deliberation (right) than the close-minded agents. When α is low, both agents are egoistic and thus their aspiration level decrease very slowly, which results in a lower

[5] The centralized aggregation method has complete knowledge. Specifically, this method receives all the cases from the two agents, receives the designer-level ingredient-based evaluation function of each agent and computes the ingredient-based utility for each recipe for each agent. Then performs exhaustive search to find the pair of courses that maximize the group welfare function W_G. This method has complete knowledge of the utility of all alternatives in the experiment and unlimited time to search all combinations, and gives an estimate of the best possible agreement in ideal conditions.

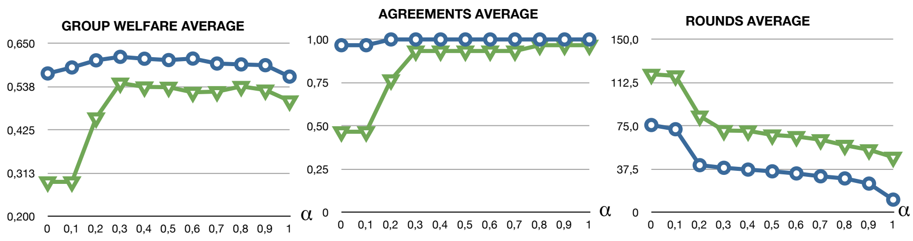


Fig. 2. Group welfare average (left), percentage of times reaching an agreement (center), and number of proposals exchanged (right), depending on the egoist ($\alpha = 0$) vs benevolent ($\alpha = 1$) spectrum for open-minded (◯) and narrow-minded strategies (∇)

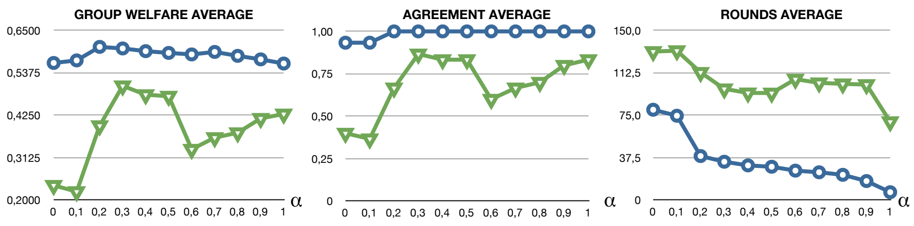


Fig. 3. Group welfare average (left), percentage of times reaching an agreement (center), and number of proposals exchanged (right) for open-minded (◯) and narrow-minded strategies (∇) agents with a 16% of shared recipes

number of agreements reached before the time limit (specially for narrow-minded). When α is higher then the narrow-minded agents improve, but they are still worse than the open-minded ones.

Another parameter affecting the outcome is the size of the case bases for an issue k and the size of the shared recipes. The experiment shown in Fig. 3 has two agents with 150 recipes each agent, a 16% of shared recipes and $M = 150$ rounds, showing the difference between open-minded and narrow-minded agents. In this setting, with smaller case bases and a smaller set of shared recipes, reaching an agreement is more difficult for the narrow-minded agents: group welfare and number of agreements decrease. Moreover, the number of rounds narrow-minded agents need to reach an agreement also increases. The open-minded agents, however, even now that the size of shared alternatives is smaller, keep a similar performance in group welfare, number of agreements, and number of rounds as before.

8 Conclusion

Case-based reasoning is a methodology that allows to address AI under incomplete knowledge by exploiting dynamically available knowledge in a more flexible way. We have addressed here the issues involved in group decisions (modeled here as agreements). Classical mathematical models in social choice assume conditions like perfect knowledge. Weakening this assumption requires a new approach, and we have shown that CBR can deal with incomplete knowledge by exploiting the dynamic exchange of information during deliberation.

Essentially, the classical approach encodes all knowledge in a utility function over known alternatives. We have shown that incorporating a similarity function over the space of possible descriptions of alternatives enables a CBR agent to cope with unknown alternatives (and thus rational ignorance). We have also included a process of deliberation, previous to the group decision itself, that allows to introduce new, unknown alternatives in an incremental way. This approach has been evaluated on the CCC dataset but could be applied to other domains where a suitable similarity over alternatives can be designed.

Future work will address group decision for more than two agents (multilateral agreement), which is a problem whose high complexity is well known. The main idea will be to use the deliberation process to increase the knowledge available to the agents, and then vote. That is to say, the goal is not to achieve an agreement by consensus, since the complexity of the problem would require a very long process. The goal will be that the deliberation process helps the agents gaining new alternatives and acquiring a better model the preferences of other agents before the final decision-making step of voting.

Acknowledgements. This research was partially supported by projects Agreement Technologies (CSD2007-0022) and Next-CBR (TIN2009-13692-C03-01).

References

1. Arrow, K.J.: Social Choice and Individual Values. John Wiley, Chichester (1951)
2. Atkinson, K., Bench-Capon, T., Mcburney, P.: Parmenides: Facilitating deliberation in democracies. Artificial Intelligence and Law 14(4), 261–275 (2006)
3. Chevaleyre, Y., Endriss, U., Lang, J., Maudet, N.: A short introduction to computational social choice. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 51–69. Springer, Heidelberg (2007)
4. Dryzek, J.S., List, C.: Social choice theory and deliberative democracy: a reconciliation. British Journal of Political Science 33, 1–28 (2003)
5. Elster, J.: Deliberative Democracy (Cambridge Studies in the Theory of Democracy), pp. 11–18. Cambridge University Press, Cambridge (1998)
6. Lang, J.: Logical preference representation and combinatorial vote. Annals of Mathematics and Artificial Intelligence 42, 37–71 (2004)
7. Ontañón, S., Plaza, E.: Case-based learning from proactive communication. In: Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 999–1004. IJCAI Press (2007)
8. Ontañón, S., Plaza, E.: Learning and joint deliberation through argumentation in multi-agent systems. In: Proc. AAMAS 2007, pp. 971–978. ACM, New York (2007)
9. Plaza, E., Ontañón, S.: Ensemble case-based reasoning: Collaboration policies for multiagent cooperative CBR. In: Watson, I., Yang, Q. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 437–451. Springer, Heidelberg (2001)

Amalgam-Based Reuse for Multiagent Case-Based Reasoning

Sergio Manzano[1,2], Santiago Ontañón[1], and Enric Plaza[1]

[1] IIIA-CSIC, Artificial Intelligence Research Institute (Spanish Scientific Research Council), 08193 Bellaterra, Catalonia, Spain
[2] Universitat Autonòma Barcelona, Bellaterra, 08193 Catalonia, Spain
{sergio,santi,enric}@iiia.csic.es

Abstract. Different agents in a multiagent system might have different solution quality or preference criteria. Therefore, when solving problems collaboratively using CBR, case reuse must take this into account. In this paper we propose ABARC, a model for multiagent case reuse, which divides case reuse in two stages: *individual reuse*, where agents generate full solutions internally, and *multiagent reuse*, where agents engage in a deliberation process in order to reach an agreement on a final solution. Specifically, ABARC is based on the idea of *amalgam*, which is a way to generate solutions by combining multiple solutions into one. We illustrate ABARC in the domain of interior room design.

1 Introduction

The multiplicity of experience sources requires to investigate new approaches to reuse and adapt them in the cycle of case-based problem-solving. This paper focuses on the collaboration of multiple CBR systems dealing with complex structure solutions. Each individual CBR system may have different biases on the solutions it prefers — bias may be due to history, preferences, priorities, etc. Therefore, each individual may come to a different solution for the same problem, and might consider a given solution as more or less satisfactory. A collective agreed-upon complex solution cannot be achieved by simple aggregation methods like voting, and requires a richer deliberation process to reach a collective solution. This deliberation process is in fact a joint reuse process that combines both the experiences and the bias of these individuals.

As a motivating example, consider two people who need to agree on the configuration of an office for having good working conditions. Naturally, each person will have their own preferences with respect to what constitutes "good working conditions," or with respect to which pieces of furniture are more desirable, etc. It is clear that proposing room configurations and voting over them is not a good approach, since the number of possible room configurations might be very high. Moreover, notice that biases such as preferences are hard to express explicitly, and thus cannot be fully communicated. For that reason, a centralized approach where each person states her preferences and then only one of them tries to find a room configuration is not viable either. In this paper we will study how can these kind of problems be addressed by presenting ABARC, a model of multiagent case reuse, which allows two agents to collaboratively reach an agreement over a structured solution.

Previous approaches to distributed CBR have mainly focused on distributed retrieval [11, 13, 9, 8], while the few approaches dealing with distributed reuse, do so in classification domains [12] by means of voting approaches. The main contribution of this paper is an approach for distributed reuse in domains with complex structured solutions.

There are two key ideas behind the ABARC (Amalgam-based Agreement for Multiagent Reuse of Cases) approach for multiagent CBR. The first is that reuse can be divided in two different stages: *individual reuse* and *multiagent reuse*. During individual reuse, each agent generates a full solution individually according to its own biases or preferences. During multiagent reuse, agents reach an agreement on a solution that is satisfactory for both agents.

The second idea is that the solutions proposed by the other agent can be used as an indication of its preferences. By combining elements from the solutions proposed by the two agents, agents could generate solutions which are satisfactory to both of them. A way to compute such combinations is through *amalgams* [10], a formal operation that generates new solutions by combining as many elements from two given solutions as possible.

The remainder of this paper is organized as follows. First, in Section 2 we present some background on generalization spaces, which is the knowledge representation formalism used in this paper. Section 3 formally introduces the idea of an amalgam. Then, ABARC is presented in Section 4. We analyze the behavior of ABARC in the task of room design in Section 5. The paper closes with related work and conclusions.

2 Background

In this paper we will make the assumption that solutions in cases are terms in some language $\mathcal{L}$, and that there exists a *subsumption* relation among terms.

We say that a term ψ_1 subsumes another term ψ_2 ($\psi_1 \sqsubseteq \psi_2$) when ψ_1 is more general (or equal) than ψ_2[1]. The subsumption relation induces a partial order in the terms in a language $\mathcal{L}$, thus, the pair $\langle \mathcal{L}, \sqsubseteq \rangle$ is a *poset* (partially ordered set) for a given set of terms $\mathcal{L}$; additionally, we assume that $\mathcal{L}$ contains the infimum element $\perp$ (or "any"), and the supremum element $\top$ (or "none") with respect to the subsumption order. In the rest of this paper we will call a pair $\langle \mathcal{L}, \sqsubseteq \rangle$ a *generalization space*.

Given the subsumption relation, for any two terms ψ_1 and ψ_2 we can define their *unification*, ($\psi_1 \sqcup \psi_2$), which is the *most general specialization* of two given terms:

$$\psi_1 \sqcup \psi_2 = \psi : (\psi_1 \sqsubseteq \psi \ \wedge \ \psi_2 \sqsubseteq \psi) \wedge (\nexists \psi' \sqsubset \psi : \ \psi_1 \sqsubseteq \psi' \ \wedge \ \psi_2 \sqsubseteq \psi')$$

[1] In machine learning terms, $A \sqsubseteq B$ means that A is more general than B, while in description logics it has the opposite meaning, since it is seen as "set inclusion" of their interpretations.

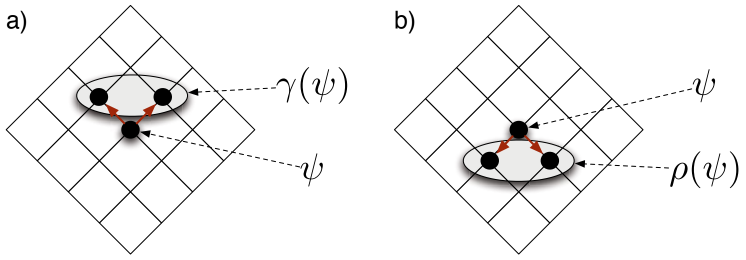


Fig. 1. A generalization refinement operator γ, and a specialization operator ρ

That is to say, the unifier's content is the addition of the content of the two original terms. However, not every pair of terms may be unified: if two terms have contradictory information then they have no unifier $\psi_1 \sqcup \psi_2 = \top$ —which is equivalent to say that their unifier is "none".

The dual operation to unification is that of *anti-unification*, that is defined as the *least general generalization* of two terms, representing the most specific term that subsumes both. If two terms have nothing in common, then $\psi_1 \sqcap \psi_2 = \bot$. Thus, anti-unification encapsulates in a single description *all* that is shared by two given terms, and is defined as follows:

$$\psi_1 \sqcap \psi_2 = \psi : (\psi \sqsubseteq \psi_1 \ \wedge \ \psi \sqsubseteq \psi_2) \wedge (\nexists \psi' \sqsupset \psi : \ \psi' \sqsubseteq \psi_1 \ \wedge \ \psi' \sqsubseteq \psi_2)$$

Notice that both anti-unification and unification might not be unique. Let us now summarize the basic notions of *refinement operator* over partially ordered sets and introduce the concepts relevant for this paper (see [6] for a more in-depth analysis of refinement operators). Refinement operators are defined as follows:

Definition 1. *A* downward refinement operator ρ *over a partially-ordered set* $(\mathcal{L}, \sqsubseteq)$ *is a function such that* $\forall \psi \in \mathcal{L} | \rho(\psi) \subseteq \{\psi' \in \mathcal{L} | \psi \sqsubseteq \psi'\}$.

Definition 2. *An* upward refinement operator γ *over a partially-ordered set* $(\mathcal{L}, \sqsubseteq)$ *is a function such that* $\forall \psi \in \mathcal{L} | \rho(\psi) \subseteq \{\psi' \in \mathcal{L} | \psi' \sqsubseteq \psi\}$.

In other words, upward refinement operators generate elements of $\mathcal{L}$ which are more general, whereas downward refinement operators generate elements of $\mathcal{L}$ which are more specific, as illustrated by Figure 1. Typically, the symbol γ is used to symbolize upward refinement operators, and ρ to symbolize either a downward refinement operator, or a refinement operator in general.

Refinement operators can be used to navigate the space of terms using search strategies, and are widely used in Inductive Logic Programming [7]. For instance, if we have a term representing "a German minivan", a generalization refinement operator would return generalizations like "a European minivan", or "a German vehicle". If we apply the generalization operator again to "a European minivan", we can get terms like "a minivan", or "a European vehicle". A specialization refinement operator would perform the opposite task, and given a term like "a German minivan", would return more specific terms like "a Mercedes minivan", or "a red German minivan".

3 Amalgams

The notion of *amalgam* can be conceived of as a generalization of the notion of unification over terms. The unification of two terms (or descriptions) is a new term, the unifier, which contains all the information in these two terms. Thus, if a term ϕ is a unifier of two other terms ($\phi = \psi_a \sqcup \psi_b$), then all that is true for one of these terms is also true for ϕ. For instance, if ψ_a describes "a red vehicle" and ψ_b describes "a German minivan" then their unification ϕ is the description "a red German minivan." Two terms are not unifiable when they possess contradictory information; for instance "a red French vehicle" is not unifiable with "a red German minivan" since being French and German at the same time is not possible for vehicles. The strict definition of unification means that any two descriptions with only one item with contradictory information cannot be unified. Now, imagine a scenario where two such descriptions have a large part of complementary information, which a CBR system would be interested in reusing: unification is not useful.

An *amalgam* of two terms (or descriptions) is a new term that contains parts from these two terms. For instance, an amalgam of "a red French vehicle" and "a German minivan" is "a red German minivan"; clearly there are always multiple possibilities for amalgams, since "a red French minivan" is another example of amalgam. In [10] we formally defined the notion of amalgam, and specifically studied the most specific amalgams of two terms. For the purposes of this paper, the following, more generic definition of amalgam suffices.

Definition 3. (Amalgam) *The set of* amalgams *of two terms ψ_a and ψ_b is the set of terms such that:*

$$\psi_a \curlyvee \psi_b = \{\phi \in \mathcal{L} - \{\top\} | \exists \phi_a, \phi_b \in \mathcal{L} | \phi_a \sqsubseteq \psi_a \ \wedge \ \phi_b \sqsubseteq \psi_b \ \wedge \ \phi = \phi_a \sqcup \phi_b\}$$

Thus, an amalgam of two terms ψ_a and ψ_b is a term that has been formed by unifying two terms ϕ_a and ϕ_b such that $\phi_a \sqsubseteq \psi_a$ and $\phi_b \sqsubseteq \psi_b$ —i.e. an amalgam is a term resulting from combining some of the information in ψ_a with some of the information from ψ_b, as illustrated in Figure 2. Formally, $\psi_a \curlyvee \psi_b$ denotes the set of all possible amalgams; however, whenever it does not lead to confusion, we will use $\psi_a \curlyvee \psi_b$ to denote one specific amalgam of ψ_a and ψ_b.

The terms ϕ_a and ϕ_b are called the *transfer* terms of an amalgam $\psi_a \curlyvee \psi_b$. ϕ_a represents all the information from ψ_a which is *transferred* to the amalgam, and ϕ_b is all the information from ψ_b which is transferred into the amalgam. This idea of transfer is akin to the idea of *transferring* knowledge from the source domain to the target domain in computational analogy [3].

Intuitively, an amalgam is *complete* when all which can be transferred from both terms into the amalgam has been transferred, i.e. if we wanted to transfer more information, ϕ_a and ϕ_b would not have a unifier.

Definition 4. (Complete Amalgam) *An amalgam $\phi = \psi_a \curlyvee \psi_b$ with transfers ϕ_a and ϕ_b is* complete *when*

$$\forall \phi'_a, \phi'_b | \phi_a \sqsubset \phi'_a \sqsubset \psi_a \wedge \phi_b \sqsubset \phi'_b \sqsubset \psi_b \Rightarrow \phi'_a \sqcup \phi'_b = \top$$

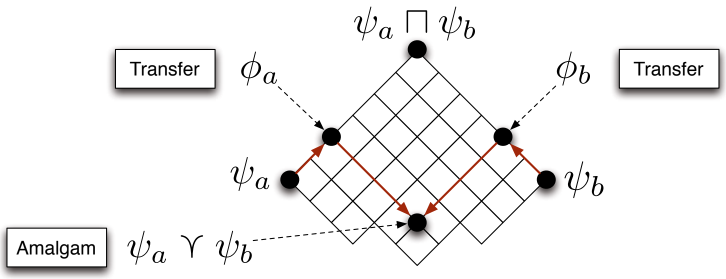


Fig. 2. Illustration of the idea of amalgam between two terms ψ_a and ψ_b

that is to say, there are no terms ϕ'_a , ϕ'_b such that $\phi_a \sqsubset \phi'_a \sqsubset \psi_a$ and $\phi_b \sqsubset \phi'_b \sqsubset \psi_b$ which have a unifier.

Finally, for the purposes of case reuse,we introduce the notion of asymmetric amalgam, where one term is fixed while only the other term is generalized in order to compute an amalgam.

Definition 5. (Asymmetric Amalgam) *The* asymmetric amalgams $\psi_s \overrightarrow{\curlyvee} \psi_t$ *of two terms* ψ_s *(called* source*) and* ψ_t *(called* target*) is the set of terms such that:*

$$\psi_s \overrightarrow{\curlyvee} \psi_t = \{\phi \in \mathcal{L} - \{\top\} | \exists \phi_s \in \mathcal{L} | \phi_s \sqsubseteq \psi_s \ \wedge \ \phi = \phi_s \sqcup \psi_t\}$$

In an asymmetric amalgam, the target term is transferred completely into the amalgam, while the source term is generalized.

4 ABARC

This section presents ABARC (Amalgam-based Agreement for Multiagent Reuse of Cases), a framework to address case reuse in multiagent CBR scenarios. Specifically, the scenario we focus on in this paper is the following.

Two CBR agents A_1 and A_2 need to solve a given problem P; the agents have s shared ontology to represent problems and solutions (e.g. solutions from one agent can be understood by the other), but each agent has its own individual biases with which to solve problems, i.e. given the same problem, each agent might prefer different solutions. In ABARC, biases are modeled as *utility functions*. Each agent A_i has a utility function U_i, which given a solution S to the problem P, returns a utility $U_i(S) \in [0, 1]$. These utility functions are private and we assume cannot be communicated (e.g. it is hard to completely communicate to another person you full preferences for food). Given a new problem, the goal of the agents is to reach an agreement over a solution, which is satisfactory to both agents (i.e. which has a high enough utility for both).

To address the previous scenario, ABARC assumes that problems to be solved have two separate parts:

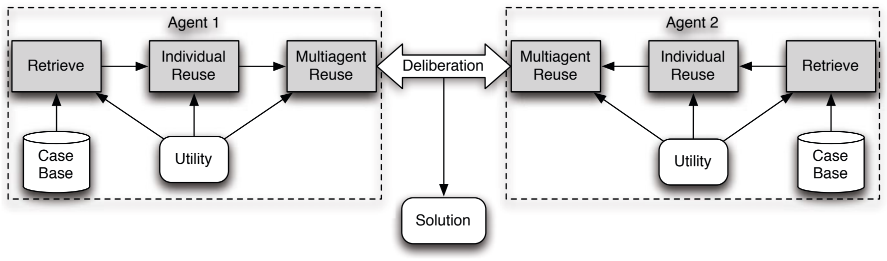


Fig. 3. In the ABARC framework, reuse is divided in two separate processes: individual reuse, and multiagent reuse

- *A shared problem specification*: the specification problem that needs to be solved, shared between the agents.
- *An individual problem bias*: the biases or preferences of a specific agent, represented as its own utility function over solutions, which is private to each agent.

Given that no agent knows the utility functions of the other agent, no one can actually propose a solution individually expecting it to have a high utility for the other agent. The main idea behind ABARC is that each agent individually can reuse cases to generate a solution which satisfies both the shared problem specification and its individual problem biases, and then combine the two solutions in order to generate a solution that satisfy at the same time the problem specification and the biases of both agents. Figure 3 illustrates this idea, where the reuse process of two CBR agents is split into two separate processes:

- *Individual reuse*: agents reuse the retrieved cases from the case base and generate a first candidate solution to problem P, which is passed on to the second reuse stage. We will call the solution generated by the individual reuse stage the *initial solution* of each agent.
- *Multiagent reuse*: agents start a deliberation process, where they propose solutions that combine parts of the agents' initial solutions (using amalgams) in order to generate a solution which satisfies as much as possible the utility functions of both agents. This is done in this way, since the initial solutions are the only indication an agent has of the utility function of the other agent.

Basically, the multiagent reuse step explores the space of amalgams between the initial solutions proposed by the agents. Given that this space might be very large, we propose a deliberation protocol that proceeds in a series of rounds where agents propose solutions, which are then evaluated by the other agent. The process continues until one agent accepts the solution of the other agent.

Assuming that solutions proposed by an agent A_i have high utility for A_i, if an agent A_j wants to generate a solution with expected high utility for A_i, such solution should be similar to the one proposed by A_i. This is the second key idea behind ABARC: initially agents generate solutions that have high utility for themselves, and at each round, the solution proposed by an agent contains more and more elements from the solution proposed by the other agent. This is done by computing amalgams that transfer more and more elements from the solution proposed by the other agent. The more elements are transferred from the other agent's solution, the less elements can be kept from the agents desired solution. Thus, the multiagent reuse process can be seen as a *concession process* where agents propose solutions that lower one's utility while trying to approach the interests of the other agent, until one agent accepts the solution of the other.

The remainder of this section first presents a specific interaction protocol that implements the deliberation process required to reach a final agreement over the solution to the problem P, and then how can the amalgam operation be used to merge solutions generated by both different agents.

4.1 Multiagent Reuse Interaction Protocol

The goal of the multiagent reuse process is to reach an agreement over a solution for the problem P. To reach such agreement, the agents explore the space of possible amalgams between their two individual solutions (which is the subspace of the whole solution space which is expected to have high utility for both). Moreover, it is possible that no solution exists that has maximum utility for both agents. Thus, it is important that agents reach an agreement, and settle for a "good enough" solution. In order to achieve that, ABARC uses a deliberation protocol where the agents are allowed to send three kind of messages:

- $propose(A, S)$: with which an agent A proposes solution S to the other agent.
- $accept(A, S)$: with which an agent A accepts the solution S previously sent by the other agent.
- $withdraw(A)$: with which an agent A withdraws from the protocol before reaching an agreement.

Given two agents, A_1 and A_2 who are trying to reach an agreement on the solution of a given problem P, the ABARC protocol is a token passing protocol, where at each round t, one of the two agents holds a token. Let us call S_1^0 and S_2^0 the solutions found by A_1 and A_2 respectively after performing individual reuse. S_1^0 is a valid solution for P and that has a high utility value for A_1, but might have a low utility value for A_2 (resp. S_2^0).

1. Initially, A_1 sends the message $propose(A_1, S_1^0)$, and A_2 sends the message $propose(A_1, S_2^0)$. Then, a token is given to one agent at random, round $t = 1$ starts, and the protocol moves to step 2.
2. The agent A_i owner of the token does the following:
   (a) Generates a new solution S_i^t. If a new solution cannot be generated (e.g. when the whole space of amalgams has already been explored), then A_i sends the message $withdraw(A_i)$ and the protocol ends without reaching any agreement.
   (b) A_i evaluates the utility of all the solutions sent by the other agent in rounds prior to t, and selects the solution S with maximum utility $U_i(S)$.

(c) If $U_i(S) \geq U_i(S_i^t)$, then A_i will send the message *accept*(A_i, S), and the protocol ends. Otherwise, A_i sends the message *propose*(A_i, S_i^t), the token is given to the next agent, a new round $t+1$ starts, and the protocol moves back to 2.

A key step in the protocol is 2.a, where agents generate new solutions. In ABARC the first solution that an agent A_i proposes, A_i^0 only takes into account its individual utility function. At each round, agents propose a new amalgam, which transfers a bit less from their solution, and a bit more from the other's solution, i.e. they propose solutions with higher expected utility for the other agent. In this way, agents *approach* each other, until they reach an agreement.

Therefore, the key component in the protocol is the mechanism for generating new solutions (new amalgams), as explained in the next section.

4.2 Merging Solutions through Amalgams

Two key functionality required in ABARC is generating solutions during the multiagent reuse protocol. Let us first explain how can this be implemented using amalgams and refinement operators.

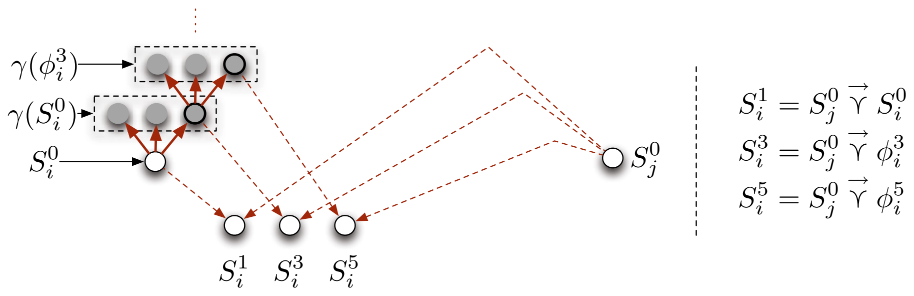
Given a solution S_i generated by an agent A_i, and a solution S_j generated by an agent A_j, if an agent A_i wants to generate a solution S^* which results from merging S_i with S_j, but without losing utility, it can do so in the following way, by using the previously introduced idea of asymmetric amalgam:

1. Let $B = \{S \in \mathcal{L} | S \sqsubseteq S_j \ \wedge \ (S_i \sqcap S_j) \sqsubseteq S\}$ be the set of potential transfers that can be used for computing the asymmetric amalgam $S_j \overrightarrow{\curlyvee} S_i$.
2. Let $\phi_j = \arg\max_{\phi \in B} U_i(\phi \sqcup S_i)$. Notice that $\phi \sqcup S_i$ is an asymmetric amalgam $S_j \overrightarrow{\curlyvee} S_i$, where ϕ is the transfer. Thus, ϕ_j is the transfer which maximizes the utility of the resulting amalgam for A_i.
3. $S^* = \phi_j \sqcup S_i$.

The previous method is generic, and can be used for any utility function. However, in the experiments reported in this paper we have modeled the utility functions in the following way. Each agent has a lists of private goals $G_1 = \{g_1^1, ..., g_n^1\}$, and $G_2 = \{g_1^2, ..., g_m^2\}$ respectively, where each goal g_j^i has an associated weight $w(g_j^i) > 0$, indicating how important it is (the higher the weight, the more important the goal is). A goal g is satisfied by a solution S if $g \sqsubseteq S$, therefore, if we have two solutions, S_1 and S_2, such that $S_1 \sqsubseteq S_2$, and g is satisfied by S_1, then g is also satisfied by S_2.

Therefore, any asymmetric amalgam will satisfy that $S_i \sqsubseteq (S_j \overrightarrow{\curlyvee} S_i)$, and thus will have, at least, the same utility as S_i. Moreover, since it is in the agent's own interest to propose solutions that maximize the other agent's utility as well, the more specific the amalgam, the higher the utility is likely to be for the other agent, and thus, it is desirable to compute complete amalgams.

In addition to merging solutions, agents in ABARC need to generate different amalgams in each round of the protocol, approaching the other agent. The way



$\gamma(\phi_i^3)$
$\gamma(S_i^0)$
S_i^0
S_j^0
S_i^1 S_i^3 S_i^5
$S_i^1 = S_j^0 \overrightarrow{\curlyvee} S_i^0$
$S_i^3 = S_j^0 \overrightarrow{\curlyvee} \phi_i^3$
$S_i^5 = S_j^0 \overrightarrow{\curlyvee} \phi_i^5$

Fig. 4. Solutions during the mutliagent reuse protocol are generated by an agent by iteratively conceding more and more, i.e. by transferring less and less from its own solution.

this is modeled in ABARC is by generating amalgams between S_i^0 and S_j^0 that are not asymmetric, i.e. where not all the information in S_i^0 is transferred to the amalgam. In an asymmetric amalgam $S_j^0 \overrightarrow{\curlyvee} S_i^0$, the transfer from S_i to the amalgam is $\psi_i = S_i$. A way to generate a solution that approaches that of the other agent is by transferring less information from S_i, i.e. having a transfer $\phi_i \sqsubset S_i$.

Moreover, A_i only wants to concede the minimum amount necessary. Thus, given that at a round t, an agent A_i proposed the solution S_i^t, with transfers ϕ_i and ϕ_j form S_i^0 and S_j^0 respectively, and given a generalization refinement operator γ, $\gamma(\phi_i)$ is a set of terms which are more general than ϕ_i. Given any term $\phi'_i \in \gamma(\phi_i)$, if we compute a complete asymmetric amalgam $S_j^0 \overrightarrow{\curlyvee} \phi'_i$, we obtain an amalgam with less transfer from S_i^0 than S_i^t (and thus may accommodate more transfer from the solution of the other agent), i.e. we obtain a solution that approaches that of the other agent.

In ABARC, agents generate new solutions during the protocol in the following way, as illustrated in Figure 4:

1. In the initial round, the agent proposes its initial solution S_i^0.
2. Then, the first time an agent A_i owns the token, it will propose the solution generated by a complete asymmetric amalgam $S_j^0 \overrightarrow{\curlyvee} S_i^0$.
3. The subsequent times an agent A_i owns the token, it will generate a solution by generalizing the last transfer term using a refinement operator:
   (a) First, compute $\phi'_i = \arg\max_{\phi \in \gamma(\phi_i)} U_i(\phi)$, which is the generalization of the previous transfer ϕ_i with highest utility for A_i.
   (b) The next solution to propose will be $S_j^0 \overrightarrow{\curlyvee} \phi'_i$.

As Figure 4 shows, by iteratively generalizing the transfer term, the solutions generated by an agent A_i contain less and less from its original solution, S_i^0, in order to accommodate more and more from the solution proposed by the other agent, S_j^0. The effect is the sequence of solutions proposed by A_i have decreasing utility for A_i, but increasing for A_j, and the solutions proposed by

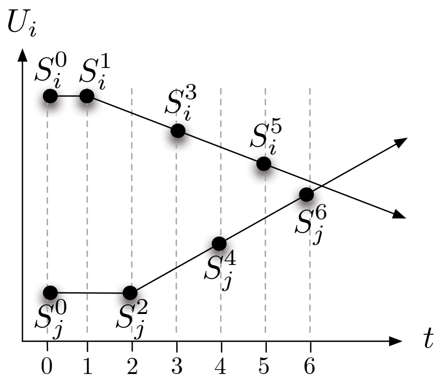


Fig. 5. The utility for one agent A_i of the solutions proposed by both agents over time

A_j have increasing utility for A_i and decreasing for A_j. As illustrated in Figure 5, the utility values will eventually cross, and the agents will reach an agreement.

5 Collaborative Generation of Room Designs

In order to illustrate ABARC, we have applied it to the domain of interior room design. In this domain the goal is to design the interior of a room (pieces of furniture to use and their layout) according to some given constraints (size and shape of the room), and a set of given goals (e.g. design a proper work environment). Moreover, we have specifically focused on the problem of designing room for two individuals. In this case, when designing the room, the two individuals (represented by agents) have to agree on a design that satisfies both.

In this section we will present an illustration with a particular instance of this problem, in which two agents need to agree on the design of a square office for work space. Agent A_1 has a "geek" personality, and agent A_2 represents an "artist" personality. In this instance, we have represented a room to be composed of 8 spaces: the four corners (which can accommodate small pieces of furniture) and the four big walls (which can accommodate large pieces of furniture). The south-west corner needs to be free, for the door to enter the room. The agents can select among 7 different small pieces of furniture (like dressers, plants, small desks, etc.) and 9 different large pieces of furniture (like couches, desks, etc.). Some pieces of furniture (like computer desks), can also have devices on top (like netbooks, laptops or computers). In total, there are about 1.3×10^9 possible room configurations. For this example, we represented solutions using *feature terms* [1], and we thus use a general feature term generalization refinement operator.

The goals and weights of the goals that compose the utility functions of the two agents are shown in Table 1. Utility is measured in the following way:

$$U_i(S) = \frac{\sum_{g \in G_i | g \sqsubseteq S} w(g)}{\sum_{g \in G_i} w(g)}$$

Table 1. Goals of two agents in a room design problem

Agent A_1		Agent A_2	
Goal	*Weight*	*Goal*	*Weight*
tower-computer	8.0	drawing table	8.0
computer desk	7.0	Tansu	7.0
plant pot	6.0	glass-door cabinet	6.0
bookcase	5.0	couch	5.0
couch	4.0	laptop	4.0
armchair	3.0	plant pot	3.0
dresser	2.0	filing cabinet	2.0
cabinet	1.0	corner desk	1.0

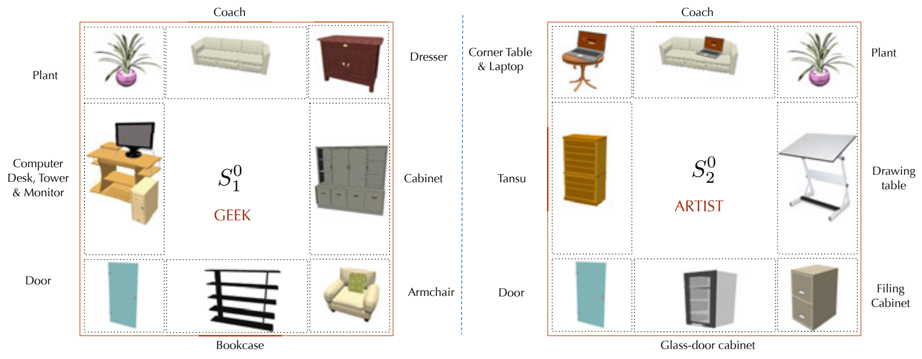


Fig. 6. The two room designs according to the Geek's and Artist's goals

i.e. as the sum of the weights of the goals that are satisfied by a solution, divided between the total sum of weights of the goals. Thus, if we have two solutions S_1 and S_2 such that $S_1 \sqsubseteq S_2$ then we know that $U(S_2) \geq U(S_1)$.

After solving the problem individually, the agents come up with the solutions shown in Figure 6[2]. If we evaluate the utilities of the solutions for both agents, we obtain that: $U_1(S_1^0) = 1.0$, $U_1(S_2^0) = 0.22$, $U_2(S_1^0) = 0.31$, and $U_2(S_2^0) = 1.0$. Notice that the solution proposed by A_2 (the artist) has a non zero utility for A_1 (the geek), since S_2^0 contains elements such as a plant pot, a couch and a glass-door cabinet (which is just a special kind of cabinet), and thus S_2^0 satisfies some of the goals of A_1.

When the ABARC protocol starts (t=1), say agent A_1 owns the token; then A_1 tries to find an asymmetric amalgam $S_2^0 \overrightarrow{\curlyvee} S_1^0$ to propose as a new solution. That is to say, A_1 tries to find a transfer $\phi_2^1 \sqsubseteq S_2^0$ that unifies with S_1^0; the result is the solution S_1^1, shown on the left hand side of Figure 7. This solution keeps all the content of S_1^0 and has incorporated the laptop, which fits on top of

[2] Notice that the 2 laptops appearing in S_0^2 means that there are 2 ways to achieve the goal of having a place for a laptop, not having 2 places or 2 laptops.

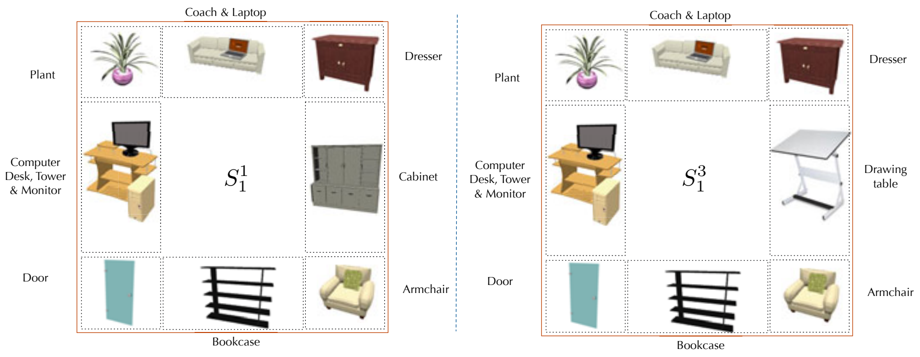


Fig. 7. Solutions S_1^1 and S_1^3 proposed by A_1 in rounds 1 and 3

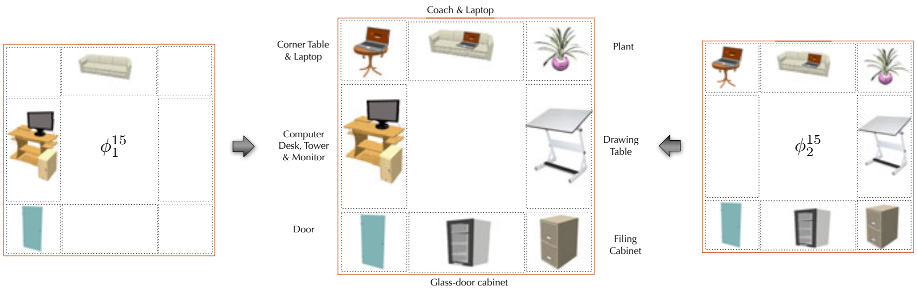


Fig. 8. Final agreed-upon solution, and the transfers from the initial solutions proposed by the agents

the couch. This solution has improved A_2's utility to $U_2(S_1^1) = 0.33$, since one more of its goals (having a laptop) is satisfied. Two rounds later, A_1 needs to generate another solution proposal, and, using a refinement operator, generalizes S_1^0 giving up the *cabinet* in the east wall, to allow any piece of large furniture (ϕ_1^3). Thus, A_1 proceeds to find another asymmetric amalgam $S_2^0 \overrightarrow{\curlyvee} \phi_1^3$, obtaining a solution S_1^3 with the drawing table on the east wall. This solution now has a slightly lower utility for A_1 (0.97), but an increased utility for A_2 (0.56).

After only 15 rounds, they arrive at the solution shown in Figure 8, which has utility 0.72 and 0.81 for A_1 and A_2 respectively. Figure 8 also shows what was transferred from the original solutions by A_1 and A_2. Notice that most of the solution comes from A_2, however, that is fine for A_1, since the part that is transferred from A_2 satisfies many of A_1's goals, such as having a plant-pot or having a cabinet. This shows ABARC can effectively and efficiently help two agents reach an agreed-upon solution even when they have completely different sets of goals, and no agent knows the goals of the other agent.

Moreover, in order to assess how good is the solution found by ABARC, we can compare it with the "best" solution that can be found in the search space. Given that there are two utility functions involved, we may use welfare functions for determining collective utility, such as the *utilitarian* welfare (the solution that maximizes the sum of utilities) or *egalitarian* welfare (the solution that maximizes the minimum utility). Considering a utilitarian welfare, the best solution that could have been found has utility values of 0.875 and 0.75 for A_1 and A_2 respectively. To find that solution, we ran an exhaustive algorithm that explored all the 1.3×10^9 possible solutions. In comparison, agents in ABARC only shared 15 different solutions (and considered 161 solutions internally). Moreover, the solution found by ABARC has a utility about 94% of the optimal solution in utilitarian welfare. This means that ABARC manages to find a very good solution very fast, thanks to the amalgam operation. On the worst case, agents using ABARC would explore the set of all possible amalgams between their initial solutions, which is a subspace of the complete solution space.

6 Related Work

Three areas of work are related to ABARC: reuse in multiagent systems, reusing multiple cases, and work on merging operations.

Concerning reuse in multiagent systems, the first work in multi-CBR systems was presented by Prasad, Lesser and Lander [13]. They focus on a system where a set of individual agents have collected experience on their own, and thus can have a local view of each problem. They present a decentralized "negotiated case retrieval" technique that allows the group of agents to retrieve the appropriate information from each individual case base and then aggregate it in order to answer a query. The aggregation, however, does not correspond to merging solutions as in our framework, but to the fact that different agents might only know some subset of the features of each problem. Similarly, other research in multiagent CBR, like Plaza, Arcos and Martin [11], McGinty and Smyth [9], or Leake and Sooriamurthi [8], focuses on distributing the retrieval stage.

To the best of our knowledge, the only work on distributed reuse is that of Ontañón and Plaza [12], which focuses on classification tasks, and on using voting mechanisms as a distributed reuse procedure. The difference with our work is that we focus here on structured solutions, and thus, voting mechanisms are not appropriate.

Concerning reusing multiple solutions, specially relevant are *compositional adaptation* techniques, which find new solutions from multiple cases and have been analyzed for configuration tasks in [15], where the approach *adaptation-as-configuration* is presented. This approach has two specific operators (compose and decompose) that achieve compositional adaptation in "conceptual hierarchy oriented configuration." *Compose* merges multiple concept instances that have already been configured, while *Decompose* gives the subconcept instances of a given concept instance. These operations work upon *is-a* and *part-of* relations in an object-oriented hierarchy.

Another application, different from configuration, is planning. Systematic Plan Adapter (SPA) [4] is an approach for adapting plans as search in a refinement graph of plans. SPA is systematic and complete and, as local search, is based on adapting a single plan. MPA (Multi-Plan Adapter) [14] is an extension which allows for reusing multiple plans. MPA breaks the different plans into smaller pieces, which are then recombined together. The complexity breaking these plans into smaller pieces, however, is very high and MPA uses only a heuristic. Compared to MPA, our approach avoids the need of this breaking down process while providing a systematic way to combine multiple solutions into a single one.

Concerning merging operations, they have been studied in *belief merging* [5], where the goal is to merge two knowledge bases (beliefs plus integrity constraints) while maintaining consistency. This approach was applied to CBR [2] by viewing case combination in reuse as a belief merging problem. Specifically, each case is viewed as a knowledge base, and the merging is generating a new knowledge base that preserves the relevant integrity constraints.

7 Conclusions

In this paper we have focused on distributed reuse in multiagent CBR systems. Specifically, we have focused on the scenario where two CBR agents with different biases or solution quality criteria need to agree on a single structured solution for a given problem. To address this scenario, we have presented ABARC, which models this problem by dividing the reuse process in two subprocesses: individual reuse, and multiagent reuse. During multiagent reuse, we use the *amalgam* operation and refinement operators in order to propose solutions that consist of different ways to merge the two individual solutions proposed by the agents.

ABARC effectively allows agents to reach agreements over structured solutions by exploring the different ways in which the solutions proposed by each agent can be merged (by using the amalgam operation). By only exploring the different ways to amalgamate the solutions proposed individually, ABARC avoids an expensive search process over the space of possible solutions trying to find a solution that maximizes the utility for both agents.

The work presented in this paper is one step towards our long term goal of enabling the application of CBR to open multiagent systems. Whereas distributed retrieval has received some attention, the topic of reusing multiple cases in multiagent CBR has remained under-explored. Our future work focuses on two main lines: on the one hand, we would like to continue analyzing the properties of the amalgam operation for multiple case reuse, and on the other hand, we are interested in the general problem of problem-solving in distributed scenarios, where agents need to reach agreements over potentially structured solutions.

Acknowledgements. This research was partially supported by projects Next-CBR (TIN2009-13692-C03-01) and AT CONSOLIDER CSD2007-0022.

References

[1] Carpenter, B.: Typed feature structures: an extension of first-order terms. In: Saraswat, V., Ueda, K. (eds.) Proceedings of the International Symposium on Logic Programming, San Diego, pp. 187–201 (1991)

[2] Cojan, J., Lieber, J.: Belief merging-based case combination. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS(LNAI), vol. 5650, pp. 105–119. Springer, Heidelberg (2009)

[3] Falkenhainer, B., Forbus, K.D., Gentner, D.: The structure-mapping engine: Algorithm and examples. Artificial Intelligence 41, 1–63 (1989)

[4] Hanks, S., Weld, D.S.: A domain-independent algorithm for plan adaptation. J. Artificial Intelligence Research 2(1), 319–360 (1994)

[5] Konieczny, S., Lang, J., Marquis, P.: Da2 merging operators. Artificial Intelligence 157(1-2), 49–79 (2004)

[6] van der Laag, P.R.J., Nienhuys-Cheng, S.H.: Completeness and properness of refinement operators in inductive logic programming. Journal of Logic Programming 34(3), 201–225 (1998)

[7] Lavrač, N., Džeroski, S.: Inductive Logic Programming. Techniques and Applications. Ellis Horwood, England (1994)

[8] Leake, D.B., Sooriamurthi, R.: When two case bases are better than one: Exploiting multiple case bases. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 321–335. Springer, Heidelberg (2001)

[9] McGinty, L., Smyth, B.: Collaborative case-based reasoning: Applications in personalized route planning. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 362–376. Springer, Heidelberg (2001)

[10] Ontañón, S., Plaza, E.: Amalgams: A formal approach for combining multiple case solutions. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 257–271. Springer, Heidelberg (2010)

[11] Plaza, E., Arcos, J.L., Martín, F.: Cooperative case-based reasoning. In: Weiss, G. (ed.) ECAI 1996 Workshops. LNCS (LNAI), vol. 1221, pp. 180–201. Springer, Heidelberg (1997)

[12] Plaza, E., Ontañón, S.: Ensemble case-based reasoning: Collaboration policies for multiagent cooperative cbr. In: Watson, I., Yang, Q. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 437–451. Springer, Heidelberg (2001)

[13] Prassad, M.V.N., Lesser, V.R., Lander, S.: Retrieval and reasoning in distributed case bases. Tech. rep., UMass Computer Science Department (1995)

[14] Ram, A., Francis, A.: Multi-plan retrieval and adaptation in an experience-based agent. In: Leake, D.B. (ed.) Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press, Menlo Park (1996)

[15] Wilke, W., Smyth, B., Cunningham, P.: Using configuration techniques for adaptation. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 139–168. Springer, Heidelberg (1998)

The 4 Diabetes Support System: A Case Study in CBR Research and Development

Cindy Marling[1], Matthew Wiley[1], Tessa Cooper[1],
Razvan Bunescu[1], Jay Shubrook[2], and Frank Schwartz[2]

[1] School of Electrical Engineering and Computer Science,
Russ College of Engineering and Technology,
Ohio University, Athens, Ohio 45701, USA
{marling,mw350004,tc273702,bunescu}@ohio.edu
[2] Appalachian Rural Health Institute, Diabetes and Endocrine Center,
College of Osteopathic Medicine,
Ohio University, Athens, Ohio 45701, USA
{shubrook,schwartf}@ohio.edu

Abstract. This paper presents the 4 Diabetes Support System^{TM} (4DSS) project as a case study in case-based reasoning (CBR) research and development. This project aims to help patients with type 1 diabetes on insulin pump therapy achieve and maintain good blood glucose control. Over the course of seven years and three clinical research studies, a series of defining cases altered the research and development path. Each of these cases suggested a new, unanticipated research direction or clinical application. New AI technologies, including naive Bayes classification and support vector regression, were incorporated. New medical research into glycemic variability and blood glucose prediction was undertaken. The CBR research paradigm has provided a strong framework for medical research as well as for artificial intelligence (AI) research. This new work has the potential to positively impact the health and wellbeing of patients with diabetes. This paper shares the 4DSS project experience.

Keywords: Medical decision support, diabetes management, CBR research paradigm.

1 Introduction

The 4 Diabetes Support System^{TM} (4DSS) project began in 2004 with the goal of producing a case-based decision support system for diabetes management. Seven years and three clinical research studies later, the research and development path has diverged considerably from that originally planned. While the medical goals have not changed, we now envision different AI integrations and new clinical applications, with potentially far greater impact on health and well-being. We hypothesize that the credit for discovering these new directions, or perhaps the blame for straying from the original path, lies squarely with the case-based reasoning (CBR) research paradigm. At each fork in the road was a case suggesting a new direction.

This paper presents the 4DSS project as a case study in CBR research and development. The next section describes the diabetes management domain, within which this work was conducted. Section 3 recaps the first clinical research study, during which the original 4DSS prototype was built. Section 4 presents the case of the bouncing blood glucose, which motivated the integration of naive Bayes classification into the 4DSS situation assessment module. The case of the just-too-late problem detection, presented in Section 5, led to the integration of support vector regression for real-time situation assessment. Section 6 describes the case of the one-fingered typist, which led to new work in user interface design. The remainder of the paper briefly touches upon future plans and related work, followed by the summary and conclusion.

2 The Diabetes Management Domain

The World Health Organization (WHO) estimates that over 220 million people have diabetes worldwide [30]. From 5 to 10% of these people have type 1 diabetes (T1D), the most severe kind. In T1D, the pancreas fails to produce insulin, an essential hormone needed to convert food into energy. Therefore, T1D patients must depend on exogenous supplies of insulin to live. T1D can not, at present, be prevented or cured; however, it can be treated and effectively managed. At the Appalachian Rural Health Institute Diabetes and Endocrine Center, T1D patients are treated with insulin pump therapy. In insulin pump therapy, a patient is continuously infused with a basal amount of insulin via a pump at all times. To account for food intake or other daily activities, the patient may instruct the pump to deliver additional boluses of insulin.

The cornerstone of diabetes management is blood glucose control. It was experimentally determined, in a landmark 1993 study, that good blood glucose control can help delay or prevent serious long-term complications of diabetes, including blindness, amputations, kidney failure, strokes, and heart attacks [11]. Achieving and maintaining good blood glucose control is a difficult task for patients, who must continuously monitor their blood glucose levels and daily activities. It is a difficult task for physicians, who must review copious quantities of blood glucose and life event data, looking for problems and making therapeutic adjustments to correct them.

Providing intelligent decision support to facilitate good blood glucose control is an endeavor in the CBR in the Health Sciences tradition [7]. The general nature of clinical diabetes management guidelines, coupled with a wide variability among individual patients, means that therapy must be customized to the needs of each individual. The strong influence of social context, including qualitative lifestyle preferences and daily life events, on quantitative outcomes means that combinations of multiple diverse features must be taken into account. The nature of chronic disease management precludes the derivation of a single solution; rather, each patient must be followed and managed over time. Other medical domains with such characteristics have provided fertile ground for CBR research and development [8]. That said, the lessons learned from the 4DSS project do not only apply in health sciences domains. We can draw analogies, for example, to continuously monitoring the "health" of an oil rig to detect problems and making "therapeutic" adjustments to correct detected problems.

3 The First Clinical Research Study

A preliminary clinical study was conducted between 2006 and 2007 to assess the feasibility of creating an intelligent decision support system for patients with T1D on insulin pump therapy. The results of this study have been previously published [15,16,24]; a synopsis is presented here. The goal was to design, build and evaluate a case-based decision support system prototype. While plentiful blood glucose data was initially available, usable cases were not. This is because the life events impacting blood glucose levels are not routinely maintained. To acquire cases for 4DSS, 20 patients were enrolled in a study, and 12 of them completed a 6-week protocol. Each patient entered daily blood glucose, insulin, and life-event data into a database via a Web browser. Each patient used a continuous glucose monitoring (CGM) system for three 3-day periods to capture supplemental blood glucose readings at 5-minute intervals.

The research team met weekly to review patient data. Physicians identified problems in blood glucose control and recommended therapy adjustments for each patient. As patients made therapy adjustments, physicians monitored to evaluate the effectiveness of the changes. These problems, solutions and outcomes were structured into 50 cases for the 4DSS system prototype. One of these cases is shown, in simplified form, in Figure 1. This figure shows a case of nocturnal hypoglycemia. Hypoglycemia, or low blood glucose, may cause weakness, dizziness, confusion, sweating, and if not promptly treated, seizures, coma, or death. Hypoglycemia that occurs while the patient is asleep, as in the sample case, is especially dangerous. Hyperglycemia, or high blood glucose, contributes to the long-term complications of diabetes. Extremely high blood gluocose levels may trigger diabetic ketoacidosis, a serious condition causing severe acute illness or death.

A significant research challenge was detecting patient problems so that useful past solutions could be retrieved and reused. Patients do not always know when problems are impending and are frequently unaware of them even after they occur. Twelve commonly occurring problems were identified by physicians, and rule-based situation assessment routines were developed to automatically recognize these problems in patient data. A 4DSS prototype was then built to: (1) detect problems in blood glucose control; (2) display detected problems to the physician, who would select the problems of interest; (3) retrieve, for each selected problem, the most similar case in the case base; and (4) display the retrieved cases as decision support for determining appropriate therapeutic adjustments.

Evaluation and feedback were obtained through a patient exit survey and two structured sessions in which diabetes practitioners evaluated the problem

Problem: Nocturnal hypoglycemia. Blood glucose levels are dangerously low all night. The patient reports feeling "totally out of it" when she wakes up. She does not eat anything to correct the hypoglycemia until noon. She had not eaten a bedtime snack the night before.

Solution: The patient should always have a mixed-nutrient snack before bed. She should lower her overnight basal rate. The combination of more food and less insulin will prevent overnight lows.

Outcome: The patient reported eating mixed nuts and crackers before bed. She set the basal rate in her pump as advised. Blood glucose data for subsequent weeks showed that the problem had been resolved.

Fig. 1. A sample case from 4DSS

detection and case retrieval capabilities of the system. Patients indicated that they would willingly accept automated decision support, but noted that the time required for data entry was a deterrent. Physicians noted that the integration of blood glucose, insulin and life-event data helped them to identify glucose trends more readily and adjust therapy more effectively. Conclusions from this study were: (1) the 4DSS system prototype provides proof of concept that intelligent decision support can assist in diabetes management; (2) additional problem/solution/outcome cases are needed to provide solutions for more blood glucose control problems; and (3) data entry time demands on the patient must be reduced.

4 The Case of the Bouncing Blood Glucose

A second clinical research study was conducted between 2008 and 2009 to enhance and evaluate the 4DSS prototype. The primary goal of this study was to streamline the user interface to reduce time demands on patients and then re-evaluate system performance. A secondary goal was to expand system functionality by implementing additional problem detection routines. Twenty-six patients with T1D on insulin pump therapy enrolled, and 23 patients completed the 5-week protocol.

Not long into this study, we encountered the case of the bouncing blood glucose. To date, we could automatically detect 12 types of blood glucose control problems, all involving hypoglycemia or hyperglycemia. Now we had a case where

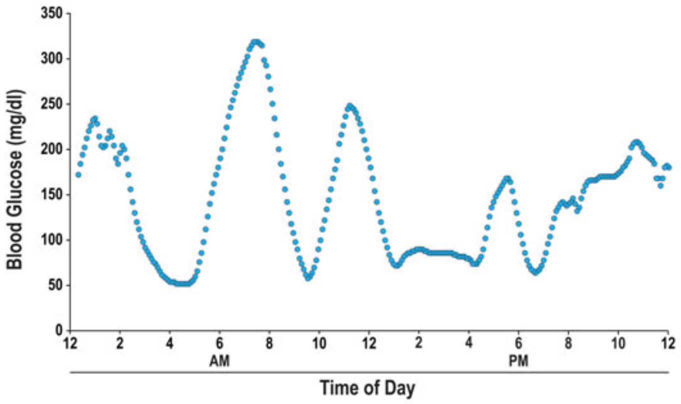


Fig. 2. Bouncing blood glucose

a patient bounced back and forth between hypo and hyperglycemia, in a roller coaster pattern, as shown in Figure 2. We thought that it would be straightforward to implement a new rule-based situation assessment routine to detect this problem. However, several attempts met with failure.

We had encountered *glycemic variability*, a current and controversial topic of diabetes research [9,13,19]. Excessive glycemic variability has been linked to hypoglycemia unawareness, an acutely dangerous condition, and to oxidative stress, which contributes to long-term diabetic complications. Its automated detection would be a major contribution to clinical diabetes management. Researchers have proposed numerous metrics for characterizing glycemic variability, but none are widely used in clinical practice [21,22]. This was a research challenge, and we were off on a detour.

While experts disagree on how to measure glycemic variability, physicians readily recognize it when they see it in blood glucose plots. We therefore considered the quantifiable aspects of glycemic variability as they relate to physicians' perception. We began with the best accepted glycemic variability metric, the Mean Amplitude of Glycemic Excursion (MAGE) [26]. MAGE captures the distance between the local maxima and minima of a blood glucose plot. We then devised two new metrics, distance traveled and excursion frequency, to capture aspects of glycemic variability not accounted for by MAGE. Distance traveled captures overall daily fluctuation, and excursion frequency counts the number of significant glucose excursions in a day.

Two physicians (JS and FS) classified blood glucose plots as excessively variable or not, based on their gestalt perceptions. The same plots were scored for MAGE, distance traveled and excursion frequency. The physician classifications and metric scores for 218 blood glucose plots were used to train multiple machine learning algorithms, via the Weka machine learning toolkit [29]. During testing, a naive Bayes classifier was able to match physician ratings 85% of the time. The preliminary results are in press [17].

In ongoing work, we are exploring other machine learning algorithms, additional glycemic variability features, and data smoothing techniques. While work continues to improve performance, machine learning classification is now an integral part of 4DSS. Within the original 4DSS framework, glycemic variability classification extends the situation assessment module. When excessive glycemic variability is detected as a problem, case retrieval can be invoked to find an applicable solution. Due to the clinical importance of glycemic variability, the new classifier also has potential as a standalone clinical assessment tool.

5 The Case of the Just-Too-Late Problem Detection

Another patient in the second clinical research study presented the next pivotal case. This patient's pump failed and stopped delivering insulin. He was aware that his blood glucose was high, and he instructed the pump to deliver more insulin. However, he did not know that the pump was not functioning, and his blood glucose continued to climb. He went into diabetic ketoacidosis (DKA) and was admitted to the hospital, where he experienced a (non-fatal) heart attack. When his data was scanned retroactively, the system automatically detected the pump problem eight hours before he was admitted to the hospital. Had the system been running in real time, the patient might have been alerted to change his infusion set, a simple adjustment that could have prevented the DKA. This case highlighted the need to predict and prevent problems instead of just detecting and correcting them. Off we went in another new research direction.

Real-time blood glucose data is not currently available from CGM systems. This is not a technical limitation; rather, the U.S. Food and Drug Administration (FDA) has not yet approved its use. There is, however, an active diabetes research program, the Artificial Pancreas project [12], which is currently seeking FDA approval for real-time problem prediction and intervention. The immediate goal (among many long-range plans) is to predict when blood glucose will drop to hypoglycemic levels and alert the patient in time to take preventive action. Blood glucose prediction is typically approached from a control theory perspective or a physiological pharmacokinetic modeling perspective. We believe that our CBR perspective gives us an advantage, in that the contextual features impacting blood glucose levels have so far been largely ignored.

Given the natural temporal ordering of blood glucose measurements, we approach blood glucose prediction as a time series forecasting problem. Here, the task is to estimate the future value of a target function based on current and past data samples. We chose Support Vector Regression (SVR) models [28] for this task, as they can easily incorporate contextual features, without any assumptions of feature independence. Furthermore, SVR models have been used successfully in numerous time series prediction problems [23].

In a preliminary experimental evaluation, an SVR model was trained to predict the blood glucose level of a T1D patient. We had three months of data available, including blood glucose measurements recorded at 5-minute intervals,

Table 1. SVR and baseline results

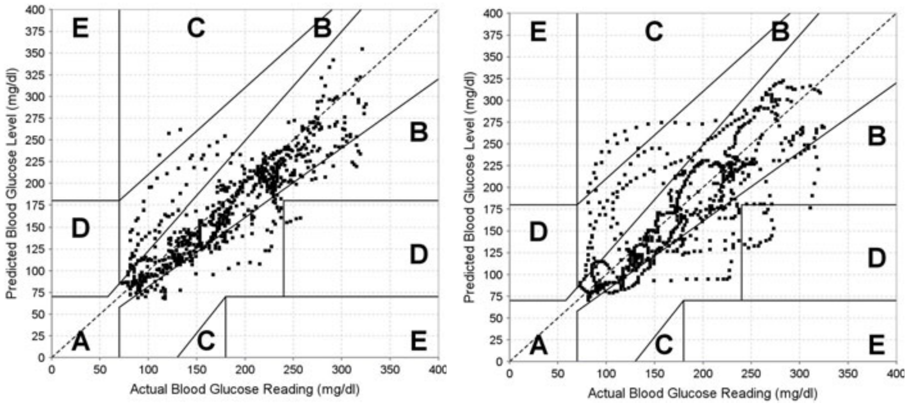
30 Minute Predictions							
Method	E_{RMS}	R^2	A	B	C	D	E
SVR	**18.0**	**0.92**	**93.0**	7.0	0.0	0.0	0.0
Baseline	25.1	0.84	87.8	11.8	0.0	0.4	0.0

60 Minute Predictions							
Method	E_{RMS}	R^2	A	B	C	D	E
SVR	**30.9**	**0.76**	**81.0**	18.1	0.4	0.5	0.0
Baseline	43.2	0.52	74.5	21.5	2.2	1.8	0.0

insulin dosages, and contextual life events. To create training and testing examples, a date was selected approximately one month into the 3-month data set. Data for seven days before that date was used as training data, while data on that date and the two subsequent days was used as test data. Testing and training examples were represented as feature vectors including: blood glucose level; the exponentially smoothed rate of change in blood glucose level; insulin dosage; carbohydrate intake; and exercise time and intensity. Two separate SVR models were trained and tested to predict blood glucose levels 30 and 60 minutes into the future. The SVR models were trained with a linear kernel; the LibSVM [10] implementation of support vector machines for regression was used.

In Table 1, we compare the performance of the two SVR models with a baseline that uses the present blood glucose level as the prediction for any future blood glucose value. This baseline, while simple, was used because it outperformed more complex moving average and rate of change baselines. We report the root mean square error E_{RMS}, the coefficient of determination R^2, and the percentage of predictions falling in the 5 areas from A to E in the Clarke Error Grid Analysis (CEGA) [14]. CEGA is a domain specific standard for assessing the accuracy of blood glucose measurements or predictions. It was originally designed to assess the quality of blood glucose sensors. Area A is the desired region: it corresponds to measured/predicted values that are within 20% of actual/observed values. Areas B through E were created in recognition that it is clinically more important for measurements or predictions to be accurate in some regions than in others. For example, if a patient's blood glucose level is predicted to be 400, but turns out to be 360, that would be an acceptable error. Both levels are very high, and the same medical treatment would apply. However, if the patient's blood glucose level is predicted to be 100, but turns out to be 60, that would be a serious error, because a hypoglycemic episode requiring immediate treatment would be masked.

The SVR models are promising, as they outperform the baseline on all performance measures. The two CEGA plots in Figure 3 show the performance of SVR and the baseline, respectively, for the 60 minute prediction. Overall, the plots show the learned SVR model making predictions that are closer to the ideal diagonal line.



(a) SVR performance (b) Baseline performance

Fig. 3. CEGA plots for 60 minute prediction of blood glucose levels

Our research in blood glucose prediction is currently focused on making the learned models more robust in the presence of measurement noise and data anomalies. We have created a modified framework for regularized spline smoothing in which blood glucose measurements obtained through traditional self-glucose monitoring (SGM) override the less accurate CGM measurements. This approach is based on the way physicians read blood glucose plots containing both CGM and SGM data. By replacing the raw blood glucose values used during training with smoothed values, we expect to mitigate the effects of noise on the final prediction performance.

When integrated into the 4DSS situation assessment module, blood glucose prediction will be an important step toward real-time case-based decision support. A preemptive solution could be retrieved from the case base as soon as a problem is predicted, 30 to 60 minutes before it might otherwise occur. Again, due to the clinical value of blood glucose prediction, this functionality has potential uses beyond the 4DSS. For example, prediction could be incorporated directly in insulin pumps, as a safety feature, to trigger alarms and warnings. Prediction would also be useful in a standalone educational tool with what-if analysis, to help patients understand the impact of their actions on blood glucose control.

6 The Case of the One-Fingered Typist

The second 4DSS clinical research study was completed in 2009. Results were not all positive [25]. Streamlining data entry entailed the following changes: (1) patients no longer interactively entered data; (2) blood glucose and insulin data stored in the pump was automatically uploaded to the database; and (3) daily life events were approximated by typical daily schedules. Time requirements were reduced for patients, but only half as many problems were detected as when the old data entry system was used. This finding was statistically significant (p=0.017), although there were no statistically significant differences between the patient populations and no reason to suspect that patients were actually experiencing fewer problems.

A third clinical research study was conducted between 2009 and 2011. The primary aims of this study were to: (1) enlarge the case base; (2) develop additional problem detection routines; and (3) add an adaptation module to tailor past solutions to the specific needs of current patients. To balance the need to collect all relevant data with the need to minimize patient time demands, it was necessary to build yet another data entry interface before enrolling patients. Seventeen patients enrolled, and twelve patients completed a 3-month protocol. For this study, patients: (1) collected CGM data for the full three months; (2) uploaded insulin pump and CGM data weekly; and (3) supplied daily life-event data that would otherwise be unavailable via a Web browser.

During this study, a patient enrolled who typed with one finger. This patient conscientiously tried to enter all required data, but the process was painful for him, and the data captured was error prone and incomplete. We had long anticipated that, when the system moved from the research laboratory to clinical practice, the user interface would be embedded in medical devices, electronic health records, and/or cell phones. We had not intended to do any research or development in user interface design. Yet, here was a case with a problem in need of a solution.

We are currently in the process of designing and implementing a smart phone data capture system for 4DSS. This system will be used and evaluated during the next clinical research study. The system is being built for a mobile browser, to accommodate any brand of smart phone. The over-arching goal is to create an interface that requires only one finger to enter data. Smart phone touch screens lend themselves to this goal. The evolving interface contains mostly buttons, check boxes, and drop down lists, all of which can be selected by the touch of a finger. The home screen for the new smart phone interface is shown in Figure 4.

We envision other potential advantages to this mode of data capture. Patients will be able to enter data as events occur, rather than waiting until the end of the day to supply a whole day's data. For example, a patient can touch a "Going to Work" button as he or she is actually going to work. Because cell phones can automatically obtain the time, the patient will no longer need to manually enter it, improving data accuracy. Furthermore, using the smart phone platform, physicians will be able to text therapy recommendations to patients, rather than relying on email. This could potentially reduce the time it takes for patients to implement physician recommendations. It is interesting to note that the first reported use of CBR for diabetes management was in support of a telemedicine application [5]. Now, telemedicine technology may support case-based diabetes management.

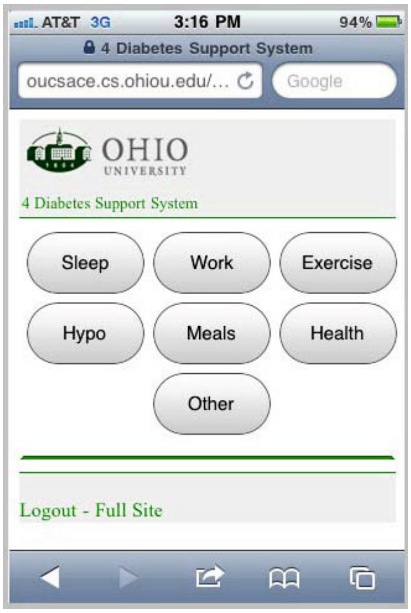


Fig. 4. New 4DSS smart phone data entry interface

7 Case-Based Decision Support Redux

Patients have concluded their participation in the third 4DSS clinical research study, but associated system development activities are ongoing. To date: (1) 30 new cases have been added to the case base; (2) six new problem detection routines have been developed; (3) an adaptation module was built; and (4) a new backend interface was built for physicians to review cases and to view system recommendations. A fourth 4DSS clinical research study is now being planned; the protocol is under development. This study will support the original project goals for case-based diabetes management, the more recent goals of detecting excessive blood glucose variability and predicting blood glucose levels, and the as-yet-to-be-determined research goals that will undoubtedly arise along the way. Parallel efforts are underway to facilitate transfer of 4DSS techology from the research laboratory to clinical use. The new smart phone interface may be viewed as part of this effort. New techology transfer collaborators are working on market assessment and product placement. This will be another chapter for the evolving 4DSS project.

8 Related Work

Twenty years ago, Stephen Slade wrote "Case-Based Reasoning: A Research Paradigm" [27]. He described CBR as a paradigm for reasoning from experience that addresses two core AI research agendas: understanding human thought, and building intelligent systems. In short, he proposed CBR as an ideal paradigm for conducting AI research. Today, it is clear that CBR is a useful paradigm for conducting diabetes research, as well. The intuition for CBR as a medical research paradigm is deeply rooted in medical history. Over 100 years ago, when Dr. Alois Alzheimer first encountered a puzzling new brain disease, he wrote that, when confronted with unknown disease patterns, "a further histological examination must be effected to determine the characteristics of each single case" [18]. While synergies between CBR research and medical research are well documented [8], the view of a CBR framework for medical research is atypical. A notable exception was the early MNAOMIA project [6], which had medical research in the field of psychiatric eating disorders as a full-fledged project goal.

A current research and development effort with many parallels to the 4DSS project is the Mälardalen stress project [3]. This project, which began in 2002 as a collaboration between CBR researchers and leading clinicians in stress diagnosis, has evolved over time to meet domain specific challenges. The initial research direction was case-based stress diagnosis, with physiological features used as defining case parameters. Discrete wavelet transformation was used to automatically extract features from the raw cardio and pulmonary signals used by clinicians in manual diagnosis [20]. The project has since grown and expanded in many directions. When it became difficult to acquire enough cases, fuzzy rules were used to generate artificial cases for the case base, which improved diagnostic performance [1]. When imprecise sensor measurements, noise, and human error impeded the performance of the similarity metric, fuzzy similarity matching was introduced to improve robustness [4]. When it became desirable to consider the social context surrounding a patient's stress, textual features were introduced. Natural language processing was therefore integrated, using WordNet and domain specific ontological knowledge [2]. The current multi-modal system incorporates CBR, rule-based reasoning, fuzzy logic and textual information retrieval to aid in the diagnosis and treatment of stress [3].

9 Summary and Conclusion

The 4 Diabetes Support System^{TM} project stands as a case study in CBR research and development. It illustrates the power of the CBR research paradigm as a framework for medical research. 4DSS was originally envisioned as a CBR system to help patients with type 1 diabetes on insulin pump therapy achieve and maintain good blood glucose control. Over the course of seven years and three clinical research studies, a series of defining cases altered the original research and development path. After the case of the bouncing blood glucose, the medical research goals expanded to include glycemic variability measurement and automated detection of excessive glycemic variability. Meanwhile, the AI approach expanded to incorporate machine learning classification. After the case of the just-too-late problem detection, 4DSS goals were extended to real-time problem prediction and prevention. Support vector regression models for blood glucose prediction were explored and integrated. After the case of the one-fingered typist, development efforts extended to smart phone interfaces for automated data capture. A concentrated focus on the salient characteristics of each individual case has led to the discovery of new research and development challenges. Meeting these challenges could positively impact the health and wellbeing of patients with diabetes.

Acknowledgments. The authors gratefully acknowledge support from Medtronic, and from Ohio University through the Russ College Biomedical Engineering Fund, the Osteopathic College of Medicine Research and Scholarly Affairs Committee, the Research Challenge Program, and the Appalachian Rural Health Institute Diabetes Research Initiative. We would also like to thank the participating patients, research nurses, and graduate research assistants for their contributions to this work.

References

1. Ahmed, M.U., Begum, S., Funk, P., Xiong, N.: Fuzzy rule-based classification to build initial case library for case-based stress diagnosis. In: Hamza, M.H. (ed.) 9th IASTED International Conference on Artificial Intelligence and Applications (AIA), pp. 225–230 (2009)
2. Ahmed, M.U., Begum, S., Funk, P., Xiong, N., von Schéele, B.: Case-based reasoning for diagnosis of stress using enhanced cosine and fuzzy similarity. Transactions on Case-Based Reasoning on Multimedia Data 1(1), 3–19 (2008)
3. Ahmed, M.U., Begum, S., Funk, P., Xiong, N., von Schéele, B.: A multi-module case based biofeedback system for stress treatment. Artificial Intelligence in Medicine 51(2), 107–115 (2011)
4. Begum, S., Ahmed, M.U., Funk, P., Xiong, N., von Schéele, B.: A case-based decision support system for individual stress diagnosis using fuzzy similarity matching. Computational Intelligence 25(3), 180–195 (2009)
5. Bellazi, R., Montani, S., Portinale, L., Riva, A.: Integrating rule-based and case-based decision making in diabetic patient management. In: Althoff, K.D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 386–400. Springer, Heidelberg (1999)
6. Bichindaritz, I.: Case-based reasoning adaptive to several cognitive tasks. In: Veloso, M., Aamodt, A. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 391–400. Springer, Heidelberg (1995)
7. Bichindaritz, I.: Case-based reasoning in the health sciences: Why it matters for the health sciences and for CBR. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 1–17. Springer, Heidelberg (2008)
8. Bichindaritz, I., Marling, C.: Chapter 7: Case-based reasoning in the health sciences: Foundations and research directions. In: Bichindaritz, I., Jain, L.C., Vaidya, S., Jain, A. (eds.) Computational Intelligence in Healthcare 4: Advanced Methodologies, Springer, Berlin (2010)

9. Ceriello, A., Ihnat, M.A.: Glycaemic variability: A new therapeutic challenge in diabetes and the critical care setting. Diabetic Medicine 27(8), 862–867 (2010)
10. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines (2001), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm` (accessed April, 2011)
11. Diabetes Control and Complications Trial Research Group: The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. New England Journal of Medicine 329(14), 977–986 (1993)
12. Juvenile Diabetes Research Foundation: Artificial pancreas project (2011), `http://www.artificialpancreasproject.com/` (accessed April, 2011)
13. Kilpatrick, E.S., Rigby, A.S., Atkins, S.L.: For debate. Glucose variability and diabetes complication risk: We need to know the answer. Diabetic Medicine 27(8), 868–871 (2010)
14. Kovatchev, B.P., Gonder-Frederick, L.A., Cox, D.J., Clarke, W.L.: Evaluating the accuracy of continuous glucose-monitoring sensors: Continuous glucose-error grid analysis illustrated by TheraSense Freestyle Navigator data. Diabetes Care 27(8), 1922–1928 (2004)
15. Marling, C., Shubrook, J., Schwartz, F.: Case-based decision support for patients with type 1 diabetes on insulin pump therapy. In: Althoff, K.D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 325–339. Springer, Heidelberg (2008)
16. Marling, C., Shubrook, J., Schwartz, F.: Toward case-based reasoning for diabetes management: A preliminary clinical study and decision support system prototype. Computational Intelligence 25(3), 165–179 (2009)
17. Marling, C.R., Shubrook, J.H., Vernier, S.J., Wiley, M.T., Schwartz, F.L.: Characterizing blood glucose variability using new metrics with continuous glucose monitoring data. Journal of Diabetes Science and Technology (in press, 2011)
18. Maurer, K., Volk, S., Gerbaldo, H.: Chapter 1: Auguste D. The history of Alois Alzheimer's first case. In: Whitehouse, P.J., Maurer, K., Ballenger, J.F. (eds.) Concepts of Alzheimer Disease: Biological, Clinical and Cultural Perspectives. Johns Hopkins University Press, Baltimore (2000)
19. Monnier, L., Colette, C.: Glycemic variability. Diabetes Care 31(supplement 2), S150 (2008)
20. Nilsson, M., Funk, P., Olsson, E.M.G., von Schéele, B., Xiong, N.: Clinical decision support for diagnosing stress-related disorders by applying psychophysiological medical knowledge to an instance-based learning system. Journal of Artificial Intelligence in Medicine 36(2), 156–176 (2005)
21. Rodbard, D.: Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control. Diabetes Technology & Therapeutics 11(s1), S-55–S-67 (2009)
22. Rodbard, D.: New and improved methods to characterize glycemic variability using continuous glucose monitoring. Diabetes Technology & Therapeutics 11(9), 551–565 (2009)
23. Sapankevych, N.I., Sankar, R.: Time series prediction using support vector machines: A survey. IEEE Computational Intelligence Magazine 4(2), 24–38 (2009)
24. Schwartz, F.L., Shubrook, J.H., Marling, C.R.: Use of case-based reasoning to enhance intensive management of patients on insulin pump therapy. Journal of Diabetes Science and Technology 2(4), 603–611 (2008)

25. Schwartz, F.L., Vernier, S.J., Shubrook, J.H., Marling, C.R.: Evaluating the automated blood glucose pattern detection and case-retrieval modules of the 4 Diabetes Support System. Journal of Diabetes Science and Technology 4(6), 1563–1569 (2010)
26. Service, F., Molnar, G., Rosevear, J., Ackerman, E., Gatewood, L., Taylor, W.: Mean amplitude of glycemic excursions, a measure of diabetic instability. Diabetes 19(9), 644–655 (1970)
27. Slade, S.: Case-based reasoning: A research paradigm. AI Magazine 12(1), 42–55 (1991)
28. Smola, A.J., Scholkopf, B.: A tutorial on support vector regression. Tech. Rep. TR-98-030, NeuroCOLT2 Technical Report Series (1998)
29. Witten, I.H., Frank, E., Trigg, L., Hall, M., Holmes, G., Cunningham, S.J.: Weka: practical machine learning tools and techniques with Java implementations. In: Proceedings ICONIP/ANZIIS/ANNES 1999: Future Directions for Intelligent Systems and Information Sciences, pp. 192–196 (1999)
30. World Health Organization: Diabetes (2011), http://www.who.int/mediacentre/factsheets/fs312/en/index.html (accessed April, 2011)

Learning More from Experience in Case-Based Reasoning

David McSherry and Christopher Stretch

School of Computing and Information Engineering,
University of Ulster, Coleraine BT52 1SA, Northern Ireland
{dmg.mcsherry,ct.stretch}@ulster.ac.uk

Abstract. Recent concerns about the effects of feedback delays on solution quality in case-based reasoning (CBR) have prompted research interest in feedback propagation as an approach to addressing the problem. We argue in this paper that the ability of CBR systems to learn from experience in the absence of immediate feedback is limited by eager commitment to the adaptation paths used to solve previous problems. Moreover, it is this departure from lazy learning in CBR that creates the need for maintenance interventions such as feedback propagation. We also show that adaptation path length has no direct effect on solution quality in many adaptation methods and examine the implications for problem solving and learning in CBR. For such "path invariant" adaptation methods, we demonstrate the effectiveness of a "lazier" approach to learning/problem solving in CBR that avoids commitment to previous adaptation paths and hence the need for feedback propagation.

Keywords: Case-based reasoning, lazy learning, adaptation.

1 Introduction

In case-based reasoning (CBR), a target problem is solved by adapting the solution from the most similar case, or simply by reusing the solution from the most similar case without adaptation [1,2]. It is this *lazy* approach to learning/problem solving that distinguishes CBR from *eager* learning algorithms that create abstractions such as decision trees from training data [3]. Another important feature of CBR is the ability to learn from experience as new cases are added to the case base. However, there is increasing awareness of the effects of feedback delays on solution quality in CBR, and maintenance strategies for addressing this problem have been proposed by several authors [4–6].

For example, Leake and Whitehead [4] investigate several approaches to propagating feedback, when received for a given case, to related and/or similar cases. In one such algorithm, feedback is propagated to all *descendants* of the reference case (defined as those cases that were generated from the reference case by a series of adaptations). Feedback propagation is guided by *case provenance* information captured by the system as each new case is added to the case base (i.e., the set of cases that contributed, directly or indirectly, to the new case's solution). The aim of feedback propagation is to reduce the effects of feedback delays on solution quality. Another problem associated with lack of feedback in CBR is that the solution for a given problem may depend on the order in which previous cases were added to the case base [4].

In this paper, we examine in depth some of the issues brought to light by recent work on case provenance and feedback propagation. The aim of our analysis is to provide a better understanding of the CBR process in the absence of immediate (or any) feedback and its susceptibility to the problems noted by Leake and his co-workers [4–6]. We argue that the ability of CBR systems to learn from experience in the absence of immediate feedback is limited by eager commitment to adaptation paths that determine case provenance but may prove to be sub-optimal in future problem solving. Moreover, it is this departure from lazy learning in CBR that creates the need for maintenance interventions such as feedback propagation. It is also a primary cause of the "order dependence" problem noted by Leake and Whitehead [4].

In previous work, we proposed a "lazier" approach to learning/problem solving in CBR, called Lazier CBR, which uses breadth-first search to discover the shortest possible adaptation path from a seed case (or other case whose solution is known to be correct) to a given problem [7]. The discovered adaptation path is then used to solve the target problem, a process that may involve generating new solutions for some of the cases in the path. In this way, Lazier CBR avoids commitment to previous adaptation paths, and hence the need for feedback propagation. An underlying hypothesis in the approach is that using the shortest available adaptation path may provide more accurate solutions in situations where solution quality tends to deteriorate as the lengths of adaptation paths increase.

However, we show in this paper that many adaptation methods are "path invariant", in the sense that any adaptation path from a given seed case to a target problem gives the same solution as adapting the seed case directly to solve the target problem. An important consequence is that adaptation path length has no direct effect on solution quality for path invariant adaptation methods. Moreover, a common feature of the path invariant adaptation methods we identify is that any case can be adapted to solve a given problem. In this situation, any seed case C provides an adaptation path $C \rightarrow P$ of the shortest possible length that can be used to solve a given problem P.

An alternative to Lazier CBR that we propose in light of this analysis, and show to be effective for a variety of estimation and classification tasks, is an even lazier approach called Lazier$^+$ CBR. In Lazier$^+$ CBR, a target problem is solved by adapting the most similar seed case (or other case whose solution is known to be correct). For path invariant adaptation methods, Lazier$^+$ CBR avoids the computational effort required for adaptation path discovery in Lazier CBR. It is also more efficient than traditional CBR in that cases with unconfirmed solutions play no part in the solution of new problems, and so do not contribute to retrieval effort.

In Sections 2 and 3, we highlight the issues that Lazier$^+$ CBR aims to address, such as the limited ability of CBR systems to learn from experience in the absence of immediate (or any) feedback. In Section 4, we show that path invariance is a property shared by many common adaptation methods, and examine the implications for CBR problem solving and learning in estimation and classification tasks. In Section 5, we examine the hypothesis that for path invariant adaptation methods, a Lazier$^+$ CBR system learns more effectively from experience, in the absence of feedback, than a traditional CBR system. Our conclusions are presented in Section 6.

2 Adaptation Paths in CBR

In this section, we introduce the basic concepts in our analysis and examine the role of adaptation paths in a traditional CBR system. To simplify the discussion, we do not consider CBR approaches in which two or more retrieved cases may contribute directly to the solution of a target problem, for example as in CBR approaches to estimation based on adaptation triples [8].

Seed and Non-Seed Cases. Before a CBR system can begin to solve new problems, it must first be provided with one or more "seed" cases with solutions that are known to be correct. Seed cases are typically provided by a domain expert or imported as legacy cases. We will refer to cases created by the system (i.e., by adapting an existing case to solve a new problem and retaining the problem and its solution as a new case) as "non-seed" cases.

Case = Problem + Solution. For any case C, we will denote by *problem*(C) the problem represented by C. We will denote by *solution*(C) the solution for C that is stored in the case base, whether or not the stored solution is correct.

Adapted Solution. For any problem P and case C that can be adapted to solve P, we will denote by *adapted-solution*(C, P) the solution for P obtained by adapting C to solve P.

Adaptation Path. A sequence of cases C_1, ..., C_n provides an adaptation path $C_1 \rightarrow \ldots \rightarrow C_n \rightarrow P$ from a seed case C_1 to a given problem P if C_i can be adapted to solve *problem*(C_{i+1}) for $1 \leq i \leq n - 1$ and C_n can be adapted to solve P. Note that the solution for a given case in an adaptation path that results from previous adaptations in the path may differ from its solution in the case base, which may have originated from a different adaptation path.

A traditional CBR system does not consider all possible adaptation paths that could be used to solve a given problem P. Instead, it retrieves the most similar case C that can be adapted to solve P and uses it to solve the problem. However, if C is a non-seed case then as we show in Theorem 1 there exists a seed case C_1 and adaptation path $C_1 \rightarrow \ldots \rightarrow C_n \rightarrow$ *problem*(C) that was used to solve problem(C). By adapting C to solve P, the system extends the adaptation path that it used to solve *problem*(C) to create a new adaptation path $C_1 \rightarrow \ldots \rightarrow C_n \rightarrow C \rightarrow P$ that now determines the solution for P.

In the proof of Theorem 1, we assume there is no deletion of cases, for example for maintenance purposes [9,10].

Theorem 1. *For any non-seed case C, there exists a seed case C_n and adaptation path $C_n \rightarrow \ldots \rightarrow C_1 \rightarrow$ problem(C) that determines the solution for C.*

Proof. If C is a non-seed case, then *problem*(C) must have been solved by adapting another case C_1, namely the adaptable case that was most similar to *problem*(C) when that problem was presented to the system. If C_1 is a seed case, then the required adaptation path is $C_1 \rightarrow$ *problem*(C). If C_1 is not a seed case, then *problem*(C_1) must have been solved by adapting another case C_2. If C_2 is a seed case, then the required adaptation path is $C_2 \rightarrow C_1 \rightarrow problem(C)$. If C_2 is not a seed case, then we can continue as long as necessary to build a sequence of cases $C_1, \ldots, C_n$ such that for $1 \leq i \leq n - 1$, C_{i+1} is the case that was adapted to solve $problem(C_i)$. Moreover, C_1 was already in the case base before C was created, and C_{i+1} was already in the case base before C_i was created for $1 \leq i \leq n - 1$. It follows that $C, C_1, \ldots, C_n$ must all be distinct cases. We also know that there can only be a finite number of cases in the case base, and at least one of them must be a seed case. So as we continue to extend our sequence of cases, it must eventually be true that the last case in the sequence (C_n) is a seed case. We have now established as required the existence of a seed case C_n and adaptation path $C_n \rightarrow \ldots \rightarrow C_1 \rightarrow problem(C)$ that determines the solution for C. □

3 Why Lazier$^+$ CBR?

Fig. 1 shows an example case base and a target problem (P) that we use in this section to highlight the issues that Lazier$^+$ CBR aims to address. There are just two attributes in the description of a case with integer values in the range from 0 to 6. Existing cases are numbered in the order in which they were added to the case base and seed cases (C_1, C_7, C_8) are shown as black circles. The adaptation paths used to generate the non-seed cases C_3, C_6, and C_{12} are also shown. For example, the adaptation path that determines the solution for C_6 is $C_1 \rightarrow C_4 \rightarrow C_5 \rightarrow problem(C_6)$.

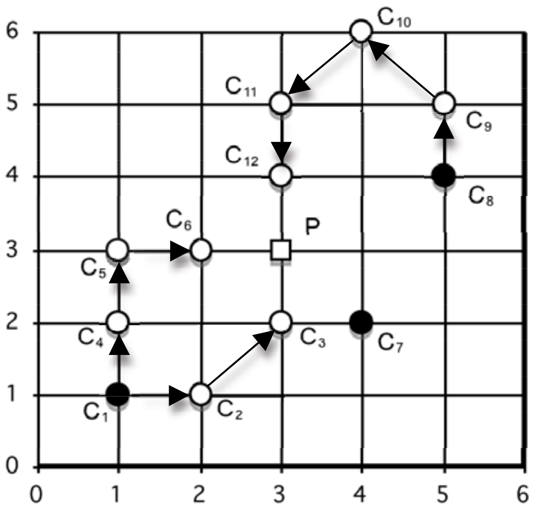


Fig. 1. An example case base in which 3 cases (C_3, C_6, C_{12}) are equally similar to a target problem P

In a traditional CBR system that adapts the most similar case, C_3, C_6, and C_{12} are equally good candidates to be used to solve P. It might be considered that C_3 would be the best choice because it was generated from a seed case by the shortest adaptation path (2 steps). Adapting C_3 to solve P amounts to extending the adaptation path $C_1 \rightarrow C_2 \rightarrow problem(C_3)$ that was used to solve $problem(C_3)$ to create a new adaptation path $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow P$. There are only 3 steps in the resulting adaptation path as opposed to an adaptation path length of 4 if C_6 is used to solve P or 5 if C_{12} is used to solve P. So using C_3 to solve P can be expected to give better results than C_6 or C_{12} if solution quality is known to deteriorate as the lengths of adaptation paths increase.

However, we show in Section 4 that for many adaptation methods used in estimation and classification tasks, the length of the adaptation path that determines the solution for a given problem has no bearing on solution quality except insofar as problems with longer adaptation paths tend to be less similar to the seed cases from which they were generated. For such "path invariant" adaptation methods, it does not matter whether C_3 or C_6 is adapted to solve P, as both cases will give the same solution. What matters in path invariant adaptation is not the length of the adaptation path, but how similar the target problem is to the seed case that determines its solution. In this context, adapting C_{12} to solve P is likely to give better results than C_3 or C_6 because the seed case from which C_{12} was generated (C_8) is more similar to P than the one from which C_3 and C_6 were generated (C_1).

A traditional CBR system will simply make some arbitrary choice between the equally similar cases C_3, C_6, and C_{12}. It will also ignore the fact that the seed case C_7 is much closer to the target problem than either of the seed cases from which C_3, C_6, and C_{12} were generated. A related issue that CBR researchers have recently begun to consider in the context of delayed/absent feedback is that the solution for a given problem may depend on the order in which previous cases were added to the case base [4].

Definition 1. *The solution that a CBR system provides for a given problem is order dependent if it depends on the order in which cases in existence at the time when the problem is solved were added to the case base.*

For example, if C_7 had been added to the example case base in Fig. 1 before *problem*(C_3) was solved, then the system would have used C_7 instead of C_2 to solve *problem*(C_3). As a result, the adaptation path that determines the solution of C_3 would now be $C_7 \rightarrow problem(C_3)$, which is likely to result in a more accurate solution than $C_1 \rightarrow C_2 \rightarrow problem(C_3)$. Moreover, if the system chooses to solve P by adapting C_3, the adaptation path now used to determine the solution for P would be $C_7 \rightarrow C_3 \rightarrow P$, which is likely to result in a more accurate solution than $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow P$. With the original ordering of cases, what prevents the system from benefiting from the addition of C_7 as a new seed case is its eager commitment to the adaptation path that it used to solve *problem*(C_3).

In a traditional CBR system, no record is kept of the adaptation paths that determine case provenance. However, the fact remains that these adaptation paths are continually reused by the system to solve new problems, with no attempt to improve or revise them as new cases are created. As the above example illustrates, this eager commitment to previous adaptation paths may limit a CBR system's ability to learn from experience. More generally, if the most similar case C to a target problem P is a seed case, then it is reasonable to expect that adapting C to solve P will provide a good solution for P. However, if C is not a seed case, there may now be several possible adaptation paths from seed cases to C that did not exist at the time when *problem*(C) was solved. Moreover, it is possible that one of these alternative adaptation paths, which the system does not consider, could provide a better solution for C, and could be extended to provide a better solution for P.

In contrast, Lazier$^+$ CBR makes no commitment to adaptation paths used to solve previous problems (which never involve more than a single adaptation step in the approach). Instead, it only allows seed cases, or other cases whose solutions are known to be correct as a result of feedback from a reliable source, to contribute to the solution of new problems. In Lazier$^+$ CBR, the target problem P in Fig. 1 would be solved by adapting C_7, the seed case that is most similar to P. Whether or not the adaptation method is path invariant, this is likely to provide a better solution for P than adapting C_3, C_6, or C_{12}. It also avoids the order dependence problem to which traditional CBR is known to be susceptible [4].

4 Path Invariant Adaptation

In this section, we show that many of the adaptation methods typically used in estimation and classification tasks are path invariant according to the following definition.

Definition 2. *An adaptation method is path invariant if any adaptation path* $C_1 \to \ldots \to C_n \to P$ *from a seed case* C_1 *to a target problem* P *gives the same solution for* P *as adapting* C_1 *directly to solve* P.

One example of path invariant adaptation is the approach sometimes referred to as *null* adaptation [11]. In this approach, the solution for the most similar case is reused (i.e., applied to a target problem) without any adaptation. Its use tends to be limited to estimation/classification tasks where the need for adaptation is less critical than in design/configuration tasks [2]. In practice, the most similar case may be required to equal or exceed a predefined similarity threshold before its solution is applied to the target problem without adaptation.

In Section 4.1, we demonstrate the path invariance of two approaches to transformational adaptation for a specific estimation task. In Section 4.2, we consider the implications of path invariant adaptation in a more general context.

4.1 Adapting Soccer Scores

Consider the idea of using a traditional CBR system to estimate the points scored by a soccer team from the numbers of matches it has won and drawn in a series of matches. We used a similar example in previous work on intelligent case authoring [12]. The correct solution for any case C, though unknown to the CBR system, is:

$$points(C) = 3 \times wins(C) + draws(C) \qquad (1)$$

In addition to one or more seed cases with correct solutions, the CBR system in our example is provided with the following rules for adapting the solution from the most similar case (C) to solve a given problem P:

Rule 1. **If** $wins(P) > wins(C)$ **then** add $k_1 \times (wins(P) - wins(C))$
Rule 2. **If** $wins(P) < wins(C)$ **then** subtract $k_1 \times (wins(C) - wins(P))$
Rule 3. **If** $draws(P) > draws(C)$ **then** add $k_2 \times (draws(P) - draws(C))$
Rule 4. **If** $draws(P) < draws(C)$ **then** subtract $k_2 \times (draws(C) - draws(P))$

The method used to assess the similarity of a given case C to a target problem P is not important in this discussion, but might for example be based on the Euclidean distance:

$$\sqrt{(wins(P) - wins(C))^2 + (draws(P) - draws(C))^2} \tag{2}$$

The CBR system solves a target problem by adapting the most similar case. It uses all applicable adaptation rules, with cumulative effect, to adapt the solution for the most similar case. The accuracy of solutions based on the adaptation rules will depend on the values of k_1 and k_2, reflecting the fact that, in practice, such rules may be based on imperfect domain knowledge. For example, Fig. 2 shows a target problem (P) with $wins(P) = 2$ and $draws(P) = 3$. The goal is to estimate $points(P)$, the points scored by a team with these numbers of wins and draws. The most similar case (C) is also shown. Its solution (11) can be seen to be correct from Eqn. 1.

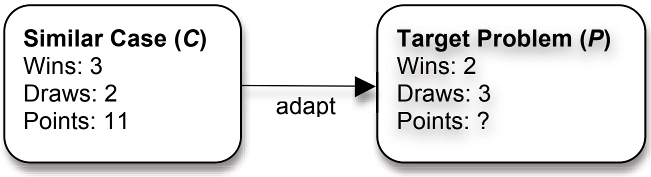


Fig. 2. Adapting the most similar case (C) to solve a target problem (P) in the soccer scores domain

We also know from Eqn. 1 that the correct solution for the target problem is $3 \times 2 + 3 = 9$. However, when Rules 1–4 are applied with $k_1 = 2$ and $k_2 = 1$, the adapted solution for P is $11 - 2 + 1 = 10$. As the solution for the most similar case is correct, it is only the adaptation process that contributes to the error in the solution for P. With $k_1 = 3$ and $k_2 = 1$, Rules 1–4 are guaranteed to give the correct solution for any problem provided the solution for the most similar case is correct. The adaptation rules can be written more concisely as an adaptation *formula* for adapting a case C to solve a given problem P:

$$points(P) = points(C) + k_1 \times (wins(P) - wins(C)) + k_2 \times (draws(P) - draws(C)) \tag{3}$$

As we show in Theorem 2, adaptation based on Eqn. 3 (or the equivalent rules) is path invariant.

Theorem 2. *In the soccer scores domain, the adaptation formula* $points(P) = points(C) + k_1 \times (wins(P) - wins(C)) + k_2 \times (draws(P) - draws(C))$ *is path invariant for all values of* k_1 *and* k_2.

Proof. For any seed case C_1, problem P, and adaptation path $C_1 \rightarrow \ldots \rightarrow C_n \rightarrow P$, the solution for P obtained by applying the adaptation formula at each step of the adaptation path is:

$points(P) = points(C_n) + k_1 \times (wins(P) - wins(C_n)) + k_2 \times (draws(P) - draws(C_n)) =$

$points(C_{n-1}) + k_1 \times (wins(C_n) - wins(C_{n-1})) + k_2 \times (draws(C_n) - draws(C_{n-1})) + k_1 \times (wins(P) - wins(C_n)) + k_2 \times (draws(P) - draws(C_n)) =$

$points(C_1) + k_1 \times (wins(C_2) - wins(C_1)) + k_2 \times (draws(C_2) - draws(C_1)) + k_1 \times (wins(C_3) - wins(C_2)) + k_2 \times (draws(C_3) - draws(C_2)) + \ldots + k_1 \times (wins(C_n) - wins(C_{n\text{-}1})) + k_2 \times (draws(C_n) - draws(C_{n\text{-}1})) + k_1 \times (wins(P) - wins(C_n)) + k_2 \times (draws(P) - draws(C_n)) =$

$points(C_1) + k_1 \times (wins(P) - wins(C_1)) + k_2 \times (draws(P) - draws(C_1))$.

That is, the solution provided by any adaptation path from C_1 to P is the same as the solution obtained by adapting C_1 directly to solve P. It follows as required that the adaptation formula is path invariant for all values of k_1 and k_2. □

Fig. 3 shows an example of path invariant adaptation based on Eqn. 3 (or the equivalent adaptation rules) in the soccer scores domain with $k_1 = 2$ and $k_2 = 1$. In this example, adapting Case 1 directly to solve the target problem gives the same solution (16) as adapting Case 1 to solve the problem represented by Case 2 and then adapting Case 2 to solve the target problem.

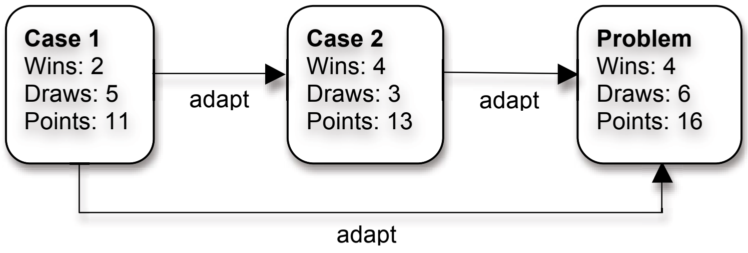


Fig. 3. Path invariant adaptation in the soccer scores domain

Another possible adaptation formula in the soccer scores domain is the following:

$$points(P) = \frac{1 + wins(P)}{1 + wins(C)} \times points(C) \qquad (4)$$

Though taking no account of the *draws* attribute, Eqn. 4 captures the idea that points scored can be expected to increase as the number of wins increases. Adding one to $wins(C)$ in the formula avoids the risk of division by zero, and ensures that any case can be adapted to solve a given problem. Also adding one to $wins(P)$ ensures that no adjustment is made when $wins(P) = wins(C)$. When Eqn. 4 is used to adapt the most similar case in Fig. 2, the solution for the target problem is $\frac{1+2}{1+3} \times 11 = 8.25$.

A similar approach to adaptation, though based on attributes with non-zero values, was used by Leake and Whitehead [4] in experiments on the Abalone and Boston Housing datasets from the UCI Machine Learning Repository [13].

Theorem 3. *In the soccer scores domain, the adaptation formula* $points(P) = \frac{1 + wins(P)}{1 + wins(C)} \times points(C)$ *is path invariant.*

Proof. For any seed case C_1, problem P, and adaptation path $C_1 \rightarrow \ldots \rightarrow C_n \rightarrow P$, the solution for P obtained by applying the adaptation formula at each step in the adaptation path is:

$$\frac{1+wins(P)}{1+wins(C_n)} \times points(C_n) = \frac{1+wins(P)}{1+wins(C_n)} \times \frac{1+wins(C_n)}{1+wins(C_{n-1})} \times points(C_{n-1}) =$$
$$\frac{1+wins(P)}{1+wins(C_n)} \times \frac{1+wins(C_n)}{1+wins(C_{n-1})} \times \ldots \times \frac{1+wins(C_2)}{1+wins(C_1)} \times points(C_1) =$$
$$\frac{1+wins(P)}{1+wins(C_1)} \times points(C_1)$$

That is, the solution provided by any adaptation path from C_1 to P is the same as the solution obtained by adapting C_1 directly to solve P. It follows as required that the adaptation formula is path invariant. □

4.2 Path Invariance in General

It might be considered that path invariance is an unusual property of the adaptation methods that we discussed in the soccer scores domain. However, the proof of Theorem 2 can be generalized to show that adaptation is path invariant for any CBR estimation task, numeric attributes $a_1, \ldots, a_r$, coefficients $k_1, \ldots, k_r$, and adaptation formula:

$$adapted\text{-}solution(C, P) = solution(C) + \sum_{i=1}^{r} (k_i \times (\pi_i(P) - \pi_i(C))) \tag{5}$$

where $\pi_i(P)$ and $\pi_i(C)$ are the values of a_i in P and C respectively.

Adaptation is also path invariant for any CBR estimation task and adaptation formula:

$$adapted\text{-}solution(C, P) = \frac{\pi_a(P)}{\pi_a(C)} \times solution(C) \tag{6}$$

where a is a numeric attribute with non-zero values and a direct relationship to the solution attribute, and $\pi_a(P)$ and $\pi_a(C)$ are the values of a in P and C respectively. As previously mentioned, null adaptation [11] is also path invariant.

As discussed in Section 3, the fact that adaptation path length has no direct effect on solution quality for path invariant adaptation methods is one of the motivating factors in our investigation of Lazier$^+$ CBR as an approach to addressing the problems caused by eager commitment to previous adaptation paths in traditional CBR. In Section 5, we examine the hypothesis that for path invariant adaptation methods, a Lazier$^+$ CBR system learns more effectively from experience in the absence of feedback than a traditional CBR system.

5 Empirical Study

In the experiments reported in this section, we assess the ability of a target CBR system to learn from experience by tracking its performance over time as new cases are added to an initially empty case base. The performance measures of interest are percentage accuracy for classification tasks and mean absolute error (MAE) for estimation tasks. We use one or other of these measures to construct a *learning curve* that shows how effectively the system learns from experience. In each experiment, we use a given dataset as a source of seed cases and problems to be solved by the target CBR system. As described in Section 5.1, the proportion of seed cases ($1/r$) in the case base is determined by an integer parameter $r \geq 2$ and remains constant at each of a series of evaluation points. In Sections 5.2 to 5.4, we use this framework to compare the performance of Lazier^{+} CBR and traditional CBR for a variety of estimation and classification tasks in the absence of feedback.

5.1 Experimental Method

Beginning with an initially empty case base, and a given dataset of size n, we repeat the following steps until the size of the case base reaches $k \times r$, where k is the largest integer such that $k \times r \leq n$.

1. Select an example (description + solution) at random from the dataset and insert it as a seed case into the case base.
2. Remove the selected example from the dataset.
3. Select $r - 1$ examples at random from the dataset and present their descriptions (one at a time) as problems to be solved by a target CBR system. As each problem is solved, add its description and the CBR system's solution to the case base as a new case before the next problem is solved.
4. Remove the examples selected in Step 3 from the dataset.
5. Calculate the percentage accuracy or MAE of the CBR system's (unrevised) solutions over all non-seed cases that are now in the case base.

For example, the system's solution for a non-seed case is deemed to be correct in a classification task if it is the same as the known solution from the dataset. At each of the k evaluation points (Step 5), the proportion of seed cases in the case base is $1/r$. Fig. 4 illustrates our approach to constructing a case base from a given dataset. In this example, $r = 3$ and evaluation points are shown as double vertical lines (||). Seed cases are shown as dark circles. The diagram also shows the adaptation paths that determine the solutions for the first six non-seed cases to be added to the case base in a traditional CBR system. However, this provenance information is not used in our experiments.

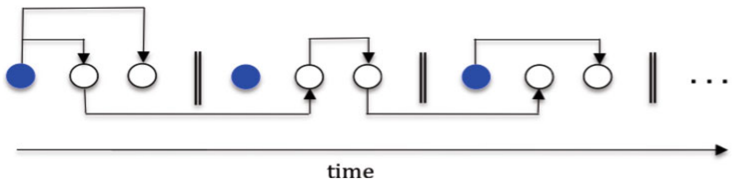


Fig. 4. An example case base incrementally constructed by adding cases to an initially empty case base in groups of $r = 3$ cases

All attributes are equally weighted for the purpose of similarity assessment in our experiments. We define the similarity of two values *x* and *y* of a numeric attribute *a* to be $sim_a(x,y) = 1 - \frac{|x-y|}{max(a) - min(a)}$, where *max*(*a*) and *min*(*a*) are the maximum and minimum values of *a* in the given dataset. We define the similarity of two values of a nominal attribute to be 0 if one or both values are missing. Otherwise, we assign a similarity score of 1 for equal values or 0 for unequal values.

5.2 Estimation in the Soccer Scores Domain

Our first experiment is based on an artificial dataset in the soccer scores domain (Section 4). The dataset contains 169 examples, one for each value of *wins* and *draws* from 0 to 12. The correct solution ($points = 3 \times wins + draws$) is stored with each example description. The resulting dataset provides a source of seed cases and problems to be solved in the construction of a case base and evaluation of a target CBR system as described in Section 5.1. The goal of the CBR system is to estimate the points scored. Adaptation is based in the experiment on the following adaptation formula, which we know to be path invariant from Theorem 2, and in which *C* is the case adapted to solve a target problem *P*.

$$points(P) = points(C) + 2 \times (wins(P) - wins(C)) + 1 \times (draws(P) - draws(C)) \quad (7)$$

Fig. 5 shows the resulting learning curves for $r = 4$ in a traditional CBR system and Lazier$^+$ CBR system. For this value of *r*, the proportion of seed cases at each evaluation point is 1/4. The MAE at each evaluation point is averaged over 100 trials. In Lazier$^+$ CBR, the MAE decreases rapidly to a minimum of 1.2 when the number of seed cases reaches 36. In contrast, it is only after 10 seed cases have been added to the case base that effective learning begins in traditional CBR. Moreover, the lowest MAE achieved by the traditional CBR system is 3.2 compared to 1.2 for the Lazier$^+$ CBR system.

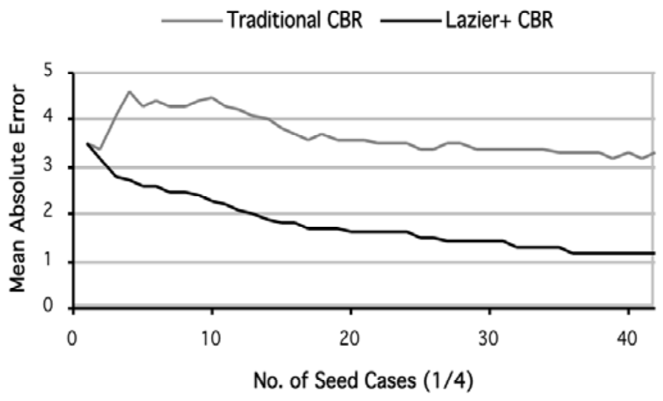


Fig. 5. Learning curves based on mean absolute error (MAE) for traditional CBR and Lazier$^+$ CBR in the soccer scores domain

Table 1. Lowest mean absolute error (MAE) achieved by traditional and Lazier$^+$ CBR systems in the soccer scores domain as the proportion of seed cases increases from 1/10 to 1/2.

	Proportion of Seed Cases (1/*r*)				
	1/10	1/5	1/4	1/3	1/2
Traditional CBR	4.1	3.1	3.2	2.4	1.7
Lazier$^+$ CBR	1.7	1.2	1.2	1.0	0.9

To assess the proportions of seed cases (1/*r*) required for effective learning in traditional CBR and Lazier$^+$ CBR, we repeated the experiment with different values of *r*. Table 1 shows the lowest MAE achieved by traditional CBR and Lazier$^+$ CBR systems as the proportion of seed cases increases from 1/10 to 1/2. The Lazier$^+$ CBR system can be seen to require much fewer seed cases for effective learning than the traditional CBR system. For example, the traditional CBR system requires a proportion of seed cases 5 times greater than the Lazier$^+$ CBR system to achieve a minimum MAE of 1.7.

5.3 Estimating Housing Values

The case base used in our second experiment is constructed from the Boston Housing dataset from the UCI Machine Learning Repository [13]. The dataset contains 506 examples, each representing a residential area in the Boston suburbs described by 13 attributes such as average number of rooms (RM) and distance to employment centers (DIS). The goal is to estimate the median value of owner-occupied homes (MEDV) in a given area. The dataset is used as a source of seed cases and problems to be solved by a target CBR system as described in Section 5.1. All attributes in the dataset are used in the experiment to assess the similarity of a given case *C* to a target problem *P*. Adaptation is based in the experiment on the adaptation formula:

$$\mathrm{MEDV}(P) = \frac{\mathrm{RM}(P)}{\mathrm{RM}(C)} \times \mathrm{MEDV}(C) \tag{8}$$

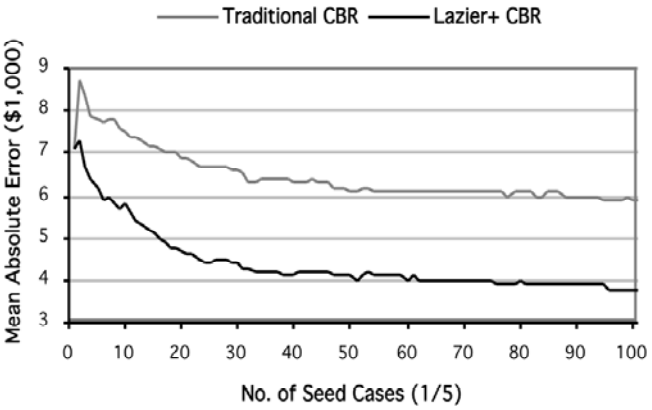


Fig. 6. Learning curves based on mean absolute error (MAE) for traditional CBR and Lazier$^+$ CBR in a case base generated from the Boston Housing dataset

Leake and Whitehead [4] used a similar approach to adaptation in experiments on the Boston Housing and Abalone datasets.

Fig. 6 shows the learning curves for $r = 5$ in a traditional CBR system and Lazier$^+$ CBR system. For this value of r, the proportion of seed cases at each evaluation point is 1/5. The MAE at each evaluation point is averaged over 10 trials. Initially, the Lazier$^+$ CBR system can be seen to learn at a much faster rate than the traditional CBR system. However, both learning curves tend to level off soon after the halfway stage is reached. From this point, the MAE for Lazier$^+$ CBR remains fairly constant at about two thirds of the MAE for traditional CBR. The lowest MAE achieved by the traditional CBR system is 5.9 compared to 3.8 for the Lazier$^+$ CBR system.

5.4 Classification with Null Adaptation

The case base in our final experiment is constructed from the Congressional Voting Records dataset from the UCI Machine Learning Repository [13]. Examples in the dataset represent the votes of 435 US Congressmen on 16 key issues as well as their political affiliations (Democrat/Republican). The goal in this classification task is to predict political affiliation from voting behavior. The dataset is used as a source of seed cases and problems to be solved by a target CBR system as described in Section 5.1. Adaptation is based in the experiment on null adaptation (i.e., the solution for the retrieved case is reused without adaptation) [11].

Fig. 7 shows the learning curves for $r = 3$ in a traditional CBR system and Lazier$^+$ CBR system. For this value of r, the proportion of seed cases at each evaluation point is 1/3. Accuracy levels are shown as percentages averaged over 10 trials. The fastest learning rates in both traditional and Lazier$^+$ CBR can be observed in the early stages of case base growth (i.e., as the number of seed cases increases from 1 to 15). Thereafter, both systems continue to learn at a slower rate until the number of seed cases reaches 100. During this phase, the difference in classification accuracy between the two systems remains fairly constant at around 4% in favor of Lazier$^+$ CBR. The maximum accuracy achieved by the traditional CBR system is 85% compared to 89% for the Lazier$^+$ CBR system. Lazier$^+$ CBR also exhibits a faster learning rate in the early stages of case-base growth than traditional CBR.

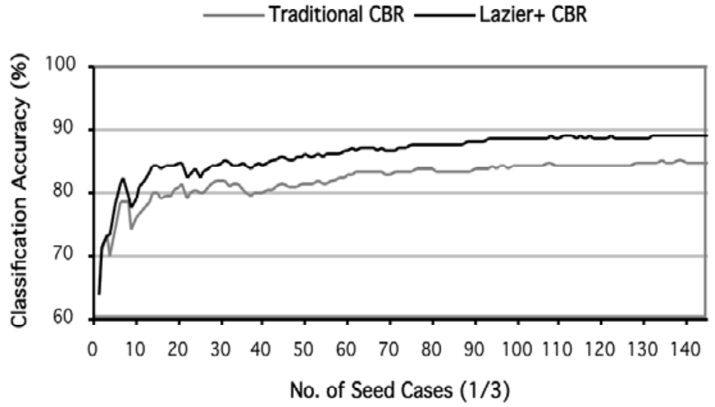


Fig. 7. Learning curves based on classification accuracy for traditional CBR and Lazier$^+$ CBR in a case base generated from the Congressional Voting Records dataset

Like the other results presented in this section, these results support our hypothesis that for path invariant adaptation methods, a Lazier$^+$ CBR system learns more effectively from experience in the absence of feedback than a traditional CBR system.

6 Conclusions

We argued in this paper that the ability of traditional CBR systems to learn from experience in the absence of immediate (or any) feedback is limited by eager commitment to the adaptation paths used to solve previous problems. This departure from lazy learning in CBR is also a primary cause of the order dependence problem brought to light by recent research on case provenance and feedback propagation [4]. We also showed that many adaptation methods used in estimation and classification tasks are path invariant in the sense that any adaptation path from a given seed case to a target problem gives the same solution as adapting the seed case directly to solve the target problem. Importantly, adaptation path length has no direct effect on solution quality for path invariant adaptation methods.

In light of this analysis, we investigated an alternative approach to learning/problem solving in CBR called Lazier$^+$ CBR that avoids commitment to previous adaptation paths, and hence the need for feedback propagation, by allowing only seed cases, or other cases with confirmed solutions, to contribute to the solution of new problems. We also demonstrated the effectiveness of Lazier$^+$ CBR for a variety of estimation and classification tasks based on path invariant adaptation methods. In the estimation/classification tasks that we studied, Lazier$^+$ CBR consistently outperformed traditional CBR in terms of mean absolute error/percentage accuracy. Lazier$^+$ CBR also exhibited faster learning rates and required fewer seed cases for effective learning than traditional CBR. Moreover, Lazier$^+$ CBR is more efficient than traditional CBR because cases with unconfirmed solutions do not contribute to retrieval effort in Lazier$^+$ CBR. In practice, such cases can be stored separately until such time as feedback on their solutions is received.

Investigation of other CBR tasks and adaptation methods that may benefit from Lazier$^+$ CBR, or variations of the basic approach, is an important direction for our future research in this area.

References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications 7, 39–59 (1994)
2. López de Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, Reuse, Revision and Retention in Case-Based Reasoning. Knowledge Engineering Review 20, 215–240 (2005)
3. Aha, D.W.: The Omnipresence of Case-Based Reasoning in Science and Application. Knowledge-Based Systems 11, 261–273 (1998)

4. Leake, D., Whitehead, M.: Case Provenance: The Value of Remembering Case Sources. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 194–208. Springer, Heidelberg (2007)
5. Leake, D., Dial, S.A.: Using Case Provenance to Propagate Feedback to Cases and Adaptations. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 255–268. Springer, Heidelberg (2008)
6. Leake, D., Kendall-Morwick, J.: External Provenance, Internal Provenance, and Case-Based Reasoning. In: Marling, C. (ed.) ICCBR 2010 Workshop Proceedings. TR-INF-2010-06-03-UNIPMN, pp. 87–94. University of Piemonte Orientale A. Avogadro (2010)
7. McSherry, D.: Towards a Lazier Approach to Problem Solving in Case-Based Reasoning. In: Marling, C. (ed.) ICCBR 2010 Workshop Proceedings. TR-INF-2010-06-03-UNIPMN, pp. 95–101. University of Piemonte Orientale A. Avogadro (2010)
8. McSherry, D.: Demand-Driven Discovery of Adaptation Knowledge. In: 16th International Joint Conference on Artificial Intelligence, pp. 222–227. Morgan Kaufmann, San Francisco (1999)
9. Smyth, B., McKenna, E.: Competence Models and the Maintenance Problem. Computational Intelligence 17, 235–249 (2001)
10. Watson, I.: Applying Case-based Reasoning: Techniques for Enterprise Systems. Morgan Kaufmann, San Francisco (1997)
11. Wilke, W., Bergmann, R.: Techniques and Knowledge used for Adaptation during Case-Based Problem Solving. In: del Pobil, A.P., et al. (eds.) IEA/AIE 1998. LNCS(LNAI), vol. 1416, pp. 497–506. Springer, Heidelberg (1998)
12. McSherry, D.: Automating Case Selection in the Construction of a Case Library. Knowedge Based Systems 13, 133–140 (2000)
13. Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences (2010)

Acquiring Adaptation Cases for Scientific Workflows

Mirjam Minor and Sebastian Görg

University of Trier,
Department of Business Information Systems, D-54286 Trier, Germany
{minor,goergs}@uni-trier.de
http://www.wi2.uni-trier.de/

Abstract. This paper addresses the automated acquisition of adaptation cases for the modification of scientific workflows. Pairs of workflow versions from community repositories are analysed to extract transformation pathways from one workflow version to another. An algorithmic solution is provided and investigated by experiments with promising results.

Keywords: scientific workflows, workflow reasoning, adaptation.

1 Introduction

Scientific workflows [3] are dedicated to support data-intensive scientific experiments by applying workflow technology. A workflow organizes work by *tasks* that descibe a human activity or a computational step. The workflow arranges the tasks and the according data in a certain execution order. This order can be specified by data dependencies forming the *data flow* as well as by routing constructs like sequences, parallel branches, sub-workflows and multiple instances that govern the *control flow* of execution. Prominent application areas for scientific workflows are molecular biology, astronomy, geology, or atom physics. Sample tasks are data transformation and analysis steps or data extraction steps like accessing measurement readings from an external database. The workflow technology controls the execution of the experimental process and operates large parts of it *in silico*, i.e. in the computer [8]. In business workflow scenarios, human tasks are rather prevalent while scientific workflows use to be computational workflows that consist exclusively of computational steps. An increasing number of Web services is available for computational steps from different fields like genome analysis in bioinformatics. BLAST (Basic Local Alignment Search Tool)[1], for example, provides many Web services on the analysis of biological sequences. It has several benefits to represent an experiment formally as a workflow instead of copying the outputs of one computational step to the inputs of another step by hand or by a program written in a script language. The scientists are enabled "to focus on domain-specific (science) aspects of their work,

[1] http://blast.ncbi.nlm.nih.gov/Blast

rather than dealing with complex data management and software issues" [8, p. 32], the execution of the experiments can be optimized on available ressources in a distributed environment, the provenance of the output data can be recorded, and the scientific workflows can be shared and reused.

Workflow reasoning is recently an emerging research field that provides automated methods for reasoning about workflows and execution traces [7,12,6,2]. This work addresses the automated adaptation of scientific workflows as reasoning method and particularly the automated acquisition of adaptation cases for scientific workflows. An *adaptation case* records experience from a previous workflow adaptation episode. Adaptation cases occur frequently in scientific workflows. For instance, an update of a Web service with slightly different input parameters may require additional data transformation steps in the scientific workflow. This is a sample for an adaptation case that occurs during workflow modeling at build time. An adaptation episode may also occur during workflow execution at run time caused by an unforeseen event, for instance if the quality of an intermediary result is not sufficient. Additional steps may be inserted, for instance a split of the data into two subsets before re-running some computational steps. Of course, changes at run time are only feasible in case of an agile workflow system [11] that is able to continue the execution of workflows that have been adapted. In our previous work on workflow reasoning [11,10], we investigated agile workflow technology including case-based adaptation of business workflows. An automated adaptation support has turned out to be benefitial for the persons that are responsile for the workflow modeling and adaptation. As an adaptation of scientific workflows has to consider the data flow in addition to the control flow, it is an even more challenging modeling task than the adaptation of business workflows that focus mainly on the control flow. Hence, an automated support by adaptation cases would be very benefitial for the scientists dealing with scientific workflows.

In this work, we aim at (I) *confirming the hypothesis that case-based adaptation methods are applicable for scientific workflows* at all and (II) *developing a novel method for extracting adaptation cases automatically* from community repositories of scientific workflows. The opportunities to apply workflow adaptation cases - may they be acquired automatically or by hand - go even beyond the traditional Case-Based-Reasoning (CBR) idea of case reuse: The transformation pathways from one workflow to another that are recorded by an adaptation case can be used to visualize or measure deviations between workflows addressing the same topic for compliance or reconcilability purposes. Scientific workflows may be checked, for instance, if teams of scientists aim to cooperate. Business workflows may be compared, for instance, after mergers and acquisitions. The remainder of the paper is organized as follows: In Sect. 2, workflow adaptation cases are introduced. Sect. 3 sketches the case-based adaptation of workflows. In Sect. 4, the automated acquisition of adaptation cases from community repositories is presented. Sect. 5 deals with an experimental evaluation. In Sect. 6, related work is discussed. A conclusion is drawn in Sect. 7.

2 Workflow Adaptation Cases

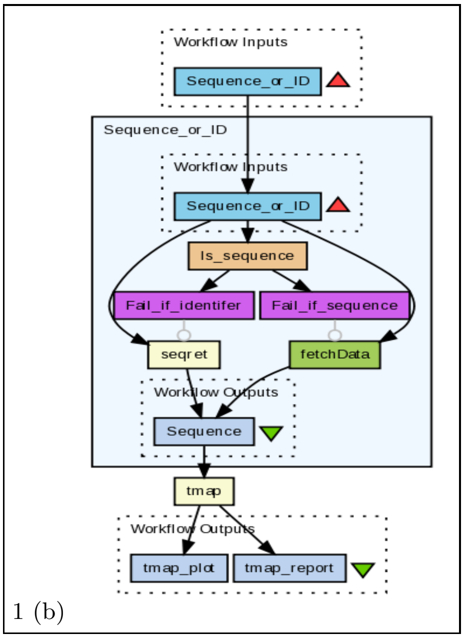
Case-based adaptation methods for workflows use adaptation cases for recording adaptation episodes. The case structure is impacted by the workflow language in which the workflows are specified. Most of the present workflow languages are graph-based in the sense that they consist of *workflow elements* as atomic parts organized in a flow-oriented manner, i.e. the elements can be represented by nodes and edges forming the flow of tasks. In this work, we focus on graph-based languages. We use the Simplified conceptual unified workflow language (Scufl) [13] as an example of an XML-based language whose elements describe nodes and edges of a workflow graph. Scufl workflows are dedicated to scientific workflows that are to be executed by the Taverna system [13]. Scufl has the following types of workflow elements that are subsumed to the set of nodes: tasks (`<s:processor>`) including placeholder tasks for sub-workflows, data objects for workflow inputs (`<s:source>`), and data objects for workflow outputs (`<s:sink>`). In the sample on the left hand side (1b) of Fig 1, the workflow has five nodes at top-level depicted by rectangular boxes: one workflow input node, two workflow output nodes, one placeholder task for the sub-workflow 'Sequence_or_ID' (containing further nodes at sub-workflow level), and one task 'tmap', which calls a Soaplab invocation as indicated by the light colour. Other types of Scufl tasks call on, for instance, a single Web service operation or a local Java function. The type of a task as well as further properties like input and output ports are specified as child elements of the XML element but do not belong to the set of nodes in the graph representation and are consequently not depicted. The edges in the workflow graph are derived from the following types of workflow elements of Scufl: Data flow edges (`<s:link>`) are depicted as arcs; coordination constraints (`<s:coordination>`) play the role of control flow edges as depicted by a connecting line with a circle. In the sample workflow (1b) of Fig. 1, such a constraint is specified, for instance, between the tasks 'Fail_if_identifer' and 'seqret', which means that task 'Fail_if_identifier' has to be completed before 'seqret' can be scheduled for execution. The details of the coordination constraints are specified again by child elements of the XML elements but are likewise not part of the graph representation.

An adaptation case can now be described as follows (compare our previous work [10]):

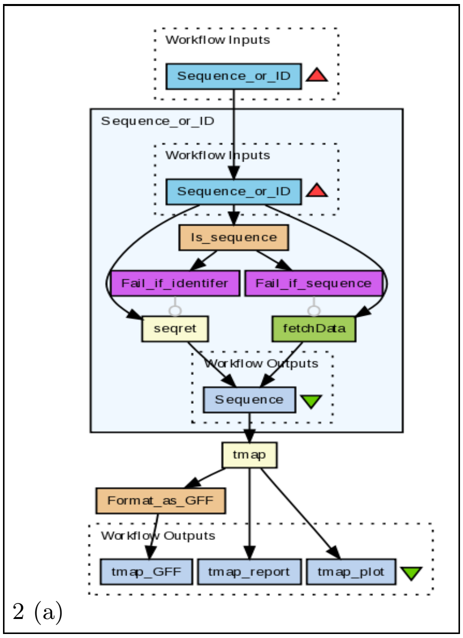
1. The *problem part* consists of
   (a) a semantic description of the change
   (b) a graph-based representation of the anterior workflow version prior to the adaptation.

2. The *solution part* contains
   (a) the posterior workflow version, i.e. the adapted workflow, in graph-based representation
   (b) the description of the adaptation steps(added and deleted workflow elements) that have been executed to transform the anterior workflow version into the posterior.

1 (a) Change description: "Add GFF output."



1 (b)



2 (a)

2 (b)

```
<ADDList>
  <Chain>
    <Pre>
      <s:processor name="tmap" refID="20" >
    </Pre>
    <WorkflowElements>
      <s:link source="tmap:outfile" sink="Format_as_GFF:tmap_output"
        sourceID="20" sinkID="52" refID="51" />
      <s:processor name="Format_as_GFF" refID="52" />
      <s:link source="Format_as_GFF:tmap_gff" sink="tmap_GFF"
        sourceID="52" sinkID="54" refID="53" />
      <s:sink name="tmap_GFF" refID="54" />
    </WorkflowElements>
    <Post>
    </Post>
  </Chain>
</ADDList>

<DELList>
</DELList>
```

Fig. 1. Sample adaptation case with two versions of a workflow for the analysis of proteine genomes (retrieved from the workflow repository www.myexperiment.org)

Fig. 1 depicts a sample adaptation case in that an additional processing step 'Format_as_GFF' with an additional data output object 'tmap_GFF' is inserted into the workflow (compare the two workflow versions depicted in (2a) and (2b)). The semantic change description characterizes the changes that have been made from the anterior to the posterior version. This part of the problem description makes use of traditional case representation approaches, e.g. a structural representation or a textual representation. The sample change description in (1a) of Fig. 1 is a text. Workflow versions are represented in a graph-based way by sets of nodes and edges as described before. The representation of the adaptation steps deserves some special attention. Similar to STRIPS operators, the adaptation steps are described by an add and a delete list. Each list contains a set of chains of workflow elements. A chain encapsulates a connected sub-graph of workflow elements that are to be added or deleted 'in a chain', i.e. the according edit operations are either fully applied or not applied at all while reusing the adaptation case. Furthermore, each chain records a pair of anchor sets that describe the positions within the workflow graph where the edit operations have taken place. The pre anchors are the workflow elements (in the anterior workflow) after which workflow elements from the chain have been added or deleted. The post anchors are the workflow elements from the anterior workflow following the last elements of the chain. Hence, the set of anchors describes the connectors at which a sub-graph has been inserted or pruned out. The XML snippet in the lower part of Fig. 1 shows a representation of an add list with scufl elements. It contains one chain of four workflow elements to be inserted namely the task 'Format_as_GFF', the output data object 'tmap_GFF' and two data links. The chain has only one connector to the anterior graph namely the 'tmap' task. It is a pre anchor as it is connected via an outgoing edge with the new sub-graph. The set of post anchors of this chain is empty as well as the delete list of the entire case. The anchors are further used during the reuse of the workflow adaptation to identify similar points in new workflows at which the proposed adaptation can be applied. (2a) is mostly redundant within the case structure as it could be reconstructed from (1b) and (2b). It is recorded for reasons of readability only (graph layout).

3 Case-Based Workflow Adaptation

Although the workflow adaptation itself is not in the scope of this work we will briefly sketch the case-based method for workflow adaptation to make the automated creation of adaptation cases more plausible to the reader. The case base consists of adaptation cases. The new problem to be solved (query) consists of a target workflow (which may be already partially executed) and the description of the current change request to the target workflow. Hence, the similarity measure must be able to assess the similarity of two change descriptions as well as of two workflows. Please refer to our previous work [11] and to the literature [9,1] for similarity measures for workflows. The similarity-based retrieval provides the most relevant adaptation cases from the case base. The best matching case is selected for reuse (either automatically according to the values of the similarity function or by user interaction). In the reuse phase, the best matching adaptation case is applied to the target workflow in order to adapt it according to the change request. This occurs in two distinct steps. First, the concrete location in the target workflow is determined that needs to be changed. This is necessary as there are usually many different positions within the workflow at which a chain from the add list can be inserted or to which the deletions in the delete list can be applied. Second, the changes are applied to the target workflow at the determined locations. The resulting adapted workflow is then the proposed solution. During the subsequent revise phase the user can validate the workflow adaptations proposed: she can either confirm them or revise them by manually performing appropriate adaptations herself. First experiments on automated, case-based adaptation of workflows have been conducted successfully (see [10] for the results). Hence, we can now turn to the case acquisition task and present a solution to automate this tedious work in the following.

4 Case Acquisition

The automated acquisition of workflow adaptation cases follows the idea of determining the difference between a pair of subsequent workflow versions and deriving a case from this delta. A huge amount of scientific workflows in different versions are available in community repositories[2]. Mostly, machine-readable representations of the workflow graphs are available and textual change descriptions are stored as revision comments. Hence, cases can be acquired from the content of such repositories by the following steps:

1. Extract pairs of subsequent workflow versions with a change description
2. Derive atomic edit operations (add an element, delete an element) from the differences of the sets of workflow elements from both versions
3. Organize the edit operations in chains with anchors.

The first step of the case acquisition is to select subsequent pairs of workflow versions and store their formal representations together with the textual description of the change in parts 1 (a),(b) and 2 (a) of a new case. Cases with an empty change description are not considered.

The second step is to gain the atomic edit operations from the formal representations. Set differences are determined for each type of workflow element as follows. The hierarchical structure of the top-level workflow with all nested sub-workflows is analysed and recorded in a *sub-workflow tree*. The root node of the sub-workflow tree stands for the top-level workflow while the other nodes represent a sub-workflow at the according level each. Fig. 2 a) illustrates this by a sub-workflow tree generated for the sample case in Fig. 1. As the hierarchical structure of the anterior workflow version may differ from the structure of the

[2] Sample repositories for scientific workfows are available at www.vistrails.org and www.myexperiment.org

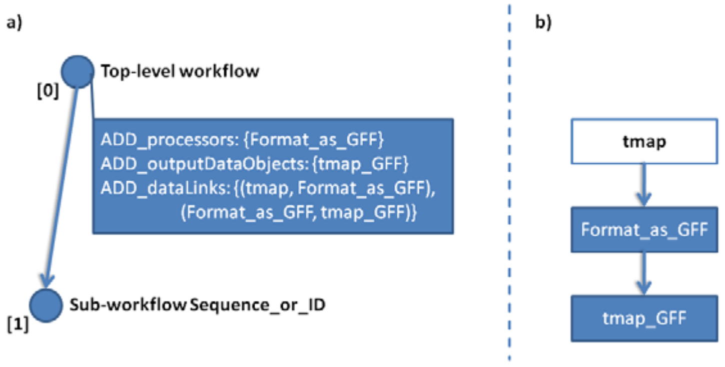

a)
Top-level workflow
[0]
ADD_processors: {Format_as_GFF}
ADD_outputDataObjects: {tmap_GFF}
ADD_dataLinks: {(tmap, Format_as_GFF), (Format_as_GFF, tmap_GFF)}
Sub-workflow Sequence_or_ID
[1]
b)
tmap
Format_as_GFF
tmap_GFF

Fig. 2. Sub-workflow tree and add graph generated for the sample case in Fig. 1

posterior version, two trees T_a, T_p have to be generated initially. The two constructed trees will be merged at the end in order to get one sub-workflow tree containing all atomic edit operations between the two versions. In depth-first search, the nodes of T_a are successively enriched by add and delete lists for each type of workflow element at the level of the actually investigated sub-workflow. The set difference between the set of data objects for workflow inputs of a sub-workflow X of the anterior version and the set of data objects for inputs of the same sub-workflow X' of the posterior version, for instance, forms the according delete list $X.DEL\_inputDataObjects$. The 'same' sub-workflow means that both sub-workflows have equal names and have the same position in T_a and T_p with respect to the path from the root node. The set $X' \setminus X$ forms the corresponding add list. A sub-workflow that has been added entirely is stored by the placeholder task in the according list for placeholder tasks as well as by an additional sub-workflow node in T_a. The sub-workflow node is enriched by add and delete lists for the inner elements of the sub-workflow. Where required, further sub-workflow nodes are inserted into T_a below the new node. From the resulting, fully expanded and enriched sub-workflow tree T'_a all edit operations that have taken place can be reproduced. The workflow elements and their position within the hierarchy of sub-workflows is recorded unambigously except for the order of sibling sub-workflow nodes. The latter is not significant as the dependencies between tasks and objects are specified explicitely by edges in the formal representation (data links and coordination constraints).

Though the sub-workflow tree is capable for a reconstruction of the change, it alone is not sufficient for a transfer of the change to another target workflow. Rather than applying the edit operations directly to the target workflow, the connected workflow elements should be grouped in order to preserve the connectivity and to enable the application of a chain of edit operations 'fully or not at all'. The anchor principle as described in Sect. 2 comes into play for mapping the positions of the chains. Thus, the third step of the automatic case acquisition is to group the atomic edit operations in chains and to determine appropriate anchors for each chain. This is done by constructing maximum connected subgraphs of the workflow graph consisting of workflow elements conjointly affected by the same type of atomic edit operation, the so-called *add graphs* and *delete graphs*. Fig. 2 b) depicts a sample add graph. An add graph consists of added data objects and tasks including placeholder tasks as nodes and added data links and coordination constraints as directed edges. As a graph can only contain edges with a source and a sink, the workflow elements that are source of an added data link or coordination constraint are included as nodes in the add graph also if they are not added themselves. The nodes of this particular set are designated as the pre anchors of the chain. The same holds for workflow elements that are sink of an added link or coordination constraint. The set of those particular nodes forms the post anchors of the chain. The delete graphs are built analogously. Add and delete graphs may span several sub-workflows.

```
Algorithm 1. Build add graphs
input
    Sub-workflow tree t (with sub-workflow nodes numbered in the
       order of a breadth first search from 0 to sizeOfTree)
output
    Forest addGraphs
begin
    dispoN[sizeOfTree]:= array of lists of workflow elements;
    dispoE[sizeOfTree]:= array of lists of workflow elements;
    anCands[sizeOfTree]:= array of lists of workflow elements;
    addTrees:=∅;
    foreach i=0..sizeOfTree do
       dispoN[i]:=unification of all add lists of type task, data
          input object, data output object, or placeholder task
          of i-th (sub-)workflow node;
       dispoE[i]:= unification of all add lists of type data link
          or coordination constraint of i-th (sub-)workflow;
       anCands[i]:= unification of all tasks, data input objects,
          data output objects and placeholder tasks of i-th (sub-)
          workflow except the elements of any add or delete list;
    od
    foreach i=0..sizeOfTree do
       addGraphs:=addGraphs ∪
          createMaxConGraphs(i,dispoN,dispoE,anCands);
    od
    return addGraphs;
end
```

```
function graphSet createMaxConGraphs(i,dispoN,dispoE,anCands)
    graphSet:=∅;
    while dispoN[i]!=∅ do
        openPH:=∅;
        currentN:=dispoN[i].firstEl;
        dispoN[i].delete(currentN);
        currentG:={currentN};
        if currentN is placeholder do
            openPH.add(currentN);
        od
        openGE:={currentN};
        while openGE!=∅ do
            currentEl:=openGE.firstEl;
            openGE.delete(currentEl);
            if currentEl is node do
              expandE(currentG,i,openGE, dispoE,currentEl);
            else
              expandN(currentG,i,openPH,openGE,dispoN,anCands,currentEl);
            od
        od
        while openPH!=∅ do
            currentPH:=openPlaceHolder.firstEl;
            expandSFW(currentG,currentPH,openPH,dispoE,dispoN);
            openPH.delete(currentPH);
        od
        graphSet.add(currentG);
    od
    while dispoE[i]!=∅ do
        currentEdge:=dispoE[i].firstEl;
        dispoE[i].delete(currentEdge);
        currentG:={currentEdge};
        expandN(currentG,i,openPH,openGE,dispoN,anCands,currentEl);
        graphSet.add(currentG);
    od
    return graphSet;

function expandE(currentG,i,openGE,dispoE,currentEl)
    foreach j=0..dispoE[i].size do
        currentEdge:=dispoE[i][j];
        if currentEdge touches currentEl do
            openGE.add(currentEdge);
            currentG.add(currentEdge);
            dispoE[i].delete(currentEdge);
        od
    od
    openGE.delete(currentEl);
function expandN(currentG,i,openPH,openGE,dispoN,anCands,currentEl)
    expanded:=false;
    foreach j=0..dispoN[i].size do
```

```
            currentN:=dispoN[i][j];
            **if** currentN touches currentEl **do**
                openGE.add(currentN);
                currentG.add(currentN);
                dispoN[i].delete(currentN);
                **if** currentN is placeholder **do** openPH.add(currentN);
                **od**
                expanded:=true;
            **od**
        **od**
        **if** !expanded **do**
            **foreach** j=0..anCands[i] **do**
                currentN:=anCands[i][j];
                **if** currentN touches currentEl **do**
                  currentG.addAnchor(currentN);
                **od**
            **od**
        **od**
        openGE.delete(currentEl);
    **function expandSWF(currentG,currentPH,openPH,dispoE,dispoN)**
        openGE:=∅;
        **foreach** el ∈ currentG **do**
            **if** el touches currentPH **do**
                openGE.add(el);
            **od**
        **od**
        **while** openGE!=∅ **do**
            currentEl:=openGE.firstEl;
            openGE.delete(currentEl);
            **if** currentEl is node **do**
                expandE(currentG,currentPH.index, openGE,dispoE,currentEl);
            else
                expandN(currentG,currentPH.index,openPH,openGE,dispoN,
                    anCands,currentEl);
        **od**
    **od**
```

Algorithm 1 details the steps of building add graphs. The input is a sub-workflow tree including the atomic edit operations that is built from the two workflow versions as described above. The algorithm passes through the sub-workflow graph in a breadth first search to start constructing the add graphs at each sub-workflow level by the function `createMaxConGraphs`. The function creates and extends one sub-graph after the other employing an 'open' list of graph elements (nodes and edges), which is a well known principle from search algorithms in artificial intelligence. If the recent add graph is extended by a workflow element the element is moved from the list of disposable elements (nodes `dispoN` and edges `dispoE`) to the list of open graph elements (`openGE`). If the element has been fully expanded in the sub-graph, i.e. if all of its touching edges (or nodes, in case of an edge) from the lists of disposable elements have been investigated, it is deleted from the list of open graph elements. The algorithm starts with the initialization of the dispo lists at each sub-workflow level (lines 7, 8, and 12 – 16 of Alg. 1). The workflow nodes that are not part of any add or delete list are stored in the `anCands` lists as they might serve as anchors (lines 9 and 17 – 19). For each node of the sub-workflow tree, the maximum connected graphs are computed by calling `createMaxConGraphs`. Within this function, an additional recursive decent into lower sub-workflow levels is required if an add graph extends into lower-level sub-workflows. However, the creation of an add graph is always started at the higest possible level. At one level, several distinct sub-workflows may start. An add graph starts with one node as initial graph (l. 34) and is extended by edges (l. 43) and nodes (l. 46) from the disponable elements. `openGE` records all graph elements (nodes and edges) of an add graph that are still to be expanded (line 38 and within the functions `expandE` and `expandN`). Placeholders are recorded in `openPH` (lines 35 – 37 and within the function `expandN`) in order to expand the add graph further in function `expandSFW` when all elements at the same level have been investigated (lines 47 – 52). After all initial graphs have been expanded, some edges might be left in openEdges. They are expanded by anchor nodes (lines 59 – 66) to form chains with one element only.

5 Experimental Results

The hypotheses posed at the beginning of this paper have been tested by experiments with scientific workflows retrieved from the community repository at `www.myexperiment.org`. Hypothesis I on the applicability of case-based adaptation methods for scientific workflows has been split into two parts investigated by manual experiments, while hypothesis II on the automated construction of adaptation cases has been investigated by comparing the results of an implementation of Algorithm 1 with the manual results gained from the experiments on hypothesis I. The following questions guided the experiments:

(Ia) *Applicability*: Can adaptation cases with chains and anchors be constructed that record the adaptation episodes of scientific workflows?

(Ib) *Applicability*: Does the reuse of the cases lead to feasible results?

(II) *Automated construction*: Can adaptation cases be captured automatically from community repositories? Is the quality of the automated results comparable to the quality of reference results acquired by experts?

We started to manually create a case base with eleven cases including the sample case in Fig. 1. The xml files retrieved from the repository only consist of atomic elements such as tasks and data objects and relational link elements, so we would have to draw the workflows manually to understand all dependencies. For that reason we used as additional source for the creation the automatically rendered representation of workflow versions because it would have been a very tedious work to compare the xml files without further utilities. Although this seems to be a quite small number of cases for a case base it took a large part of time of this work. This is purely owed the complexity of scientific workflows. We ensured that the workflows in our adaptation cases use all workflow elements from the Scufle modeling language. In some adaptation cases we have chains with more than 45 conjoined workflows elements which are connected over several hierarchical levels. In such cases it is difficult for a human to follow all links of a complex workflow bi-directional. Nevertheless the eleven adaptation cases could be created successfully. This provides a reference solution for the computed cases in (II) and confirms hypothesis (Ia).

Hypothesis (Ib) has been investigated by grab samples only. One sample target workflow is depicted in Fig. 3 which contains the same tmap Soaplab service as the anterior workflow of our sample adaptation case in Fig. 1. Hence, the change described by the adaptation case could be transferred to the sample target workflow. Admittedly, the grap samples give a first hint only that the automated adaptation would work in principle. Further evidence is required from experiments that will be part of our future work. The next step in our work is to test if the chains in the automatically acquired cases can be used to reconstruct the given cases. For this purpose we will investigate whether the mapping of anchors [10] can be applied to scientific workflows too. If these tests will be successful we will be able to create a large case base with reusable adaptation cases and conduct further experiments on re-purposing the adaptation episodes on other target workflows.

For hypothesis (II), we implemented Algorithm 1 in Java. Running the algorithm lead to slightly different results from the reference solution. Only six cases were identical to the reference solution. Surprisingly, four of the other five cases showed up small failures in the reference solution. The algorithm can also detect port changes in a link between data objects and tasks. Because these details are not illustrated in the graphical representation as described in Sect.2 it is hardly possible to determine such changes manually. Other failures that orrcurred in the reference solution chains are miss-spellings of element names that have been overseen (e.g. 'organsim' instead of 'organism') and small changes between workflow versions which only lead to very small chains in a case. The anchors of the reference solution only differred in one case from the automatically created solution, but also in this case it was due to a human error that an anchor was overseen in the reference solution. Only in one case a minor deviation from the 'optimal' solution occurred: Changes crossing a sub-workflow by a continuous path but not affecting the entire sub-workflow led to the effect that the placeholder task for this sub-workflow did not appear within the change lists. As a consequence, the algorithm divided the edit chain into two chains, while the reference solution modeled one chain only. In order to deal with such special cases, the algorithm could be extended by unifying chains that contain paths through a sub-workflow. As a result from the automatic acquisition of adaptation cases we got the experimental confirmation that the algorithm works very well. It is

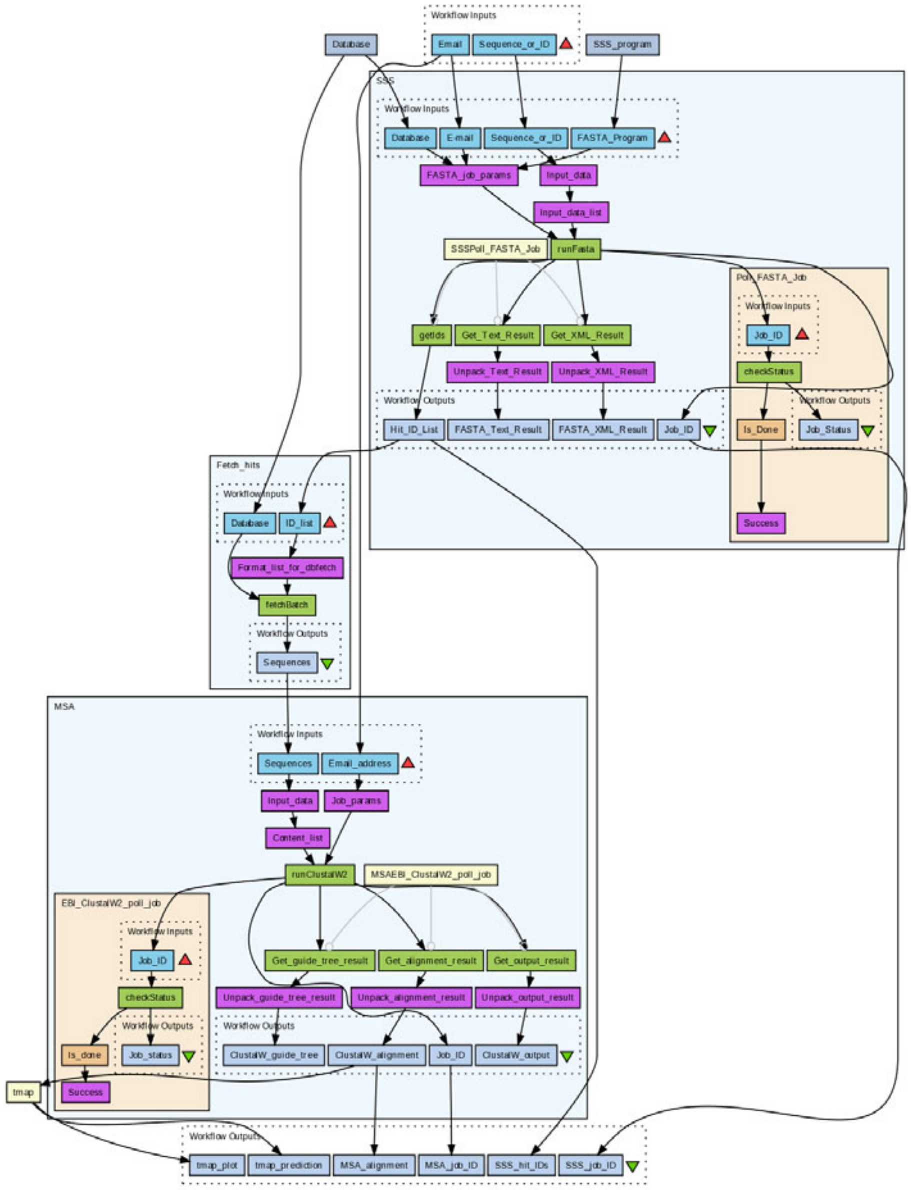


Fig. 3. Sample target workflow retrieved from `www.myexperiment.org` for that the case from Fig. 1 would be applicable

evident that the manual acquisition is not only time consuming but also fault-prone for small changes in workflow versions. Concluding we regard hypothesis II to be confirmed by the experimental results.

6 Related Work

Related work from the field of scientific workflow reasoning and from the field of case-based workflow adaptation will be discussed in the following. Goderis [4] deals with the discovery of scientific workflows and workflow elements by techniques of information retrieval, ontology reasoning, and graph matching. Workflow adaptation is not addressed. Gil [2] develops a vision of a knowledge level view on workflows for reasoning tasks like workflow generation and validation, automated data/parameter selection or metadata generation. The workflow system Wings assists scientists in some of these tasks by means of semantic metadata. Adaptation support is not mentioned but is very closed to workflow generation. The following approaches as well as our own previous work [11] employ case-based methods to give adaptation advice to the users. Leake and Morwick [7] provide case-based support for workflow generation by evaluating the execution traces of scientific workflows. Weber et al. [15] employ conversational CBR for the adaptation of health workflows. Montani and Leonardi [12] focus on case-based workflow monitoring based on execution traces also in the health domain. Kapetanakis et al. [6] employ case-based workflow monitoring by means of temporal relationships in the area of business workflows. In our recent previous work [10], we extend the scope towards automated adaptation of workflows. In contrast to this work, the adaptation knowledge is still captured by experts.

7 Conclusion

In this paper, the automated acquisition of adaptation cases from community repositories of scientific workflows has been investigated. First, the work resulted in the confirmation that case-based adaptation methods developed for workflow reasoning on business workflows can be transferred to scientific workflows in principle. Data dependencies can be included into the methods that have been control flow-oriented so far. However, details of data dependencies like the data ports of the concerned Web services have not yet been addressed by the solution provided so far. Future work on the internals of workflow elements could close this gap. Second, an algorithm for the automated acquisition of adaptation cases led to feasible experimental results. The overall outcome of this work is promising for an application of the case-based workflow adaptation methods to further fields like workflows in computer games[14] or personal workflows[5].

References

1. Becker, J., Bergener, P., Breuker, D., Räckers, M.: On measures of behavioral distance between business processes. In: Bernstein, A., Schwabe, G. (eds.) Proceedings of the 10th International Conference on Wirtschaftsinformatik, vol. 2, pp. 665–674 (2011)
2. Gil, Y.: From data to knowledge to discoveries: Artificial intelligence and scientific workflows. Scientific Programming 17(3), 231–246 (2009)
3. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the challenges of scientific workflows. Computer 40(12), 24–32 (2007)
4. Goderis, A.: Workflow Re-use and Discovery in Bioinformatics. PhD thesis, University of Manchester (2008)
5. Hwang, S.Y., Chen, Y.F.: Personal workflows: Modeling and management. In: Chen, M.-S., Chrysanthis, P.K., Sloman, M., Zaslavsky, A. (eds.) MDM 2003. LNCS, vol. 2574, pp. 141–152. Springer, Heidelberg (2003)
6. Kapetanakis, S., Petridis, M., Knight, B., Ma, J., Bacon, L.: A case based reasoning approach for the monitoring of business workflows. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 390–405. Springer, Heidelberg (2010)
7. Leake, D.B., Kendall-Morwick, J.: Towards Case-Based support for e-Science workflow generation by mining provenance. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 269–283. Springer, Heidelberg (2008)
8. Ludäscher, B., Weske, M., McPhillips, T., Bowers, S.: Scientific Workflows: Business as Usual? In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 31–47. Springer, Heidelberg (2009)
9. Madhusudan, T., Zhao, J.L., Marshall, B.: A case-based reasoning framework for workflow model management. Data & Knowledge Engineering 50(1), 87–115 (2004)
10. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards Case-Based adaptation of workflows. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 421–435. Springer, Heidelberg (2010)
11. Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 224–238. Springer, Heidelberg (2007)
12. Montani, S., Leonardi, G.: A case-based approach to business process monitoring. In: Bramer, M. (ed.) IFIP AI 2010. IFIP AICT, vol. 331, pp. 101–110. Springer, Heidelberg (2010)
13. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20(17), 3045–3054 (2004)
14. Ram, A., Nón, S.O., Mehta, M.: Artificial intelligence for adaptive computer games. In: Twentieth International FLAIRS Conference on Artificial Intelligence (2007)
15. Weber, B., Wild, W., Breu, R.: CBRFlow: Enabling Adaptive Workflow Management Through Conversational Case-Based Reasoning. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 434–448. Springer, Heidelberg (2004)

Combining Expert Knowledge and Learning from Demonstration in Real-Time Strategy Games*

Ricardo Palma, Antonio A. Sánchez-Ruiz, Marco Antonio Gómez-Martín, Pedro Pablo Gómez-Martín, and Pedro Antonio González-Calero

Dep. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid, Spain
rjpalma@estumail.ucm.es, {antsanch,marcoa,pedrop,pedro}@fdi.ucm.es

Abstract. Case-based planning (CBP) is usually considered a good solution to solve the knowledge acquisition problem that arises when developing AIs for real-time strategy games. Unlike more classical approaches, such as state machines or rule-based systems, CBP allows experts to train AIs directly from games recorded by expert players. Unfortunately, this simple approach has also some drawbacks, for example it is not easy to refine an existing case base to learn specific strategies when a long game session is needed to create a new trace. Furthermore, CBP may be too reactive to small changes in the game state and, at the same time, do not respond fast enough to important changes in the opponent's strategy. We propose to alleviate these problems by letting experts to *inject* decision making knowledge into the system in the form of *behavior trees*, and we show promising results in some experiments using Starcraft.

1 Introduction

Real-time strategy (RTS) games are very demanding in terms of AI complexity. They require fast pathfinding algorithms for moving large numbers of units through extensive levels, which need to be manually or procedurally annotated with tactical information. Regarding decision making, RTS require a multi-tiered AI approach, with decisions made at a low level for individual characters, at an intermediate level for a formation of characters, and at the high level of a whole side in the game. Usually simple techniques, such as state machines, are used for low level decision making, while some form of rule-based system is the most common approach for decision making at higher levels [8] .

Building a rule-based system for decision making at the tactical and strategic level of an RTS game is a complex task for game designers, who are not used to deal with knowledge representation. In order to alleviate this authoring effort, there is an open line of research on the automatic acquisition of decision making knowledge from recorded traces of human experts playing the game. Such approaches, through the application of machine learning techniques, seek

* Supported by the Spanish Ministry of Science and Education (TIN2009-13692-C03-03).

to make possible a form of *programming by demonstration*, where the human author shows to the game AI how to play.

We propose to bring the game designer back into the loop, by allowing him to explicitly inject decision making knowledge in the form of behavior trees to complement the knowledge obtained from the traces. Behavior trees are the technology of choice for representing decision making knowledge in commercial videogames. They can be built by both programmers and designers, and provide the ability to react to urgent goals or a fine-grained control over alternative courses of action.

The work presented in this paper seeks to extend the techniques demonstrated in the Darmok and Darmok 2 (D2) systems [10,11], which are mature and well-tested systems within this line of research. We show through an experimental evaluation in the Starcraft game, how we can easily increase the efficacy of a case-based planner significantly. Moreover, we show that very simple behavior trees can make a big difference in terms of the AI quality.

The rest of the paper runs as follows. Sections 2 and 3 briefly describe the two technologies we propose to integrate: learning from demonstration and behavior trees. Section 4 discusses the integration of both technologies into a single architecture, which is empirically evaluated by means of some experiments described in Section 5. Last section presents related work and concludes the paper.

2 Learning from Demonstration in Real-Time Strategy Games

Learning from demonstration tries to replace the time-consuming and hand-made task of programming behaviors by an automatic process in which an expert shows the system how to achieve some goal. Afterwards, a learning system analyses the actions performed by the expert and tries to extract useful knowledge. This type of techniques has traditionally been applied to build behaviors in robots, but the complexity of current video games makes them perfect candidates to apply these techniques as well.

There are many different approaches to extract decision making knowledge from the traces of expert players, for example to consider sequences of actions as reusable *plans*. In *case based planning* (CBP) these plans are stored in a plan base for later reuse, and the planner retrieves and adapts the most promising one according to the current goal and state of the problem. In our context, the goal is usually something like *win* and the state of the problem is the state of the game.

Being games dynamic, static CBP suffers in this context from lack of reactivity. In order to respond properly to the continuous changes in the game state, CBP systems usually include an *on-line* component that supervises the execution of the proposed plan. This way, instead of just considering the planning problem as a single-shot process where execution and problem solving are decoupled, *on-line case-based planning* (OLCBP) systems are also partly in charge of the plan execution. Note that OLCBP systems can discover low level aspects at the plan expansion stage that were not considered originally and which might require to perform some changes in order to prevent failure.

Although planning in OLCBP is performed in *real-time*, planning decisions are based on an expensive *learning* process that is made previously off-line from game *traces* produced by human expert players. Depending on the CBP engine, the learning phase might require the assistance of a human expert to *label* the traces in order to enrich them with extra knowledge about, for example, pursued goals or important game state variables. The other option is to make the system intelligent enough to perform the labelling task automatically and thereby avoid that tedious task to the expert. Unfortunately, there is no magic bullet here; OLCBP systems using this last approach usually require a more detailed domain description in order to extract useful plans from plain traces.

Once the plan base has been fed with plans (cases) extracted (automatically or not) from traces, it is time for the *on-line* component of the system to take control. The game will be launched as normal and connected with an external AI that, in our architecture, is implemented using an OLCBP system. The game engine will broadcast its state using a game-dependent protocol, and, on the other way, the OLCBP system will inject the primitive actions that should be executed in the game environment.

One of such OLCBP systems is Darmok, presented in [11]. It has been used for playing real-time strategy games like Wargus (an open source clone of Warcraft II) and other games. As we will see in a later section, the combination of domain knowledge automatically extracted from game traces with hand-made expert knowledge is an effective solution to some of the problems identified in this type of OLCBP.

3 Expert Knowledge in Games

Regarding how to create behaviors for the non-player characters that appear in videogames, many techniques have been used. The goal here is to translate the knowledge that game experts have (those who know the best way to defeat the other player) into instructions to be executed by the machine. The approach is different depending on the level of abstraction and game genre but when focusing on low level decision making for controlling units on strategy games the most commonly used techniques have been finite state machines (FSMs) and more recently behavior trees (BTs).

Though BTs were initially proposed as a tool for programmers, they have been proved to be useful also for professional *game designers* to create the behaviors of the entities from scratch [4,5]. One of the key points is that, just like FSMs, BTs open up the possibility of developing tools to create and edit behaviors by means of a graphical user interface, something that cannot be done in other techniques like scripts.

A BT is a hierarchical and declarative representation of a behavior where every node (and its subtree) can be seen as a simpler behavior. Therefore, BTs promote reuse and allow designers to define complex behaviors in an incremental way.

Using visual tools, game designers and game programmers create the BTs that eventually will define the behavior of the game AIs. The available toolset for this authoring process depends on the actual AI engine implementation and every game studio has its own set of tree nodes that can be combined to encode complex behaviors from simpler ones. There are, therefore, many types of BT nodes but we will only cover those we have used in our experiments.

During the execution of the game, the behavior trees are loaded into memory and a module known as the *BT interpreter* controls its execution. The BT interpreter manages which branch of the current BT is *active* (complex systems may allow BTs to have more than one branch in execution) and how much CPU quantum should be assigned to each node. In addition, when a node complete its execution either successfully or failed, the BT interpreter decides, according to the semantics of its parent node in the tree, which branch should be expanded next.

In pure systems all the decisions are performed based on BTs, while mixed systems may have a higher level AI module to select the best BT for each entity according to some global policy or strategy. This high level module must receive continuous notifications to properly manage the behavior of the different entities in a coordinated manner. In turn, the BT execution model might rest either on a single global BT interpreter or there might be several BT interpreters, one for each entity. As we will see in Section 4, our system has been implemented using a mixed architecture and a single BT manager.

Regarding the types of behavior a BT may encode, its expressiveness and versatility depends on the amount and types of nodes available at design time. The simplest BT is compound by just one node representing a *primitive action*, that is, a game action to be executed by one character. These primitive actions may have parameters whose values are either hard-coded on the BT itself or computed at run time depending on the context. When the BT interpreter decides to execute one of these nodes, the primitive action is sent to the game engine and the interpreter must wait until its execution ends either with success or failure. Note that although from the point of view of the AI system the action is atomic, in a real-time game its execution might take several seconds. Besides, the action can fail if something happens during that time that prevents its execution.

The simplest way of combining behaviors is using *sequential nodes*, inner nodes whose child nodes (or subtrees) are executed sequentially. When all of them finish successfully their execution, the behavior succeeds, failing otherwise. Sequential nodes are usually represented as a node with an arrow inside.

Dynamic priority lists bring more expressive power to BTs, adding the idea of *conditions*. Every child node has not only a behavior but also a guard or condition that checks some aspect of the game state that is relevant for the sub-behavior. When the interpreter reaches a dynamic priority list, evaluates the guards of the children nodes from left to right and starts the execution of the first behavior whose guard is met. If no condition is met, the entire behavior fails. Once a child node is active, the interpreter keeps evaluating the conditions of the left siblings and whenever one of them becomes true, the current behavior is aborted and the child with the higher priority takes control. In the figures shown later in the paper guards appears as nodes in dotted lines with one child representing the behavior that must be executed when the guard becomes true.

The last node we consider is the *query node* [3] that promotes even further reusability. All the nodes described so far are hard-coded in the BT, but this new node allows to dynamically attach BTs as subtrees of other BTs at run time. This way, BTs can delegate to achieve a particular goal to other BTs without having to specify to which one in particular. The idea is that different BTs are created independently during the game design phase and stored in a behaviors base and, at run time, the query nodes serve as joint points to combine behaviors like assembly parts.

The technology behind query nodes is borrowed from case-based reasoning. The behavior base may be seen as a case base storing different solutions to achieve a goal. These BTs are described using a semantic label from a behavior ontology, a set of variables and constraints which encode the game states where the BT is optimal, and the entities the BT is meant to (this is needed because a soldier may have a different behavior to protect/cover itself than a tortoise).

The *query node* in turn contains the CBR query which describes the desired behavior along with a number of variables and constraints using the same vocabulary that was used to index the behavior base.

At run time the query is extended with the current game state and the current entity executing the BT and, using similarity measures similar to those used on CBR, it retrieves the best BT for the current situation. Interested reader may refer to [3] for more details. As we will see later, we have used these nodes in different experiments. In the rest of the paper, this kind of nodes is drawn using thick lines.

4 Extending Case-Based Planning with Behavior Trees

Three main flaws [12] have been identified in on-line case based planning when used with plan extraction from expert traces:

– *Poor reactivity at the plan level:* when a plan is chosen, it will not be abandoned until it is completely performed or a *low level action* fails. World changes that should fire a new replanning phase are usually ignored because they are not recognized as a failure condition for the current action in the *working* plan.
– *Excessive reactivity at the action level:* when a small low level action fails, a big plan could be entirely discarded because the planner would assume that also the *complete plan* has failed and cannot be fixed. A new planning phase would be fired, which could end up with a complete opposite approach for the current goal. In other words, small local problems could have unreasonable big responses.
– *Learning by demonstration is hard to fine-tune:* When an expert identifies behaviors that should be improved, the nature of learning by demonstration

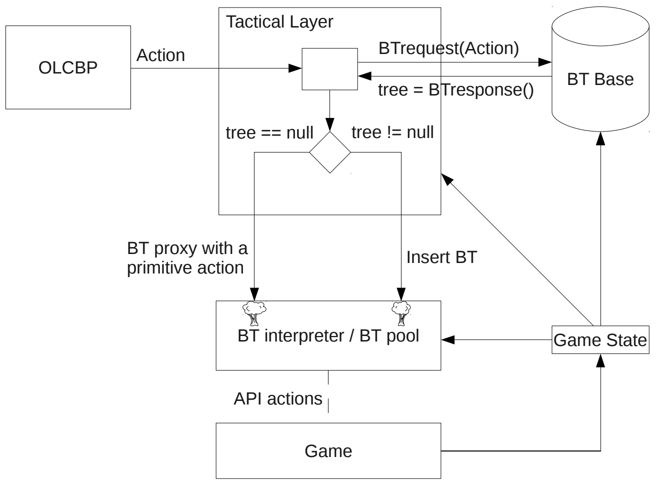


Fig. 1. Low level tactical layer

and the fact that he has to provide new traces to have the new behavior learnt, makes the process really hard. This is due to the complexity and randomness inherent in strategy games, where it is difficult to get a suitable scenario where the expert's actions are relevant enough for being incorporated as a plan in the plan base.

Our solution is to incorporate expert knowledge into the process. This knowledge takes the form of behavior trees presented in the previous section. As we will describe shortly, BTs have such an important role in our approach that the BT interpreter is the only one that injects primitive actions into the game. This interpreter manages a *BT pool* that contains all the BTs that are currently in execution.

Between the OLCBP and this *BT pool* we place a *tactical layer*. The planner still emits primitive actions to be executed, but instead of directing them into the game they pass through this layer. Its functionality is based on a BT base that is populated by behaviors built previously by the experts. When the OLCBP wants to execute an action, the tactical layer retrieves the most suitable BT that performs that action in the current game state and sends it to be executed by the interpreter, adding it to the BT pool. If there is no such a BT, it builds a small BT with just the node that executes that action.

The tactical layer and the BT base allow the expert to *overwrite* the execution of primitive actions when needed. For example, there will be specific situations where a primitive action should be enriched with a more *error-safe* plan that is able to gracefully react to some small situations that would cause the primitive action to fail. This would avoid the problem of excessive reactivity at the action level, but, unfortunately, *training* (and teach) the CBP system to use these so concrete plans is quite hard (if possible at all). This extension to the basic architecture where only the game and OLCBP system coexist does not require changes in any of them. Figure 1 shows a simplified view of the architecture so far. The OLCBP system emits a primitive action to the tactical layer which tries to get a BT through a query to the BT base. Should the query return *null*, the original primitive action is packed on a BT containing a single node. In other case the BT is inserted into the interpreter that manages its execution sending a primitive action to the game.

A second extension of pure OLCBP systems consists on allowing experts to overwrite *complete* plans. The idea is similar to that of low level actions but requires changes on the OLCBP system. The BTs created by the expert contain a description of the game state specifying when it is appropriate to use them. When the OLCBP system in its plan expansion reach a new goal, instead of just retrieving the most similar plan stored in the plan base, it performs a query to the BT base to check if a hand-made BT has been created for that specific situation. If such a BT exists the goal is replaced by it, proceeding through the plan base otherwise.

Therefore, with this extension the output of the OLCBP system may be just a primitive action (that goes through the tactical layer and it is converted to a BT) or an entire BT that is directed to the BT interpreter.

This extension aims to solve the poor reactivity at the plan level problem, because *BT guards* keep an eye on the high-level game conditions that makes the plan suitable for running, and fires a new replanning phase when they are not longer met, even if the primitive actions are not failing.

Figure 2 shows the detailed architecture regarding this last extension. When the plan expansion module is processing the plan for goal 1 and detects goal 3, instead of directly trying to use the plan library, it firstly query the BT base looking for a BT created by the expert suitable for the current situation of the game. In that case there is a BT that replaces the goal 3 and which eventually will be sent to the tactical layer to be inserted in the BT interpreter.

5 Experiments

Starcraft [2] is a famous real-time strategy game that has captivated millions of players in the last decade. Although the goal of the game is very simple (to build an army and to defeat the other players), this game offers hundreds of hours of fun thanks to the huge number of different strategies that can be created combining different types of units and technologies.

Players can choose among three different races (*Terran*, *Protoss* and *Zerg*), each one of them with its unique strengths and weaknesses. The chosen race will determine the type of units and technologies that will be available during the game. In order to win, players have to wisely manage a limited number of resources and invest them to get more resources, to develop new technology or to build defensive and offensive units.

A normal game goes through three different stages: to harvest raw materials, to build and develop your base, and to attack the enemy. During the *harvesting phase*, players focus on building a good number of gathering units or *workers* and, that way, to ensure a good income rate of raw resources. In the second

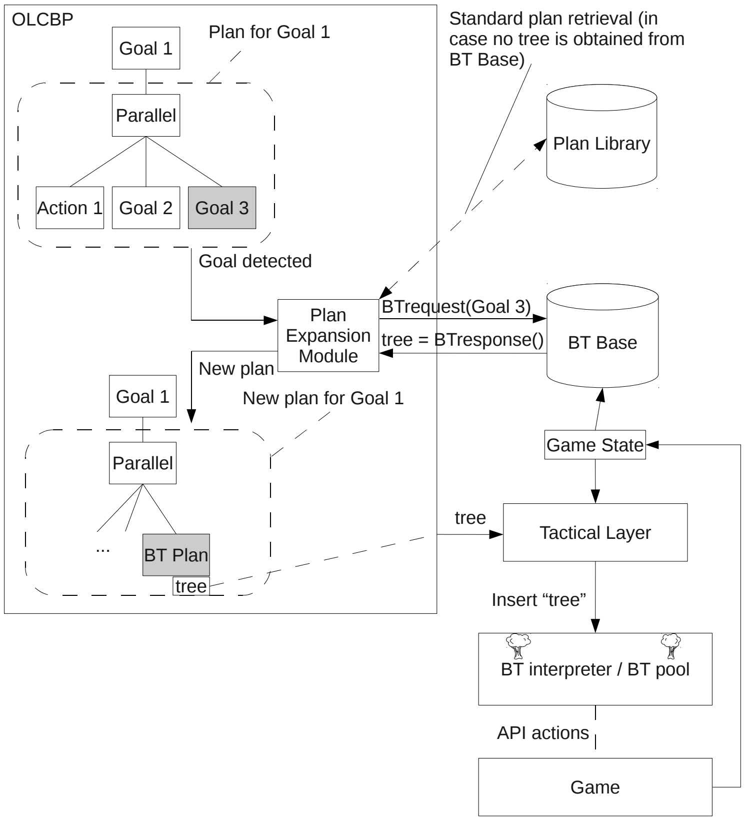


Fig. 2. High level tactical layer

phase, players use those resources to *build their bases*, that is, to create different types of buildings that will allow them to train stronger units and to research new technologies to improve their army. Finally, during the *fighting phase*, players try to destroy the enemy bases using their forces. Of course, these three stages are not really independent and players need to pay attention to the game development in order to decide the best course of action.

We focus on the battle aspects in the Starcraft game. It is consider one of the most challenging features in the game because it requires being to combine different units and skills in a effective way and, at the same time, to react quickly when the enemy changes his strategy.

In this section we describe three simple experiments that will let us evaluate: (1) the viability of the hybrid architecture presented in this paper; (2) the significant improvement in terms of AI quality we obtain when we allow experts to complete the case base with simple tactical knowledge; and (3), how easily we can represent that type of knowledge using behavior trees. The experiments take place in three battle scenarios proposed in a recent Starcraft AI competition [1], and each one of them requires a different strategy in order to win.

Fig. 3. Starcraft game (Blizzard Entertainment)

5.1 Experiment 1: Marines Battle

This scenario involves a square battleground without obstacles in which two teams of 12 *terran marines* fight against each other until one of them is exterminated. The terran marine is a basic infantry unit with a medium range attack. Each team starts in a corner of the battlefield so the main tactical decisions in this scenario concern the movement of troops and which enemies should be the first targets.

In this scenario we find an example of the poor reactivity at the plan level problem. When Darmok decides to send one of its units A to kill an enemy unit B, and another enemy unit C intercept A on the way, it is not uncommon to see how A dies under the enemy fire without even reacting to the attack.

In order to improve this behavior we used the two BTs in figures 5 and 6. The first one uses a priority list to detect when a travelling unit is in danger and must defend itself. The second BT is used to attack the threat, giving priority to the initial target over other possible enemies. Finally, the *search BT* node is used to attach the second BT as a branch of the first one at run time. This on-line mechanism give us great flexibility because the search can use knowledge about the current state of the game in order to select the best BT to attack the enemy (there could be different BT to describe different attack strategies).

In this experiment Darmork was trained using the traces of 3 real games played in this scenario. Then, we made the Darmok system to play 1000 battles against the game AI, using Darmork with and without the BT layer. In this experiment, these two simple BTs produced an improvement of 61.94% in terms of battles won.

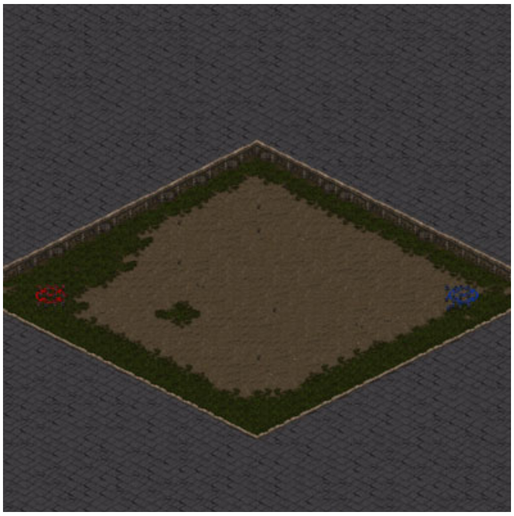


Fig. 4. Starcraft game

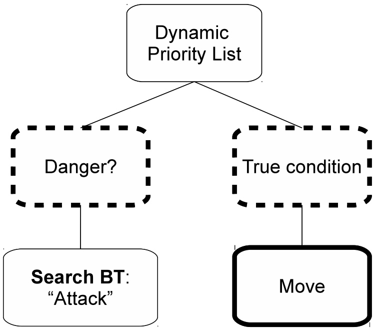


Fig. 5. BT to *move* troops

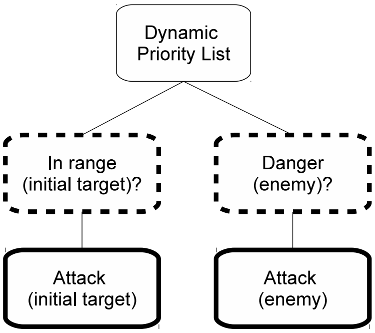


Fig. 6. BT to *attack* enemies

	Darmok	Darmok with BTs
victories	26.2% (262)	42.3% (432)
defeats	73.8% (738)	57.7% (577)
improvement	-	**61.94%**

5.2 Experiment 2: Bunker Defense

The goal of the second scenario is to defend a base as long as possible, while the enemy AI sends waves of enemies every now and then. In order to do it, the player counts with 18 terran marines and some *bunkers* strategically located in the only entrance to the base. A bunker is a defensive structure that can accommodate up to four terran infantry units. Units inside the bunker benefit from a longer range attack and suffer no damage until the bunker is destroyed and the units are expelled unharmed.

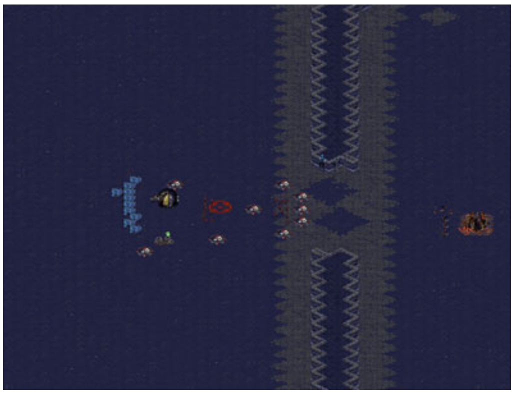


Fig. 7. Bunker Defense scenario

The best strategy in this scenario is to use the bunkers in the first line and to retreat to another bunker when the current one is destroyed. Darmok, however, does not always protect the troops using the bunkers and, sometimes, the system sends the marines to attack the enemy in the open field. In order to improve this behavior, we designed an extended version of the previous *attack BT* (figure 8) that takes into account the presence of near bunkers.

In this experiment Darmok was also trained using the traces of 3 real games. Then, we used Darmok to defend the base 200 times and we counted the number of waves that was able to resist before the base was destroyed. In order to make the simulation more real, the defender only sees the part of the map close to the base and, therefore, it does not know when the next wave will arrive.

In this experiment the use of a simple BT produced an improvement of 24.38% in terms of resisted waves.

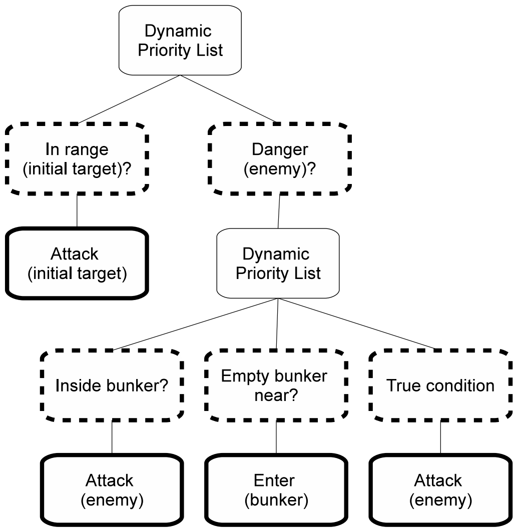


Fig. 8. BT to *attack* enemies using available *bunkers*

	Darmok	Darmok with BTs
resisted waves	402	500
μ waves / simulation	2.01	2.5
σ waves / simulation	0.5	0.7
improvement	-	**24.38%**

5.3 Experiment 3: Vultures vs. Firebats

One essential skill to be a good Starcraft player is to know how to use the special powers of some units to produce several casualties in the enemy army. One of these special units is the *terran vulture*, an advanced motorbike and the fastest unit in the game. Vultures produce a very small amount of damage when they attack directly the enemy, but they have a very special weapon, the *spider mines.* Vultures can hide these mines in the ground, were the mines will stay unnoticed until some enemy passes near, producing a big explosion that will kill all the surrounding units (allays and enemies).

In this last experiment we brought face to face 6 terran vultures and 24 *firebats.* Firebats are close range terran units able to make a big amount of damage to all the enemies that stay in front of them and close enough. A good strategy for the player controlling the vultures is to scatter the mines on the ground in front of the fire bats, and then to retreat to a safe position taking advantage of their speed.

The Darmok planner was trained using the traces of 3 games also in this case. It is interesting to realize how difficult is for Darmok to learn automatically how to use the spider mines from some game examples and, however, how easily this

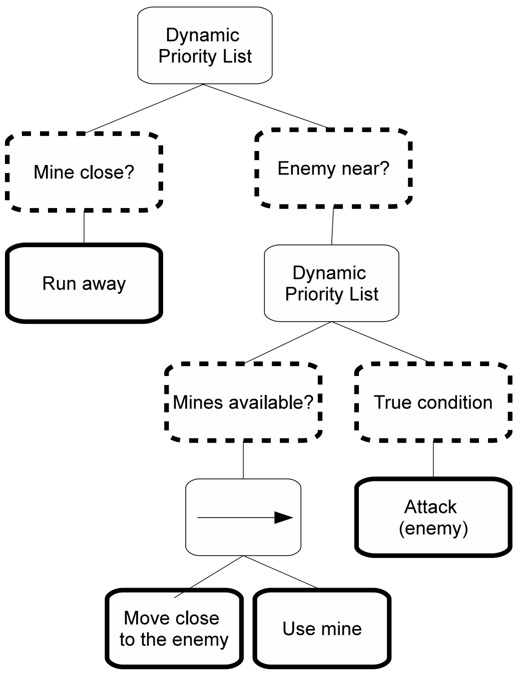


Fig. 9. BT to *attack* enemies using *spider-mines*

type of knowledge can be represented using a BT like the one in figure 9. This BT uses a priority list to check some conditions in a certain order and respond appropriately. First, we check if there is a mine close to the vulture so we can move the unit away to avoid the imminent explosion. Then, we check if the vulture has some mines left and in that case we move it closer to the enemy to use them. Finally, if none of the above conditions are met, we use the vulture to attack the enemy using its default (and weak) attack.

In this experiment we simulated 500 battles in which Darmok controlled the vultures, first without BTs and then using them. We should explain that it is very difficult for 6 vultures to defeat 24 firebats even using the mines properly, so we decided to measure the success of the experiment in terms of the number of firebats killed. In this experiment, just one BT produced an improvement of 34.10%.

	Darmok	Darmok with BTs
firebats killed	4927	6607
μ firebats killed / simulation	9.85	13.21
σ firebats killed / simulation	5.42	5.83
improvement	-	**34.10%**

6 Related Work and Conclusions

There is no, to the best of our knowledge, other work combining Case-based planning and BTs. Nevertheless, this approach can be considered as an example of a more general AI trend of combining domain theory and empirical data: BTs encode a partial view of the expert's domain theory, and cases in the plan library are empirical data. From this point of view, multiple examples of integrations of a domain theory, usually in the form of a set of rules, and Case-based reasoning (CBR) can be found in the research literature [13]. Some systems take the output of a rule-based component as the input for a case-based one, such as the one described in [6], a bank audit system that automatically detects abnormal, irregular, risky, and violated transactions from the standards at the first screening stage, and then applies CBR, which scrutinizes the detected transactions and provides the punishment levels at the second stage. Other systems take an approach closer to the one presented here, where the output of a case-based module feeds a rule-based one, such as the system described in [7], a medical system for Alzheimer's Disease patients, where the case-based module is invoked to determine whether a neuroleptic drug should be prescribed to a patient and if this is so, the rule-based is invoked to select one of five drugs.

Considering BTs as a kind of planning artifact that stores hand-written plans, we can also find related work on the combination of case-based planning and other planning approaches. The SiN system [9] uses a case-based planning algorithm that combines conversational case retrieval with generative planning. SiN can generate plans given an incomplete domain theory by using cases to extend that domain theory, which is given in the form of a planning domain. SiN can also reason with imperfect world-state information by incorporating preferences into the cases. While the case-based module and the domain theory are independently developed in SiN, we propose a more efficient approach by purposely developing a domain theory to fill the holes in the empirical data.

Also in the Starcraft domain, [14] proposes some preliminary ideas to combine expert knowledge in the form of hand-written ABL plans with knowledge automatically extracted from traces using statistical techniques. Although this work is very related to ours, it is at the same time in a very early stage. Besides, using BTs instead of a planning language we facilitate the process of injecting expert knowledge to the system because the experts, in this case game designers, are used to them.

Regarding future work, we intend to explore possible techniques for facilitating the task of identifying those areas in the plan library that require the expert intervention. At this point, a major drawback of the proposed approach is that the expert needs to analyse the plan library in order to identify those plans, sub-plans or basic actions that require improvement. We envision a computer-assisted identification process, where by generating traces of the AI controlled by the plan library the system can automatically pinpoint actions and goals that usually fail as places for improvement.

References

1. AIIDE: StarCraft AI competition. As Part of the Program of the Artificial Intelligence and Interactive Digital Entertainment Conference (2010)
2. Blizzard: Starcraft game (1998), http://us.blizzard.com/en-us/games/sc
3. Flórez-Puga, G., Llansó, D., Gómez-Martín, M.A., Gómez-Martín, P.P., Díaz-Agudo, B., González-Calero, P.A.: Empowering Designers with Libraries of Self-validated Query-enabled Behaviour Trees. In: Artificial Intelligence for Computer Games, pp. 55–82. Springer, Heidelberg (2011)
4. Isla, D.: Halo 3 - building a better battle. In: Game Developers Conference (2008)
5. Krajewski, J.: Creating all humans: A data-driven AI framework for open game worlds. Gamasutra (February 2009)
6. Lee, G.H.: Rule-based and case-based reasoning approach for internal audit of bank. Know.-Based Syst. 21(2), 140–147 (2008)
7. Marling, C.R., Whitehouse, P.: Case-based reasoning in the care of alzheimer's disease patients. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 702–715. Springer, Heidelberg (2001)
8. Millington, I., Funge, J.: Artificial Intelligence for Games, 2nd edn. Morgan Kaufmann, San Francisco (2009)
9. Muñoz-Avila, H., Aha, D.W., Nau, D.S., Weber, R., Breslow, L., Yamal, F.: Sin: integrating case-based reasoning with task decomposition. In: IJCAI 2001: Proceedings of the 17th International Joint Conference on Artificial Intelligence, pp. 999–1004 (2001)
10. Ontañón, S., Bonnette, K., Mahindrakar, P., Gómez-Martín, M.A., Long, K., Radhakrishnan, J., Shah, R., Ram, A.: Learning from human demonstrations for real-time case-based planning. In: Kuter, U., Muñoz-Avila, H. (eds.) Proceedings of the IJCAI 2009 Workshop on Learning Structural Knowledge From Observations (2009), http://www.cs.umd.edu/~ukuter/struck09/index.html
11. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: On-line case-based planning. Computational Intelligence 26(1), 84–119 (2010), http://dx.doi.org/10.1111/j.1467-8640.2009.00344.x
12. Palma, R., González-Calero, P.A., Gómez-Martín, M.A., Gómez-Martín, P.P.: Extending case-based planning with behavior trees. In: 24th Florida Artificial Intelligence Research Society Conference (to appear, 2011)
13. Prentzas, J., Hatzilygeroudis, I.: Categorizing approaches combining rule-based and case-based reasoning. Expert Systems 24(2), 97–122 (2007)
14. Weber, B.: Integrating expert knowledge and experience. In: AAAI Doctoral Consortium (2010)

Selective Integration of Background Knowledge in TCBR Systems

Anil Patelia[1], Sutanu Chakraborti[1], and Nirmalie Wiratunga[2]

[1]Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai–600036, India
[2]School of Computing, The Robert Gordon University,
Aberdeen AB25 1HG, Scotland, UK
pateliaaj@cse.iitm.ac.in, sutanuc@iitm.ac.in, n.wiratunga@rgu.ac.uk

Abstract. This paper explores how background knowledge from freely available web resources can be utilised for Textual Case Based Reasoning. The work reported here extends the existing Explicit Semantic Analysis approach to representation, where textual content is represented using concepts with correspondence to Wikipedia articles. We present approaches to identify Wikipedia pages that are likely to contribute to the effectiveness of text classification tasks. We also study the effect of modelling semantic similarity between concepts (amounting to Wikipedia articles) empirically. We conclude with the observation that integrating background knowledge from resources like Wikipedia into TCBR tasks holds a lot of promise as it can improve system effectiveness even without elaborate manual knowledge engineering. Significant performance gains are obtained using a very small number of features that have very strong correspondence to how humans describe the domain.

1 Introduction

Textual Case Based Reasoning (TCBR) aims at solving new problems by reusing past experiences recorded in the form of free form (or semi-structured) text. The effectiveness of TCBR systems is critically dependent on the method used to estimate semantic relatedness between two pieces of text. As humans, we are skilled at arriving at representations that capture deeper meanings of texts that may not have a direct bearing with the surface level word forms. In doing so, we not only use an elaborate knowledge of language, but also implicitly and seamlessly integrate common-sense and background knowledge. It is thus natural to suppose that TCBR systems would also benefit from a principled integration of background knowledge. This paper reports experiments we conducted towards testing this hypothesis. A comparative study on text classification shows that background knowledge as is readily available in resources like Wikipedia can lead to improvements in retrieval effectiveness.

We extend the Explicit Semantic Analysis (ESA) [2] approach to allow easy integration of Wikipedia knowledge into instance based learners. The key idea is to treat Wikipedia articles as concepts and construct representation of documents as feature vectors over these concepts. Intuitively, the relevance of a concept to a document is estimated by measuring the overlap of the words present in the document and those present in the Wikipedia article corresponding to the concept. This approach lends itself easily to a TCBR framework since it allows for lazy incremental learning that relies on local models. Also, true to the spirit of CBR, the ESA representations are easily interpretable and retrieval or classification results can be easily explained. There are two significant questions that remain unanswered: how do we identify the set of Wikipedia articles (concepts) relevant to a given task? and can we do better by relaxing the assumption that the Wikipedia concepts are unrelated to each other? In other words, can we enrich the retrieval performance of the system by modelling the relatedness of Wikipedia concepts?

The paper is organized into the following sections. Section 2 presents a background to our work and identifies related works. Section 3 introduces four different Wikipedia article selection strategies, and section 4 describes an approach to estimate semantic relatedness between Wikipedia articles, and integrate this knowledge to obtain revised representation of cases. Section 5 presents empirical evaluation of our approaches.In Section 6, we deliberate on the key ideas behind this paper and reflect on certain research directions that are motivated by this work. Section 7 summarizes our key contributions.

2 Background and Related Work

Let us consider an example to motivate the importance of background knowledge in estimating relatedness of documents. Considering two short documents describing chess moves, one containing the word "rook" and another containing the word "bishop". If these documents share no other term, a TCBR system may not be able to relate the two documents. However, the two words rook and bishop co-occur in the Wikipedia article on chess. In this way Wikipedia knowledge can help in arriving at better models of semantic similarity between words, as well as between documents. The key idea behind ESA [2] is to treat words (and phrases) like chess as general concepts and express documents (textual cases) in terms of these concepts. More specifically, each Wikipedia article is thought of as representing a concept and each document, as well as each word, is a vector over a space defined by these concepts.

Figure 1 shows an example to illustrate the idea behind ESA. The sentences "US President summarizes his position on the Middle East" and "Israel and Palestine respond to Obamas foreign policy note" share no words. Yet we know that these sentences are strongly related to each other. The Wikipedia articles to which the words in sentence 1 has strong correspondence include {Barack_Obama, Foreign_policy_of_the_United_States, Middle_East, Afganistan}, which has a good overlap with the set of Wikipedia articles { Barack_Obama, Middle_East, Foreign_policy_of_the_United_States } that are related to sentence 2. The two sentences were orthogonal to each other in the original vector space spanned by words, but display high similarity when represented in the new vector space, where each dimension corresponds to a concept which maps on to a

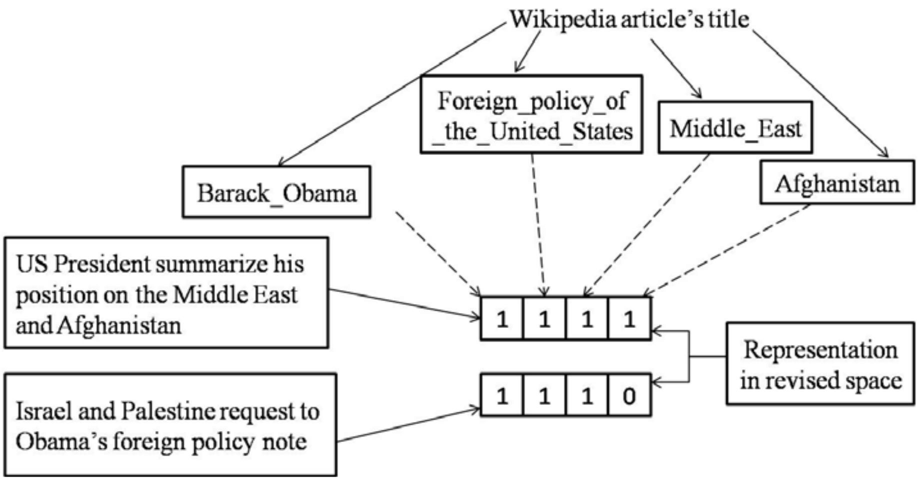

Wikipedia article's title
Foreign_policy_of _the_United_States
Middle_East
Barack_Obama
Afghanistan
US President summarize his position on the Middle East and Afghanistan
1 1 1 1
Representation in revised space
Israel and Palestine request to Obama's foreign policy note
1 1 1 0

Fig. 1. Representation of news sentences in revised space

Wikipedia article name. For a given domain, we would like to restrict attention to a small set of relevant concepts.

Whilst concepts can be constructed introspectively (e.g. Latent Semantic Indexing) it is only possible when concepts needed to construct rich representations in a domain are present within the document descriptions themselves. Consider the two sentences in Figure 1, which can only be related if we know something about politics in the US and in the Middle East. ESA has access to the world that defines the context, and can overcome this limitation. Also, concepts are Wikipedia article names, which humans find easy to relate to. Accordingly ESA provides an elegant means to incorporate background knowledge in a transparent manner.

There have been much research aimed at creating revised representations of documents based on linguistic or background knowledge. Scott et al [11] used the synonymy and hypernymy relations from WordNet[3] to revise bag-of-words representations. Zelikovitz et al. [10] present a case for transductive learning, whereby test documents (without their class labels) were treated as a source of background knowledge to make up for the inadequacy of labeled examples. Others have also attempted to mine relationships between entities in Wikipedia. This is useful for tasks like constructing domain specific resources like thesauri, taxonomies and ontology. In the CBR community, Propositional Semantic Indexing [7] has been proposed as an alternative to approaches like LSA [5]. PSI features are more expressive than those derived from LSI in that they are logical combinations of words (as opposed to linear combinations in LSI), however they can only be composed out of existing words. This means that a compact concept descriptor like, chess, or US Politics, is highly descriptive of how we view the domain cannot emerge as new features. An extracted feature in PSI can at best be a disjunction over conjunctions of several terms related to chess, and this may become unwieldy and hard to understand as the descriptions grow longer. PSI is an introspective learner and has no access to background knowledge.

2.1 Using ESA for Classification

Figure 2 illustrates how ESA can be used to represent cases for text classification tasks. Each training document is mapped to a concept representation. A concept corresponds to a Wikipedia article. The semantic similarity of each term to a concept is estimated by observing how strongly (say in terms of a tf-idf measure) the term is present in a Wikipedia article corresponding to that concept. Once we have representation of each term as a concept vector, a document (case) can be represented as a concept vector as well. The concept vector representing the document is simply the vector sum of the concept vectors corresponding to each term present in the document. The unseen test document is mapped to its concept representation, which is compared against the concept vectors of training documents, and the top k training documents according to a cosine similarity measure are used to decide the class label of the test document.

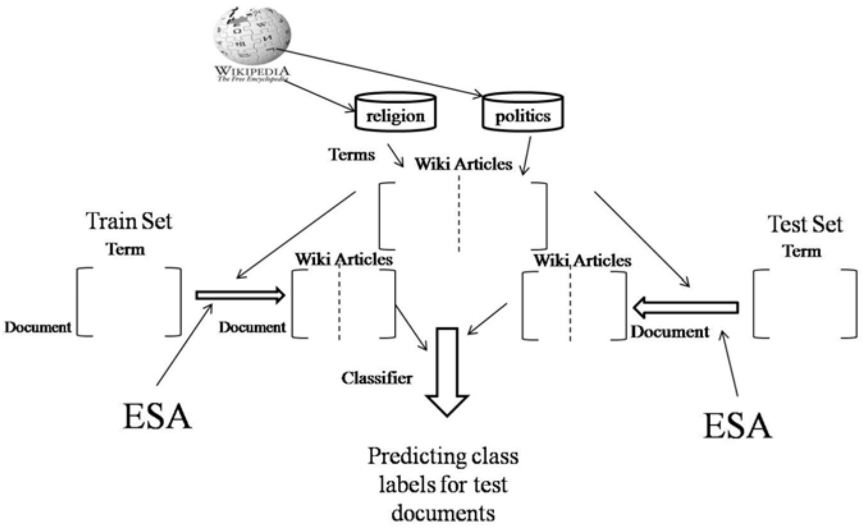


Fig. 2. Explicit Semantic Analysis for classification

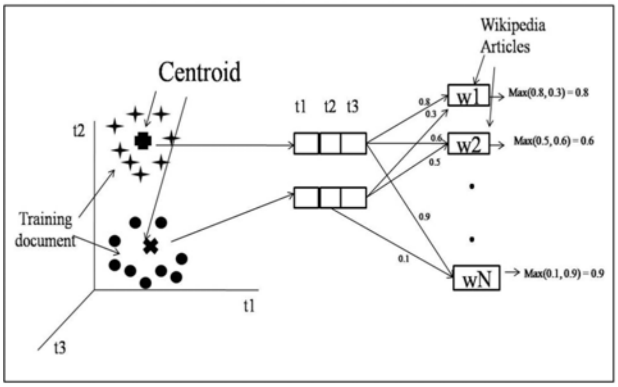
3 Informed Selection Strategies for Wikipedia Articles

While incorporation of background knowledge from Wikipedia can be useful in improving system effectiveness, it is also important to know which Wikipedia pages to actually use for modeling concepts, given a specific task like text classification and a corpus of documents (cases). One option is to look at all Wikipedia articles that contain any of the terms used in the training corpus. This may result in accumulating web-pages that are also remotely relevant to the classification task. Interestingly, Wikipedia pages are tagged with knowledge of categories (drawn from a hierarchy), and this can act as a preliminary filter. For example, in a text classification scenario where we want to discriminate between documents of classes Religion and Politics, we may only consider Wikipedia pages belonging to those categories. Sometimes, the category labels in Wikipedia will not have neat correspondence to the class labels of the domains, but it is often possible to establish a mapping. This approach of considering all pages under certain Wikipedia categories is often not adequate. Since not all Wikipedia pages tagged with the relevant category labels will help in discriminating between the classes. Thus, we may still have a large number of redundant Wikipedia pages being considered. We proposed and experimented with four different Wikipedia article selection strategies with the goal of addressing these shortcomings.

Baseline Approach. The Baseline algorithm used for our comparisons is one that compiles a collection of Wikipedia articles that have category labels relevant to the classification task, and randomly selects pages from this collection to generate ESA representations. Therefore the baseline algorithm is top-down and solely driven by category labels in the collection. Essentially it completely disregards bottom-up clues from the words actually used in the training corpus.

3.1 Centroid Strategy

The centroid strategy is founded on a vector space that is spanned by the union of all distinct words in the domain, and those that appear in Wikipedia articles relevant to the class labels. For each class, we compute the centroid of training documents in that class. Wikipedia articles are ranked based on maximum cosine similarity they have with any cluster centroid, and the top k articles are selected. Figure 3 illustrates the idea and summarizes the algorithm. The basic idea behind this approach is to select web-pages prototypical of the



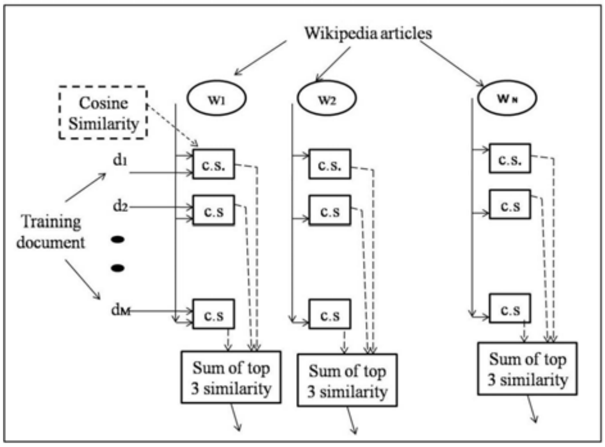
Step 1: Compute the centroids of training documents in each class.
Step 2: Compute cosine similarity between each Wikipedia article and the centroid of each class.
Step 3: Sort articles in the descending order of the maximum cosine similarity they have with any class centroid, and select the top n articles from each class.

Fig. 3. Centroid strategy for Wikipedia article selection

categories. However, one downside of this approach is that we could imagine pathological situations where certain categories starve. In other words, we are not guaranteed to obtain adequate number of representative Wikipedia pages for each category. The second limitation is that a Wikipedia article could be very prototypical of more than one class, in which case it may not be very good at discriminating between classes, even if it is ranked highly. A third limitation arises from the observation that there may be scenarios where the cluster centroids are not adequately representative of the Wikipedia pages in the corresponding categories. This situation is common in complex classification tasks where Wikipedia pages in disjoint well separated clusters are labelled with the same category tag.

3.2 k-Nearest Neighbour Strategy

In this approach, we no longer use the centroid as a representative of a class. Instead, corresponding to each Wikipedia article we identify the training documents that are closest to it in terms of the cosine similarity. A rank is assigned to a Wikipedia article based on the sum of the top three cosine similarities. The top ranked Wikipedia articles are treated as concepts for classification. Figure 4 illustrates the idea and summarizes the algorithm. This approach overcomes a key limitation of the centroid approach, in that it can handle complex classification problems where local neighbourhoods are more indicative of correct category than proximity to class centroid. A limitation of this could be that Wikipedia pages that are extremely similar to each other can get selected, leading to redundancy.



Step 1: For each Wikipedia article, find cosine similarity with each training document.

Step 2: For each Wikipedia article, sort the training documents in ascending order according to their cosine similarities and sum top three cosine similarities and assign as score it to Wikipedia article.

Step 3: Sort the Wikipedia articles according to their score.

Fig. 4. kNN strategy for Wikipedia article selection

3.3 k-Nearest Neighbour with Discrimination Strategy

This strategy is very similar to the kNN approach, except that we ensure that each class is assigned representative articles. Thus we select the top m Wikipedia articles for each class having highest cosine similarities with the three nearest neighbours in training documents of that class. This overcomes the second limitation of the Centroid strategy in that it guarantees that no class suffers from starvation.

3.4 Probability Ratio Strategy

This strategy evaluates the relative importance of a Wikipedia article to a class using probability estimates computed from the training corpus using add-1 smoothing. Given a class c drawn from a set of n categories, the posterior probability $\mathrm{P}(c|wk)$ of c given a Wikipedia article, wk is estimated. A Naive Bayes Classifier that assumes conditional independence of the features is used [9]. The Wikipedia article is assigned to the class which gives rise to the highest posterior estimate. The top few Wikipedia articles of each category are selected.

3.5 Augmented ESA

In addition to the four Wikipedia article selection strategies described above, we also carried out an experiment where a representation of a textual case was formed using a mix of words and concepts derived from Wikipedia. This was motivated by the observation that we do not wish to loose those words that are already good in discriminating between classes. This approach is referred to as Augmented ESA.

We tried an approach that attempts to directly estimate the discriminating power of a Wikipedia page, and selects those pages that allow for best discrimination between classes. A Wikipedia page is represented in terms of a vector of real valued tf-idf values over the feature space of words. So we need a discretization (binning) method to evaluate the Information Gain of the concept feature corresponding to that page. When only two bins are used, we get a binary-valued feature corresponding to each concept. The details of this binning approach are available in [13]. These discretrized concept features are stacked along with binary valued features derived from words, as in Augmented ESA, and the Information Gain of the word-level features as well as those of concept level features are evaluated. The concept features (articles) having highest Information Gain are selected. It may be noted that while using the Information Gain idea, no filtering mechanism is used to prune the set of Wikipedia articles. Rather, a huge number of relevant articles are evaluated for their Information Gain, without consideration of whether they have significant correspondence to the documents in the training corpus.

4 Modelling Similarity between Wikipedia Articles

The ESA approach assumes that each Wikipedia article represents a concept that is unrelated to all other concepts (Wikipedia articles). It is easy to see that this is at best a convenient approximation. In this section, we discuss how we incorporate the knowledge of similarity between Wikipedia articles into the revised representation of terms and documents.

A Case Retrieval Network (CRN) is used to capture the pair wise concept similarities which are used to revise the document representations. Let us consider a document as being represented as a vector, each component of which represents the relevance of the document to a concept. We can assume that these relevances are zero when a concept is not relevant to a document and 1 when it is relevant. In the CRN framework, we have similarity arcs connecting every pair of concepts as shown in Figure 5. The relevant concepts are allowed to "activate" other concepts which have non-zero similarity to it, using a process of spreading activation. At each concept node the incoming activations are aggregated and the revised document representation is a vector compromising the aggregated activation at each concept node. For example assume an initial representation of document D is 1, 1, 0, 0, 0 in the vector space of Wikipedia-based concepts W1 through W5. Let us consider the pair wise similarity values as shown in Figure 5. If the aggregation function at each node is a simple summation, the resulting representation of D ought to be 1.9, 1.9, 1.1, 0, 0. This new representation can be seen as a result of a matrix operation. Let R_i and R_n be initial and new representation of the document respectively and S be a symmetric matrix of concept pair similarities. The new representation can be given as $R_n = R_i$S.

The similarities between Wikipedia articles are estimated using Latent Semantic Analysis (LSA). This is in line with an earlier work where LSA was used for introspective knowledge acquisition in CRNs[1]. In particular we use Sprinkled LSA[8] whereby category knowledge is incorporated into the process of obtaining revised lower dimensional representations. This is motivated by the observation that while LSA dimensions capture significant variances in the data, they are not guaranteed to be the ones with highest discriminatory power. The central idea behind sprinkling is to augment a document representation with additional terms, each representative of a particular category to which the document belongs. This has the effect of pulling together documents belonging to the same category and emphasizing the distinction between documents belonging to

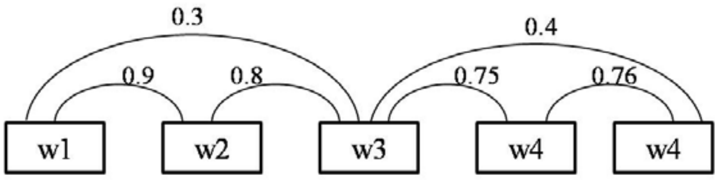


Fig. 5. Case retrieval network

different categories. The number of sprinkled (augmented) terms can be varied to control the degree to which category knowledge is emphasized. The details of this procedure are explained in [8].

5 Evaluation

We evaluated the effectiveness of our proposed integration of background knowledge from Wikipedia in the context of text classification.

5.1 Datasets and Methodology

We tried classification on four datasets created from the 20 Newsgroups [6] corpus. There are a total of twenty different news-group categories in this dataset. Each category has thousand articles drawn from postings of discussions, queries, comments etc. Four datasets were formed from the news-group:

- HARDWARE group from two hardware categories, one on MAC and the other on PC.
- RELPOL, from two groups, one concerning religion, the other politics in the middle-east.
- SCIENCE from four science related groups
- REC from four recreation related groups.

Thus HARDWARE and RELPOL are two class problems, and SCIENCE and REC are multi-class problems. Each sub-corpus was divided into train and test sets. Sizes of train and test sets are equal. Each partition contains 20% of documents randomly selected from the original corpus, and is stratified in that it preserves the class distribution of the original corpus. Fifteen such train-test splits (alternately called trials) were obtained for each of the four datasets mentioned above. It may be noted that the documents were pre-processed by removing stop words (noise words) like functional words which are frequent throughout the collection and ineffective in discriminating between classes. Weighted kNN classifier is used with $k = 3$.

Table 1 compares the accuracies of ESA against a naive bag-of-words Vector Space approach and the Baseline on 4 sub category from the 20Newsgroup. Table 2 reports the accuracies obtained when Augmented ESA representation (see Section 3.5) using a mix of concepts and words were used. As we can see, ESA techniques yield substantial improvements over Vector Space Model in each category. ESA with various Wikipedia article selection strategies also achieves much better accuracy compared to the Baseline approach that relied on an adhoc selection procedure. Results presented in Figures 6 to 9 generally suggests that classification accuracy increases as a function of the number of Wikipedia pages. This is particularly evident RELPOL and HARDWARE, where the increase is steeper than with random selection of Wikipedia articles. This shows that we can attain conspicuous improvements using fewer pages, if we adopt a principled approach to selection of Wikipedia articles.

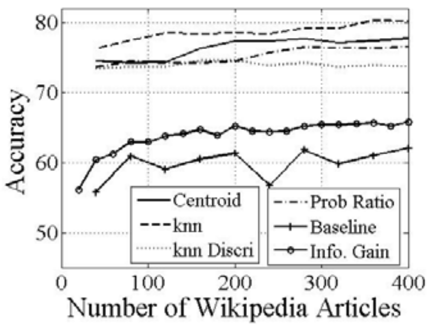


Fig. 6. ESA on HARDWARE

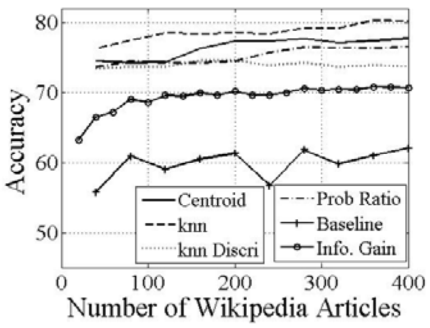


Fig. 7. Augmented ESA on HARDWARE

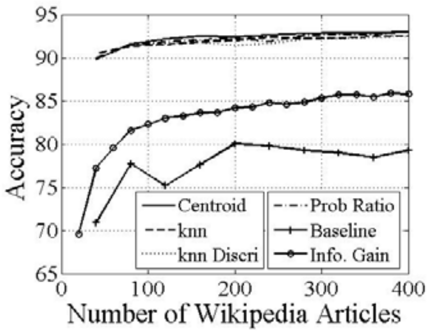


Fig. 8. ESA on RELPOL

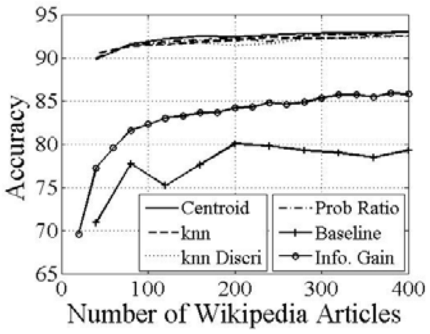


Fig. 9. Augmented ESA on RELPOL

Table 1. Comparison of performance of ESA, Vector Space Model (VSM) and Baseline

Dataset	Wikipedia document Selection strategy				Info gain	VSM	Baseline
	Centroid	Knn	knnDiscri	probRatio			
HARDWARE	77.72	76.51	74.60	**80.28**	65.79	59.51	65.77
RELPOL	92.94	92.98	92.43	92.54	85.76	70.51	80.88
SCIENCE	**81.01**	76.76	78.34	77.98	68.90	54.89	60.22
RECREATION	**83.02**	76.76	79.72	77.26	67.99	62.79	66.54

Table 2. Performance of Augmented ESA

Dataset	Wikipedia document Selection strategy				Info gain	VSM	Baseline
	Centroid	Knn	knnDiscri	probRatio			
HARDWARE	74.75	76.70	76.51	75.84	70.69	59.51	65.77
RELPOL	93.13	93.09	93.04	93.09	85.93	70.51	80.88
SCIENCE	77.76	78.16	76.44	77.63	71.88	54.89	60.22
RECREATION	**82.68**	77.45	77.31	79.23	70.32	62.79	66.54

Table 3. Performance of ESA with knowledge of concept similarities

Dataset	Wikipedia document Selection strategy				Info gain	VSM	Baseline
	Centroid	Knn	knnDiscri	probRatio			
HARDWARE	75.37	75.37	72.38	**78.12**	69.98	59.51	65.77
RELPOL	**93.08**	92.64	91.04	92.49	86.45	70.51	80.88
SCIENCE	**79.69**	77.91	76.13	75.00	70.56	54.89	60.22
RECREATION	**80.05**	72.89	76.43	75.66	70.37	62.79	66.54

Table 4. Performance of Augmented ESA with knowledge of concept similarities

Dataset	Wikipedia document Selection strategy				Info gain	VSM	Baseline
	Centroid	Knn	knnDiscri	probRatio			
HARDWARE	**76.57**	74.23	74.23	75.77	72.56	59.51	65.77
RELPOL	94.49	94.31	93.66	94.23	86.88	70.51	80.88
SCIENCE	**82.49**	80.83	79.95	79.78	72.38	54.89	60.22
RECREATION	**80.87**	78.56	77.82	79.89	73.21	62.79	66.54

5.2 Modeling Similarity between Wikipedia Articles

We empirically evaluated the impact of modelling similarity between Wikipedia pages as described in Section 4. We use Latent Semantic Indexing for modelling similarity between Wikipedia articles. The main parameter in LSI is the number of dimensions used, which should ideally be set using cross validation. The results reported in this section correspond to choice of dimensions that led to best LSI performances. Table 3 shows the classification accuracy using the revised case representation which incorporates knowledge of similarities between concepts. Table 4 shows the results when Augmented ESA representation is used, along with knowledge of similarities between concepts.

5.3 Summary of Observations

Paired one tailed t-test with 95% confidence was used to analyse the observed differences between accuracies reported by each pair of methods over the 15 train test pairs. We observe that after integration of background knowledge, effectiveness of text classification improves conspicuously. The improvements are more pronounced when the principled article selection strategies described in Section 3 are used. Importantly significant gains are seen even when fewer concept-level features are used. As shown in Table 1, Baseline algorithm performs significantly better than naive Vector Space model for each category and differences vary from 3% to 10%. Each Wikipedia article selection strategy performs significantly better than the baseline. The classification accuracy of RELPOL dataset increases from around 80% in Baseline to more than 90% for each of the different Wikipedia article selection strategies as shown in Table 1.

Comparing the Wikipedia article selection strategies, we can see that the centroid strategy is, on the whole, better than the rest. As shown in Table 2, Augmented ESA performs better than ESA on RELPOL dataset. Similarity modelling between Wikipedia articles does not improve the result for ESA presented in Table 3. In particular, the highest accuracy is 94.49% when centroid strategy is used for augmented ESA representation with similarity modelling. Augmented ESA with similarity modelling performs better than case representations that ignore original features for SCIENCE dataset, over all Wikipedia articles selection strategies. For the HARDWARE dataset, performance decreases as we try to model similarity between Wikipedia articles. A closer look suggests that there are many common Wikipedia articles belonging to both categories. For instance we have observed that articles like persona_computer, history_of_computing_hardware are selected for both categories of hardware (IBM and Mac). These pages seem to be related to both hardware.ibm and hardware.apple, and hence cannot help in the classification task. Justification for additional similarity modelling as discussed in Section 4 remains weak. For instance we found that there was a significant difference between the similarity modelling versions of ESA (except on the RELPOL dataset), where augmented ESA was found to be better.

It is interesting to note from Tables 1 through 4 that the four principal Wikipedia article selection strategies described in Section 3 far outperform the Information Gain based measure outlined in Section 3.5. This can be attributed to the fact that the Information Gain measure ignores the bottom-up information suggested by the actual words in the training corpus. The article selection strategies appear to strike a decent trade-off between selecting features that are inspired by the corpus, and those that actually contribute positively to discrimination between classes. Also, Figures 6 through 9 show that all four article selection strategies lead to performance improvements even with fewer Wikipedia articles (of the order of 50 to 100). The Information Gain based measure is less robust and shows a sharper increase as more Wikipedia based concepts are included. Improvements with injecting semantic similarity between concepts was not very pronounced, except in a few domains over select article selection strategies. In retrospect, we perhaps need to be more conservative in linking up Wikipedia articles. We may like to add measures of similarity after evaluating their potential impact on classification accuracies. The semantic similarity computation may also need to be refined by incorporating knowledge of hyperlink associations and category links attached to Wikipedia pages.

The improvement gains over complex datasets like Hardware are really encouraging. It may be noted that Apple and Mac classes in HARDWARE have good overlap of terms they share, but the classes get more easily separable when background knowledge is in place. The kNN strategy outperforms the rest in Hardware, whereas in RELPOL the differences between strategies are less pronounced. This hints at the fact that local models work better in complex domains as opposed to global ones (like centroid based strategies), as they lend themselves to modeling more complex decision boundaries.

6 Discussion and Outlook

An interesting aspect of the integration of background knowledge using principled Wikipedia article selection strategies is the fact that we can achieve significant gains in effectiveness using very few features. The graphs in the previous section illustrate that knowledge of around 30-40 semantically rich concepts constituting background knowledge in the domain is perhaps more worth knowing than blindly acquiring thousands of word-level features with the hope of learning statistical models that are severely constrained by the representativeness of the data they are presented with, and are hard to train, interpret and maintain. It is important to know which concepts will make the most impact given the task and the dataset at hand; article selection strategies presented in this paper are designed with this goal in mind.

Having very few features has implications in terms of improving retrieval efficiency. While efficiency and effectiveness are often viewed as conflicting goals, it turns out that having fewer concepts can contribute positively to realising both these goals at the same time. Hubert Dreyfus observes: AI researchers have long recognized that the more a system knows about a particular state of affairs, the longer it takes to retrieve the relevant information, and this presents a general problem when scaling up is concerned. Conversely, the more a human being knows about a situation or an individual, the easier it is to retrieve other relevant information. Additionally having access to a knowledge repository as large as the WWW can help CBR systems do better than just look at the set of cases it has immediate access to. Several research strategies view the WWW as a means to provide access to many more cases. This view can be restrictive in that it can slow down the system since the search at retrieval time has now to deal with a larger number of cases. However, if the integration of background knowledge is done intelligently, it can also help it condense the set of features that are useful in arriving at a revised representation of cases that is more reflective of their similarities. This appears more intuitive and in concordance with Dreyfus observation above. In the current paper, we are restricted to a set of features derived from the cases and from the Wikipedia articles. This is the view of the system at a given timestamp. We can extrapolate this view to a situation where the system acquires more and more cases, and as it grows in the size of the case-base, the feature set also evolves with time. The set of features can even reduce in number if we discover that all cases are about just a few underlying topics (concepts). In the context of the current paper, this implies that the cases can be meaningfully interpreted if we have access to a small number of Wikipedia articles. Since the performance of kNN-based approaches is more critically dependent on the dimensionality of the space than on the number of cases (the curse of dimensionality[6]), this progressive reduction in the dimensionality can lead to faster retrieval with fewer features.

In a general setting, the problem of determining the right set of Wikipedia articles is an optimization problem. The objective function in the supervised case corresponds to classification effectiveness averaged over the several folds created out of the training data. In an unsupervised setting, we can aim at finding features (articles) that minimize the case-base complexity. In other words we would like to construct representations of the problem and solution components of cases such that in problems close to each other in the revised problem space, correspond to solutions that are close to each other in the revised solution space. This corresponds, for example, to a TCBR system where both problem and solution components are textual. The Wikipedia articles used to describe the problem space may be very different from the Wikipedia articles used to represent the solution space.

7 Conclusion

The effectiveness of a TCBR system is critically dependent on the representation of cases, and the measure of semantic relatedness between cases. There have been several studies into introspectively learning strategies that exploit co-occurrence patterns between words and phrases. In this paper, we present an approach to selectively exploiting background knowledge to construct richer case representations. We present empirical evidence to suggest that this approach can achieve significant effectiveness gains with very few features. We compare several article selection strategies to identify Wikipedia articles that can potentially have high impact on classification effectiveness. We also examine the effects of incorporating knowledge of relatedness between these features. We hope that the paper will stimulate further research that aim at constructing TCBR systems that are knowledge rich, easy to maintain and can be adapted to capture as much domain knowledge as is needed to suit the requirements of the specific task at hand, while compensating for the lack of cases that "cover" the problem domain adequately.

References

1. Chakraborti, S., Ambati, S., Balaraman, V., Khemani, D.: Integrating knowledge sources and acquiring vocabulary for textual CBR. In: UK-CBR Workshop, pp. 74–84 (2004)
2. Gabrowich, E., Markovith, S.: Computing semantic relatedness using Wikipedia based explicit semantic analysis. In: Proc. of Int. Joint Conference on AI, pp. 1606–1611 (2007)
3. Miller, G.A., Beckwith, R., Fellbaum, C.D., Gross, D., Miller, K.: WordNet: An online lexical database. Int. J. Lexicograph, 235–244 (1990)
4. Lenz, M.: Case Retrieval Nets as a Model for Building Flexible Information Systems, PhD dissertation, Humboldt Uni. Berlin. Faculty of Mathematics and Natural Sciences (1999)
5. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science, 391–407 (1990)
6. Mitchell, T.: Machine Learning. McGraw Hill International (1997)
7. Wiratunga, N., Lothian, R., Chakraborti, S., Koychev, I.: A propositional approach to textual case indexing. In: Proc. of European Conference on Principles and Practice of KDD, pp. 380–391 (2005)

8. Chakraborti, S., Lothian, R., Wiratunga, N., Watt, S.: Sprinkling: Supervised Latent Semantic Indexing. In: Proc. of Annual European Conference on Information Retrieval, pp. 510–514 (2006)
9. Sebastiani, F.: Machine Learning in automated text categorization. ACM Computing Surveys, 1–47 (2002)
10. Zelikovitz, S., Hirsh, H.: Using LSI for Text Classification in the Presence of Background Text. In: Proc. of International Conference on Information and Knowledge Management, pp. 113–118 (2001)
11. Scott, S., Matwin, S.: Text classification using Wordnet Hypernyms. In: Workshop on Usage of WordNet in NLP Systems, pp. 45–51 (1998)
12. Rodriguez, M., Gomez-Hidalgo, Z., Diaz-Agudo, B.: Using WordNet to Complement Training Information in Text Categorization. In: The Proc. RANLP, pp. 25–27 (1997)
13. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Machine Learning: Proceedings of the Twelfth International Conference (1995)

User Satisfaction in Long Term Group Recommendations*

Lara Quijano-Sánchez, Juan A. Recio-García, and Belén Díaz-Agudo

Dep. Ingeniería del Software e Inteligencia Artificial,
Universidad Complutense de Madrid, Spain
{lara.quijano,jareciog}@fdi.ucm.es, belend@sip.ucm.es

Abstract. In this paper we introduce our application *HappyMovie*, a Facebook application for movie recommendation to groups. This system takes advantage of social data available in this social network to promote fairness for the provided recommendations. Group recommendations are based in the individual satisfaction of each individual. The (in)satisfaction of users modifies the typical aggregation functions used to estimate the value of an item for the group. This paper proposes a memory of past recommendations to compute the satisfaction of users when similar items (movies, in this case) are recommended several times.

1 Introduction

Recommender systems were born from the necessity of having some kind of guidance when searching through complex product spaces. More precisely, group recommenders were built to help groups of people, who share a common activity, decide in conflict situations. Our previous works [1,2,3,4] presented our approach of making recommendations for groups of people connected through social network structures. In them we introduced a method, based on three important features: personality, social trust and memory of past recommendations to ensure social fairness. We have proven that our theories for making recommendations to groups of people connected through social network structures improve the current existing methods.

Our current work consists on providing an Facebook application reachable to a great deal of people, where we can continue our research and experiments and also extend the group recommendations to different domains. Besides, by having the application located in a social network, we can extract the information regarding the users from it. In that way we don't have to bombard our users with questionnaires in order to build the personal profile that we need to do the recommendation, because we can extract from their Facebook profiles a great deal of the information we need. We have moved our standalone group recommendation application to a public application where everybody can benefit of it. Our method includes the analysis of the group personality composition and the trust

* Supported by the Spanish Ministry of Science and Education (TIN2009-13692-C03-03).

between users to improve the accuracy of group recommenders. This way we simulate in a more realistic way the argumentation process followed by groups of people when agreeing on a common activity. Our recommendation method and the architecture of our Facebook application are valid for any domains with rated products. However, we have applied them to an specific domain, the movie domain, and we have created *Happy Movie*.

The main goal of this paper is to analyse the impact of the last social factor involved in our group recommendation method: group satisfaction to ensure social fairness. The use of group satisfaction is based on some results from organizational behaviour and social psychology that have highlighted the concept of *emotional contagion* [5]. This social aspect states that the satisfaction of an individual is likely to depend on other individuals of the group [6,7]. In this context, social fairness is understood as the intention of maximizing the personal satisfaction of every user in the group and minimizing their differences. To achieve this goal, we propose a memory of past recommendations that is used to compute an individual satisfaction value that is later combined to estimate the global satisfaction of the group for the provided recommendations. This group satisfaction is critical in recommender systems that propose the same kind of items several times for the same group of people. Some examples are movies, music, leisure trips and any other social activity.

Several works have focused in avoiding repeated recommendations and recommendations that tend to be detrimental for the same group members repeatedly. MusicFX [8] employs a weighted random policy for selecting one of the top radio stations selected by the recommender, instead of always selecting the top category. Another solution is taking into account the history of the results produced by the recommender. For example, in FlyTrap [9] the previous selections are taken in consideration. This way, when they choose the next song to be played, abrupt changes of genre do not appear. Another system that takes into account the previous selections is PoolCasting [10]. It uses a Case-Based Reasoning system to generate a sequence of songs customized for a community of listeners. To select each song in the sequence, first a subset of songs musically associated with the last song of the sequence is retrieved from a music pool; then the preferences of the audience expressed as cases are reused to customize the selection for the group of listeners; finally, listeners can revise their satisfaction (or lack of) for the songs they have heard.

The process that we have followed when making group recommendations which ensure social fairness is very similar to the Case Base Reasoning (CBR) cycle [11]. CBR is a successful and established methodology in Artificial Intelligence that has inspired us in the implementation of our recommender with memory. In our system each recommendation provided by the group recommender will be stored as a new case that can be used later to improve the next recommendation. This fact corresponds with the *retain phase* in the CBR cycle. This way, we acquire the experiences that will be useful for the resolution of future recommendations because we will know which products have been recommended to a group. We also store how much satisfied each member of the group is with this recommendation, so we are able to adjust the satisfaction factor in future recommendations. Before making the following recommendation we will check the previous situation, which in the CBR cycle will be the *retrieve phase*. Once we obtain that information we can perform a new recommendation, but taking into account what we have retained (the products that we have already recommended and how satisfied each of the members of the group are). This will be equivalent to the *reuse phase* in the CBR cycle. Last but not least, we will modify the recommendation so that the proposed products are not repeated and we assure a certain degree of fairness when we benefit the preferences of each users. This last phase, the *revise* one, will close de CBR cycle.

The next section details our recommender application. In Section 3 we explain the recommendation techniques used by the application to select items for the group. Experimental evaluation is exposed in Section 4 and finally conclusions are detailed in Section 5.

2 Facebook Application: Happy Movie

HappyMovie is a Facebook application where we provide a group recommendation for groups of people that wish to go together to the cinema. Although this application has been initially designed for recommending movies, this domain will be extended as our method is valid for any other domain with rated products.

In the following sections we are going to explain the uses of the application and its architecture.

2.1 Uses of Happy Movie

In order to use our application, users only have to start their Facebook account and look for *HappyMovie* in the applications section. We explain the uses of the application through an example of a given group of people connected in the social network. The necessary steps to obtain a movie recommendation for a group with Happy movie are explained below:

– **Creating the user profile in the application:** Before any user has access to the movie recommendation results we have to create their individual "recommendation profile" which is necessary for our recommendation method. This profile is based on three different aspects: personality, individual preferences and trust with other users. To obtain the personality and preferences, users must answer two different tests. The first one is the personality test, where users have to choose a series of characters to whom they feel identified, as shown in Figure 1 (left image). Once they have answered the personality test, users have to rate a set of movies (at least 20 movies), where they suggest their personal preferences , as shown in Figure 1 (right image). Finally, to obtain the last factor –trust– the application reads the information stored in the Facebook personal profile. It calculates the trust that the user has with all the other users that have joined the event up to now.

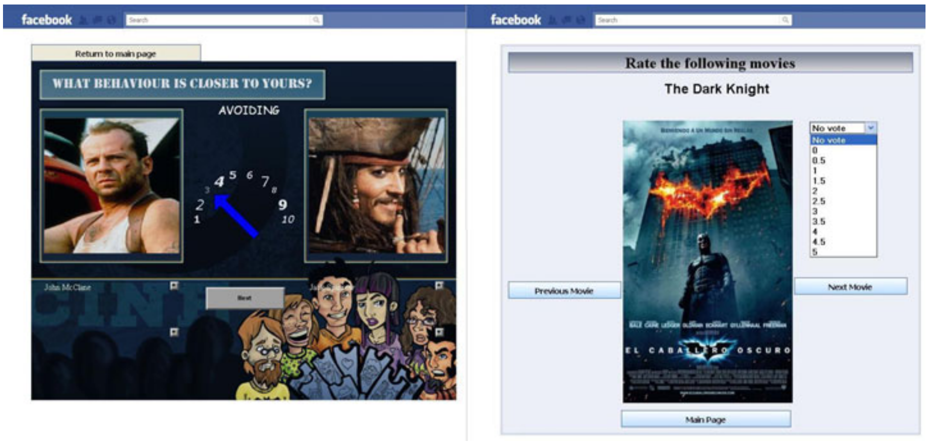
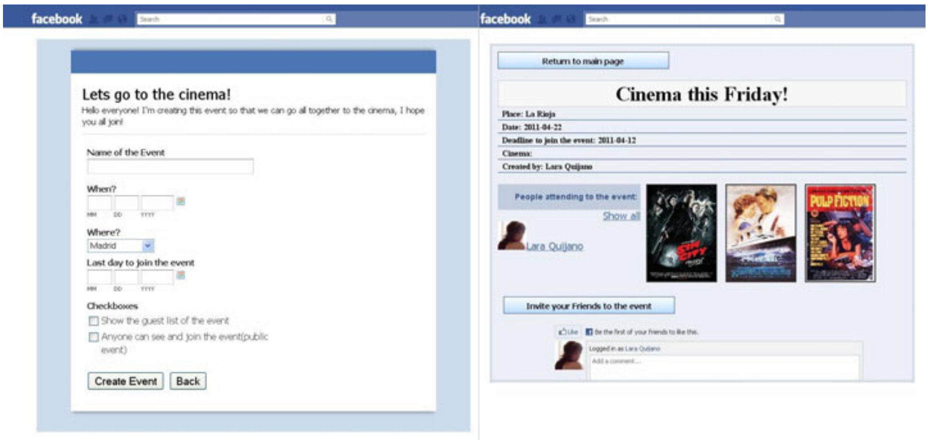


Fig. 1. Personality and Preferences test in *HappyMovie*

- **Creating the activity:** The organizer user, U_1 decides to organize an activity and starts the application to create a new event "*Let's go to the cinema*" as shown in Figure 2 (left picture). To create the event, organizers must establish some data like place, date or invited users. Once the event has been created any user attending the event can see the date and place of the event and a proposal of three movies, that are the best ones that our group recommender has found for the current group of users attending the event. For example, when U_1 starts the application for the first time she has the role of *organizer*. As an *organizer* she firstly invites some of her Facebook friends to the event. Lets say that she invites U_2 and U_3. Next, she chooses the place and date where the event will take place.

  Once the *organizer* finalizes this initial configuration she will continue with a role of *common user*. When users participate in the event as *common users* they are also able to invite any Facebook friends they wish and they can retire from the event at any time. For example, U_2 accepts the invitation of the event and later she invites her Facebook friend U_4. On the other hand, U_3 initially accepts the invitation and joins the event but later she decides against going, so she retires from it.
- **Recommendation method:**When the application has obtained the three factors that identify each user that joins the event (personality, individual preferences and trust between other users) it provides a group recommendation using our concrete method which is explained in Section 3.
- **Having the recommendation made:** When the event is created it looks up for the current movie listing from the selected city and provides a list of 3 movies, which represent the best 3 movies that the recommender has found in the movie listing for the users that have joined the event up to now, this is shown in Figure 2 (right picture). This list is automatically updated every time a user joins the event or retires from it. This process keeps going on until the day that the *organizer* has selected as final date. In our example, it


facebook
Search
Create Event
Back
Return to main page
People attending to the event:
Show all
Lara Quijano
Invite your Friends to the event
Like
Be the first of your friends to like this.
Add a comment...

Lets go to the cinema!

Hello everyone! I'm creating this event so that we can go all together to the cinema, I hope you all join!

Name of the Event

When?

MM DD YYYY

Where?

Madrid

Last day to join the event

MM DD YYYY

Checkboxes

☐ Show the guest list of the event

☐ Anyone can see and join the event(public event)

Cinema this Friday!

Place: La Rioja

Date: 2011-04-22

Deadline to join the event: 2011-04-12

Cinema:

Created by: Lara Quijano

Logged in as Lara Quijano

Fig. 2. How to create an activity in *HappyMovie* and how events look like in *HappyMovie*

initially provides a recommendation for users U_1, U_2 and U_3 when they first join the event. Later, when U_3 retires from the event a new recommendation is made for users U_1 and U_2. Finally when U_4, who was invited by U_2, joins the event another new set of 3 movies appears for users U_1, U_2 and U_4. When the expiration date is reached users can see the final movie list. In that moment they are allowed to vote each of the movies individually. This process lets us decide which movie they are finally watching and, more important, it provides the required feedback to evaluate the quality of our recommendation.

2.2 Happy Movie's Modular Architecture

The architecture of *HappyMovie* is represented in Figure 3. We can see that the application is divided in seven different modules: TKI Metaphor, Facebook Profile Analysis, Satisfaction Data Base, Web Test, Web Crawling, Content Based Estimation and *HappyMovie*'s group recommender. Next sections explain what are the basis of each of these modules.

– **TKI Metaphor:** This module fulfils the task of obtaining a value that represents the personality of each user. To do so, each user must answer to a personality test that measures people's behavior in conflict situations. In our previous works [1,2,3] we used the TKI personality test [12], that consists on 30 questions where the user has to decide how she will react in the exposed situations. As we have prove in our previous works this is a tedious test to answer but its results are very reliable. To make the application more easy going we have introduced a movie metaphor as an alternative method to obtain the users personality in conflict situations. This interactive metaphor consists on displaying two movies characters with opposite personalities for five

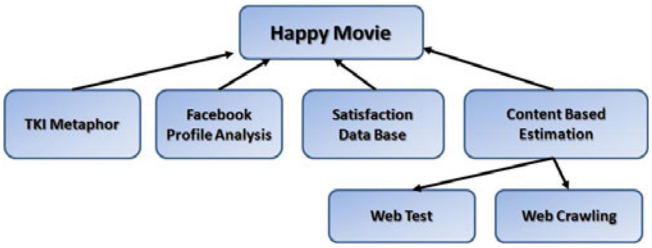


Fig. 3. Facebook application arquitecture

different modes of responding to conflict situations (this personality modes are the ones presented in the TKI experiment). One character represents the essential characteristics of the category, while the other one represents all the opposite ones. What the user has to do is to choose with whom of each pair of characters she feels more identified by simple moving a graded arrow, as seen in Figure 1 (left image).

– **Facebook Profile Analysis:** Once *Happy Movie* is running, the trust module must perform its estimation every time a user joins the event. When this happens the trust module explores the users who are currently in the list of attending users and calculates the trust of each of them with the user who has recently joined the event. To do so, the profiles of the two users will be analysed, comparing different social factors. The Trust Module is the module that has more benefits due to embedding the application in a social network. Previously, with a standalone application, the task of obtaining the data required to compute the trust between users was very tedious. Now, we are able to calculate the trust between users extracting the specific information from each of their own profiles in the social network. Users in Facebook can post on their profiles a huge amount of personal information that can be analysed to compute the trust to other users: distance in the social network, number of shared comments, likes and interests, personal information, pictures, games, duration of the friendship, ... We analyse 10 different trust factors comparing the information stored in their Facebook profiles. Next, these factors are combined using a weighted average. A detailed explanation of the trust factors obtained from Facebook and the combination process is provided in [3].
– **Satisfaction Data Base:** In this module we store all the recommendations that have been made for every user and every group. This specific feature of our application is fully detailed in Section 3.1. This module manages the following information: Each group to whom each user has participated on, each movie that each group has watched and the satisfaction of each user with each of her groups.
– **Web Test:** It consists on a test of the individual preferences of each user. Each time that the user uses the application she can modify her preferences or evaluate more, however it is only compulsory the first time she uses the application. In our specific domain, movies, the user will see the title of the

movie and the movie poster. Our users are provided with 50 movies from which they have to rate (in a rank of 0-5) at least 20, this is shown in Figure 1 (right image). These preferences are stored as the individual case base of each user.

- **Web Crawling:** This module searches the web and finds the movie listing of the city that the *organizer* has selected. Once it has that information it searches the complete file of each of the movies in the movie listing. Later, it analyses the file and extracts all the data required to define the movie. Each specific data is a field that the individual recommender contrasts. For example, in our particular case of study, these fields are main actors, director, year,etc. The recovered items, with all their specific information, are sent to the individual recommender module and to the group recommender module as they are the products to be recommended.
- **Content Based Estimation:** This is the individual recommender module, it is built using the jCOLIBRI framework extension to build recommender systems [13] and follows a content based approach [14] that uses descriptions of the products to be recommended and returns the collection of products that are more similar to the aimed product. In these particular case of study, *HappyMovie*, it returns the best three movies a user should watch individually. As it is a content based recommender system it has stored a case base, and the recommender compares all the considered items to be recommended with this case base. This case base is different for each user and has the information retrieved from the Web Test module.

3 Group Recommendation Methods

We have developed a group recommendation method which is based on the typical preference aggregation approaches. These approaches [5,15] aggregate the users individual predicted ratings $pred(u,i)$ to obtain an estimation for the group $\{gpred(G,i)|u \in G\}$. Then the item with the highest group predicted scoring is proposed.

$$gpred(G,i) = \bigsqcup_{\forall u \in G} pred(u,i) \tag{1}$$

Here G is a group of users, which user u belongs to. This function provides an aggregated value that predicts the group preference for a given item i. By using this estimation, *our group recommender proposes the set of k items with the highest group predicted scoring.*

In our proposal, we modify the individual ratings with the personality, trust and satisfaction factors. This way, we modify the impact of the individual preferences as shown in Equation 2.

$$\begin{aligned} gpred(G,i) &= \bigsqcup_{\forall u \in G} pred'(u,i) \\ pred'(u,i) &= \bigsqcup_{\forall v \in G} f(\, pred(u,i)\, ,\, p_u\, ,\, t_{u,v}\, ,\, s_u) \end{aligned} \tag{2}$$

where $gpred(G, i)$ is the group rating prediction for a given item i, $pred(u, i)$ is the original individual prediction for user u and item i, p_u is the personality value for user u, $t_{u,v}$ is the trust value between users u and v, and s_u is the satisfaction of user u within the group.

A wide set of aggregation functions has been devised for combining individual preferences [16], being the average, least misery and most pleasure strategies the most commonly used:

– **Average Satisfaction:** Refers to the common arithmetic mean, which is a method to derive the central tendency of a sample space. It computes the average of the predicted ratings of each member of the group. The function that represents this strategy is:

$$gpred(G, i) = \frac{1}{|G|} \sum_{u \in G} pred'(u, i) \tag{3}$$

Where $pred'(u, i)$ is the predicted rating for each user u, and every item i. $gpred(G, i)$ is the final rating of item i for the group.

– **Least Misery:** This strategy considers that a group is as happy as its least happy member. The final list of ratings is the minimum of each of the individual ratings. A disadvantage can be that if the majority really likes one item, but one person does not, then it will never be chosen.

$$gpred(G, i) = \min_{u \in G} pred'(u, i) \tag{4}$$

– **Most Pleasure Strategy:** It is the opposite of the previous strategy, Least Misery, it chooses the highest rating for each item to form the final list of predicted ratings.

$$gpred(G, i) = \max_{u \in G} pred'(u, i) \tag{5}$$

Once we have introduced the typical aggregation approaches we can explain the estimation functions. We use three different methods to compute $pred'_{i,u}$, that as we have explained before, is a modification of the predicted rating for a user according to the personality, trust and satisfaction factors. The main ideas of these approaches are explained below:

– **Personality-based method (*pbm*):** Takes into account the differences in the personalities between pairs of individuals in the group. It is based on the modified average satisfaction employed in our previous work [1]. This strategy reflects that assertive characters will have more influence in the aggregated scoring than the cooperative characters. Our approach uses the type of personality to weight the influence of the estimated ratings during the recommendation process.

– **Delegation-based method (*dbm*):** The idea behind this method is that users create their opinion based on the opinions of their friends. It tries to simulate the following behaviour: when we are deciding which item to choose within a group of users we ask the people who we trust. Moreover, we also take into account their personality to give a certain importance to their opinions (for example, because we know that a selfish person may get angry if we do not choose her preferred item).

- **Influence-based method (*ibm*):** Simulates the influence that each friend has in a given person. It supposes that the user may modify her preference for an item depending on the preferences of her friends to the same item. For example, if our rating for an item is 3 and our friend has a 5 rating for the same item, we could think on modifying our rating to 4. Depending on the trust with this friend, we decide the level of variation for our rating (i.e. 3.5 if the trust is low, and 4.5 if trust is high). Furthermore, the variation of our rating also depends on our personality. If we have a strong personality we will not be willing to change our rating, but if we have a weak personality we could be easily influenced by other users.

Once the estimation methods are outlined, next section details how to include the memory of users satisfaction in the recommendation process.

3.1 Including User Satisfaction in the Recommendation Process

Our approach for including fairness is based on the satisfaction of the users that conform a group. We propose a modification of the previous methods including a satisfaction parameter that measures the degree of happiness of every user regarding past recommendations. Our goal is to maximize the satisfaction among the group by promoting the items preferred by most dissatisfied users. To achieve this goal we need to keep track of past recommendations and the evolution of the satisfaction of each user. In *HappyMovie* this task is delegated to the Satisfaction Data Base module (see Figure 3).

Having recommendations with memory means that we are able to create a system that remembers all the previous recommendations for a given group. It is a Case-Based reasoning approach to the recommendation process. and a necessary step when providing a whole set of fair recommendations. This way, if one member accepts a proposal that she was not interested in, next time she will have some kind of priority in the recommendation process. This means that her opinion will have a higher weight next time. These weights will also be influenced by the different personalities of each group member. For example, someone with a strong personality that has been negatively affected would be immediately compensated next time; however someone with mild personality would not have problems giving in several times.

The satisfaction value s_u is the level of satisfaction of a user u. A user who is extremely happy with the recommendations will have this satisfaction measure value close to 1. However, the more upset with the recommendations she is, the more that this value will decrease, reaching down to 0 in the worst case. An important and interesting issue of this approach is the time scope of the memory of user's satisfaction. We can update the s_u value to reflect the satisfaction according to the last immediate group recommendation or to take into account previous ones. Therefore, the satisfaction value for an execution t of the recommender may depend on the satisfaction of the user with the items recommended in t but it also depends on her satisfaction in the previous recommendations $t-1, t-2, \ldots$. Therefore we manage two satisfaction values:

- Instant satisfaction (is_u): reflects the immediate satisfaction of the user with the last recommendation. This is, her conformance with the last item recommended to the group. Its value can be obtained by estimating the preference of the user for the item selected to the group among all the items available.
- Global satisfaction: (s_u): measures the average satisfaction of the user among time. It is updated every time a recommendation is made:

$$s_u(t) = (1-\delta) \cdot is_u(t) + \delta \cdot s_u(t-1) \quad (6)$$

  In this equation we use the $\delta \in [0..1]$ parameter to adjust the impact of the previous satisfaction when updating that value. Somehow, this parameter measures the degree of forgetfulness about past (in)satisfacion. For example, some people could easily remember that they were not taken into account for the last recommendation when facing a new decision making process to select a similar item. On the other hand, other users won't ever take it into account. The measurement of this value belongs to the domain of the social sciences and is out of the scope of this paper. We have estimated it experimentally as exposed in next section.

It is important to note that the instant satisfaction value is also weighted depending on the personality of the user to reflect the importance of satisfying that concrete user.

In next Section we explain the details of our experimental evaluation to measure the impact of memory in the recommendation process of *HappyMovie*.

4 Case Study: Experimenting with Memory

Our goal is to estimate what is the best recommendation strategy for long term recommendations in *HappyMovie*. This strategy will be updated once we have real user data available. However, initially we must estimate this strategy to provide recommendations to our users. Therefore we have designed an experiment with synthetically generated data about users and movies. We must note that the validity of experimenting with this synthetically generated users has been already proven in our previous studies [3]. We simulate user preferences and personality to simulate different scenarios where several groups of users choose a movie for going to the cinema. In our previous work [4] we have evaluated the estimation strategies –personality (*pbm*), delegation (*dpm*) and influence(*ibm*)– without taking into account the memory of past recommendations. These experiments were performed with data from real volunteers that simulated going to the cinema together. Results shown that *dbm* and *ibm* provide better results than *pbm* when including the trust factor (t_u). However for our simulation it is impossible to generate synthetically that value and therefore, we have focused on the *pbm* to evaluate which is the best aggregation function: average satisfaction, least misery or most pleasure.

4.1 Experimental Set-Up

The experiment configuration runs as follows. We have simulated 1000 groups of individuals going to the movies together 15 times. Each group consists of 10 individuals. Although the composition of the group does not change, having such a large number of groups let's us include in the simulation any kind of variation in their composition. Movies are described by means of a vector that represents the degree of conformance with several genres (terror, action, romance). These genres were obtained from the MovieLens database [17]. Correspondingly, cinema preferences of each individual are represented in the same way. Movies and individual preferences are generated randomly and compared using the cosine distance to obtain what would be the real rating of an individual for a given movie. This real rating (referred as $rr(u,m)$ with range [0..10]) will be later used to evaluate the performance of our recommender.

Our recommendation method is based on an individual recommender that estimates the rating $pred(u,m)$ given by a user to a movie. This recommender has been implemented using the jCOLIBRI framework [13] and follows the collaborative filtering approach described in [18] based on the Pearson Correlation. This method requires a population of previous individuals that have rated several items (movies). These users and their ratings are also generated synthetically.

Finally, the personality of each user is assigned according to the probability distribution inferred from the 50 volunteers that took part in our first experiments. It is important to note that we could also apply the distribution used by the original TKI test[1].

On each round of the simulation (there are 15 rounds per group) we generate a random movie listing composed of 10 movies. Our group recommender predicts what is the best movie for the group $gpred(G,m)$. Then, the proposed movie is compared with the real preferences of each individual to compute their instant satisfaction and the global satisfaction of the group. To obtain the instant satisfaction we order the movie listing for each individual according to the real rating that she would assign to each movie. Next we compare what is the position of the movie selected for the group in that list. Instant satisfaction will be higher if $gpred(G,m)$ is in the first positions and lower it is at the end of the list. Instead of using directly the position of the movie recommended for the group in the individual ordered list, we slightly modify that position according to the personality of the individual. A user with a strong personality won't be happy if the movie is in the second or third position of her preference list because she will expect to see her first favourite movie. On the other hand, an individual with mild personality won't mind to watch a movie in the middle of her preferences list. We refer to this value as the *dislike* factor. And it is linearly weighted to compute the instant satisfaction:

$$is_u = (size(ml) - dislike(u,m))/size(ml) \tag{7}$$

$$dislike(u,m) = index(m,ml) + p_u \tag{8}$$

[1] TKI personality distribution is obtained from a population of 8000 individuals from U.S.A.

where ml is the movie listing proposed to the users, and $size(ml)$ returns its size. $m \in ml$ is the movie recommended by our system for the group. The position of a movie in the movie listing once ordered according to the user preferences is obtained by means of the $index(m, ml)$ function. Finally, p_u is the personality of the user with range $[-1, 1]$.

Once is_u is obtained, the global satisfaction s_u is obtained. We have configured a δ value of 0.5 to represent balanced impact of previous satisfaction.

4.2 Results

We have run the experiment with the three aggregation functions: average satisfaction (AS), least misery (LM) and most pleasure (MP). These aggregation functions combine the individual prediction for each movie. This prediction is obtained by means of the Personality-based method. It is computed as follows: (1) the individual estimation of the rating given to the movie is returned by the collaborative individual recommender. (2) This rating is weighted according to the personality of the user p_u. (3) Resulting rating is again modified according to the user satisfaction s_u. Step (3) tries to promote those movies that have a high estimated rating for an unsatisfied user. On the other hand it decreases the final rating of movies with low estimation (to avoid their selection for the group). If a user is satisfied, ratings are slightly modified. Analogously, step (2) takes into account the personality of the user to promote the movies preferred by users with strong personalities.

In our evaluation we have studied the effect of the previous modifications. We refer to BASE when we only apply step (1) to obtain the individual prediction (note that it is the standard aggregation function and the baseline of our metric). The measures including only steps (1) and (2) are referred as PERSONALITY as they only take into account the personality factor. Finally, the complete method including steps (1), (2) and (3) is named as (PERSONALITY_MEMORY) because it includes both the personality factor and the satisfaction memory.

To measure the performance of the group recommender we use the average accumulative satisfaction of the group. The accumulative satisfaction is the sum of the individual satisfaction s_u of a user after n cinema events. This way, a user that had a high satisfaction in several events will finish with a high accumulative satisfaction and a user that was not taken into account will have a low value. To reflect the satisfaction of the group we compute the mean of the accumulative satisfaction of every user in that group.

However, our goal is to maximize the mean satisfaction but minimizing the standard deviation. The mean represents the global satisfaction of the group and reflects the goodness of the items recommended by our system. On the other hand, the standard deviation reflects the differences in the satisfaction levels within the group. It is a measure of the social fairness. As conclusion, our evaluation function is the mean value minus the standard deviation of the accumulative satisfaction of the group ($\bar{x} - \sigma$), where a higher is better.

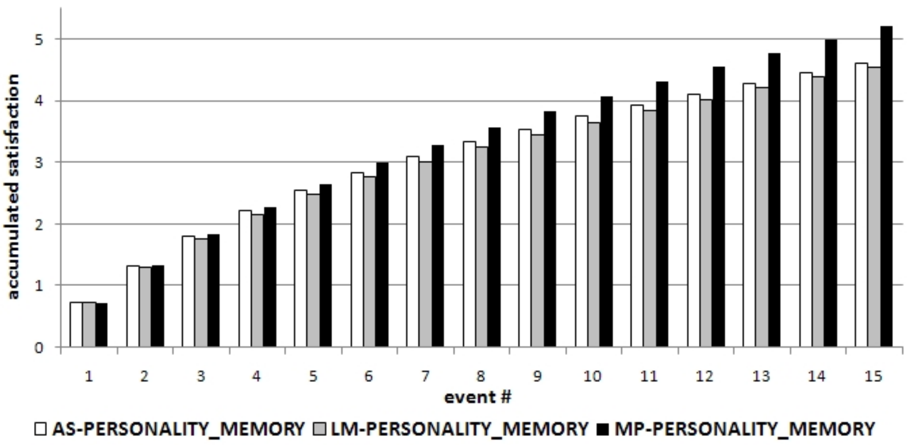


Fig. 4. Evaluation results

Results of this evaluation function are shown in Figure 4. For clarity reasons BASE and PERSONALITY methods are not shown. All of them obtain worse results than the PERSONALITY_MEMORY approach. Using just personality is slightly better than the BASE methods. As we could expect, the inclusion of personality and memory provides the best performance, being the most pleasure (MP) aggregation function the optimal approach. It is followed by average satisfaction (AS) and finally we find least misery (LM). This behaviour can be explained thinking about the nature of these aggregation functions and the bias that we promote when including the memory of user satisfaction. The main consequence of including the memory of user satisfaction is that we minimize the standard deviation within the group (i.e. maximize the fairness). By definition, LM gives not very good results in average but maximizes the fairness. However, AS and MP provide respectively good and very good recommendations in average but don't care about the fairness. With the inclusion of the satisfaction in the estimation functions we remove this drawback found in AS and MP. Therefore, they provide high values in average and minimize the standard deviation. Concretely, MP returns the item with the higher rating for the user, even if that item is a very worse recommendation for other users. This fact maximizes the mean satisfaction, and our bias ensures that others' satisfaction does not decrease too much. This is the reason it obtains the highest performance.

When analysing the mean and the standard deviation separately, we have not found very significant differences in the mean satisfaction. However the standard deviation is highly influenced by the inclusion of the memory. We have also measured the impact of modifying the size of the group, the movie listing length and the number of events, not finding any correlation between these variables and the system performance.

5 Conclusions

In this paper we have introduced our Facebook application Happy Movie. It is a group recommender for the movies domain that takes advantage of the social variables available in social networks that can be exploited to improve the performance of the system. We propose the inclusion of the following social factors: personality of every group member, trust between users, and a memory of users satisfaction to promote fairness. In this paper we have focused in this last factor –memory of users satisfaction– as we propose a CBR approach to modify the items presented to the group depending on the evolution of this satisfaction.

Our approach can be applied with several aggregation functions –that provide global recommendation for the group– and different estimation measures that predict the rating a user would assign to a given item. We have run an experiment with synthetic data to obtain the best approach for the *HappyMovie* application. Results show that optimal performance is obtained by means of the most pleasure aggregation function together with the inclusion of personality and memory in the estimation. Our future work consists on confirming these results with real users that could use our application to organize their cinema events.

Acknowledgments. Supported by Spanish Ministry of Science & Education (TIN2009-13692-C03-03) and Madrid Education Council and UCM (Group 910494).

References

1. Recio-García, J.A., Jimenez-Diaz, G., Sánchez-Ruiz, A.A., Díaz-Agudo, B.: Personality aware recommendations to groups. In: Bergman, L.D., Tuzhilin, A., Burke, R.D., Felfernig, A., Schmidt-Thieme, L. (eds.) Procs. of the 2009 ACM Conference on Recommender Systems, pp. 325–328. ACM, New York (2009)
2. Quijano-Sánchez, L., Recio-García, J.A., Díaz-Agudo, B.: Social based recommendations to groups. In: Procs. of the 14th UK Workshop on Case-Based Reasoning, pp. 46–57. CMS Press, University of Greenwich (2009)
3. Quijano-Sánchez, L., Recio-García, J.A., Díaz-Agudo, B.: Personality and social trust in group recommendations. In: Procs. of the 22th International Conference on Tools with Artificial Intelligence, ICTAI 2010 (to appear, 2010)
4. Quijano-Sánchez, L., Recio-García, J.A., Díaz-Agudo, B., Jiménez-Díaz, G.: Social factors in group recommender systems. In: ACM-TIST, TIST-2011-01-0013 (to be published, 2011)
5. Masthoff, J., Gatt, A.: In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. User Modeling and User-Adapted Interaction 16, 281–319 (2006)
6. Barsade, S.G.: The ripple effect: Emotional contagion and its influence on group behavior. Administrative Science Quarterly 47, 644–675 (2002)
7. Hatfield, E., Cacioppo, J., Rapson, R.: Emotional Contagion. Studies in Emotion and Social Interaction. Cambridge University Press, Cambridge (1994)

8. McCarthy, J.F., Anagnost, T.D.: MusicFX: An arbiter of group preferences for computer aupported collaborative workouts. In: CSCW 1998: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, pp. 363–372. ACM, New York (1998)
9. Crossen, A., Budzik, J., Hammond, K.J.: Flytrap: intelligent group music recommendation. In: IUI, pp. 184–185 (2002)
10. Baccigalupo, C., Plaza, E.: A case-based song scheduler for group customised radio. In: Weber, R., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 433–448. Springer, Heidelberg (2007)
11. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues,methodological variants, and system approaches. Artificial Intelligence Communications 7, 39–59 (1994)
12. Thomas, K., Kilmann, R.: Thomas-Kilmann Conflict Mode Instrument, Tuxedo, N.Y. (1974)
13. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J.A., Sánchez-Ruiz-Granados, A.A.: Building cbr systems with jcolibri. Sci. Comput. Program. 69, 68–75 (2007)
14. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007)
15. O'Connor, M., Cosley, D., Konstan, J.A., Riedl, J.: Polylens: a recommender system for groups of users. In: ECSCW 2001: Proceedings of the Seventh Conference on European Conference on Computer Supported Cooperative Work, pp. 199–218. Kluwer Academic Publishers, Norwell (2001)
16. Masthoff, J.: Group modeling: Selecting a sequence of television items to suit a group of viewers. User Modeling and User-Adapted Interaction 14, 37–85 (2004)
17. Bobadilla, J., Serradilla, F., Hernando, A.: Collaborative filtering adapted to recommender systems of e-learning. Knowl.-Based Syst. 22, 261–265 (2009)
18. Kelleher, J., Bridge, D.G.: An accurate and scalable collaborative recommender. Artif. Intell. Rev. 21, 193–213 (2004)

Using Personality to Create Alliances in Group Recommender Systems[*]

Lara Quijano-Sánchez, Juan A. Recio Garcia, and Belén Díaz-Agudo

Dep. Ingeniería del Software e Inteligencia Artificial,
Universidad Complutense de Madrid, Spain
jareciog@fdi.ucm.es, belend@sip.ucm.es

Abstract. Our recent work analyses the accuracy of group recommenders when using information about the personality and the social connections between the members of the group. The goal in this paper is the use of personality and trust as the mean to define alliances to reach agreements inside a group of people. The approach reproduces the behaviour of real users when negotiating a common item to consume using three variables: personality, trust and personal preferences. We run an experiment in the movie recommendation domain where we use a personality test to identify the group leaders and test the number of people they are able to convince about a certain item to consume.

1 Introduction

Recommender systems have been one of the main application areas of the techniques commonly used in the Case-Based Reasoning field [1,2]. The analogies between Case Based Reasoning (CBR) and recommenders are obvious. Recommender systems manage items instead of cases but the retrieval methods are very similar. Once the best item is obtained it is proposed directly to the user without requiring adaptation. Moreover, both techniques pay an important attention to the learning processes that improve the performance of the systems by taking into account the preferences or experiences of the users. In a general way we could apply two different approaches. Collaborative recommenders use the ratings already assigned by the users to several products. Users are selected according to their similarity with the target individual (by comparing the ratings given to the products). Most similar users are used as predictors and their ratings are combined to estimate the rating that the target user would assign to a new product. On the other side, the content-based approach compares each item to be proposed with the items already rated by the target user. Then the ratings of the most similar rated items are combined to provide an estimation.

Our recent work [3,4,5,6] analyses the accuracy of group recommenders when using information about the personality and the the social connections between the members of the group. Typically a group recommender uses several subsets

[*] Supported by the Spanish Ministry of Sci. and Ed. (TIN2009-13692-C03-03).

of preferences -one per person- that are combined to create a global recommendation suitable for everyone in the group. Simpler existing works on group recommender systems are based on the aggregation of the preferences of every member of the group, where each member is considered with the same degree of importance [7,8]. However, groups of people can have very different characteristics like size and can be made of people with similar or antagonistic personal preferences. It is a fact that when we face a situation in which the concerns of people appear to be incompatible, a conflict situation arises.

Our previous approaches determine that the general satisfaction of the group is not always the aggregation of the satisfaction of its members as different people have different expectations and behaviour in conflict situations. The *personality* factor reflects the cooperativeness or selfishness of each user when selecting a product for the whole group. This fact is taken into account in recent works that agree on the need of adapting the recommendation process to the group composition. Furthermore, it is also known that the user preferences can be affected by other people of the group and can change over the time [9,2,10]. Personality allows us measuring the degree of acceptance of the products proposed by other users and the way of solving conflicts. Our research characterizes people using the Thomas-Kilmann Conflict Mode Instrument (TKI) [11] that describes a person behavior in conflict situations.

The concept of *trust* [12], can be defined as the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible. Trust networks consist of transitive trust relationships between people, organizations and software agents connected through a medium for communication and interaction. Note that trust is also related to tie strength and previous works have reported that both are conceptually different but there is a correlation between them [13].

In this paper we describe a new approach to solve conflict situations by modeling users interaction in group recommender systems. Instead of computing a global recommendation for the group of people based on the individual preferences and personality of its members, we propose a model where each user negotiates to convince other members about a common item to consume. We exploit the principle of *homophily*, people that share interest with their friends and tend to be friends with people who share their interests. This feature has been shown to exist in many social networks [14,15]. In our model, users with strong personalities try to create *alliances* with other users to support their personal preferences. This way, influencer users obtain the required votes to get their proposal chosen by the group. These influencers, or leaders, try to influence other users and they use their leadership to create the alliance.

Influencers, are typically characterized as thought leaders, or just plain interesting personalities who have the ability to influence potential users. In practice, these individuals may be identified as highly connected individuals or individuals that bridge (also called *connectors* [16]) two relatively large sub-communities. This social behaviour has been extensively researched in the social sciences over the past few decades [17],[14],[18].

Our new approach uses personality and trust as the mean to define *alliances* inside a group of people. An alliance is defined as a subgroup that agree about the same recommendation result. A leader creates alliances with other users (s)he trusts in order to support a concrete product p . The product in the alliance with the bigger number of members is chosen as the global recommendation result. A total agreement situation leads to an alliance including all the people of the group.

Summing up, in this paper we propose a model based on alliances to provide recommendations to groups. We identify leaders by a personality test. Potential allies are obtained by computing the trust between users. Leaders negotiate with their closer friends to conform an alliance that has the majority of votes required to get the influencer's favourite items.

The paper runs as follows. Section 2 introduces related work. In Section 3 and 4 we explain an overview of our previous research, a generic architecture for group recommendations, ARISE, that uses personality and trust values in order to improve group recommendations. Section 5 describes the method based on alliances that we propose in this paper. Section 6 describes a case study in the movie recommendation domain and presents some results on the use of alliances in the group decision making. Section 7 concludes the paper.

2 Related Work

Related works about creating alliances and the role of influencers are shown in some online social communities. A *coalition* from social agents area is defined as a temporary association between agents in order to carry out joint projects. The aim is to achieve complex projects by using a better distribution of competencies. An example is the approach of [19] to solve a cooperative game. Different works study automatic methods for coalition formation [20] or properties like efficiency, optimality or stability of the coalition structure [21,22]. Our approach is also related with voting games [23], a popular model of collaboration in multiagent systems. In such games, each agent has a weight (intuitively corresponding to resources he can contribute), and a coalition of agents wins if its total weight meets or exceeds a given threshold.

Our theory is based on the idea of a distributed group recommender system based on previous research on distributed Case Based Reasoning. Distributed CBR assumes multi-case base architectures involving multiple processing agents differing in their problem solving experiences [24]. In this new scenario each case base contains a list of contents, like products, rated by the user. These ratings represent the users explicit preferences that belong to the user model. These individual ratings are later combined with the ratings from other users to obtain a joint recommendation for the group. CBR literature proposes several ways to combine several experiences to obtain improved solutions in distributed architectures. One important method is the *ensemble effect* explained in [25] which proves that the argumentation of two agents improves the results obtained by one only agent working with the same experiences. This conclusion was the precursor of a research line focused on finding the best argumentation protocols to allow CBR agents to discuss about a common problem. In [25] they came up with the AMAL protocol that enables several CBR agents to argue about a common problem by means arguments and counterarguments. We have adapted the idea of agents giving arguments to validate their proposal, to an approach where the agents are influencers who give arguments to try to convince other users they are close to, to support their proposal.

The motivation and main contribution of this work is to use the ideas of alliances formation and collaboration between agents to improve group recommender systems. However in our model people of the same alliance do not collaborate to solve a complex project but reach an agreement on the item to be consumed by the whole group. So, our model does not represent knowledge about agent competencies or resources to contribute. It represents information about people's preferences, personality and trust that are used to convince the other members in the group.

In our method, leaders, who we call influencers, try to wield influence over friends to achieve their own goals. This must be taken into account when recommending items to groups of friends. The main problem when applying this model is the identification of potential influencers and influenced friends. However social networks provide (partially) these data. We can compute the trust between users to measure the closeness of their relationship and therefore the possibility of influence. However, social connections aren't enough for identifying influencers. To do so, we propose to measure the personality of the users.

3 ARISE: Generic Architecture for Group Recommenders Using Social Elements

Our approach, presented in [4,5,6] determines that the general satisfaction of the group is not always the aggregation of the satisfaction of its members, as groups of people can have very different characteristics. The inclusion of social elements into a group recommendation strategy is what we call ARISE[1] (Figure 1). This architecture allows us to simulate in a more realistic way the decision process followed by groups of people when choosing a joint activity.

The architecture of ARISE [6] is divided in six different modules: personality, trust, memory and satisfaction individual preferences estimation, explicit individual preferences, and product data. The information provided by each module is combined by the ARISE's group recommendation methods described in Section 4. Next, we summarize modules functionality:

– **Personality Module.** When making group decision processes there are situations where the concerns of people appear to be incompatible and *conflict situation* arises. Different people have different expectations and behaviour in conflict situations that should be taken into account. We have studied the different behaviours that people have in conflict situations according to their

[1] ARISE stands for Architecture for Recommenders Including Social Elements.

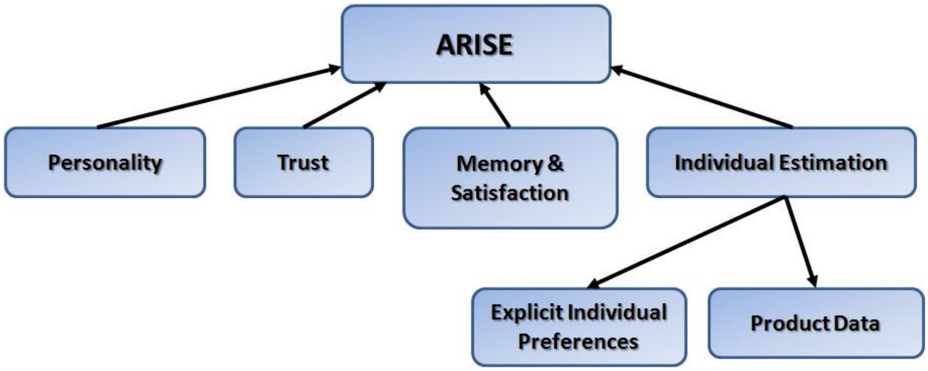


Fig. 1. Facebook application architecture: ARISE

personality. Personality module fulfils the task of obtaining a value that represents the personality of each user. This value, p, is a number $\epsilon[0, 1]$ where 1 represents a very strong personality and 0 a very easy going personality. In the ARISE architecture it is described as a high-level module that can be implemented in different ways. We obtain this factor using a popular personality test called TKI [11]. We have chosen this test because it takes very little time to answer it and the questions about the users personality are asked in an indirect way, not digging into too personal questions. In that way users do not resent from a excessively tedious test to answer.

– **Trust Module.** Current research has pointed out that people tend to rely more on recommendations from people they trust (friends) than on recommendations based on anonymous ratings [26]. In this module we evaluate information stored in our users profiles inside a social network, Facebook. With this information we compute the trust between users. Examples of these *social factors* are distance in the social network, number of common friends, intensity, intimacy or duration of the relationship.

  The details of the trust and the personality computation are fully detailed in [4,5].

– **Memory and Satisfaction Module.** After applying the personality and trust factors we must assure a certain degree of satisfaction between all the members of the group. We propose the use of a memory of past recommendations. Having recommendations with memory means that we are able to create a system that remembers all the previous recommendations for a given group. We believe that this is a necessary step when providing a whole set of fair recommendations.

– **Individual Preferences Estimation.** Our recommendation strategies predict the rating that each user would assign to every item in the catalogue and then these estimated ratings are combined to obtain a global prediction for the group. Finally, the product with the highest prediction is proposed. Therefore, a basic building block of the architecture is the module in charge of the computation of the individual predictions. For the construction of the individual recommender we use the jCOLIBRI framework [27]. jCOLIBRI is currently a reference platform in the Case-Based Reasoning (CBR) community that facilities the design of different types of CBR applications and it has a specific extension for developing recommender systems.

Independently of the approach chosen to implement this generic module of the ARISE's architecture, there are two components (or submodules) that are always required by the individual recommender: A) the explicit individual preferences, which spans any kind of information about the user that is required to predict the rating for a new item. Commonly, it just consists on the ratings given to some products in the catalogue. B) the product data set, which provides the information about the items in the catalogue that should be recommended to the group.

4 Group Recommendation Methods in ARISE

Our group recommendation method is based on the typical preference aggregation approaches. These approaches [7,8] aggregate the users individual predicted ratings $pred(u,i)$ to obtain an estimation for the group $\{gpred(G,i)|u \in G\}$. Then the item with the highest group predicted scoring is proposed, this group recommendation method is what we call a base group recommender.

$$gpred(G,i) = \bigsqcup_{\forall u \in G} pred(u,i) \tag{1}$$

Here G is a group of users, which user u belongs to. This function provides an aggregated value that predicts the group preference for a given item i. By using this estimation, *our group recommender proposes the set of k items with the highest group predicted scoring.*

In our proposal, we modify the individual ratings with the personality and trust factors. This way, we modify the impact of the individual preferences as shown in Equation 2.

$$\begin{aligned} gpred(G,i) &= \bigsqcup_{\forall u \in G} pred'(u,i) \\ pred'(u,i) &= \bigsqcup_{\forall v \in G} f(\, pred(u,i)\, ,\, p_u\, ,\, t_{u,v}\,) \end{aligned} \tag{2}$$

where $gpred(G,i)$ is the group rating prediction for a given item i, $pred(u,i)$ is the original individual prediction for user u and item i, p_u is the personality value for user u and $t_{u,v}$ is the trust value between users u and v.

There are several ways to modify the predicted rating for a user according to the personality and trust factors. These strategies will be depicted in Section 4.2. Next, we will explain the aggregation functions that can be applied to combine the individual estimations.

4.1 Aggregation Functions

A wide set of aggregation functions has been devised for combining individual preferences [9], being the average and least misery strategies the most commonly used. In the experiments presented in this paper we use the average satisfaction strategy, ir refers to the common arithmetic mean, which is a method to derive the central tendency of a sample space. It computes the average of the predicted ratings of each member of the group. The function representing this strategy is:

$$gpred(G,i) = \frac{1}{|G|} \sum_{u \in G} pred'(u,i) \tag{3}$$

Where $pred'(u,i)$ is the predicted rating for each user u, and every item i. $gpred'(G,i)$ is the final rating of item i for the group.

4.2 Modifying Individual Predictions with Social Elements

Our recommendation approaches [5] consist on evaluating the different behaviours that people have when reaching a decision making process. To do so we modify the predictions made by the individual recommender with the personality and trust factors. In that way not all the predictions are taken into account equally. We use two different methods to compute the new individual rating ($pred'(i,u)$) used in Equation 2.

– **Delegation-based method:** The idea behind this method is that users create their opinions based on the opinions of their friends. The estimation of the delegation-based rating ($dbr(u,i)$) given an user u and an item i is computed in this way:

$$pred'(u,i) = dbr(u,i) = \frac{1}{|\sum_{v \in G} t_{u,v}|} \sum_{v \in G \wedge v \neq u} t_{u,v} \cdot (\ pred(v,i) + p_v\ ) \tag{4}$$

In this formula, we take into account the recommendation $pred_{v,i}$ of every friend v for item i. This rating is increased or decreased depending on her personality (p_v), and finally it is weighted according to the level of trust ($t_{u,v}$). Note that this formula is not normalized by the group size and uses the accumulated personality. Therefore, this formula could return a value out of the ratings range. This is simply managed by the recommender by choosing the closest value within the valid range.

– **Influence-based method:** This method simulates the influence that each friend has in a given person. Instead of creating a new preference, it supposes that the user may modify her preference for an item depending on the preferences given by her friends to the same item, as shown in the following formula:

$$pred'(u,i) = ibr(u,i) = pred(u,i) + (1-p_u)\frac{\sum_{v\in G \wedge v\neq u} t_{u,v}\cdot (\, pred(v,i) - pred(u,i)\,)}{|G|-1} \tag{5}$$

In this formula, the individual rating for the item ($pred_{u,i}$) is modified according to its difference with the ratings of other users ($pred_{v,i} - pred_{u,i}$). This difference takes into account the trust between users ($t_{u,v}$). Finally, the accumulated difference is weighted according to our personality in an inverse way $(1 - p_u)$.

Next section presents the main contribution of this paper, a new group recommendation strategy, that uses the information retrieved by the ARISE architecture, personality, trust and personal preferences in order to provide a group recommendation based on alliances. It consists on a new approach to modify individual predictions with social elements, different from the *delegation-based* and *influence-based* methods that we have just explained.

5 Alliance Based Approach

Alliance based approach first computes personality and trust for every user in the group as explained in section 3. Next step uses this information to identify the leader users and her close friends set. Every user with a personality higher than a threshold α is considered a group leader. In Section 6 we use α as the 85% of the highest personality value in the group. Note that the number of group leaders is not fixed. We have empirically discovered in our case of study that our method performs better when we obtain a number of leaders close to half of the size of the group. For every leader in the group l, we obtain her close friends set $cfs(l)$. This set is obtained using the trust values computed between the leader and every other user in the group and then selecting the users that the leader trusts higher. This set represents all the "possible alliance mates". If the trust between a user, u_i and the leader l is higher than another threshold, β, she is included in her $cfs(l)$.

Negotiation between l and $cfs(l)$ begins to agree on a common product that the leader l likes. This negotiation process allows us to determine whether the proposal made by the leader is accepted or not. It runs as follows:

1. For every user in the group we obtain the individual estimation of ratings of the products in the catalogue. We use the *Individual preferences estimation* module of the ARISE architecture (see Section 3) by applying an individual recommendation approach with the information retrieved in the *explicit individual preferences* module. The construction of the recommender runs as follows.
2. Analyze the recommendations made to the leaders and identify which are their favourite items. This set of items, $lfi(l)$ (leaders favourite items), are the ones that each leader proposes to her close friends set $cfs(l)$ in order to

form the alliance. Note that the size of $lfi(l)$ is not fixed, it can be adjusted depending on the size of the catalogue of items. There are n ($n = |l|$) sets of leaders favourite items ($lfi(l)$), one for each leader in the group.

3. Propose the leaders individual favorite items $lfi(l)$ to leader l "possible alliance mates". A proposal is accepted if the estimated rating that a user u_i, with $u_i \epsilon cfs(l)$, has of the proposed item p_i, with $p_i \epsilon lfi(l)$, is higher than a certain threshold δ. This threshold δ is modified depending on the users personality (it will be bigger with stronger personalities) and also depending on the trust with the leader (if the user has a strong trust on the leader the threshold will be lower). See Equation 6 in Section 6.
4. When an user accepts the proposal we include her in the alliance of that leader. We note that the leader has θ ($\theta = |lfi(l)|$) attempts to "persuade" each one of the users in her $cfs(l)$, one attempt for every item in the set of the leaders favourite items. To be part of the alliance a user just has to accept one item of the proposed list. As we have said before, a leader l creates alliances $alli(l, p)$ with other users supporting a concrete product p_i. If the size of the alliance $|alli(l, p)|$ is greater than a half of the group, the items in $lfi(l)$ are directly chosen as the items for the group. If there is no majority we will choose the items proposed by the larger coalition.

6 Case Study: Movie Recommendation

In this section we evaluate the alliance based approach for group recommendation using the movie recommendation domain. The goal of the experiment is improving other group recommender approaches. We compare the results obtained using alliances with a base group recommender system using the average satisfaction aggregation function and also with our previous approaches using personality and trust [4,5]. The construction of the alliances recommender involves the processing of several factors that are obtained in different ways. The personality values are obtained through the TKI tests [11], whereas trust values are directly extracted from a social network where all the users belong to. Next we explain how we extract the information required from our users, how we measure the results, the configuration of our alliances recommender and the results of the experiment.

6.1 Experimental Setup

In order to perform our experiment in the movie recommendation domain, we created two events in two different social networks, Facebook[2] and Tuenti[3]. In these events we asked some of our users to complete three questionnaires[4]. The first questionnaire serves to obtain the personality of each user, is the one run by the *personality module*. Second questionnaire gets the individual preferences

[2] http://www.facebook.com
[3] http://www.tuenti.com. The most popular social network in Spain.
[4] Questionnaires are accessible at http://www.lara.warhalla.com/ (in Spanish.)

of the user about cinema. Users have to evaluate 50 heterogeneous movies from the MovieLens data set [28](rating them with a range of 0.0 to 5.0). These 50 movies are the list of products that are assigned to each agent, and they are stored in the *Explicit individual preferences module*.

Finally, third test asks users to choose their 3 favourite movies from a list of 15 *recent* movies (of the 2009 year), that represents a movie listing from a cinema. This list of 15 products is the one gathered by the *Product Data module*. The movie listing was chosen from movies of the MovieLens database using a diversity function. The 3 movies selected by each user are included as her individual favourites, *if*. These movies are the ones she would actually like to watch or had enjoyed best. The answers to these questionnaires are analysed to define the user profile of each participant. 58 real users have participated in our experiment.

To measure the accuracy of the group recommendation we brought our users together in person and ask them to mix differently several times and simulate that they are going to the cinema together, forming different groups that would actually come out in reality. We provide them the 15 movies that represent our movie listing and we ask them to choose in the group which 3 movies in order they actually would watch together. We manage to gather 10 groups: 6 groups of 5 members and 4 groups of 9 members. The three movies that each group chooses are stored as the *real group favourites* set –*rgf*–. This way, to evaluate the accuracy of our recommender we can compare the set proposed by the recommender –the *pgf* set– with the real preferences *rgf*. The evaluation metrics applied to compare both sets are explained in Section 6.2.

Our group recommendation strategies combine individual recommendations to find an item (movie) suitable for any user in the group. This individual recommender is built using the jCOLIBRI framework [29] and follows a content based approach [30] to find the most similar movie rated by the user. It uses product descriptions and returns the collection of products that are more similar to the aimed product, assigning the rating given by the user as a prediction. This set of movies is different for each user and it has the information retrieved from the second questionnaire.

6.2 Evaluation Metrics

Our experiment requires an evaluation function to measure the accuracy of the group recommendation. To do so, we compare the results of our recommender system to the real preferences of the users (that is, what would happen in a real life situation). When we started our evaluation process we took into account the number of estimated movies that we were going to take into account. We are not interested on a long list of ordered items that estimates movies a user or group should watch. Real users are only interested on a few movies they really want to watch. This fact discards several evaluation metrics that compare the ordering of the items in the real list of favourite movies and the estimated one (MAE, nDCGs, etc.). On the other hand, the number of relevant and retrieved items in our system is fixed. Therefore, we cannot use general measures like recall or precision. However, there are some metrics used in the Information Extraction field [31] that limit the retrieved set. This is the case of the *precision@n* measure that computes the *precision* after n items have been retrieved. In our case, we can use the *precision@3* to evaluate how many of the movies in pgf are in the rgf set (note that $|rgf| = 3$). This kind of evaluation can be seen from a different point of view: we are usually interested on having at least one of the movies from *pgf* in the *rgf* set. This measure is called *success@n* and returns 1 if there is at least one hit in the first n positions. Therefore, we could use *success*@3 to evaluate our system computing the rate of recommendations where we have at least one-hit in the real group favourites list. For example, a 90% of accuracy using *success*@3 represents that the recommender suggests at least one correct movie for the 90% of the evaluated groups. In fact, *success*@3 is equivalent to having *precision*@3 $> 1/3$. We can also define a 2*success*@3 metric (equivalent to *precision*@3 $> 2/3$) that represents how many times the estimated favourites list *pgf* contains at least two movies from *rgf*. Obviously, it is much more difficult to achieve high results using this second measure.

6.3 Alliance Recommender System

For each group we build the alliance recommender using the following steps:

1. We obtain the members of the group and we calculate an estimation of their individual preferences with content based individual recommender system. After this process what we have is an estimated rating of each user for each of the 15 movies in the movie listing from the cinema.
2. We identify the leaders of the group, which are those who have a personality that is higher than the 85% of the personality value of the user with the strongest personality in the group (threshold α).
3. For each of the leaders we try to find alliances. To find the possible candidates that could form the alliance we select those users who have a trust with the leader higher than the 75% of the trust value of the most trusted user of the leader (threshold β).
4. To accept a user as part of the coalition, we propose the 3 movies that the leader of the group has with the higher rating, that as we remember we obtained from the individual recommender. If the users predicted rating for that movies is higher than threshold δ then the user is accepted as part of the alliance. Threshold δ is obtained with the following formula:

$$\delta = ir_{u,5} - t_i + p_r \tag{6}$$

where $ir_{u,5}$ is the predicted rating of the best fifth item for the user, $t_i = \mu * trust_{u,leader}$, and $p_r = \lambda * p_u$. $trust_{u,leader}$ represents the existing trust between the user and the leader, p_u is the personality value of the user and μ and λ have been experimentally obtained. ($\mu > 0.4$ and $\lambda < 0.5$).

We have built another alliance recommender system simplifying this last formula, we call it *Alliance-based Recommender simpler version*, we have done this to study the influence and benefits of using the trust and personality factors in order to vary the threshold of acceptance of a proposed item δ. This variation of our method obtains the threshold δ with this simplified formula:

$$\delta = ir_{u,5} \tag{7}$$

where $ir_{u,5}$ is the predicted rating of the best fifth item for the user.

5. After forming all the alliances we compare the sizes of the alliances. If the size of the alliance is greater than a half of the group we propose as selected items, the favourites of the leader, which are the 3 movies that the leader of the group has with the higher rating.

6.4 Experimental Results

In Figure 2 we have analyzed the performance of the base recommender, a group recommender using the same data-set but applying our influence-based recommendation method, a group recommender using the same data-set applying our delegation-based recommendation method, a group recommender with the simplified version of our alliances approach (the one that does not use personality and trust factors in order to calculate the threshold of acceptance of each item) and finally a recommender with our alliances approach. We can see that we have improved the performance of the basic recommender in a 10% with the success@3 and in a 40% with the 2success@3. Results also show that with the 2success@3 measure the alliances approach obtains the best results. As we have explained before this measure is much more difficult to obtain than the success@3 measure, so with this results we validate our alliances method and conclude that

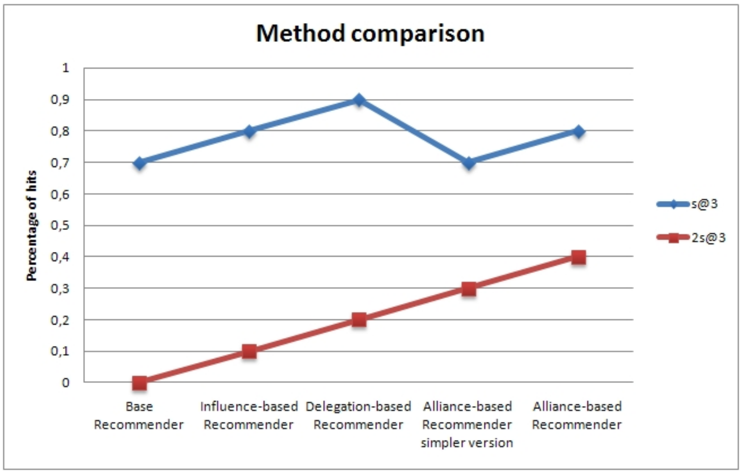


Fig. 2. Comparison of the results obtained with the base recommender, the influence-based recommender, the delegation-based recommender, the alliance-based recommender simpler version and alliance-based recommender

with it we improve our previous group recommendation strategies. From this Figure we also observe that it is essential to include the personality and trust factors in order to calculate the threshold of acceptance of each item, because with the simplified version of our alliance approach the results with the success@3 measure are equal to the base recommender so we do not improve with it the group recommendation. We must note that we still can validate our strategy because for the 2success@3 even with the simplified version of our alliance approach results are better than the ones obtained by the base, influence-based, and delegation-based recommenders.

7 Conclusions

In this paper we have proposed and evaluated a group recommendation strategy based on alliances for the recommendation of products in social networks. In previous papers we have already experimented with a novel method of making recommendations for groups taking into account the group personality composition and the social structure of the group. Once shown that personality profiles can improve a recommendation for a group of people, we have extended this approach by reflecting in a more realistic way the social relationships between the users involved in the recommendation. We have tested our method in the movie recommendation domain and shown that group recommendation using alliances improves the base group recommender system using the average satisfaction aggregation function. Results also have shown that with the 2success@3 measure the alliances approach obtains the best results and improve our previous group recommendation strategies. We have also observed that it is essential to include the personality and trust factors in order to calculate the threshold of acceptance of each item in the recommender system. Our proposed alliance based approach for group recommendation is based on identifying users with strong personalities try to create *alliances* with other users to support their personal preferences. This way, influencer users obtain the required votes to get their proposal chosen by the group. These influencers, or leaders, try to influence other users and they use their leadership to create the alliance. The proposed method first computes personality and trust for very user in the group and then uses this information to identify the leader users and her close friends set. Negotiation between the leader and people from her close friends set begins to agree on a common product that the leader likes. This negotiation process allows us to determine whether the proposal made by the leader is accepted or not. Our ongoing work consists on making further evaluations of our alliances method by embedding it into a social network application, where we will be able to continue our experiments with larger and more general populations.

References

1. Bridge, D., Göker, M.H., McGinty, L., Smyth, B.: Case-based recommender systems. Knowledge Engineering Review 20, 315–320 (2006)

2. Jameson, A., Smyth, B.: Recommendation to groups. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 596–627. Springer, Heidelberg (2007)
3. Recio-García, J.A., Jimenez-Diaz, G., Sánchez-Ruiz, A.A., Díaz-Agudo, B.: Personality aware recommendations to groups. In: Procs. of the 2009 ACM Conference on Recommender Systems, pp. 325–328. ACM, New York (2009)
4. Quijano-Sánchez, L., Recio-García, J.A., Díaz-Agudo, B.: Personality and social trust in group recommendations. In: Procs. of the 22th Int. Conference on Tools with Artificial Intelligence, ICTAI 2010, pp. 121–126. IEEE Computer Society, Los Alamitos (2010)
5. Quijano-Sánchez, L., Recio-García, J., Díaz-Agudo, B., Jiménez-Díaz, G.: Social factors in group recommender systems. In: ACM-TIST, TIST-2011-01-0013 (in press, 2011)
6. Quijano-Sánchez, L., Recio-García, J.A., Díaz-Agudo, B., Jiménez-Díaz, G.: Happy movie: A group recommender application in facebook. In: 24th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2011 (2011)
7. Masthoff, J., Gatt, A.: In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. User Modeling and User-Adapted Interaction 16, 281–319 (2006)
8. O'Connor, M., Cosley, D., Konstan, J.A., Riedl, J.: Polylens: a recommender system for groups of users. In: ECSCW 2001: Proceedings of the Seventh Conference on European Conference on Computer Supported Cooperative Work, pp. 199–218. Kluwer Academic Publishers, Norwell (2001)
9. Masthoff, J.: Group modeling: Selecting a sequence of television items to suit a group of viewers. User Modeling and User-Adapted Interaction 14, 37–85 (2004)
10. Chen, Y.L., Cheng, L.C., Chuang, C.N.: A group recommendation system with consideration of interactions among group members. Expert Syst. Appl. 34, 2082–2090 (2008)
11. Thomas, K., Kilmann, R.: Thomas-Kilmann Conflict Mode Instrument, Tuxedo, N.Y (1974)
12. Josang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision, pp. 618–644 (2007)
13. Levin, D.Z., Cross, R., Abrams, L.C.: The strength of weak ties you can trust: the mediating role of trust in effective knowledge transfer. Management Science 50, 1477–1490 (2004)
14. Miller Mcpherson, L.S.L., Cook, J.M.: Birds of a feather: Homophily in social networks, pp. 415–444 (2001)
15. Lazarsfeld, P., Merton, R.: Friendship as a social process: A substantive and methodological analysis. In: Berger, M., Abel, T., Page, C. (eds.) Freedom and Control in Modern Society, pp. 18–66. Van Nostrand, New York (1954)
16. Gladwell, M.: The tipping point: How little things canmake a big difference. Little Brown, Boston (2000)
17. Burt, R.S.: Toward a structural theory of action: Network models of social structure, perception and action, pp. 1336–1338 (1982)
18. McPherson, M., Smith-Lovin, L.: Homophily in voluntary organizations: Status distance and the composition of face-to-face groups, pp. 370–379 (1987)
19. Chalkiadakis, G., Elkind, E., Markakis, E., Polukarov, M., Jennings, N.R.: Cooperative games with overlapping coalitions. J. Artif. Intell. Res. (JAIR) 39, 179–216 (2010)

20. Barton, L., Allan, V.H.: Methods for coalition formation in adaptation-based social networks. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676, pp. 285–297. Springer, Heidelberg (2007)
21. Airiau, S., Sen, S.: On the stability of an optimal coalition structure. In: Proceedings of 19th European Conference on Artificial Intelligence, ECAI 2010, Lisbon, Portugal, August 16-20, pp. 203–208 (2010)
22. Nuno, D., Ao, S.J.S., Helder, C.: Agent-based social simulation with coalitions in social reasoning. In: Moss, S., Davidsson, P. (eds.) MABS 2000. LNCS (LNAI), vol. 1979, pp. 244–265. Springer, Heidelberg (2001)
23. Elkind, E., Chalkiadakis, G., Jennings, N.R.: Coalition structures in weighted voting games. In: 18th European Conference on Artificial Intelligence, ECAI 2008. Frontiers in Artificial Intelligence and Applications, vol. 178, pp. 393–397. IOS Press, Amsterdam (2008)
24. McGinty, L., Smyth, B.: Collaborative case-based reasoning: Applications in personalised route planning. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 362–376. Springer, Heidelberg (2001)
25. Ontañón, S., Plaza, E.: An argumentation-based framework for deliberation in multi-agent systems. In: Rahwan, I., Parsons, S., Reed, C. (eds.) Argumentation in Multi-Agent Systems. LNCS (LNAI), vol. 4946, pp. 178–196. Springer, Heidelberg (2008)
26. Sinha, R.R., Swearingen, K.: Comparing recommendations made by online systems and friends. In: DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries (2001)
27. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Prototyping Recommender Systems in jCOLIBRI. In: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 243–250. ACM, NY (2008)
28. Bobadilla, J., Serradilla, F., Hernando, A.: Collaborative filtering adapted to recommender systems of e-learning. Knowl.-Based Syst. 22, 261–265 (2009)
29. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J.A., Sánchez-Ruiz-Granados, A.A.: Building cbr systems with jcolibri. Sci. Comput. Program. 69, 68–75 (2007)
30. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007)
31. Oard, D.W., Baron, J.R., Hedin, B., Lewis, D.D., Tomlinson, S.: Evaluation of information retrieval for e-discovery. Artif. Intell. Law 18, 347–386 (2010)

Analogy-Making for Solving IQ Tests: A Logical View

Henri Prade and Gilles Richard

IRIT, Université Paul Sabatier, 31062 Toulouse Cedex 09, France
{prade,richard}@irit.fr

Abstract. Among the diverse processes at work in human cognition, the ability to establish analogies plays a crucial role and is often evaluated via IQ tests where an incomplete sequence has to be completed with a suitable item. This has motivated the AI community for developing various computational models of analogy-making. A Boolean logic view of analogical proportions (a basic form of analogical statements of the form "*a* is to *b* as *c* is to *d*") has been recently proposed and extended to another logical proportion, namely paralogical proportion (stating that "what a and b have in common, c and d have it also"). When used in combination, these 2 proportions provide an enhanced power to complete IQ tests. This Boolean modeling essentially relies on the assessment of the differences and similarities between the items involved, and in the case of analogy, satisfies the expected properties of an analogical proportion. An extension to multiple-valued features has also been defined, reinforcing their scope of applications. It is then possible to complete, in a deterministic manner, some incomplete proportions where the last item *d* is missing. In this paper, we show how this can be the basis of a simple inference paradigm that provides a rigorous way to solve representative analogy-based IQ tests by computing the missing items rather than by choosing in a list of options. The result of the analogical/paralogical inference depends on the way the items are represented. The paper discusses how this approach can be used in analogy-making for both determining missing items in proportions and laying bare the relation linking the components of such proportions. The novelty of the approach is stressed w.r.t. other proposals existing in the literature.

Keywords: analogical reasoning, analogical proportion, IQ test.

1 Introduction

The capability of human mind to reason by analogy has been considered for a long time as a distinctive feature associated with the idea of intelligence. Indeed IQ tests include tasks where this type of reasoning should be at work. This is probably why, rather early in the history of artificial intelligence, researchers have started to try to incorporate some analogical reasoning capabilities in their inference machineries [11] and to design programs able to solve IQ test-like tasks

[4]. Roughly speaking, following [5], one may distinguish between symbolic approaches and approaches incorporating connectionist features. The first ones, e.g. [22],[6], use appropriate symbolic representations of the items involved and try to match them in order to reveal some hidden analogical relation. For instance, in case-based reasoning [1], a target problem has to be mapped with a source problem (whose solution is known). The connectionist view [8],[7],[9] uses constraint satisfaction networks, explores competing hypotheses, and rather insists on the optimization process at work for determining the best solution. Moreover, this type of approach naturally embeds a graded view of similarity, while symbolic approaches have generally difficulties to handle similarity beyond mere identity. Although symbolic representations use logic-like encodings of the items, they do not usually rely on a logical inference process for establishing analogies and getting plausible conclusions on this basis. However, a recent proposal [15,18] has shown that it makes sense to have a logical view of analogical proportions that are statements of the form "a is to b as c is to d". Due to its very generic setting in Boolean logic, this framework highlights the fact that other proportions, distinct from analogy, are also of interest. This is particularly the case for one of them, named *paralogical proportion*, which is invariant when we switch the 2 items a and b for instance, and which may be also encountered in IQ tests. The logical view makes possible to derive d from a, b and c. Having such a tool at hand, it is interesting to see how this approach can be applied to some representative IQ tests, without limiting ourselves to tests that appear as direct applications of analogical proportions because they ask for completing a series of 3 items. This is the topic of Section 4 and Section 5, after providing the necessary background on analogical and paralogical proportions in Section 2 and deriving a rigorous inference process in Section 3. In Section 6, the approach is compared to the main other existing methods.

2 A Logical View of Analogical Proportions

An analogical proportion is a statement of the form "a is to b as c is to d", and is usually denoted by $a : b :: c : d$. A formal study of analogical proportions has been first proposed by Lepage in [12] in a computational linguistics perspective some years ago, and further developed in [20]. These authors have proposed a definition of analogical proportions, which was restated in a different, simpler but equivalent way in [15]. The underlying idea is that "a differs from b as c differs from d" and that "b differs from a as d differs from c" (see also [18]).

2.1 Binary Case

We use a, b, c, d to denote propositional variables, $\overline{a}$ to denote the negation of a, $\wedge$ for conjunction, $\rightarrow$ and $\equiv$ the associated implication and equivalence, i.e. $a \rightarrow b = \overline{(a \wedge \overline{b})}$ and $a \equiv b = (a \rightarrow b) \wedge (b \rightarrow a)$. The definition then writes:

Definition 1. $(a : b :: c : d)$ *iff* $((a \to b) \equiv (c \to d)) \wedge ((b \to a) \equiv (d \to c))$,

or equivalently $$(a \wedge \overline{b} \equiv c \wedge \overline{d}) \wedge (\overline{a} \wedge b \equiv \overline{c} \wedge d) \quad (1),$$
highlighting the fact that analogical proportion relies on dissimilarities between the 2 first items which have to be identical to the dissimilarities between the 2 last items. These two equivalent definitions satisfy the three postulates considered as being characteristic of analogical proportions in the literature [12,18]:

– $(a : b :: a : b)$ (and $(a : a :: b : b)$) (identity)
– $(a : b :: c : d) \Rightarrow (c : d :: a : b)$ (symmetry)
– $(a : b :: c : d) \Rightarrow (a : c :: b : d)$ (central permutation).

Considered as a quaternary Boolean connective with truth value in $\mathbb{B} = \{0, 1\}$, the six patterns of truth values that make $a : b :: c : d$ true are given in Table 1. For the 10 other 4-tuples of binary values, $(a : b :: c : d)$ is false. As can be seen, the analogical proportion $(a : b :: c : d)$ holds either when a and b (resp. c and d) are identical, or when the changes from a to b, and from c to d take place in the same *direction* simultaneously (see the two last lines of Table 1). In Table 1, changing the 1's into 0's and vice versa leaves the table unchanged: the evaluation of an analogical proportion does not depend on the way it is encoded. Namely, it would be the same, when describing a situation, to state that a property holds or that its negation is false.

Table 1. The 6 cases where the proportions hold as true

	analogy				**paralogy**			
	a	b	c	d	a	b	c	d
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	1	1	0	0	1	0	0	1
4	0	0	1	1	0	1	1	0
5	1	0	1	0	1	0	1	0
6	0	1	0	1	0	1	0	1

2.2 Paralogy

When moving all the negation operators appearing in formula (1) into one equation, we get a new proportion now defined by:

$$((a \wedge b) \equiv (c \wedge d)) \wedge ((\overline{a} \wedge \overline{b}) \equiv (\overline{c} \wedge \overline{d})) \quad (2)$$

This proportion, studied in [18], and whose truth table is shown in Table 1 has been called *paralogical proportion* and denoted $a; b :: c; d$. The proportion enjoys the properties of analogical proportion except the central permutation, but it can be checked that $a : d :: c : b$ iff $a ; b :: c ; d$. Nevertheless, as can be seen from the definition, paralogy focuses on the similarities (i.e. $a \wedge b$ and $\overline{a} \wedge \overline{b}$)

between items, capturing other intuitions than the standard analogy. In fact we have $a\ ;\ b :: c\ ;\ d$ iff $a\ ;\ b :: d\ ;\ c$ which means that paralogy is not sensitive to the ordering inside the pair of elements that are compared. This will prove to be of interest when dealing with IQ tests solving.

2.3 Multiple-Valued Case

The evaluation of analogical and paralogical proportions has been extended to multiple valued scales, i.e., when $a, b, c, d \in [0, 1]$. It has been shown in [17] that an appropriate extension of the definitions is obtained by choosing $\overline{x} = 1 - x$, $x \wedge y = \min(x, y), x \rightarrow y = \min(1, 1 - x + y)$ and $x \equiv y = 1 - |x - y|$. It can be checked that $(a : b :: c : d) = 1$ if and only if $|a - b| = |c - d|$ and $a \leq b \Leftrightarrow c \leq d$. This expresses that the amount of change from a to b, and from c to d is the same and is in the same direction. It also applies to discrete scales.

2.4 Vectors Case

It is an easy task to extend the previous definitions to the case where we deal with vectors $\overrightarrow{a} = (a_1, \ldots, a_i, \ldots, a_n)$, $\overrightarrow{b}$, $\overrightarrow{c}$, $\overrightarrow{d} \in \mathbb{B}^n$ instead of Boolean atoms. We just extend the definition componentwise by putting:

$$\overrightarrow{a} : \overrightarrow{b} :: \overrightarrow{c} : \overrightarrow{d} \text{ iff } \forall i \in [1, n],\ a_i : b_i :: c_i : d_i$$

The same scheme applies to define $\overrightarrow{a}; \overrightarrow{b} :: \overrightarrow{c}; \overrightarrow{d}$. Obviousy, this will be useful when dealing with complex items which are often represented as vectors of binary properties that they satisfy or do not satisfy. This extends as well to non-binary features. Let us examine now how we can use this framework to provide the basis of an automatic analogy-making process.

3 Inference Principle

We start with the Boolean viewpoint and extend it to more general situations.

3.1 Equation Solving

When a sequence of 3 items has to be completed by a 4th one in order to make an analogical (resp. paralogical) proportion, from a Boolean viewpoint, the problem can be stated in that way:

- Given 3 Boolean values a, b, c, does it exist a value x such that the proportion $a : b :: c : x$ (resp. $a; b :: c; x$) holds (we then speak of "solvable equation")?
- and, in such a case, is the value unique?

For each of the two proportions, it can be easily seen that there are cases where the equation has no solution. For instance, the patterns $1 : 0 :: 0 : x$ and $0 : 1 :: 1 : x$ have no analogical solution, but they have a unique paralogical solution which is $x = 0$. When the solution exists for analogy, it is unique and is given by $x = c \equiv (a \equiv b)$ as first suggested in [10] (see [15]). A similar result comes for paralogy (see [18] for instance). When we examine the truth table of analogical proportion, we see that, when an equation is solvable, in 4 situations among 6, the solution is just equal to the second element of the proportion (e.g., 101 is completed with 0). Then it amounts to have the pattern corresponding to (c, d) as being an exact copy of the one associated with (a, b).

In the multiple-valued case, it can be checked that there exists x such that $a : b :: c : x$ holds at degree 1 if and only if $x = c + b - a \in [0, 1]$ (see [17]). For instance, we can solve the analogical equation $0.7 : 0.5 :: 0.2 : x$ with $x = 0.2 - (0.5 - 0.7) = 0$, but obviously $0.7 : 0.5 :: 0.1 : x$ cannot be solved exactly: for, e.g., $x = 0$, the truth value of the proportion is 0.9.

When it exists, the solution is still unique. Let us note that the limit situations of the multiple-valued case with only the 2 Boolean values 0 and 1 appearing in the equation strictly fit with the Boolean definition. From a practical viewpoint, when 3 items a, b and c, described with only one feature, fit with the pattern of a solvable equation, we can predict the value of the 4th item d as being the solution of this equation.

This equation solving process can be extended to Boolean vectors just by repeting it componentwise: when 3 Boolean vectors $\overrightarrow{a}$, $\overrightarrow{b}$ and $\overrightarrow{c}$ are such that they componentwise match solvable equations, we may predict the 4th Boolean vector $\overrightarrow{d}$ by assuming that a proportion holds and then solving each equation. However, in practice, one may need to observe that the proportion holds for some known components of $\overrightarrow{d}$ for assessing the idea that the proportion should hold and may be applied to the remaining unknown components for $\overrightarrow{d}$.

Let $(a_1, \ldots, a_n, \ldots, a_{n+p}), (b_1, \ldots, b_n, \ldots, b_{n+p}), (c_1, \ldots, c_n, \ldots, c_{n+p})$ 3 Boolean vectors providing complete descriptions of three situations according to $n+p$ variables, which are all valued in $[0, 1]$ (or in a discrete subset of it including 0 and 1), and let $d = (d_1, \ldots, d_n)$ be a fourth piece of data, which is incomplete in the sense that $d_{n+1}, \ldots, d_{n+p}$ are unknown. In case this new piece of data forms a proportion in a component-wise manner for each of the first n variables, one inductively assumes that this proportion still holds for the p remaining variables. In the case of analogy, it amounts to adopt the following general pattern:

$$\frac{\forall i \in [1, n], \quad a_i : b_i :: c_i : d_i \text{ holds}}{\forall j \in [n+1, p], \quad a_j : b_j :: c_j : d_j \text{ holds}}$$

Then, if $\forall j \in [n+1, p], a_j, b_j, c_j$ match the pattern of a solvable equation, we allocate to d_j the solution of this equation. The information about the initial components from $i = 1$ to n is just a kind of "insurance" that we can infer the remaining values in a plausible way. A similar pattern applies with paralogical proportion. This is useful in a classification task where we have only to predict the missing information for $\overrightarrow{d}$ (namely its class) from a triple of training items $\overrightarrow{a}$, $\overrightarrow{b}$, $\overrightarrow{c}$ whose class is known and whose initial attributes build a proportion componentwise with $\overrightarrow{d}$ (possibly in an approximate way); see [14,18,19]. This is obviously a form of reasoning that is not logically sound, but which may be useful for trying to guess unknown values.

Concerning IQ tests, we may be faced to situations where i) we have no information at all about the last item to be guessed, i.e. $n = 0$, and all the features of $\overrightarrow{d}$ are unknown, or ii) the features defining the 3 first items $\overrightarrow{a}$, $\overrightarrow{b}$, $\overrightarrow{c}$ may not constitute a solvable analogical proportion, but only a solvable paralogical proportion. Then we just solve the equations componentwise. If an equation is a solvable analogical equation, we allocate the unique solution of this equation to the missing feature. If not, but it is solvable as a paralogical equation, we allocate the unique solution of this equation. Finally, if an equation is not solvable for any proportion, we are in a case of failure and we cannot predict a solution.

3.2 Extended View

If some IQ tests can be directly encoded within a Boolean framework, many of them escape this simple modeling and involve the use of functions in order to complete the sequence. A typical example would be to complete the sequence $1 : 2 :: 7 :?$. In that case, we are naturally led to apply to the third item 7 the function which has been applied to 1 to get 2. This leads us to take as granted an analogical pattern such as

$$x : f(x) :: y : f(y) \quad (3)$$

In fact, considering only the syntactic items x, y and f appearing in formula (3), and representing each of the first 3 terms of this proportion as Boolean vectors in a natural manner, Figure 1 exhibits the fact that we get $f(y)$ as a solution of the analogical equation just by solving the equation column per column. It is then advisable to extend our analogical inference principle with the following rule: given 3 items a, b, c such that $b = f(a)$, infer the missing item d as $f(c)$ if f applies to c. It should be clear that if f does not apply to c as it is the case when we want to complete the proportion "abc":"abd":: "xyz": ?, we just fail to predict the missing item. However, we have to note that, even when we deal with numbers, a simple Boolean encoding can be sufficient as it is for $2 : 2^3 :: 3 :?$ which can be coded as $(2, 1) : (2, 3) :: (3, 1) : ?)$ leading to $(3, 3)$. Besides, although a paralogical counterpart of (3) does exist ($x : f(y) :: f(x) : y$), it looks as a pattern that is cognitively too complicate for being really useful in IQ tests.

	x	y	f
x	1	0	0
f(x)	1	0	1
y	0	1	0
?	?	?	?

Fig. 1. A Boolean representation of $x : f(x) :: y :?$

4 Automatic Solving of Geometric IQ Tests

As clearly stated in [5], the human ability "to see a particular object or situation in one context as being the same as another object or situation in another context" is the main process at work when making analogy, and this ability is one of the main features of the human intelligence. This is probably why a noticeable part of the IQ tests are based on providing incomplete analogical proportions where the 3 first items a, b, c are given and where the 4th item d has to be chosen among diverse plausible options in such a way that the analogical proportion $a : b :: c : d$ holds. It is largely agreed, in order to avoid the bias of a cultural background, that these tests have to be mainly picture-based instead of vocabulary-based: ultimately such tests are supposed to be able to measure the "intelligence" of an illiterate person. To fit with this philosophy, we start with picture-based examples. It appears that what is at work in geometrical proportional analogies may be also transposed to other types of scenes such as topographic maps, see [16] on this issue.

4.1 Basic Characteristic Examples

In that exercise, we have 3 perfectly described items a, b, c involving diverse geometric figures and the 4th one d has to be guessed "in accordance with the logic of the 3 first figures". In that case, we are expecting a triangle containing a circle, above a black disk. A simple Boolean encoding of this example is provided with 5 features denoted R for rectangle, T for triangle, C for circle, St for Star and Bd for black disk. The 4 pictures are denoted a, b, c and d where d is totally unknown. Applying our inference principle, we immediately get the unique solution $d = (0, 1, 1, 0, 1)$ leading to the expected answer. Moreover, the computation for obtaining the solution can be traced back and exploited for generating an *explanation* of the solution, e.g., the star disappears and the circle appears when going from a to b as when going from c to d.

Nevertheless, some graphical IQ-tests are not only built-up via analogical proportion as it is the case for Figure 4. In that case, the column defining d_5 (Black disk Bd) is $1 : 0 :: 0 :?$ which is not a solvable analogy, but is still a solvable paralogy $1; 0 :: 0; ?$ with solution $d_5 = 1$. This approach would clearly extend to multiple valued features allowing for intermediary values, e.g., 'big circle', 'regular circle', 'small circle'.

As suggested by the above examples, and easily checked, the approach enforces sequences of the form (P_1, P_1, P_1, P_1), (P_1, P_1, P_2, P_2), or (P_1, P_2, P_1, P_2), for analogy, or even (P_1, P_2, P_2, P_1) (showing a "mirror" effect) for paralogy, when

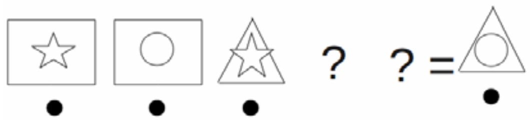


Fig. 2. IQ test 1: Graphical analogy

cell	R	T	C	St	Bd
a	1	0	0	1	1
b	1	0	1	0	1
c	0	1	0	1	1
d	d_1	d_2	d_3	d_4	d_5

Fig. 3. Boolean coding of IQ test 1: analogy only

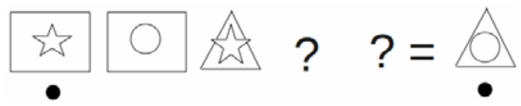


Fig. 4. IQ test 1: Analogy and paralogy mixed

applied to a couple of elementary patterns P_1, P_2 (e.g., 'Star', 'Circle'). It is only by composing elementary patterns that a sequence may exhibit 4 global patterns that are different (or even 3, with a proportion of the form $a : b :: b : c$ in a multiple-valued setting). However, here we do not cover patterns that can be handled by (3) only, where some function would be applied to a geometric picture like, e.g., in A : ∀ :: R : ꓤ.

In the treatment above, the encoding is done via a set of symbolic high level features. But if we adopt a more fine-grained view, where each picture is described as a Boolean matrix of image pixels for instance, the approach still works and is able to build pixel by pixel the image of the same solution, as shown now.

4.2 Implementation of the Procedure at the Pixel Level

Having in mind that information representation is of high importance for this kind of tests, it is challenging to wonder if the approach can still work with a standard bitmap representation of geometric figures, i.e., viewing these figures as ultimately represented by a matrix of pixels, and thus without an higher level representation in terms of mathematical concepts (circle, triangle, center, radius, etc.). The algorithm is quite simple: starting from 3 input images a, b, c represented with matrix of pixels $M_a[i,j]$, $M_b[i,j]$, $M_c[i,j]$ having the same size $n \times m$, we build the 4th image d represented as a matrix $M_d[i,j]$ simply by solving the equations $M_a[i,j] : M_b[i,j] :: M_c[i,j] : x$ and allocating the solution x (if any) as value for pixel $M_d[i,j]$ (or in a paralogical way if there is no analogical solution). This is done via the simple algorithm (linear in the size of the picture) below:

```
Input: Ma, Mb, Mc;
for i=0 to n {
for j=0 to m {
 if solvableAnalogicalEq(Ma[i,j],Mb[i,j],Mc[i,j]) {
    Md[i.j] = solAnalogicalEq(Ma[i,j],Mb[i.j],Mc[i,j]) }
    else {if solvableParalogicalEq(Ma[i,j],Mb[i,j],Mc[i,j]) {
        Md[i.j] = solParalogicalEq(Ma[i,j],Mb[i,j],Mc[i,j]) }
            else {Md[i,j] = error} } } }
Output Md
```

As we can see, when there is neither an analogical nor a paralogical solution, an error message is returned, and one cannot complete the IQ test.

It can be shown that such a low level view always agrees with the high level view in case we deal with sequences of 4 pictures that are made of elementary patterns that are repeated 2 or 4 times (according to the patterns of analogy and paralogy). Indeed, let us consider a high level description where a picture is a collection of geometric figures such as S, T, C, St and Bd in the example, and represented as a vector of Boolean values: 1 meaning the figure is in the picture, 0 meaning the opposite. Let us take, for instance, a figure $F \in \{S, T, C, St, Bd\}$: if F appears in a, it means that all the pixels $M_a[i, j]$ corresponding to F (i.e., the projection of M_a over F that we denote M_a^F) are equal to 1. If F does not belong to a, it means that at least one pixel of M_a^F is equal to 0. Now let us suppose that F belongs to a and c, but not to b: so F should not belong to d following the high level representation, since w.r.t. F, we have the pattern 101 for the 3 first items. In the low level view, it means that all the pixels $M_a^F[i, j]$ and $M_c^F[i.j]$ are equal to 1. Now considering the incomplete Boolean proportion $M_a^F[i, j] : M_b^F[i, j] :: M_c^F[i, j] : x$ that the algorithm solves, it obeys the pattern $1 : M_b^F[i, j] :: 1 : x$, and the solution is just $x = M_b^F[i, j]$. If F does not appear in b, then for a given (i, j), the corresponding pixel $M_b^F[i, j] = 0$: this leads to the solution $M_d^F[i, j] = x = M_b^F[i, j] = 0$, meaning that F does not belong to d. This is exactly the result we get from the high level description. Now, the same reasoning applies if F belongs to b, showing that F belongs to d in that case.

This is why the program automatically builds triangles, circles, and more generally geometric figures without having any knowledge of what is a triangle or a circle or any geometric concept (which might seem amazing at first glance). This kind of background knowledge (generally implemented in analogy-solvers) is not necessary in the approach proposed here. The low level (Boolean) view of analogy-making, when implemented, generates the pictures that can be interpreted at the high (conceptual) level.

5 Other Analogical IQ Tests

In this section, we show that our logical approach applies as well to two other kinds of analogical IQ tests, either involving symbols that should be interpreted beyond their mere external appearance (thus requiring some cultural background), or having a structure that does immediately fits the four terms of an analogical proportion.

5.1 Copycat Example

Copycat [7] is one of the best well-known analogy-making system and will be discussed a little bit deeper in section 6.1. It applies to what is called a micro

$$abc \Rightarrow abd, \quad iijjkk \Rightarrow ?$$

Fig. 5. IQ test 2: Analogy between strings

abc	(a,1)	(b,1)	(c,1)
abd	(a,1)	(b,1)	(d,1)
iijjkk	(i,2)	(j,2)	(k,2)
???	?	?	?

Fig. 6. IQ test 2 encoding

world. In this world, we deal with sequences of letters from the Roman alphabet, and starting from a sequence of 3 strings, a, b, c, the system has to find how to build up a 4th one d in such a way c is transformed into d in the same way a is transformed into b. Below is an example where a = '*abc*', b = '*abd*' and c = '*iijjkk*' and where the transformation leading from a to b has to be guessed, and then applied to c. In the literature, a set of plausible answers is generally proposed for d, e.g. $iijjll, iijjkl, iijjkd, iijjkk$ in the example. Each one is supposed to be the result of a possible reading of the transformation process $abc \Rightarrow abd$ which is then applied to $iijjkk$. In fact, we claim that the option $iijjkk$ is not a plausible analogical option simply because there is no valid analogical proportion $a : b :: c : c$ where the 2 first items are distinct and the 2 last ones are identical. And then there is no need to refer to any plausible transformation process. In that case, the expected answer is d = '*iijjll*' since the last letter in the first string a has been transformed into its successor letter in the second string b. In this case, it is relatively straightforward to follow our Boolean approach. In fact each string can be encoded in the way described in Figure 4. And using our inference principle, since $d = succ(c)$, we immediately get $(i, 2), (j, 2), (l, 2)$ as analogical solution, where we have to apply the $succ$ function to k: this is the encoding of $iijjll$. We have to notice that

- in this kind of example, the last item d is entirely unknown and our method builds it from the 3 other items, once a transformation has been identified.
- the method would work in the same way for, e.g.,'*iijjkkk*' or '*iiijjkkkk*'. In fact, the same process is relevant for similar examples coming from Copycat world, where the pattern (3) is at work.
- as any analogy-making process, our method strongly relies on the coding of the available items and the knowledge. For instance, it would be possible, by changing the initial coding, to get $iijjkl$ also as a solution. In that case, a kind of "simplicity" principle should be used to determine the best option as discussed in section 6.2.

5.2 A More Sophisticated Example Involving 9 Cells

In this type of example, often encountered in IQ tests (e.g. [2]), we have a matrix of 9 cells, 8 of the cells being filled in with a compound picture, and we have to predict the 9th one. Generally, one has to choose it among several options (here 3 options). The IQ test of Figure 7 exhibits simple drawings that can be easily described by the 4 following binary features: Circle (C), Square (S), Point (P), Triangle (T). Thus, the cells of Figure 7 can be described in 2 ways: either we try to complete the test by examining the rows or by examining the columns.

We can apply an analogy-based reasoning by considering that cell 1 and cell 2 always determines cell 3, i.e. (with obvious notation) $cell3 = f(cell1, cell2)$. In that case, the 2 first rows (or columns) are just examples of the output of the function to be guessed to predict the final cell. With that view, in the case of the row encoding, we can build up a global analogical proportion by saying that *the pair* $(cell1, cell2)$ *is to* $cell3$ *in row 1 as* $(cell1, cell2)$ *is to* $cell3$ *in row 2* (and also as the pair (cell1, cell2) is to cell3 in row 3), which can be written as (still with obvious notations and thanks to the transitivity of analogy):

$$(cell11, cell12) : cell13 :: (cell21, cell22) : cell23$$
$$(cell21, cell22) : cell23 :: (cell31, cell32) : cell33$$

The last analogical proportion should be completed to get $cell33$ which is unknown (or partially unknown). This writes for each feature:

Circle: (0, 1):1 :: (0, 0):0 :: (0,1):? Square: (1, 0):1 :: (0, 0):0 :: (1, 0):?
Point: (0,1):1 :: (0, 1):1 :: (0, 0):? Triangle: (0, 0):0 :: (1, 1):0 :: (1, 1):?

Solving the analogical equations here leads to $(C, S, P, T) = (1, 1, ?, 0)$ since the equation for the variable P is not solvable from an analogy viewpoint. So let us go for a column point of view,

(cell 11, cell 21) : cell 31:: (cell 12, cell 22) : cell 32 :: (cell 13, cell 23) : cell 33

Circle: (0, 0):0 :: (1,0):1 :: (1,0):? Square: (1, 0):1 :: (0, 0):0 :: (1, 0):?
Point: (0, 0):0 :: (1, 1):0 :: (1, 1):? Triangle: (0, 1):1 :: (0, 1):1 :: (0, 0):?

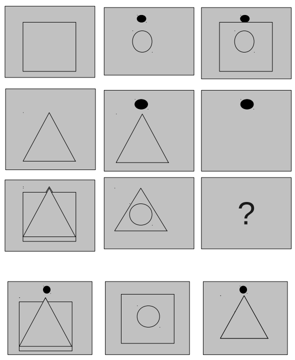


Fig. 7. IQ test 3: 9 cells example

coding by row		cell 1	cell 2	cell 3
row 1	C	0	1	1
	S	1	0	1
	P	0	1	1
	T	0	0	0
row 2	C	0	0	0
	S	0	0	0
	P	0	1	1
	T	1	1	0
row 3	C	0	1	x_1
	S	1	0	x_2
	P	0	0	x_3
	T	1	1	x_4

coding by column	column 1				column 2				column 3			
	C	S	P	T	C	S	P	T	C	S	P	T
cell 1	0	1	0	0	1	0	1	0	1	1	1	0
cell 2	0	0	0	1	0	0	1	1	0	0	1	0
cell 3	0	1	0	1	1	0	0	1	x_1	x_2	x_3	x_4

Fig. 8. Boolean encoding of IQ test 3 by rows and by columns

Solving the analogical equation for P gives $P = 0$ thus completing the scheme $(C, S, P, T) = (1, 1, 0, 0)$. This is exactly the coding of the 2nd option among the 3 proposed ones, which is the expected solution. Note that we need to investigate the rows and the columns in order to get the complete solution. In this case, we do not switch to paralogy since this is not necessary. Note also that our process does not even try here to guess a kind of function $F = (f_C, f_S, f_P, f_T)$ taking the 2 first cells in a row (resp. column) as input and giving the 3rd cell in the same row (resp. same column) as output, each component function being simply a Boolean valued function. In that sense, the process is close to a transductive machinery where one predicts the class of a new element just by investigating other known cases, without any inductive step. But, in this particular case, we can get the truth table of these "hidden" functions (f_C, f_S, f_P, f_T). For f_C for instance, considering the rows first, we get that 01 leads to 1, 00 leads to 0, then with the column, we get 10 leads to 1, which leaves two options (since 11 is not given). But in this particular example, it appears that XOR is the unique function that is compatible with each of the components f_C, f_S, f_P and f_T. This function F can be considered as a "general explanation" of the process and extracted (or induced or learned) from the solution of the analogical process.

It is worth noticing that the solving of the equations that leads to the expected solution corresponds here to what may be termed as being a lazy reasoning approach. Indeed, in order to fill the missing values, it just amounts to reproducing the previous patterns observed for the corresponding feature. For instance, whatever the coding procedure (row or column), considering the missing value x_1 for feature C in the final cell, we have to complete a pattern $(01x_1)$ (row coding) which appears above as (011) so we decide $x_1 = 1$, or the pattern $(10x_1)$ (column coding) which appears before as (101) so we decide again $x_1 = 1$. When a pattern does not appear, as it is the case with $(00x_3)$ for the P attribute in the row coding, one can consider the column equation $(11x_3)$ and conclude here $x_3 = 0$ by copying the column pattern (110) that appears for P. Note that in that case, we are lucky since each pattern appearing for the definition of the missing values previously appears either in a row or in a column (or in both). Obviously when the pattern does not appear, we cannot conclude for the missing value.

6 Discussion and Related Works

We start the discussion by going back to the analogy-making system Copycat, before considering other approaches.

6.1 Copycat

The Copycat project has been developed by Hofstadter and Mitchell (see [7] for an exhaustive description). This is mainly a computer program, operating in a micro world of short strings and designed to be able to discover insightful analogies, in a way supposed to be close to the human being cognitive process. Hofstadter and Mitchell claimed that their project was not only to simulate analogy-making, but more fundamentally, to simulate the very core process at work in human cognition when dealing with real world concepts. Despite the fact that the system operates on string like ‘abc’, ‘abd’, etc., these strings are viewed as idealized instances of relationship like ‘parent of’, ‘neighbor of’. It appears that a human brain is able to *slip* from one relationship to another, in order to highlight deeply hidden links between items coming from apparently distinct contexts. On top of that, due to the highly parallel processing performed in a human brain, it is likely that we have to introduce non-determinism in the system and then randomness. The problems solved with Copycat follow the same pattern similar to Figure 9, where *abc* is the prototype, *iijjkk* the target, *abd* is the result and ? (the thing we are looking for) is the goal. Using primitive operators, the prototype and the target are compared and a rule is chosen among diverse options, a rule transforming the prototype into the result. By applying this rule to the target, the system gets a plausible analogy.

$$abc \Rightarrow abd, \quad iijjkk \Rightarrow ?$$

Fig. 9. IQ test 2: Analogy with Copycat

6.2 A Complexity Approach to Analogy

In [3], the author develops a point of view where a model of analogy should be consistent with other theoretical frameworks developed to model cognitive processes and especially induction. Starting from that, he suggests a criterion suitable for evaluating candidate analogical proportions which would take into account the complexity of the available options, relying on a kind of Minimum Description Length principle. The best candidate analogy should optimize this criterion. Let us come back to the previous Copycat example where a=‘*abc*’,

b=‘*abd*’ and c= ‘*iijjkk*’. If we consider: d_1 = ‘*iijjll*’, d_2 = ‘*iijjkl*’, d_3 = ‘*iijjkd*’, d_4 = ‘*iikjkk*’, d_5 = ‘*iijjkk*’ and d_6 = ‘*iijjdd*’ as candidate solutions for d to complete the proportion, it appears that the most “obvious” d from a cognitive viewpoint (which is $d = d_1$) is also ranked as the least complex from the author’s viewpoint. Other experiments suggest that this principle often provides the solutions chosen by a human being.

A precise definition of the principle is ultimately based on Kolmogorov complexity theory, which provides a universal definition of the complexity $K(s)$ of a given string s. Despite this complexity $K(s)$ is an ideal number, there are ways to approximate it and then to make the minimization criterion an operational principle. Obviously, this approach is not constructive since it does not allow to suggest a value for the missing d in a proportion $a : b :: c : d$: it just allows to suggest the best one among a set of plausible available options.

6.3 E-Unification

In [21], a completely different approach is used to solve analogical proportions. In that approach, each item a, b, c is considered as a structured term in the sense of first order logic for instance (or in the Prolog sense to make it clear). For such kind of terms, there is the well-known unification algorithm and the anti-unification which is the dual notion. By unifying 2 terms a and c, we make them identical by finding a substitution σ such that $\sigma(a) = \sigma(c)$ (where = denotes the syntactic identity). By anti-unifying 2 terms a and c, we look for the most specific generalization of both terms i.e. a term u such that there exists 2 substitutions σ_1 and σ_2 satisfying $\sigma_1(u) = a$ and $\sigma_2(u) = c$. u is a generalization of both a and c (a so-called anti-instance), but we want u to be the most specific in the sense that if another u' satisfies the same property w.r.t. a and c, then u should be an instance of u'. This anti-unification process allows the essential mapping (similarity) between the terms a and c appearing in the proportion. By reference to case-based reasoning (CBR), u represents the similarities between the 2 cases to consider namely a and c. First we start to get the common features between a and c by anti-unification. We then get a and c as particular instances of a global scheme u. What is important in this process are the resulting substitutions σ_1 and σ_2. Since $\sigma_1(u) = a$, we apply σ_1^{-1} to the term b and then we get a new term or scheme v. Applying the substitution σ_2 to this generic term v leads to a new term $\sigma_2(v)$. This new term $\sigma_2(v)$ can be described as the result of “doing the same thing to b at it was done to a to get c. As a consequence, $\sigma_2(v)$ can be considered as a plausible solution d to make the analogical proportion $a : b :: c : d$ valid. Back to the CBR context, d can be viewed as the adaption of the solution b of a to the new case c [13]. In order to extend the scope of this method, the authors deal with unification/anti-unification modulo an equational theory E, getting the so-called E-generalization process. They do not only deal with syntactic equality between terms, but they accept equality of terms modulo some axioms (like $x + y = y + x$ when the target domain is the set of natural numbers).

This technique of E-generalization applies very straightforwardly to the Copycat world of strings. It can also be applied to solve predictive analogies which are described via the SME model [6]. The equational theory E is just the proper coding of the background knowledge related to the target domain. This method clearly falls in the category of symbolic methods where the initial representation of the domain as a set of first-order terms is crucial.

6.4 Adaptation and CBR

When we have 2 situations / cases at hand, x and y, the basic pattern of analogical reasoning amounts to conclude, if x and y share a property P, i.e. $P(x)$ and $P(y)$ hold, and if x also satisfies another property Q, that $Q(y)$ also holds, by an "analogical jump". This can be captured in our approach, just as we did for the analogical proportion $x : f(x) :: y : f(y)$; namely $P(x) : Q(x) :: P(y) : Q(y)$ holds. This can be viewed as a very simple example of adaptation. More generally, when, instead of representing pictures or IQ diagrams, our Boolean vectors represent standard binary database items and encode cases, each one represented as a vector of features, the same inference principle and algorithm described in section 4.2 apply. Let us consider for instance a database of homes to rent, containing houses (1) and flats (0), which are well equipped or not (1/0), which are cheap or expensive (1/0), where you have to pay a tax or not (1/0). Then a house, well equipped, expensive and taxable will be represented as a vector $A = (1, 1, 0, 1)$. Having 2 other cases $B = (1, 0, 1, 1)$, $C = (0, 1, 0, 1)$, we can predict the price and taxation status of a new case D which is a flat not well equipped i.e. $D = (0, 0, x, y)$ where 2 values are unknown. Applying the presented approach, and considering that an analogical proportion $A : B :: C : D$ holds for the 2 first components of each vector, we "infer" that this proportion should hold for the 2 last components giving $x = 1$ and $y = 1$ (i.e. cheap and taxable). As previously explained, the same approach remains suitable when dealing with multi-valued attributes (e.g., very cheap, cheap, expensive, very expensive, for the price). As can be seen in our approach, a form of adaptation is at work through the simultaneous comparison of the considered case D with 3 other cases A, B, C, while classical CBR compares it which each case in the repertory, separately. A full discussion of the potentials of the approach for case adaptation in CBR is a topic for further study.

7 Conclusion

In this paper, we have suggested a Boolean approach to analogy-making, starting with the truth table of the analogical proportion and using the equation-solving principle to build up a missing item d in an incomplete proportion involving a, b, c. This approach is generalized to another proportion named paralogy having a different truth table, but where the same constructive principle applies. It appears that, by using the 2 proportions, this approach allows us to successfully build up the 4th term, incomplete or even missing, of a sequence of 3 terms, when the underlying domain is picture-based as in IQ tests, or a Copycat-like domain. Our approach is constructive in the sense that it does not choose the most plausible option among a finite set of options: it builds up the missing element starting from the previous ones.

Obviously, the main issue is to properly encode the problem at hand: despite the encoding is simple at the end of the process, it is not clear how we can make it without background knowledge in general. Nevertheless, there is a domain where we could rely on the existing encoding: when we deal with pictures (bitmap, GIF or even PNG), we can consider a pixel as the atomic part available to build up low level Boolean proportions. With this in mind, it is possible to build up a new image d, starting from 3 images a, b, c, then leaving apart the encoding process. Our program automatically provides this 4th image which is, in the tested cases, a plausible picture that completes the sequence, i.e. the human answer to the problem. The proposed approach has still to be investigated in various respects: from a cognitive science, experimental validation point of view on the one hand, and from a possible CBR application point of view on the other hand.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning; foundational issues, methodological variations, and system approaches. AICom 7(1), 39–59 (1994)
2. Carter, P., Russell, K.: IQ Firepower. Robinson (1995)
3. Cornuéjols, A.: Analogie, principe d'économie et complexité algorithmique. In: Actes des 11èmes Journées Francaises de l'Apprentissage, Sète, France (1996)
4. Evans, T.G.: A heuristic program to solve geometry-analogy problems. In: Proc. A.F.I.P. Spring Joint Computer Conference, vol. 25, pp. 5–16 (1964)
5. French, R.M.: The computational modeling of analogy-making. Trends in Cognitive Sciences 6(5), 200–205 (2002)
6. Gentner, D.: The Mechanisms of Analogical Learning. In: Similarity and Analogical Reasoning, pp. 197–241. Cambridge University Press, Cambridge (1989)
7. Hofstadter, D., Mitchell, M.: Copycat project: a model of mental fluidity and analogy-making, pp. 205–267. Basic Books, Inc., New York (1995)
8. Holyoak, J., Thagard, P.: Analogical mapping by constraint satisfaction. Cognitive Science 13, 295–355 (1989)
9. Hummel, J.E., Holyoak, K.J.: Distributed representations of structure: a theory of analogical access and mapping. Psychological Review 104(3), 427–466 (1997)
10. Klein, S.: Culture, mysticism & social structure and the calculation of behavior. In: Proc. 5th Europ. Conf. in Artif. Intellig. (ECAI 1982), Orsay, pp. 141–146 (1982)
11. Kling, R.: A paradigm for reasoning by analogy. In: Proc. IJCAI, pp. 568–585 (1971)
12. Lepage, Y.: Analogy and formal languages. In: Proc. FG/MOL 2001, pp. 373–378 (2001), http://www.slt.atr.co.jp/~lepage/pdf/dhdryl.pdf.gz
13. Lieber, J.: Reformulations and adaptation decomposition. In: Proc. ICCBR Workshops, pp. 27–34 (1999)
14. Miclet, L., Bayoudh, S., Delhay, A.: Analogical dissimilarity: definition, algorithms and two experiments in machine learning. JAIR 32, 793–824 (2008)
15. Miclet, L., Prade, H.: Handling analogical proportions in classical logic and fuzzy logics settings. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS, vol. 5590, pp. 638–650. Springer, Heidelberg (2009)

16. O'Donoghue, D.P., Bohan, A.J., Keane, M.T.: Seeing things: Inventive reasoning with geometric analogies and topographic maps. New Generation Comput. 24, 267–288 (2006)
17. Prade, H., Richard, G.: Multiple-valued logic interpretations of analogical, reverse analogical, and paralogical proportions. In: Proc. 40th IEEE Int. Symp. on Multiple-Valued Logic (ISMVL 2010), Barcelona, pp. 258–263 (2010)
18. Prade, H., Richard, G.: Reasoning with logical proportions. In: Proc. Int. Conf. on Principles of Knowledge Repres. and Reas (KR 2010), Toronto, pp. 546–555 (2010)
19. Prade, H., Richard, G., Yao, B.: Classification by means of fuzzy analogy-related proportions: A preliminary report. In: Proc. 2nd IEEE Int. Conf. on Soft Computing and Pattern Recognition (SocPar 2010), Evry, pp. 297–302 (2010)
20. Stroppa, N., Yvon, F.: Analogical learning and formal proportions: Definitions and methodological issues. ENST Paris report (2005)
21. Weller, S., Schmid, U.: Solving proportional analogies by generalization. In: Freksa, C., Kohlhase, M., Schill, K. (eds.) KI 2006. LNCS (LNAI), vol. 4314, pp. 64–75. Springer, Heidelberg (2007)
22. Winston, P.H.: Learning and reasoning by analogy. Com. ACM 23, 689–703 (1980)

Using Case-Based Tests to Detect Gray Cygnets

Edwina L. Rissland and Xiaoxi Xu

Department of Computer Science, University of Massachusetts,
Amherst, MA, USA 01003
rissland@cs.umass.edu

Abstract. Black Swans are surprising, exceptional, provocative cases that instigate major change. Gray Cygnets follow a Black Swan, are highly similar to it, are also exceptional in outcome, and continue to provoke change. We discuss experiments with a family of tests designed to detect Gray Cygnet (GC) cases in a stream of cases following a known Black Swan case. Using the two classic CBR measures of lattice-based and nearest neighbor similarity, the tests use positional information about the Black Swan in the analysis of a new case, such as its being a supreme on-point case (sopc), a Level 1 (L1) case, or in the first ring of nearest neighbors (NN#1), to determine if it is a potential GC. The idea is that a case very similar to a known Black Swan might be a GC. Experiments performed on a corpus of cases from a well-known episode in legal history spanning the era from mid-1800's to mid-1900's showed tests using sopc's were very precise, while those using L1 cases had good recall.

Keywords: Case-Based Reasoning, Black Swans, Gray Cygnets, Surprising cases, Case Similarity, Supreme on-point case (sopc), Classification tests.

1 Introduction

A *Black Swan* is an unanticipated, surprising, exceptional event (case) that provokes profound change or upheaval in the accepted order of things. It sets into motion transformative change in a domain: its concepts, rules, models, policies, protocols, etc. A *Gray Cygnet* (GC) is a case that follows a Black Swan, is *highly* similar to it, is also exceptional in outcome, and continues to provoke havoc or change. (Obviously a GC cannot be quite as surprising or novel as its antecedent swan.) The notion of a Gray Cygnet was introduced by Rissland at ICCBR09 [14] and is so called because GCs tend to follow along in the wake of a leading swan like so many little baby swans, which happen to be gray as juveniles. The notion of Black Swan arises from philosophical discussions in Britain where white swans are the rule and a black one is rare indeed. Had the notion arisen in Australia, the colors might well have been reversed. The book *The Black Swan* by Nassim Taleb discusses the ideas in some detail [17].

Black Swans seem to be everywhere these days given the rash of exceptional, surprising events that have had profound (negative) impact. Examples include 9/11, the Great Recession of 2008-2009, the BP Gulf of Mexico oil spill, the uprisings in the Arab world, and the on-going catastrophe in Japan. As one *New York Times* reporter put it when writing about events in Japan, "all the swans seemed black" [16].

While these events are often called Black Swans, some argue that *we should have seen them coming* since there had already been similar exceptional, antecedent events with enormous impact – past depressions (the Great one of 1929), past bombings of US assets (the *USS Cole*), past oil rig blowouts, past earthquakes, past tsunamis, etc. – that should have alerted us to the possibility of such an event occurring again.

The still unfolding tripartite tragedy in Japan unleashed by the convergence of earthquake, tsunami, and nuclear disasters is often called a surprise, but should it be? There is strong evidence that we should have seen it coming: Japan has a history of earthquakes and tsunamis and of overly optimistic decisions in regulating its nuclear power plants [8]. Official records of tsunamis and earthquakes go back centuries (e.g., to a tsunami in 869). As the *New York Times* reporter wrote [16]:

> "The details of this catastrophe were unforeseeable, leading some to conclude this was a black swan event – something so wildly unexpected, so enormous in its impact, that it seems to defy our understanding and expose the fragility of our knowledge of the world. How could anyone have predicted *this*?
>
> And yet in 2007, Tokyo Electric Power or Tepco, escaped a disaster at its huge Kashiwazaki-Kariwa nuclear power station, on the opposite side of Japan, when that plant was damaged by a 6.8-magnitude earthquake. Tepco basically lucked out last time."

The author of a *New York Times* Op-Ed piece described a hike he took with a group of his middle-school students when he was a teacher in a small fishing village on the northeast coast of Japan [6]. They came across a "mossy stone marker" on a spot of level ground on a steep mountainside high above the village. It marked the high-water mark for the area's biggest tsunami: more than 50 feet above the valley floor. When asked when it happened, the schoolboys couldn't say. "They had learned about it in school…Everything seemed like ancient history to them, but the thought of a wave reaching so high over the homes of my friends sent a chill down my spine." He investigated. He learned that major tsunamis have hit the Sanriku coast every few decades over the last century and a half (e.g., 1896, 1933, 1960). The monument marked the crest of the 1896 tsunami, which killed 20,000, and entirely destroyed the little village. The 1933 tsunami also destroyed the village. Such markers are engraved, "When there is an earthquake, watch for tsunami." In 1958, a massive seawall 30 feet high and stretching across the bay was built to hold back the sea surge from tsunamis. But the height chosen was less than that of known past tsunamis. As in the moral of the story of King Canute, it was a futile effort.

The *New York Times* reporter wrote [16], "The risks are clear enough in hindsight." Sadly, as the Op-Ed author opined, they were also clear enough – save perhaps with regard to the nuclear component – in foresight. Clairvoyance was not required. But use of case memory was.

While Black Swans might be hard to recognize contemporaneously as they are occurring, Gray Cygnets might be amenable to such detection using CBR. The idea is that a case very highly similar to a known Black Swan might also be exceptional and provocative, that is, a GC. This should cause us to consider it very carefully so as not to be surprised as with the Black Swan. Devising case-based tests to spot potential GCs is the focus of this paper. We do not consider the detection of Black Swans.

In Section 2, we discuss the legal domain used in our studies; in Section 3, the tests we devised; in Section 4, details of the case corpus used in our experiments; in Section 5, our experimental results; and in Section 6, our conclusions.

2 The Test Domain

Our test domain is the well-known historical legal episode in which an ‘old’ doctrine that we will call the *privity doctrine* was displaced by a *new* one [5]. The episode began in 1852 and concluded in the 1900’s; the exact date depends on the jurisdiction, for instance, 1916 in New York, 1932 in the United Kingdom, and 1946 in Massachusetts. In the old doctrine only a party who was in a direct contractual relation (“privity”) with the maker of a good that caused harm could recover from the maker for the harm. If there was an intermediate party, like a retail market or drugstore, no recovery (i.e., money damages) was allowed. In the new doctrine, privity is not required. Modern consumer protection laws take this point of view.

2.1 The Old Rule

The leading case for the old privity doctrine was the 1842 British case of *Winterbottom v. Wright*, 152 Eng. Rep. 402, 10 Maule & Selwyn 109. In this case, the plaintiff (P) Winterbottom was a coachman on a mail route from Hartford, in the county of Chester, to Holyhead, a port town in North Wales. He was employed as a coach driver by Atkinson who had a contract with the Postmaster General to supply horses and coachmen to drive mail coaches. The defendant (D) Wright had a contract with the Postmaster General both to supply the coaches and to keep them in a “fit, proper, safe, and secure state and condition.” Wright was thus in a contract with Winterbottom’s employer’s employer. In August 1840 while driving an “unsafe and unfit” coach that broke down, Winterbottom was thrown from his seat and lamed for life. He sued Wright.

In this case, there was thus a chain of three contracts (denoted K1, K2, K3) with two intermediate parties between plaintiff Winterbottom and defendant Wright:

Winterbottom (P) $-_{K1}-$ Atkinson $-_{K2}-$ The Postmaster General $-_{K3}-$ Wright (D)

The British court considered this a case of “first impression” – that is, a case about an issue not considered before – and refused to find for Winterbottom. He lost because he was not in a direct contract with Wright. In his famous opinion, Lord Abinger stated:

> “There is no privity of contract between these parties; and if the plaintiff can sue, every passenger, or even any person passing along the road, who was injured by the upsetting of the coach, might bring a similar action. … [T]he most absurd and outrageous consequences, to which I can see no limit, would ensue.”

Another judge concurring with Lord Abinger wrote, “The only safe rule is to confine the right to recover to those who enter the contract: if we go one step beyond that, there is no reason why we should not go fifty.” Another concurring judge wrote, “Hard cases, it has been frequently observed, are apt to introduce bad law.” Indeed.

Unknowingly prophetic, the defendant’s counsel said, “If a gentleman’s coachman were injured by the breaking down of his carriage…he might bring his action against the smith or the coachmaker” even though his contract is with his master.

In summary, *Winterbottom v. Wright* created the rule of the privity doctrine: *no privity, no recovery.*

2.2 The Black Swan

A decade later in the US, an important case made its way through the New York state court system. In 1852, in *Thomas & Wife v. Winchester*, 6 N.Y. 397 (1852), the NY Court of Appeals, the highest appellate court in New York, held that a remote consumer injured by a mislabeled drug could recover from the drug's manufacturer. This clearly violated the privity rule.

The facts were these. In March 1849, Mr. Thomas went to his local druggist Foord in Cazenovia, a town in upstate New York, to fill a prescription for dandelion extract, which had been prescribed as a treatment for his sick wife. Druggist Foord filled the prescription from a jar labeled as containing extract of dandelion. Foord had purchased it from another druggist named Aspinall, who had purchased it from Winchester, whose employee had made the extract and applied the label. As it turned out what was in the jar was belladonna. While belladonna and dandelion extract may look similar, their effects are markedly different. Dandelion extract is harmless; belladonna is not. The mislabeled medicine caused Mrs. Thomas to become gravely ill. She suffered severely with muscle spasms and "derangement of mind," and "for a short time her life was thought to be in great danger." Fortunately she recovered.

Winchester was thus a remote vendor to the remote buyers, the Thomases. As in *Winterbottom*, there are two intervening parties and three intermediate contracts:

$$\text{Mr/Mrs Thomas (P)} \; -_{K1}- \; \text{Foord} \; -_{K2}- \; \text{Aspinall} \; -_{K3}- \; \text{Winchester (D)}$$

Likewise, the goods provided were defective and the intermediate parties acted as mere middlemen who did not alter them. The defects were not obvious to anyone. The products were used in the way that they were intended to be. Luckily neither defective product caused death but both did cause severe harm. However similar the facts, the legal outcomes were exactly opposite.

Mr. and Mrs. Thomas (P) sued Winchester (D). They won at trial. Winchester appealed. The lower court's verdict was upheld and the Thomases won again. Winchester appealed to the highest appellate court in New York. The Thomases prevailed again.

The outcome of *Thomas & Wife v. Winchester* was totally surprising. It flew directly in the face of the prevailing doctrine. If the New York court had followed the rule, the Thomases would have lost since they were not in privity of contract with Winchester. Instead, the NY court held that an exception should be made because the defendant's act of selling incorrectly labeled poisonous drugs put human life in "imminent danger." It distinguished this case from *Winterbottom* by saying that the coach was not an example of a thing that was imminently dangerous to human life.

Thomas & Wife v. Winchester created an extraordinary exception to the old rule and opened a way to escape the privity requirement. If a plaintiff could successfully argue that the good that caused the harm was imminently dangerous, then the plaintiff could argue by analogy that it too should be able to recover, even if not in a direct contractual relation with the defendant. *Thomas* thus rent a gigantic hole in the fabric of the old doctrine.

This case was provocative in that it set into motion profound change in the accepted order of things. Modern consumer law can be viewed as the end result of the revolution it set in motion. In short, *Thomas & Wife v. Winchester* was a Black Swan.

2.3 Gray Cygnets

In the wake of *Thomas* there followed many cases whose outcomes also violated the privity rule by allowing remote consumers to recover. Obviously, these exceptional cases couldn't be as totally surprising as *Thomas*, but they did continue to advance the doctrinal shift in the law and in this sense they were Gray Cygnets.

Of course, remote buyers did not always win. It depended on the circumstances, and whether the court felt that the product fit its evolving notion of what was an "imminently" or "inherently" dangerous object and could thus fall under the exception. Edward Levi in his well-known account of this episode in legal history discusses several post-*Thomas* NY appellate cases to show how the law goes through several distinct stages [5]. Levi's cases in which the plaintiff (sometimes an estate) prevailed include: *Devlin v. Smith*, 89 N.Y. 470 (1882), in which a defective 90 ft. scaffold caused death; *Torgeson v. Schultz*, 12 N.Y. 156 (1908), in which an exploding bottle of aerated water caused the loss of an eye; and *Statler v. Ray*, 195 N.Y. 478 (1909), in which an exploding coffee urn caused scalding. Levi's cases in which the plaintiff lost include: *Loop v. Litchfield*, 42 N.Y. 351 (1870), in which a defective fly wheel caused death; *Losee v. Clute*, 51 N.Y. 494 (1873), in which an exploding boiler damaged property; and *Cadillac v. Johnson* (1st appeal), 221 F. 801 (1915), in which a defective wheel on a car caused an accident and injuries

Max Radin in a 1933 law journal article also discusses this episode [12]. He focuses more on what we now call concept learning through generalization and specialization in response to positive and negative instances – cases in which the plaintiff won or lost. His account sounds remarkably like Version Spaces [7][15].

Then in 1916 came *MacPherson v. Buick Motor Car*, 217 N.Y. 382 (1916). It is the capstone case of this line of cases. In this case, Donald MacPherson bought a Buick from a dealership in New York. One of its wheels was made of defective wood and its spokes crumbled into fragments and caused an accident in which MacPherson was thrown from the car and injured. Aside from the fact that this case is about a car and *Winterbottom*, a coach, these cases are 'isomorphic' or, as lawyers say, 'on all fours.'

In *MacPherson*, the highest court in NY settled the issue. It held that injured remote buyers could recover from remote manufacturers for harm caused by their products. As Cardozo wrote in his famous opinion for *MacPherson*:

> "The question to be determined is whether the defendant owed a duty of care and vigilance to any one but the immediate purchaser. The foundations of this branch of law, at least in this state, were laid in *Thomas v. Winchester* (6 N.Y. 397).
>
> …
>
> *Thomas v. Winchester* became quickly a landmark of the law. In the application of its principle there may at times have been uncertainty or even error. There has never in this state been doubt or disavowal of the principle itself. The chief cases are well known. … [E]arly cases suggest a narrow construction of the rule. Later cases, however, evince a more liberal spirit."

After discussing cases like *Devlin*, *Torgeson*, and *Statler*, Cardozo wrote:

> "We hold, then, that the principle of *Thomas v. Winchester* is not limited to poisons, explosives, and things of like nature. … The principle that the danger must be imminent does not change, but the things subject to the principle do change. They are whatever the needs of life in a developing civilization require them to be."

In discussing *Winterbottom* and the unsettled state of the law in England, he wrote:

> "There is nothing anomalous in a rule which imposes upon A, who has contracted with B, a duty to C and D and others according as he knows or does not know that the subject-matter of the contract is intended for their use."

Immediately following *MacPherson*, *Cadillac v. Johnson,* 261, F. 878 (1919), a case originally decided for Cadillac on appeal, was reversed on a second appeal and the injured plaintiff Johnson recovered. There were no longer any doubts, at least in New York, about whether a remote buyer could recover.

Meanwhile in the UK, the courts were sticking to their doctrine and were in case after case consistently ruling against remote plaintiffs injured by defective products: "putrid rabbits" in *Beer v. Walker*, [1847-1880] All ER Rep 1139 (1877); "quite putrid" partridges in *Burrows v. Smith*, T.L.R. 1893-1894 (Q.B. 1894); spoiled tinned salmon that caused the death of a child in *Gordon v. McHardy*, (1903) 6 F. 210; a burst brazing lamp in *Blacker v. Lake & Eliot*, (1912) 106 L.T. 533 (K.B.); a burst bottle of ginger beer in *Bates v. Batey*, (1913) 3 K.B. 351; and a decomposed mouse in a bottle of ginger beer in *Mullen v. Barr & Co.*, (1929) S.C. 461.

However in 1932, everything in the UK changed with *Donoghue v. Stevenson,* [1932] A.C. 562, [1932] UKHL 100, which involved a decomposed snail in a bottle of ginger beer – the so-called "Paisley snail"– that caused Ms. Mary Donoghue to suffer "shock and severe gastroenteritis" when she consumed an ice cream soda made with the tainted ginger beer at the Wellmeadow Café in Paisley, Scotland. The landmark opinion by Lord Atkin relies heavily on the holding and reasoning set forth by Cardozo in *MacPherson*. Lord Atkin famously asked, "Who is my neighbor?" For the purposes of recovery, he concluded there is no neighborhood beyond which manufacturers can go to hide from those seriously injured by their defective goods.

Some states were slow to come around to the new doctrine. It took until 1946 for the Supreme Judicial Court of Massachusetts to announce it as the law of the state in the case of *Carter v. Yardley & Co.*, 319 Mass. 92 (1946), which involved a second degree burn caused by perfume made by Yardley and sold to the plaintiff in a retail store. Ultimately, the US and the UK enacted laws (e.g., *Food, Drug and Cosmetic Act of 1938*) to protect consumers. Most consumers today cannot even conceive of the legal regime in existence before the case brought by Mr. and Mrs. Thomas.

3 Tests for Detecting Gray Cygnets

To address the problem of detecting a potential Gray Cygnet (GC) once a Black Swan (BS) has been recognized, we devised a number of tests. The idea is to run a new case against a case base that includes the Black Swan and *use the position of the BS in the analysis of the new case* to signal whether it is potentially a GC because of their high similarity, as always a key issue in CBR [13]. The positions used are:

BS is a sopc (a supreme on-point case)
BS is a mopc (a most on-point case but not a sopc)
BS is a L1 case (a Level 1 case but not a mopc)
BS is a NN#1 case (a case in the first ring of nearest neighbors)

For convenience we just use the short forms *sopc, mopc, L1, NN#1* for these predicates without repeatedly writing "BS is a…"

The first three predicates are based on classic HYPO-style "claim lattices" [2]. These use the standard set-theoretic partial ordering of set inclusion on the power set of a set S to sort relevant cases. Here S is the set of dimensions applicable to the problem case. Any case whose own set of applicable dimensions has a non-empty intersection with S is considered relevant. The relevant cases are sorted using the intersections of their sets of applicable dimensions with S. This creates a lattice with the problem case residing at the root. The closer to the root a case is, the more on point it is considered to be.

Traditionally in HYPO-style systems, any case that is maximal in this sorting is called a *most-on point case* (mopc).[1] There can be several mopc's. Typically, mopc's appear one level below the root in Level 1. However, it is possible for mopc cases to occur in the root node; such cases have all of the problem case's applicable dimensions. (Of course, they probably have some that the problem case doesn't.) Since such root-node mopc's are *supremely* similar to the problem case, we give them a special name: **supreme on-point case** (**sopc**). In this paper, we confine use of the term mopc to maximal cases that are not sopc's, that is, maximal cases found just below the root node in Level 1. A case that occurs in Level 1 but because of the existence of a sopc is not maximal is called simply an **L1** case. In this project, we found that supreme on-point cases occur frequently (in more than half the problem cases), certainly more than in our past projects. That is why we singled them out and also created the L1 predicate.

In this project we also use Hamming Distance (HD) to compute similarity. The Hamming Distance of two equal length vectors is simply the number of components on which their values differ. A case that is at 0 distance (HD=0) is as close as a case can be to the problem case: they are *exactly* the same on all features.

When we analyze a new problem case, we compute the Hamming Distances between it and all the cases in the data set we are using. (Usually this includes only cases that came before the problem case.) Cases fall into *rings* of cases at the same distance from the problem case; there are typically many cases in each ring.

Those cases that occur at the minimum distance are the closest cases – all are *nearest* neighbors – and are said to be in the *first ring* of cases. Typically there are many of them and typically their distance from the problem case is greater than 0. Any such case is called a **NN#1** case, no matter what its distance is, no matter how many others are in the first ring with it.

If there is only one case in the first ring – that is, a *solo* NN#1 – it is truly *the* nearest neighbor. A solo NN#1 case is roughly analogous to a solo sopc. A solo HD=0 NN#1 case is analogous to a solo sopc that has *exactly* the same set of applicable dimensions as the problem case (i.e., the symmetric difference of their sets of dimensions is empty). In these experiments, solo NN#1 cases occurred surprisingly often (in about 40% of the cases run); HD=0 NN#1 cases, less frequently (about 30%); and solo HD=0 NN#1 cases, the least frequently (about 15%). With respect to lattice-based similarity, about half the problem cases had sopc's, and about a quarter, solo sopc's.

[1] In some HYPO-style systems, case disposition is also used. We ignore case disposition here.

Of course, what we are examining is the *position of a known Black Swan* in the nearest neighbor rings or lattice: the more on-point and/or closer it occurs to a new case, the more likely, we posit, the new case will turn out to be a GC. In this study *Thomas & Wife v. Winchester* is our Black Swan and thus we are using its position to spot potential GCs.

Our tests in order of decreasing strictness of pre-conditions are:

Test 1: sopc and NN#1

Test 2: sopc or NN#1

Test 3: (sopc or mopc) and NN#1

Test 4: (sopc or mopc) or NN#1

Test 5: (sopc or mopc or L1) and NN#1

Test 6: (sopc or mopc or L1) or NN#1

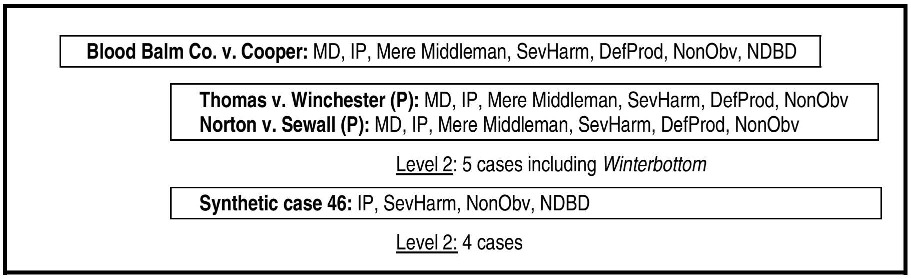
For convenience, we use *smopc* as shorthand for the disjunct "*sopc or mopc"* and *slopc* for "*sopc or mopc or L1*". In essence, the less strict the test, the further down in the lattice it looks for the Black Swan.

We also created three other tests, a majority vote and two single-predicate tests:

MVT: the Majority Vote of Tests 1-4, with a tie considered a Yes
Test 00: sopc
Test 01: NN#1

We started out with Tests 1-4 and the Majority Vote Test. We later added Test 5 and Test 6 to see if looking at L1 cases would be beneficial; it was. We added the single predicate tests, Test 00 and Test 01, as strawmen to see if either alone would be as good at spotting GCs as other tests, which require more computation; Test 00 was.

For example, running *Norton v. Sewall*, 106 Mass. 143 (1870), which involved death caused by tincture of opium dispensed instead of rhubarb, as a problem case, resulted in *Thomas* appearing as a solo sopc and a solo HD=0 NN#1 case. To be both was exceedingly rare. This means *Norton* and *Thomas* are as similar as can be. *Norton* was classified as a GC by *all* of the tests. (The plaintiff won in *Norton* and thus *Norton* was indeed an exceptional case: that is, it violated the old privity rule.)



Blood Balm Co. v. Cooper: MD, IP, Mere Middleman, SevHarm, DefProd, NonObv, NDBD

Thomas v. Winchester (P): MD, IP, Mere Middleman, SevHarm, DefProd, NonObv
Norton v. Sewall (P): MD, IP, Mere Middleman, SevHarm, DefProd, NonObv

Level 2: 5 cases including *Winterbottom*

Synthetic case 46: IP, SevHarm, NonObv, NDBD

Level 2: 4 cases

Fig. 1. Top levels of the lattice for *Blood Balm Co. v. Cooper* with case names, outcomes, and applicable features (see Section 4.2 for details on case features).

As another example, consider *Blood Balm Co. v. Cooper*, 83 Ga. 457 (1889), which involved a defective blood "purifier" that severely sickened the plaintiff. In the nearest neighbor analysis, *Thomas* is a NN#1 case (with HD=2); it shares the first ring with two other cases, one of which is *Norton*. In the lattice, there are no other cases (sopc's) sharing the root node with the problem case. *Thomas* occurs in Level 1 (with *Norton*) and is a mopc (as is *Norton*). See Fig. 1. *Blood Balm* is thus a GC by all tests except Test1 and Test 00. (The plaintiff won in this case as well.)

4 The Case Knowledge Base

We compiled a corpus of cases concerning the privity issue. We began by including cases from Levi's book [5] and cases gleaned from law review articles (e.g., [3, 4, 9, 10]) and Prosser's classic book [11]. We then expanded out from this core by chasing citations forwards and back, for instance, by including cases that the Levi cases cite. This led us to include some very old English cases. We read, summarized, and then represented the cases using features we thought were relevant to this body of law.

4.1 Cases and Doctrinal Eras

Our Case Knowledge Base (CKB) includes 133 real cases from the US and the UK plus 2 additional cases: one alternate encoding of a real case and one hypothetical based on a real case.[2] These were assigned the same year as their original base cases. In addition we created 48 purely made up "synthetic" cases – the **Syn** set – involving food or drink, medicine or drugs, or machines.

In the entire corpus, there are 29 cases from the UK; the rest are from the US. There are 28 cases from New York, 36 from Massachusetts, and the rest from other jurisdictions including 5 Federal appellate level cases. (See Table 1 in Section 4.4.) The two earliest cases *Chandelor v. Lopus*, 79 Eng. Rep. 3 (1603) and *Roswel v. Vaughn,* 79 Eng. Rep. 171 1378-1865 (1607) are English cases. Among the more recent cases is *Carter v. Yardley*, 319 Mass. 92 (1946), which announced the acceptance of the new doctrine in Massachusetts.

We used the landmark cases *Thomas*, *MacPherson*, *Donoghue*, and *Carter* to partition our CKB into five doctrinal eras. (See Table 1 for more data on each era.)

(1) ***Pre-Thomas era*** – before 1852: 12 cases (all from the UK)
(2) ***Post-Thomas era*** – 1852 to 1916 (including *Thomas*, *MacPherson*): 64 cases
(3) ***Post-MacPherson era*** –1917 to 1932 (including *Donoghue*): 36 cases
(4) ***Post-Donoghue era*** – 1933 to 1946 (including *Carter*): 17 cases
(5) ***Post-Carter era*** – after 1946: only 6 cases

With regard to doctrinal evolution, particularly in the US (aside from slow adopters like Massachusetts), the periods 1852-1916 and 1917-1932 are of great interest.

[2] The landmark 1932 case *Donoghue v. Stevenson* was encoded alternatively with severe harm and without it since the plaintiff suffered gastroenteritis, which didn't seem to be on the same level of severity as the loss of an eye or death even though the court called it "severe." The hypothetical is based on the 1887 case *Davis v. Guarnieri* and alters Mrs. Guarnieri dying to her surviving.

4.2 Case Representation

We used 15 features to represent the cases. They encoded the type of product involved, whether the parties were in Privity (P) or there were any Intervening Parties (IP), whether these acted as a Mere Middlemen (MM), whether the product caused Severe Harm (SevHarm), whether it was a Defective Product (DefProd), whether the defect was Non-Obvious (NonObv), whether the product was Not Defective But Dangerous anyway (NDBD), and whether it was imminently, intrinsically or Inherently Dangerous (InhDang).

We used seven product types: Food or Drink (FD), Medicine or Drugs (MD), Chemicals, Personal Care items, Cars & Carriages, Machines, and Other stuff. Some examples of items in the category of Chemicals are naphtha and insecticide; in Personal Care, hair wash and perfume; in Machines, threshing machines and coffee urns. Things that for want of a better category were placed in Other included a brazing lamp and a side saddle.

If the feature was true in a case, it was coded as a 1. If it was definitely not true, it was coded as –1. If we didn't know, it was left blank. (In processing, blanks were turned into 0's.) This 15-component, three-valued representation was used when computing Hamming Distance. For lattice computations, we treated any non-blank (non-zero) feature as being "applicable," whether it was encoded as 1 or –1 (e.g., SevHarm = 1 and SevHarm = –1 both map to "applicable"). The rationale was that either way there was enough information to support discussion about the feature. Our features are like "factual predicates" in HYPO-style systems [2]. In effect we are generating a lattice using applicable factual predicates rather than dimensions.

This case representation is exceedingly simple and certainly not rich enough to support full HYPO-style argumentation. However, we felt it was adequate for these initial experiments. Further, we felt that many features (e.g., SevHarm, MM) could be turned into more traditional dimensions since they do have gradations, key "focal" data, and a sense of direction favoring plaintiff or defendant. For instance, one could imagine dimensions that say: the more severe the harm, the more a case favors the plaintiff; the less an IP acts as a mere middleman, the more a case favors the defendant. Some features (e.g., IP) could function as CATO-like "factors" since their mere presence tends to favor one side or the other [1]. Noticing changes in the dimensions used or their directionality might be used to catch doctrinal change (e.g., IP favors the defendant in the old doctrine, but doesn't matter in the new).

4.3 The Case Base Used in the Experiments

Our CKB can be partitioned into various subsets using jurisdiction and date. All of the problem cases in these experiments were run against the following case base:

MA&NY&UK (above) + Syn.

Above means only cases that came before a problem case were used. (This caused the numbers of cases actually used in problem cases to vary.) Syn is the set of 48 synthetic cases.

4.4 The Answer Key

We wanted to test our ideas by running cases *de novo* as problem cases and seeing how well the various tests performed in spotting Gray Cygnets. However, since there is no official listing of Gray Cygnets, we needed to create an objective definition in order to label them.[3] This would provide a gold standard for our experiments. In crafting this definition, we assumed a "contemporaneous" standpoint of one trying to reason about a new case knowing only about cases preceding it and the recognized Black Swan but without knowledge of how the case or the law ultimately turned out.

In our domain, *Thomas* is the Black Swan. Since *Thomas* concerned medicine and drugs, we were tempted to confine our GC definition to cover cases involving only medicine and drugs (i.e., cases with product type MD). However, from the precedents, particularly old English cases, we knew that certain professions like those supplying food and drink to others (e.g., inn keepers, victualers) were considered to have a professional duty to the public to provide safe and wholesome goods. Professional duty was established early in English law, and is discussed in cases like *Everard v. Hopkins*, 80 Eng. Rep 1164 1378-1865 (1615), in which a "chyrurgeon" (surgeon) dispensed unwholesome medication; *Tessymond's* case, 1 Lewin's Crown Cases, 169 (1826) in which laudanum (opium) was mislabeled and dispensed as paregoric; and *Burnby v. Bollet*, 153 Eng. Rep. 1348 1220-1865 (1847), which involved the re-sale of a rotten pig carcass unfit for human consumption. So instead of creating a definition of GCs that covered only MD cases, we chose a broader disjunctive definition that also included food and drink (FD) cases. Cases involving other types of products (e.g., coffee urns, cars) are not covered.

In these experiments, a GC is defined as a case that satisfies:

(FD OR MD) AND IP AND SevHarm AND DefProd AND NonObv.

The IP condition insures that our GCs do not satisfy the privity requirement. The definition rules out harm that is not severe or permanent (e.g., stomachache, rash). Our definition does not take into account which party won the actual case. It covers a much narrower category than the one for which recovery was eventually allowed; many of Levi's cases are not GCs under our definition. To craft a definition covering all of these would have assumed too much knowledge of what ultimately obtained.

Table 1. Numbers of cases for doctrinal eras

Era	Dates	Total	NY	MA	UK	GC	MD	FD
Pre-*Thomas*	Before 1852	12	0	0	12	—	—	—
Post-*Thomas**	1852-1916	63	12	16	13	6	4	2
Post-*MacPherson*	1917-1932	36	9	12	3	5	0	5
Post-*Donoghue*	1933-1946	17	4	7	1	1	0	1
Post-*Carter*	After 1946	6	3	1	0	0	0	0
Synthetic Cases	—	48	—	—	—	2	1	1
Totals (*excl. *Thomas*)		182	28	36	29	14	5	9

[3] Actually, we are labeling *potential* GCs, because determination of true GCs depends on whether they actually contributed to change. This requires more analysis of legal history than we were able to perform.

In the entire corpus (sans *Thomas*) of 134 non-synthetic cases, only 12 qualify as GCs including the version of *Donoghue* with severe harm, and the one hypothetical. There are also 2 synthetic GCs: Synth 47 (a FD case), and Synth 48 (an MD case). Of the 11 real GC cases, 9 were won by the plaintiff and 2, by the defendant. All 9 violated the old privity rule. That the plaintiff prevailed in so many of the cases we label as GCs gives us some confidence that we have the definition about right.

5 Results

We present our results in terms of True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN) in matrices in the standard way. We use Precision (P), Recall (R), and Accuracy (i.e., percentage of correct answers) defined for classification problems as:

Acc = (TP + TN)/N where N is the total number of cases (N=TP+FP+TN+FN)

Precision = TP/(TP + FP)

Recall = TP/(TP + FN)

We also use two standard "F" measures:[4] $F = 2\,[P\bullet R/(P + R)]$, $F_2 = 5\,[P\bullet R/(4P + R)]$

5.1 Results in the Post-*Thomas* Era: 1852-1916

We ran all 63 cases from this era. In this set, 6 are GCs (4 MD, 2 FD). All but Test 1 and Test 00 spotted all 4 MD GCs. All but Test 6 missed both the FD GCs.

Test 1 sopc	Test=Y	Test=N
GC=Y	TP=3	FN=3
GC=N	FP=0	TN=57
Acc=95.2% **Prec=100%** Recall=50%	**F=66.7%** F_2=55.6%	

Test 2	Test=Y	Test=N
GC=Y	4	2
GC=N	5	52
Acc=88.9% Prec=44.4% Recall=66.7%	F=53.3% F_2=60.6%	

Test 00 sopc	Test=Y	Test=N
GC=Y	3	3
GC=N	1	56
Acc=93.7% Prec=75% Recall=50%	F=60% F_2=53.6%	

Test 3 mopc	Test=Y	Test=N
GC=Y	4	2
GC=N	4	53
Acc=90.5% Prec=50% Recall=66.7%	F=57.1 F_2=62.5%	

Test 4	Test=Y	Test=N
GC=Y	4	2
GC=N	22	36
Acc=61.9% Prec=15.4% Recall=66.7%	F=25% F_2=40%	

Test 01 NN#1	Test=Y	Test=N
GC=Y	4	2
GC=N	4	53
Acc=90.5% Prec=50% Recall=66.7%	F=57.1 F_2=62.5%	

Test 5 L1	Test=Y	Test=N
GC=Y	4	2
GC=N	3	54
Acc=92.1% Prec=57.1% Recall=66.7%	F=61.5% **F_2=64.5%**	

Test 6	Test=Y	Test=N
GC=Y	5	1
GC=N	42	15
Acc=31.8% Prec=10.6% **Recall=83.3%**	F=18.9% F_2=35.2%	

MVT 1-4	Test=Y	Test=N
GC=Y	4	2
GC=N	5	52
Acc=88.9% Prec=44.4% Recall=66.7%	F=53.3% F_2=60.6%	

[4] The general measure that values recall β times more than precision is $F_\beta = (1 + \beta^2)\,[P\bullet R/(\beta^2 P + R)]$.

5.2 Results in the Post-*MacPherson* era: 1917-1932

We ran 35 cases from this era. This included 5 GCs (all FD cases). Only Test 6 found any of them. On all the other tests, TP=0, both Precision and Recall were 0, and neither F-test was applicable. All results, except those for Test 4 and Test 6, were identical.

All but…	Test=Y	Test=N
GC=Y	TP=0	FN=5
GC=N	FP=1	TN=29
Acc=82.9%	F=NA	
Prec=0%	F_2=NA	
Recall=0%		

Test 4	Test=Y	Test=N
GC=Y	0	5
GC=N	4	26
Acc=74.3%	F=NA	
Prec=0%	F_2=NA	
Recall=0%		

Test 6	Test=Y	Test=N
GC=Y	4	1
GC=N	25	5
Acc=25.7%	F=23.5%	
Prec=13.8%	F_2=40.8%	
Recall=80%		

5.3 Results in the Overall Era: 1852-1932

We ran 98 cases (excluding *Thomas*) from the *overall* era (1852-1932) that combines the 1852-1916 and 1917-1932 eras. It includes 11 GCs (4 MD, 7 FD).

Test 1 sopc	Test=Y	Test=N
GC=Y	TP=3	FN=8
GC=N	FP=1	TN=86
Acc=90.8%	F=40%	
Prec=75%	F_2=31.3%	
Recall=27.3%		

Test 2	Test=Y	Test=N
GC=Y	4	7
GC=N	6	81
Acc=86.7%	F=38.1%	
Prec=40%	F_2=37%	
Recall=36.4%		

Test 00 sopc	Test=Y	Test=N
GC=Y	3	8
GC=N	2	85
Acc=89.8%	F=37.5%	
Prec=60%	F_2=30.6%	
Recall=27.3%		

Test 3 mopc	Test=Y	Test=N
GC=Y	4	7
GC=N	5	82
Acc=87.8%	F=40%	
Prec=44.4%	F_2=37.7%	
Recall=36.4%		

Test 4	Test=Y	Test=N
GC=Y	4	7
GC=N	26	61
Acc=66.3%	F=19.5%	
Prec=13.3%	F_2=27%	
Recall=36.4%		

Test 01 NN#1	Test=Y	Test=N
GC=Y	4	7
GC=N	5	82
Acc=87.8%	F=40%	
Prec=44.4%	F_2=37.7%	
Recall=36.4%		

Test 5 L1	Test=Y	Test=N
GC=Y	4	7
GC=N	4	83
Acc=88.8%	**F=42.1%**	
Prec=50%	**F_2=38.5%**	
Recall=36.4%		

Test 6	Test=Y	Test=N
GC=Y	9	2
GC=N	67	20
Acc=29.6%	F=20.6%	
Prec=11.8%	F_2=37.4%	
Recall=81.8%		

MVT 1-4	Test=Y	Test=N
GC=Y	4	7
GC=N	6	81
Acc=86.7%	F=38.1%	
Prec=40%	F_2=37%	
Recall=36.4%		

5.4 Discussion

All but Test 4 and Test 6 were conservative in the sense of producing few positives (TP, FP) and many negatives (TN, FN). Tests 4 and 6 were the most profligate in terms of FPs. All tests performed best, and had highest scores, in the post-*Thomas* era. All did poorly in the post-*MacPherson* era where only Test 6 had any TPs.

Table 2. Best and worst performance scores in post-*Thomas* and overall eras

Best (Worst) for 1852-1916		**(N=63)**		**Best (Worst) for 1852-1932**		**(N=98)**	
Best Acc= 92.5%	(31.8%)	Test 1	(Test 6)	Best Acc= 90.8%	(29.6%)	Test 1	(Test 6)
Best Prec= 100%	(10.6%)	Test 1	(Test 6)	Best Prec=75%	(11.8%)	Test 1	(Test 6)
Best Recall= 83.3%	(50%)	Test 6	(Tests 1, 00)	Best Recall=81.8%	(27.3%)	Test 6	(Tests 1, 00)
Best F= 66.7%	(18.9%)	Test 1	(Test 6)	Best F=42.1%	(19.5%)	Test 5	(Test 4)
Best F_2= 64.5%	(35.2%)	Test 5	(Test 6)	Best F_2=38.5%	(27%)	Test 5	(Test 4)

The pervasive pattern was that our tests missed GCs that are FD cases. This happened because *Thomas* is an MD case and both lattice and nearest neighbor similarity take product differences into account. Thus in FD cases, *Thomas is pushed lower and further away*. In fact, the similarity measures worked exactly as they should; it was our disjunctive GC definition that led to the misses, since it brought in FD cases. After 1903, all GCs are FD cases; this hurt performance in later eras.

Performance also degraded in later eras due to the increased number of highly similar relevant cases and the concomitant accretion of cases showing up as sopc's and in the first ring. These also *pushed Thomas lower and further away*. For instance, there is on average about 1 sopc and 2 NN#1's per problem case in 1852-1916, but 3.7 and 3.3, respectively, in 1917-1932. This reflects the actual evolution in the law: as time goes on, cases violating the rule occur more often, and eventually the exceptions "swallow the rule" as the new doctrine displaces the old.

Our results exhibit the classic trade off between precision and recall. Test 1 was always most accurate and precise and Test 6, always least. Test 6 always had highest recall and Test 1 (and Test 00), always the lowest. Test 1 (a *sopc* test) is our strictest test. Test 6 (a *slopc* test) is our least strict. Test 1 and Test 00 performed similarly and always produced fewest FPs. Tests 4 and 6 performed similarly and always produced the most FPs. Sorting the tests by precision/accuracy and by recall gives the same orderings in post-*Thomas* zand overall eras (they collapse in post-*MacPherson*):

Prec & Acc:	Test 1 > Test 00 > Test 5 > Test 01, Test 3 > Test 2, MVT >> Test 4 > Test 6
Recall:	Test 1, Test 00 < Tests 2, 3, 4, 5, 01, MVT << Test 6

Not surprisingly, the "and" tests (1, 3, 5) produced higher precision and accuracy and lower recall than their "or" counterparts (2, 4, 6). Often, there is not much difference between the scores. However, the pair Test 4 and Test 6 does stand apart from the others. Both are much worse on Prec and Acc, and Test 6 is much better on Recall.

With respect to the F-measure, which equally blends precision and recall, Test 1 was the best in the post-*Thomas* era, but was hurt by low in the overall era. For F_2, which values recall twice as much as precision, Test 1 lost out, again due to its recall.

We conclude that with respect to precision and accuracy, Test 1 is the best overall, but its weakness is recall. Test 00, which uses only the *sopc* predicate, ranks next and does nearly as well. They differ in accuracy by only about 1%; they differ more on precision. This suggests that the simpler Test 00 could serve as a proxy for Test 1.

This still leaves the recall problem. Test 6 (*slopc*) is far better than the others on recall. However, it has very low precision, truly a sloppy test. Test 6 achieves much better recall than Test 5 and Test 01 (NN#1 alone). This suggests that its recall is being driven by L1 cases. Thus considering Level 1 is indeed helpful for recall.

6 Conclusions

Tests that examine the position of a known Black Swan in the similarity analysis of a new problem case offer effective ways to detect potential Gray Cygnets. Tests, like Test 1 and Test 00, that require the Black Swan to appear as a supreme on-point case (*sopc*) exhibit high precision and raise few false alarms. However, such tests are too strict if our goal is not to miss GCs, since they suffer from low recall. Less stringent tests, like Test 6, which require the BS to appear only in Level 1, produce good recall but suffer low precision. Such high recall/low precision tests should be supplemented by closer examination of a possible GC, for instance, with arguments pro and con, before deciding how to proceed (e.g., raise a red flag alert). These experiments demonstrate the enduring value of lattice-based similarity and reinforce the intuition that the action in lattices occurs near the top: the root for precision, Level 1 for recall.

In summary, we believe CBR techniques offer a good way to guard against being surprised by (nasty) Gray Cygnets that follow a Black Swan. We speculate these ideas could be useful in other applications: emergency mitigation, epidemiology, etc. While case bases require capital investment, we feel they are worth it. What better way is there to avoid being surprised by events we should have seen coming.

Acknowledgments. This research was supported in part by grants from the National Science Foundation (IIS-0938504) and the Microsoft Research Corp. The first author also acknowledges additional support of the NSF in preparing this article; any opinions, findings, conclusions, or recommendations expressed in it are her own and do not necessarily reflect the views of the NSF. We also thank the reviewers for their helpful comments.

References

1. Aleven, V.: Using background knowledge in case-based legal reasoning: A computational model and an intelligent tutoring environment. Artificial Intelligence 150(1-2), 183 (2003)
2. Ashley, K.D.: Modeling Legal Argument: Reasoning with Cases and Hypotheticals. MIT Press, Cambridge (1990)
3. Bohlen, F.H.: The Basis of Affirmative Obligations in the Law of Tort. American Law Register (1905)
4. Bohlen, F.H.: Fifty Years of Tort. 50 Harv. L. Rev. 725 (1937)
5. Levi, E.H.: An Introduction to Legal Reasoning. U. Chicago Press, Chicago (1949)
6. Miller, I.J.: Bitter Legacy, Injured Coast. New York Times, Op-Ed. (March 20, 2011)
7. Mitchell, T.M.: Machine Learning. McGraw-Hill, Boston (1997)
8. Onishu, N, Glanz, J.: Nuclear rules in Japan relied on old science. New York Times, Sunday Business Section, p. 1 (March 27, 2011)
9. Prosser, W.L.: The Assault Upon the Citadel (Strict Liability to the Consumer). 69 Yale L.J. 1099 (1960)
10. Prosser, W.L.: The Fall of the Citadel (Strict Liability to the Consumer). 50 Minn. L. Rev. 791 (1965-1966)
11. Prosser, W.L.: Law of Torts, 4th edn. West Publishing, St. Paul, Minn (1971)
12. Radin, M.: Case Law and Stare Decisis: Concerning Präjudizienrecht in Amerika. 33 Columbia Law Review 199 (1933)

13. Rissland, E.L.: AI and Similarity. IEEE Information Systems 21(3), 39–49 (2006)
14. Rissland, E.L.: Black Swans, Gray Cygnets, and Other Rare Birds. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 6–13. Springer, Heidelberg (2009)
15. Rissland, E.L., Collins, R.C.: The Law as a Learning System. In: Eighth Annual Cognitive Science Society Conference, pp. 500–513. Morgan Kaufman, San Francisco (1986)
16. Sommer, J.: A Crisis That Markets Can't Grasp. New York Times, Sunday Business, p. 1 (March 20, 2011)
17. Taleb, N.N.: The Black Swan. Random House, New York (2007)

Recommending Case Bases: Applications in Social Web Search

Zurina Saaya, Barry Smyth, Maurice Coyle, and Peter Briggs

CLARITY: Centre for Sensor Web Technologies,
School of Computer Science and Informatics,
University College Dublin, Ireland
firstname.lastname@ucd.ie
http://www.clarity-centre.org

Abstract. For the main part, when it comes to questions of retrieval, the focus of CBR research has been on the retrieval of cases from a repository of experience knowledge or case base. In this paper we consider a complementary retrieval issue, namely the retrieval of case bases themselves in scenarios where experience may be distributed across multiple case repositories. We motivate this problem with reference to a deployed social web search service called *HeyStaks*, which is based on the availability of multiple repositories of shared search knowledge, known as *staks*, and which is fully integrated into mainstream search engines in order to provide a more collaborative search experience. We describe the case base retrieval problem in the context of HeyStaks, propose a number of case base retrieval strategies, and evaluate them using real-user data from recent deployments.

Keywords: Social search, context recommendation.

1 Introduction

This paper is about social search and the use of case-based reasoning (CBR) techniques to develop social search technologies that work in tandem with mainstream web search engines. Case-based reasoning is well suited to this problem because CBR methods provide us with a framework for reasoning with experiences and, at its heart, social search is about harnessing the search experiences of others in order to improve web search. To this end we will focus on the HeyStaks social search system, which has been described and evaluated in some detail previously [15,16]. Briefly, HeyStaks provides for a range of search engine enhancements to support collaborating searchers, as well as deeper algorithmic components in order to identify relevant search experiences from a community of collaborators. In short, HeyStaks makes result recommendations to searchers at search time, based on the past searches of their social network. It assumes asynchronous, remote collaboration: searchers do not need to be co-located and collaboration can occur overtime as recent searchers benefit from recommendations that originate from earlier search sessions.

The HeyStaks recommendation engine borrows many ideas from case-based reasoning work and in this paper we focus on a particular challenge for HeyStaks and its users. Specifically, the central concept in HeyStaks is the notion of a *search stak*, which acts like a folder for our search experiences. Briefly, a user can create a search stak on a topic of their choosing and they can opt to share this stak with other users. As they search (using HeyStaks in combination with their favourite mainstream search engine) the results that they select (or tag or share) will be associated with their active stak. These results can be subsequently recommended to other stak members in the future when appropriate. In this way, stak members can benefit from the past searches of friends or colleagues who share their staks.

Search staks are effectively case bases of search knowledge. As described in [15] each stak is made up of a set of *search cases* that reflect the history of search on a particular page. HeyStaks reuses these cases at search time as a source of recommendations, by suggesting pages that match their queries and that are contained within staks that they have joined or created. In addition, as users locate pages of interest as they search, HeyStaks adds this information to relevant staks and so search experience grows through usage. And thus the relevance to case-based reasoning is that HeyStaks is a multi-case-base CBR system and the stak selection problem outline above amounts to a case base selection problem.

A key problem for HeyStaks is to ensure that the right stak is chosen for a given search session. One way to address this is to ask users to pick their stak at the start of their search session, but since many users forget to do this, this is not a practical solution in reality. The alternative is to use information about the user's current search session as the basis for automatically selecting and/or recommending an appropriate stak at search time, which if successful provides for a much more reliable solution. In this paper then we focus on this stak selection (or case base selection) problem and in what follows we describe and evaluate a recommendation-based strategy that works well enough in practice to automatically suggest relevant staks to the user at search time, or even automatically switch users into a likely stak without their intervention.

2 Related Work

Ultimately this work is focused on the application of case-based reasoning concepts and techniques to support web search. Of course CBR researchers have already recognised the opportunity for case-based techniques to improve information retrieval and web search. For example, the work of Rissland [13] looks at the application of CBR to legal information retrieval, and [4] describe a case-based approach to question-answering tasks. Similarly, in recent years there has been considerable research looking at how CBR techniques can deal with less structured textual cases. This has led to a range of so-called *textual CBR* techniques [10]. In the context of Web search, one particularly relevant piece of work is the *Broadway* recommender system [7], and specifically the Broadway-QR query refinement technique, which uses case-based techniques to reuse past query refinements in order to recommend new refinements to searchers. The work of [5] applies CBR techniques to Web search in a different way, by combining user profiling and textual case-based reasoning to dynamically filter Web documents according to a user's learned preferences. This paper focuses on how CBR techniques can be applied to conventional Web search, as opposed to related information retrieval tasks. It builds on previous work [1,2,3] which has already demonstrated the benefits of reusing search experiences within community-based search case bases; each case base representing the prior search experiences of a community of like-minded searchers.

An interesting feature of this work is the fundamental role that multiple case bases play in search support. Conventionally, most CBR systems have assumed the availability of a single case base, focusing on issues of case representation, retrieval, adaptation, and learning with respect to this single case base. However, some researchers have considered the potential and challenges for the use of multiple case bases during problem solving. For example the work of [17] introduced the idea of *multi-case-base reasoning* (MCBR) and proposed a novel distributed case-based reasoning architecture to supplement local case base knowledge, by drawing on the case bases of other CBR systems. Building on this concept, [9] explore, in detail, the issues arising out of MCBR, summarizing key component processes, the dimensions along which these processes may differ, and some of the key research issues that must be addressed for successful MCBR systems. The work of [8] goes on to explore varies strategies for implementing MCBR techniques and specifically proposes methods for automatically tuning MCBR systems by selecting effective dispatching criteria and cross-case-base adaptation strategies. The methods require no advance knowledge of the task and domain: they perform tests on an initial set of problems and use the results to select strategies reflecting the characteristics of the local and external case-bases.

In the present paper we are also concerned with a form of multi-case base reasoning. As per the introduction our case base are repositories of search knowledge (search staks), which a particular user has subscribed to, and the specific task that we focus on is the selection of the right case base (stak) for a given query, which of course represents just one of the many processes involved in multi-case-base reasoning. In the case of our work, however, it is a vital process since the lack of an effective case base recommendation technique seriously limits the effectiveness of the HeyStaks system and can lead to a contamination effect across search staks since off-topic content may be added to staks if recommendation failures occur.

3 A Review of HeyStaks

In designing HeyStaks our primary goal has been to provide social Web search enhancements, while at the same time allowing searchers to continue to use their favourite search engine. HeyStaks adds two basic features to any mainstream search engine. First, it allows users to create *search staks*, as a type of folder for their search experiences at search time, and the creator can invite members

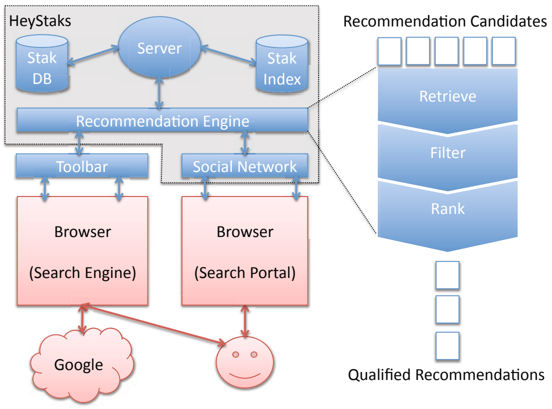


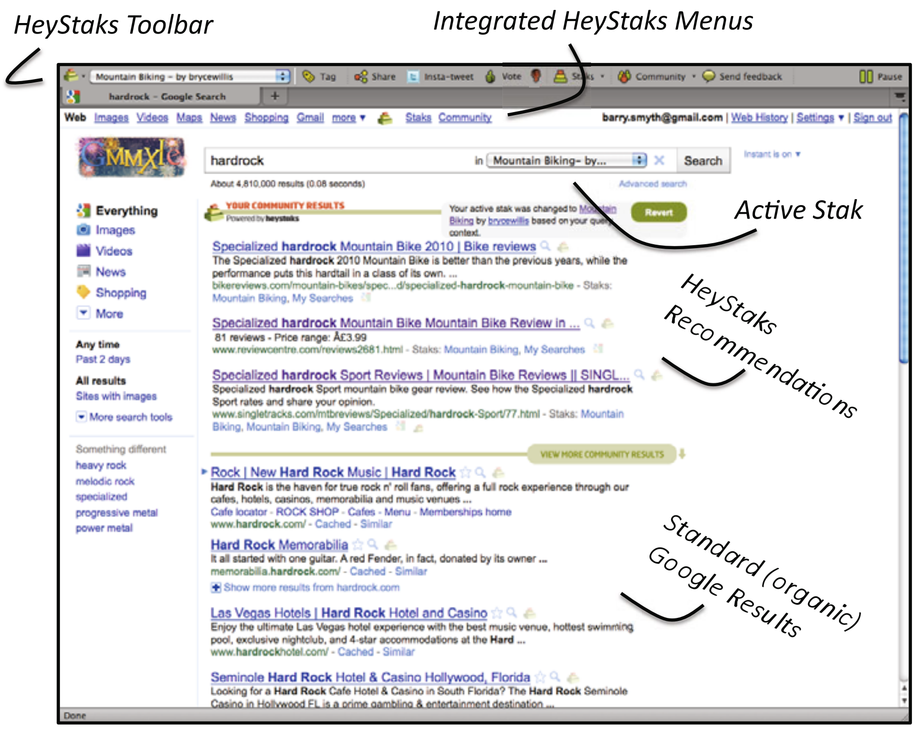
Fig. 1. The HeyStaks system architecture and outline recommendation model

directly. Staks can be configured to be *public* (anyone can join) or *private* (invitation only). Second, HeyStaks uses staks to generate recommendations that are added to the search results that come from the underlying mainstream search engine. These recommendations are results that stak members have previously found to be relevant for similar queries and help the searcher to discover results that friends or colleagues have found to be interesting, results that may otherwise be buried deep within Google's default result-list.

As shown in Figure 1, HeyStaks takes the form of two basic components: a client-side *browser toolbar* and a back-end *server*. The toolbar (see Figure 2) allows users to create and share staks and provides a range of ancillary services, such as the ability to tag or vote for pages. The toolbar also captures search result click-thrus and manages the integration of HeyStaks recommendations with the default result-list. The back-end server manages the individual stak indexes (indexing individual pages against query/tag terms and positive/negative votes), the stak database (stak titles, members, descriptions, status, etc.), the HeyStaks social networking service and, of course, the recommendation engine. In the following sections we review how HeyStaks captures search activities within search staks and how this search knowledge is used to generate and filter result recommendations at search time; more detailed technical details can be found in [16].

3.1 Profiling Stak Pages

Each stak in HeyStaks captures the search activities of its stak members within the stak's context. The basic unit of stak information is a result (URL) and each stak (S) is associated with a set of results, $S = \{r_1, ..., r_k\}$. Each result is effectively a *case* that is anonymously associated with a number of implicit and explicit interest indicators including: the total number of times a result has been selected (Sl), the query terms ($q_1, ..., q_n$) that led to its selection, the terms



hardrock in Mountain Biking– by...

About 4,810,000 results (0.08 seconds)

YOUR COMMUNITY RESULTS
Powered by heystaks

Your active stak was changed to Mountain Biking by brycewillis based on your query context.

Specialized **hardrock** Mountain Bike 2010 | Bike reviews
The Specialized **hardrock** 2010 Mountain Bike is better than the previous years, while the performance puts this hardtail in a class of its own. ...
bikereviews.com/mountain-bikes/spec...d/specialized-**hardrock**-mountain-bike - Staks: Mountain Biking, My Searches

Specialized **hardrock** Mountain Bike Mountain Bike Review in ...
81 reviews - Price range: Â£3.99
www.reviewcentre.com/reviews2681.html - Staks: Mountain Biking, My Searches

Specialized **hardrock** Sport Reviews | Mountain Bike Reviews || SINGL...
Specialized **hardrock** Sport mountain bike gear review. See how the Specialized **hardrock** Sport rates and share your opinion.
www.singletracks.com/mtbreviews/Specialized/**hardrock**-Sport/77.html - Staks: Mountain Biking, Mountain Biking, My Searches

Rock | New **Hard Rock** Music | **Hard Rock**
Hard Rock is the haven for true rock n' roll fans, offering a full rock experience through our cafes, hotels, casinos, memorabilia and music venues ...
Cafe locator - ROCK SHOP - Cafes - Menu - Memberships home
www.**hardrock**.com/ - Cached - Similar

Hard Rock Memorabilia
It all started with one guitar. A red Fender, in fact, donated by its owner ...
memorabilia.**hardrock**.com/ - Cached - Similar
Show more results from hardrock.com

Las Vegas Hotels | **Hard Rock** Hotel and Casino
Enjoy the ultimate Las Vegas hotel experience with the best music venue, hottest swimming pool, exclusive nightclub, and 4-star accommodations at the **Hard** ...
www.**hardrock**hotel.com/ - Cached - Similar

Seminole **Hard Rock** Hotel & Casino Hollywood, Florida
Looking for a **Hard Rock** Cafe Hotel & Casino in South Florida? The **Hard Rock** Seminole Casino in Hollywood FL is a prime gambling & entertainment destination ...

Fig. 2. The searcher is looking for information from a specialist mountain biking brand, Hard Rock, but Google responds with results related to the restaurant/hotel chain. HeyStaks recognises the query as relevant to the the searcher's*Mountain Biking* stak and presents a set of more relevant results drawn from this stak.

contained in the snippet of the selected result $(s_1, ..., s_j)$, the number of times a result has been tagged (Tg), the terms used to tag it $(t_1, ..., t_m)$, the votes it has received (v^+, v^-), and the number of people it has been shared with (Sh) as indicated by Equation 1.

$$r_i^S = \{q_1...q_n, s_1...s_j, t_1...t_m, v^+, v^-, Sl, Tg, Sh\} . \tag{1}$$

Thus, each result page is associated with a set of *term data* (query terms and/or tag terms) and a set of *usage data* (the selection, tag, share, and voting count). The term data is represented as a Lucene (*lucene.apache.org*) index, with each result indexed under its associated query and tag terms, and this provides the basis for retrieving and ranking *recommendation candidates*. The usage data provides an additional source of evidence that can be used to filter results and to generate a final set of recommendations.

3.2 Recommending Results: Relevance and Reputation

At search time, the searcher's query q_T and current, active stak S_T are used to generate a list of recommendations to be returned to the searcher. A set of *recommendation candidates* are retrieved from S_T by querying the corresponding Lucene index with q_T. This effectively produces a list of recommendations based on the overlap between the query terms and the terms used to index each recommendation (query, snippet, and tag terms). These recommendations are then filtered and ranked. Results that do not exceed certain activity thresholds are eliminated as candidates; e.g., results with only a single selection or results with more negative votes than positive votes (see [16]). The remaining recommendation candidates are then ranked according to two key factors: *relevance* and *reputation*. Essentially each result is evaluated using a weighted score of its relevance and reputation score as per Equation 2; where w is used to adjust the relative influence of relevance and reputation and is usually set to 0.5.

$$score(r, q_T) = w \times rep(r) + (1 - w) \times rel(q_T, r) \ . \tag{2}$$

The relevance of a result r with respect to a query q_T is computed based on Lucene's standard *TF*IDF* metric [6] as per Equation 2. The reputation of a result is a function of the reputation of the stak members who have added the result to the stak. And their reputation in turn is based on the degree to which results that they have added to staks have been subsequently recommended to, and selected, by other users; see [11] for additional information.

4 Recognising Context and Recommending Staks

In this paper we are not concerned with recommending individual result pages to HeyStaks users since this has been covered in [16] already. Rather, our focus is on the aforementioned *stak selection* task: which of a user's search staks (search case bases) are appropriate for their current search query or session. The success of HeyStaks depends critically on this. As in the example in Fig. 2, as the user searches for mountain bike related information they need to choose *Mountain Biking* as their current stak. If they do this consistently then HeyStaks will learn to associate the right pages with the right staks, and be in a position to make high quality recommendations for stak members. However, the need to manually select a stak at the start of a new search session is an extra burden on the searcher. To make this as easy as possible, HeyStaks integrates its stak-lists as part of the mainstream search engine interface (see Fig. 2) but still many users forget to do this, especially during the early stages, and this means that a majority of search sessions are associated with the searcher's default stak (*My Searches*), or an incorrect stak as part of an earlier session.

The central contribution of this paper is to provide a practical solution to this problem, one that avoids requiring the user to manually select staks at search time. To do this we draw on ideas from recommender systems, case based reasoning, and traditional information retrieval. Each stak is effectively a case base

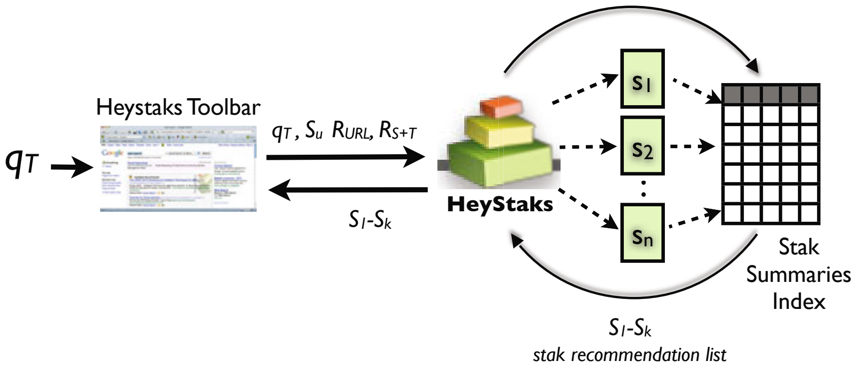


Fig. 3. Stak Recommendation

of search cases, each case representing a page that has been selected, tagged, and/or shared by stak members. For the purpose of stak recommendation we treat the combinations of the cases in each stak/case base as a type of *summary* document to reflect the stak's topic. Effectively the terms and URLs collectively contained in the stak cases become the terms of the summary document and in this way a collection of staks (case bases) can be represented as a collection of documents. Using Lucene, these documents (each one representing a single case base) can then be transformed into a *stak summary index* (or *SSI*); see Fig. 3. Then, at search time, we can use the searcher's query as a probe into this stak summary index to identify those staks/case bases that are most relevant to the query; in this work we focus only on staks that the user is currently a member of but a similar technique could be used to recommend other third-party staks in certain circumstances. These recommended staks can then be suggested directly to the user as a reminder to set their appropriate stak context; or, alternatively, we can configure HeyStaks to automatically pre-select the top ranking recommendation as the current stak context, while providing the searcher with an option to undo this if they deem the stak to be incorrect.

In the above we assume that the user's own search query (q_T) is used as the *stak query*, but in fact there are a number of additional sources of information that could be usefully harnessed for this. For example, at search time, the initial set of search engine results represents a valuable source of additional context information. Our approach to getting additional context information is similar to the technique proposed by [18,12,14], as we exploit local sources of context by using the result of a search as the basis for context assessment, extracting context terms that can then be used to augment the user's original query. However rather than use the context information in query augmentation, we are using the context to find similar staks.

Specifically we use the terms in the search engine result titles and snippets (R_{S+T}), and URLs (R_{URL}) in addition to the user's short search query. Accordingly, we refer to three basic *types* of stak recommendation strategy – *query, snippet, URL* – depending on which sources of information form the user's stak query (S_Q). We might also consider a recommendation strategy based on the *popularity* of the stak for the user so that staks that they use more frequently are more likely to be recommended.

At stak recommendation time we use Lucene's standard TF*IDF weighting model as the basis for scoring recommended staks as shown in Equations 3 and 4. Effectively, terms in the stak summary index (SSI) representing each case base are scored based on the TF*IDF model, preferring terms that are frequent within a given case base but infrequent across the user's staks/case bases (S_U) as a whole.

$$RecList(S_Q, S_U, SSI) = \begin{array}{l} SortDesc(Score(S_Q, S, SSI)) \\ \forall S \epsilon S_U \end{array} \tag{3}$$

$$Score(S_U, S, SSI) = \sum_{t \epsilon S_U} tf(t, S) \times idf(t, SSI) \tag{4}$$

In this way we can generate different recommendation lists (RL_{URL}, RL_{query}, RL_{S+T}) by using different sources of data as the stak query (S_Q); for example, we can use the terms in result titles and snippets as the stak query, which will lead to staks being recommended because they contain lots of distinctive title and snippet terms. Of course we can also look to combine these different sources of terms, for example, by ranking recommended staks according to their position across the recommendation lists produced by different sources of query terms. For instance, we can define the *rank score* of a given stak, across a set of recommendation lists, to be the sum of the positions of the stak in the different recommendation lists with a simple penalty assigned for lists that do not contain the stak as per Equations 5 and 6. The final recommendation list is then sorted in ascending order of the rank scores of recommended staks.

$$RankScore(s, RL_1 - RL_n) = \sum_{RL_i \epsilon RL_1 - RL_n} PositionScore(s, RL_i) \tag{5}$$

$$PositionScore(s, RL) = \begin{cases} Position(s, RL) & \text{if } s \epsilon RL; \\ Length(RL) + 1 & \text{otherwise.} \end{cases} \tag{6}$$

In summary then, HeyStaks is based on the idea of search staks which are effectively case bases of search experiences. Users can be members of many staks and at search time we need to know which stak is most likely to match their current search needs, without having to ask them directly. This is a case base retrieval problem, which we address by treating the case bases themselves as cases in their own right. Each of these 'case base' cases is made up of the combination of its individual search cases. The advantage of this approach is that we can now apply a wide range of conventional retrieval techniques to help select the right case base, and therefore search stak, at search time.

This provides for a general purpose approach to stak recommendation, which accommodates different sources of query data, and provides for a flexible way to combine multiple recommendation lists to generate an ensemble recommendation list. The intuition of course is that by combining different sources of query data we will generate better recommendations, which we shall look at in the following evaluation.

5 Evaluation

In this section we evaluate the different forms of our stak recommendation approach, based on live-user search data, and focusing in particular on the overall recommendation accuracy of the different techniques, and combinations of techniques, across different stak types.

5.1 Setup

The data for this evaluation stems from HeyStaks usage logs generated during the period October 2008 - January 2011. For the purpose of this evaluation we limit our interest to only those activities that are associated with non-default search staks submitted by 29 active users who submitted a minimum of 100 queries through HeyStaks; this means that we focused on search sessions where the user actively selected a specific stak, which we can use as the ground-truth against which to judge our techniques. The test dataset covers 4343 individual searches. The activity levels and stak memberships of these users is presented in Figure 4; we can see that the average activity level per user is 150 activities (result selections, tags, shares etc) and the average user is a member of 19.6 staks. Overall these users were members of 229 unique staks and the size of these staks (by numbers of unique URLs) is presented in Figure 5; we further divide these staks base on their relative size as shown. For the purpose of this study we evaluate different recommendation strategies based on our five basic techniques, namely,

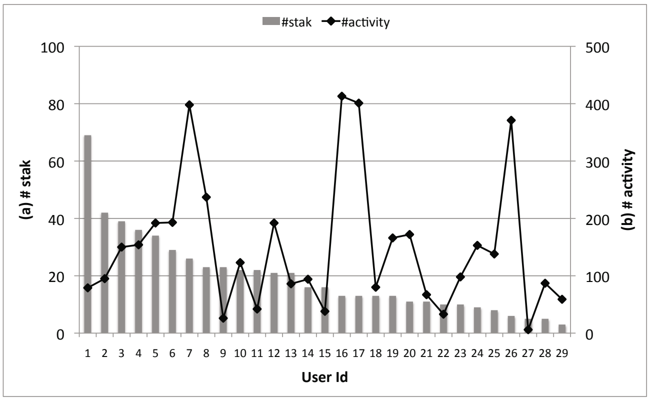


Fig. 4. Summary user data: a) a histogram of user stak membership; b) user activity levels

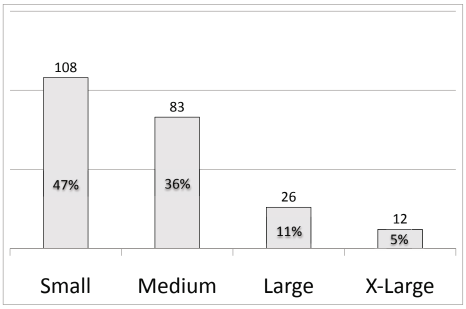


Fig. 5. A histogram of stak sizes. The staks are partitioned into groups based on their size: *small* - 1-10 pages; *medium* - 11-100 pages; *large* - 101-500 pages; *xlarge* - 500+ pages.

Query, Snippet, URL, Popularity and including all combinations of these techniques. In addition we also evaluate a baseline *random* recommendation strategy, which suggests staks at random from the user's stak-list, and also the *popularity* strategy, which recommends the user's most popular stak. This leads to a total of 16 different recommendation alternatives. To evaluate these alternatives, we generate a recommendation list for each of 4343 search instances and compute the percentage of times that the known active stak is recommended among the top k stak recommendations ($k = 1 - 5$).

5.2 Overall Recommendation Precision

To begin with we will look at the overall *success rate* across the different recommendation alternatives; in other words, how often do we recommend the correct stak (case base) to the searcher given their query? Remember we know the ground-truth from our test data since in each case the user did manually select a stak for their search. Thus by comparing the recommended staks to the ground-truth information we can calculate the success rate — how often one of the recommended staks matches the ground-truth — for the various different stak recommendation strategies and for different sizes of recommendation-lists. This data is presented in the graphs of success rate against recommendation-list size (k). For clarity we split the result data across two graphs show one set of techniques in Fig. 6 and the remaining techniques in Fig. 7.

The results indicate a wide variety of success rates across the various techniques. In Fig. 6 we can see that techniques such as *URL*, *Query*, and the combination of *URLQuery* perform poorly, recommending the correct stak about 10-20% of the time regardless of the stak-list size. In other words URL and query information does not provide a sufficiently strong signal for accurate stak recommendation on their own. In contrast, using the snippet text of results provides a much strong signal as evidenced by the superior success rates enjoyed by the *Snippet* technique in Fig. 6 , which achieves a 50% success rate when only a single stak is recommended ($k = 1$), growing to about 70% for larger recommendation-list sizes.

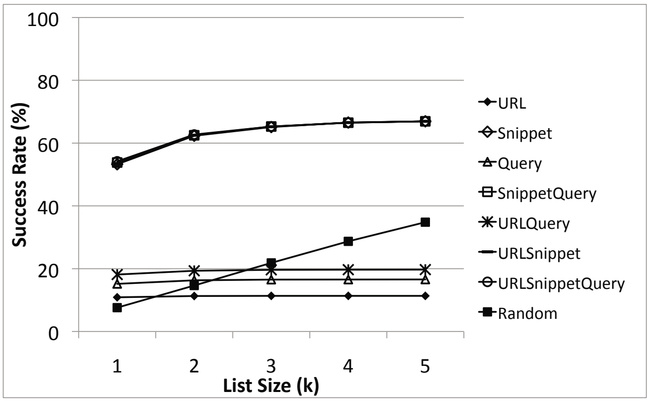


Fig. 6. Recommendation success rate for *Query, Snippet, URL* and ensemble technique

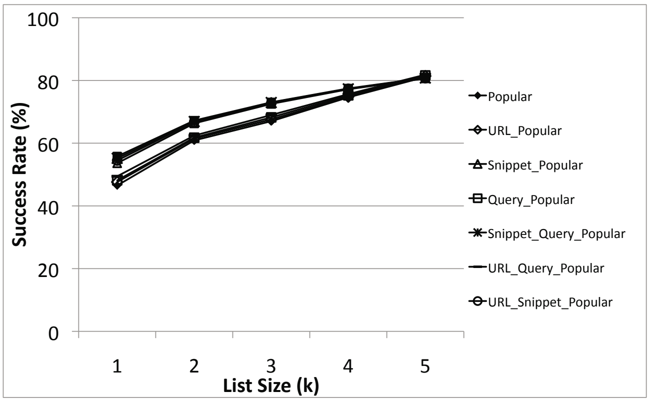


Fig. 7. Recommendation success rate for *Popularity* and ensemble technique

Fig. 6 and Fig. 7 presents the combination strategies, those that combine multiple signals during stak recommendation and we can see clear benefits across the board, with all of the combination techniques shown delivering success rates up to 80%. Importantly, we find that the combination of all signals (*Query, URL, Snippet, Popularity*) tends to deliver the best performance across different values of k. This is more clearly seen in Fig. 8 where we present the average success rate (averaged across all values of k) for the different combinations of techniques. The best performing combination combines *Query, URL, Snippet, Popularity* to achieve an average success rate of almost 71%. Interestingly, it is worth highlighting that using *Popularity* on its own delivers impressive success rates (66% on average in Fig. 8). This is in part as result of the fact that many of our test users tend to use one or two dominant staks and so a popularity-based mechanism can often make good predictions. Over time, as users spread their searches more evenly across more staks we can expect this technique to decline.

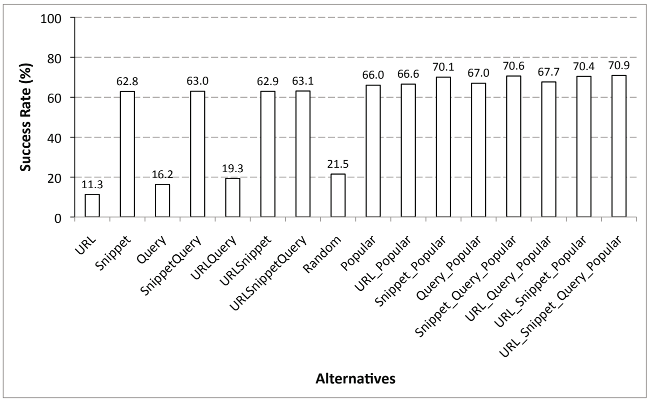


Fig. 8. Mean average success rate

Regardless, by adding additional signals to popularity we are able to further improve the stak recommendation success rate as shown.

In the above it is interesting to pay special attention to the $k = 1$ results because the ideal strategy for HeyStaks would be to automatically switch the user into a correct stak, rather than present a set of stak options for the searcher to choose from. This would require a reasonably high success rate at $k = 1$ to avoid user frustration in the case of incorrect stak switches. Unfortunately, it appears from the results in Fig. 6 and Fig. 7 that the success rates at $k = 1$ do not quite support such an automatic switching approach. For example, the best performing strategy at $k = 1$, which combines *URL*, *Snippet*, *Query* and *Popularity* techniques, achieves a success rate of 56%, which does not seem high enough to support an automatic stak switching.

5.3 Success vs. Stak Size

Of course the above results refer to average success rates across all staks. But not all staks are created equally. For example, as per Figure 5, the majority of staks in this study (47%) contain relatively few URLs (1-10 URLs) which surely provides a much weaker signal for their retrieval. It seems likely that this should impact on stak recommendation effectiveness when compared to larger staks. As HeyStaks matures we can expect users to develop more mature staks and so it is appropriate to evaluate the relationship between recommendation success and stak size. To test this, we present the recommendation success rate for each of the recommendation alternatives, by stak size (comparing *small, medium, large* and *extra-large* staks) for recommendation lists of size 1 (Fig. 9) and 3 (Fig. 10). In these graphs, each basic recommendation technique is represented by four separate bars showing the success rate of this technique for the four different stak types (*small, medium, large,* and *xlarge*). For convenience we also present a line-graph of the average success rate (drawn from Fig. 8) as a reference across the techniques.

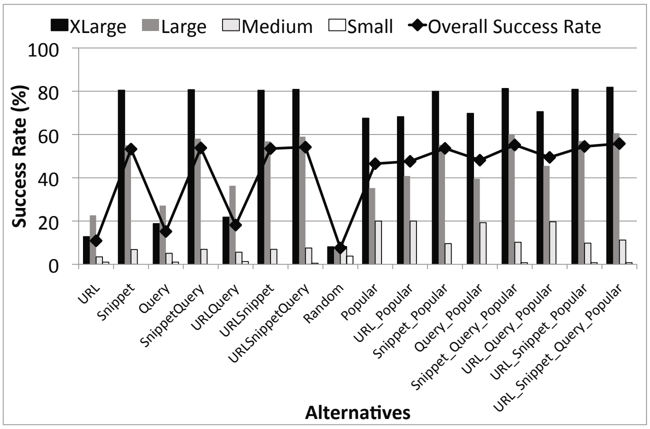


Fig. 9. Success rate by stak size where $k = 1$

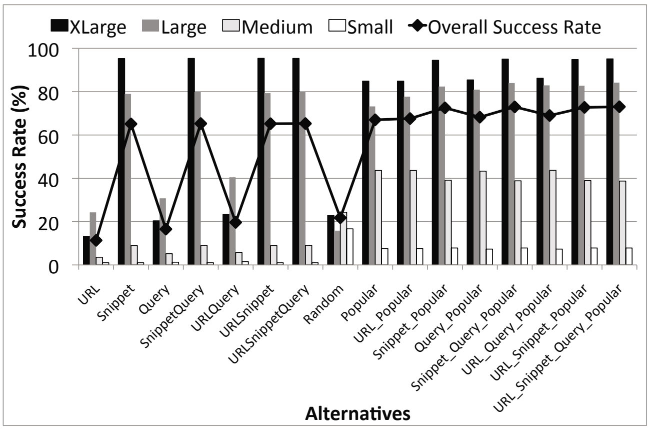


Fig. 10. Success rate by stak size where $k = 3$

It is clear that there are significant differences in recommendation accuracy across the various stak sizes. As expected the larger, more mature staks benefit from higher success rates across the various recommendation techniques and combinations. Once again the combination of all techniques does marginally better overall than any other combination. For example, looking at the combination *URL*, *snippet*, *query* and *popularity* we see a success rate of about 82% at $k = 1$ for the extra-large staks and 61% for the large staks, compared to only 11% and 1% for the medium and small staks respectively. This is encouraging because, from a engineering standpoint, it suggests that it may be practical to implement a reliable automatic stak switching policy, at least for large staks which contain more than 100 URLs. When we look at the results for $k = 3$ (see Fig. 10) we see similar effects, only this time many combination techniques are achieving success rates in excess of 95%, for a number of recommendation combinations across the extra-large staks.

5.4 Conclusions

HeyStaks is a deployed social web search service that used ideas from case-based reasoning to help users to search more effectively online. Users can create and join so-called search staks, which are collaborative case bases of search knowledge, in order to receive community recommendations at search-time. The main contribution of this work has been to highlight a practical problem facing HeyStaks — the need to automatically predict the right stak for users at search time — and to propose and evaluate potential solutions in the form of stak recommendation strategies To this end we have described a general framework for stak recommendation, based on the indexing of staks. The approach accommodates a variety of different recommendation alternatives, using different types of query data at search time, such as search query terms, the titles, URLs, and snippet terms of search results, for example. We have described the results of a comprehensive evaluation of a variety of recommendation strategies, based on live user search data. The success rates achieved for the larger staks in particular speak to the potential for a reliable automatic stak switching mechanism, and at the very least it is possible to generate a short-list of stak recommendations that are accurate up to nearly 95% of the time.

One important limitation of this work is that recommendations are based on a single query (or the results of a single query), when we know that most searchers engage in sessions that extend beyond a single query; we will often submit 2 or 3 queries variations before we find what we are looking for. Obviously, by looking at an extended search session it may be possible to leverage more data (more queries, more URLs, more snippets) in order to better guide stak recommendation. So, for example, while it might not be possible to recommend the correct stak on the first query, we may find that the addition of a second query (and its associated URLs and snippets) greatly improves recommendation quality. Another opportunity could see the use of third-party services to enhance our understanding of the user's search context. For example, by leveraging WordNet or Wikipedia it may be possible to elaborate a users query and to better identify context and this context information could also be used for stak recommendation. These ideas will form the basis for the next steps in this research.

Acknowledgement. This work is supported by Science Foundation Ireland under grant 07/CE/I1147, HeyStaks Technologies Ltd, Ministry of Higher Education Malaysia and Universiti Teknikal Malaysia Melaka.

References

1. Balfe, E., Smyth, B.: Case-based collaborative web search. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 489–503. Springer, Heidelberg (2004)

2. Boydell, O., Smyth, B.: Enhancing case-based, collaborative web search. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 329–343. Springer, Heidelberg (2007)
3. Briggs, P., Smyth, B.: Provenance, trust, and sharing in peer-to-peer case-based web search. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 89–103. Springer, Heidelberg (2008)
4. Burke, R., Hammond, K., Kulyukin, V., Tomuro, S.: Question Answering from Frequently Asked Question Files. AI Magazine 18(2), 57–66 (1997)
5. Godoy, D., Amandi, A.: PersonalSearcher: An Intelligent Agent for Searching Web Pages. In: Monard, M.C., Sichman, J.S. (eds.) SBIA 2000 and IBERAMIA 2000. LNCS (LNAI), vol. 1952, pp. 43–52. Springer, Heidelberg (2000)
6. Hatcher, E., Gospodnetic, O.: Lucene in action. Manning Publications (2004)
7. Kanawati, R., Jaczynski, M., Trousse, B., Andreloi, J.-M.: Applying the Broadway Recommendation Computation Approach for Implementing a Query Refinement Service in the CBKB Meta-search Engine. In: Conférence Française sur le Raisonnement á Partir de Cas, RáPC'99 (1999)
8. Leake, D.B., Sooriamurthi, R.: Automatically selecting strategies for multi-case-base reasoning. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 204–233. Springer, Heidelberg (2002)
9. Leake, D.B., Sooriamurthi, R.: Managing multiple case bases: Dimensions and issues. In: FLAIRS Conference, pp. 106–110 (2002)
10. Lenz, M., Ashley, K.: AAAI Workshop on Textual Case-Based Reasoning, AAAI Technical Report WS-98-12 (1999)
11. McNally, K., O'Mahony, M.P., Smyth, B., Coyle, M., Briggs, P.: Towards a reputation-based model of social web search. In: IUI 2010: Proceeding of the 14th International Conference on Intelligent User Interfaces, pp. 179–188. ACM, New York (2010)
12. Mitra, M., Singhal, A., Buckley, C.: Improving automatic query expansion. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1998, pp. 206–214. ACM, New York (1998)
13. Rissland, E.L., Daniels, J.J.: A hybrid CBR-IR Approach to Legal Information Retrieval. In: Proceedings of the 5th International Conference on Artificial Intelligence and Law, pp. 52–61. ACM Press, New York (1995)
14. Shen, D., Pan, R., Sun, J.-T., Pan, J.J., Wu, K., Yin, J., Yang, Q.: Query enrichment for web-query classification. ACM Trans. Inf. Syst. 24, 320–352 (2006)
15. Smyth, B., Briggs, P., Coyle, M., OḾahony, M.: A case-based perspective on social web search. In: McGinty, L., Wilson, D. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 494–508. Springer, Heidelberg (2009)
16. Smyth, B., Briggs, P., Coyle, M., O'Mahony, M.P.: Google shared. a case-study in social search. In: User Modeling, Adaptation and Personalization, pp. 283–294 (2009)
17. Sooriamurthi, R.: Multi-case-base reasoning. PhD thesis, Indiana University, Indianapolis, IN, USA, AAI3278223 (2007)
18. Xu, J., Croft, W.B.: Query expansion using local and global document analysis. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1996, pp. 4–11. ACM, New York (1996)

Measuring Similarity in Description Logics Using Refinement Operators[⋆]

Antonio A. Sánchez-Ruiz[1], Santiago Ontañón[2], Pedro Antonio González-Calero[1], and Enric Plaza[2]

[1] Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid, Spain
antsanch@fdi.ucm.es, pedro@sip.ucm.es
[2] IIIA-CSIC, Artificial Intelligence Research Institute
Campus Univ. Aut. Barcelona, 08193 Bellaterra, Catalonia, Spain
{santi,enric}@iiia.csic.es

Abstract. Similarity assessment is a key operation in many artificial intelligence fields, such as case-based reasoning, instance-based learning, ontology matching, clustering, etc. This paper presents a novel measure for assessing similarity between individuals represented using Description Logic (DL). We will show how the ideas of *refinement operators* and *refinement graph*, originally introduced for inductive logic programming, can be used for assessing similarity in DL and also for abstracting away from the specific DL being used. Specifically, similarity of two individuals is assessed by first computing their *most specific concepts*, then the *least common subsumer* of these two concepts, and finally measuring their distances in the refinement graph.

1 Introduction

Description Logic (DL) [4] is becoming a de facto standard for knowledge representation in many application areas. DL constitutes a family of different logics, which have been carefully characterized in terms of expressivity and computational complexity of their deduction algorithms. Gaining momentum through the Semantic Web initiative, DL popularity is also related to a number of tools for knowledge acquisition and representation, as well as inference engines, that have been made publicly available. For these reasons, DL has also become the technology of choice for representing knowledge in knowledge-intensive case-based reasoning systems [23,9].

In the last few years, there has been a growing interest in defining similarity measures for expressive representation formalisms, such as DL. For example, Amato et al. [10] propose to measure concept similarity as a function of the intersection of their interpretations, which is, in fact, an approximation to the semantic similarity of concepts. The approximation is better or worse depending on how good is the sample of individuals used for assessing similarity. Thus, a good sample of individuals is required.

Other approaches have been proposed in order to assess similarity between individuals or concepts without requiring the use of a good sample of individuals. González

[⋆] Partially supported by the Spanish Ministry of Science and Education project Next-CBR (TIN2009-13692-C03-01 and TIN2009-13692-C03-03).

et al. [12] present a similarity measure for description logic designed for case-based reasoning systems. This similarity measure is based on the idea of hierarchical aggregation, in which the similarity between two instances is computed as an aggregation of the similarity of the values in their roles. Like most hierarchical aggregation measures, however, this measure has problems with roles which create cycles and they are ignored during similarity assessment. Related to the work of González et al. other similarity measures have been proposed using the hierarchical aggregation principle for other representation formalisms such as Horn Clauses [14], Feature Terms [1], or object-oriented representations [6]. Description logic has also been used to model CBR case retrieval mechanisms not based on similarity, but on the subsumption order [22].

The work presented in this paper is most related to that of Ontañón and Plaza [20], where they introduced two similarity measures for feature terms based on refinement graphs. The large differences between feature terms and description logic imply that their ideas cannot be applied directly, however. For instance, there are ideal refinement operators for feature terms, and also there is no distinction between concept and individual like in DL. In this paper we borrow the basic ideas and extend them in order to define similarity measures for description logic.

The rest of the paper runs as follows. The next section briefly introduces the basic concepts of DL and refinement operators used in the paper. Section 3 defines the proposed similarity measure, along with the algorithms used to compute it. Section 4 exemplifies the application of the similarity measure for a particular domain, while Section 5 presents the results of an empirical evaluation. Finally Section 6 concludes the paper and elaborates on future work.

2 Background

This section briefly summarizes basic concepts regarding description logic, refinement operators and similarity assessment, which we will use in this paper.

2.1 Description Logic

Description Logic (DL) is a family of knowledge representation languages which have received a lot of attention due to the development of the Semantic Web. DL is the logical foundation of OWL [13], the W3C standard ontology language, and consequently there is great interest in the creation and maintenance of knowledge bases coded using this formalism.

DL represents knowledge using three types of basic entities: *concepts*, *roles* and *individuals*. Concepts provide the domain vocabulary required to describe sets of individuals with common features, roles allow to describe relationships between individuals, and individuals represent concrete domain entities. DL expressions are built inductively starting from finite and disjoint sets of atomic concepts (N_C), atomic roles (N_R) and individual names (N_I).

An interpretation is a vector $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set called the *interpretation domain*, and $\cdot^{\mathcal{I}}$ is the interpretation function. The interpretation function relates each atomic concept $A \in N_C$ with a subset of $\Delta^{\mathcal{I}}$, each atomic role $R \in N_R$ with a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each individual $a \in N_I$ with a single element of $\Delta^{\mathcal{I}}$.

Table 1. $\mathcal{EL}$ concepts and semantics

Concept	Syntax	Semantics
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Atomic concept	A	$A^{\mathcal{I}}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

Table 2. TBox axioms

Axiom	Syntax	Semantics
Concept inclusion	$A \sqsubseteq B$	$A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$
Disjointness	$A \sqcap B \equiv \bot$	$A^{\mathcal{I}} \cap B^{\mathcal{I}} = \emptyset$
Role domain	$domain(R) = A$	$(x, y) \in R^{\mathcal{I}} \rightarrow x \in A^{\mathcal{I}}$
Role range	$range(R) = A$	$(x, y) \in R^{\mathcal{I}} \rightarrow y \in A^{\mathcal{I}}$

Table 3. ABox axioms

Axiom	Syntax	Semantics
Concept instance	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role assertion	$R(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
Same individual	$a = b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
Different individual	$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

There are many DL with different expressive power and reasoning complexity depending on which concept constructs are allowed in the language[1]. In this paper we will focus on the $\mathcal{EL}$ logic, a light-weight DL with polynomial reasoning time that has proven to be useful to manage large knowledge bases in real world applications [8,2]. Table 1 shows the $\mathcal{EL}$ concept constructs as well as the extension of the interpretation function to complex concepts. Later in this paper we will use $\mathcal{C}(\mathcal{EL})$ to denote the set of all possible concept expressions that can be built in the $\mathcal{EL}$ logic.

A DL knowledge base (KB), $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, consists of two different types of information: the *TBox* or terminological component, which contains concept and role axioms and describes the domain vocabulary, and the *ABox* or assertional component, which uses the domain vocabulary to assert facts about individuals. For the purposes of this paper, a TBox is a finite set of concept and role axioms given in Table 2, and an ABox is a finite set of axioms about individuals shown in Table 3. We say that a *TBox* is *acyclic* if no concept definition depends directly or indirectly on itself. Note that we only allow concept inclusion axioms between atomic concepts and, therefore, these TBoxes will always be acyclic.

An interpretation $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K}$ iff the conditions described in Tables 2 and 3 are fulfilled for every axiom in $\mathcal{K}$. A concept C is *satisfiable* w.r.t. a knowledge base $\mathcal{K}$ iff there is a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$.

The basic reasoning operation in DL is *subsumption*. Let $\mathcal{K}$ be a knowledge base and C and D be two concepts, we say that C is subsumed by D w.r.t. $\mathcal{K}$ ($C \sqsubseteq_{\mathcal{K}} D$)

[1] See http://www.cs.man.ac.uk/~ezolin/dl/ for further information.

iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. When the knowledge base $\mathcal{K}$ is known we can simplify the notation and write $C \sqsubseteq D$. Finally, an *equivalence axiom* $C \equiv D$ is just an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$, and a *strict subsumption axiom* $C \sqsubset D$ is simply $C \sqsubseteq D$ and $C \not\equiv D$.

2.2 Refinement Operators

This section briefly summarizes the notion of *refinement operator* and introduces the concepts relevant for this paper (see [15] for a more in-depth analysis of refinement operators). Refinement operators are defined over *quasi-ordered sets*.

Definition 1. *A* quasi-ordered set *is a pair* $(S, \leq)$*, where* S *is a set, and* $\leq$ *is a binary relation among elements of* S *that is* reflexive *(*$a \leq a$*) and* transitive *(if* $a \leq b$ *and* $b \leq c$ *then* $a \leq c$*).*

If $a \leq b$ and $b \leq a$, we say that $a \approx b$, or that they are *equivalent*.
Refinement operators are defined as follows:

Definition 2. *A* downward refinement operator ρ *over a quasi-ordered set* $(S, \leq)$ *is a function such that* $\forall a \in S : \rho(a) \subseteq \{b \in S | b \leq a\}$.

Definition 3. *An* upward refinement operator γ *over a quasi-ordered set* $(S, \leq)$ *is a function such that* $\forall a \in S : \gamma(a) \subseteq \{b \in S | a \leq b\}$.

In other words, upward refinement operators generate elements of S which are "bigger" (which in this paper will mean "more general"), whereas downward refinement operators generate elements of S which are "smaller" (which in this paper will mean "more specific"). Typically, the symbol γ is used to symbolize upward refinement operators, and ρ to symbolize either a downward refinement operator, or a refinement operator in general. A common use of refinement operators is for navigating sets in an orderly way, given a starting element. Typically, the following properties of operators are considered desirable:

- A refinement operator ρ is *locally finite* if $\forall a \in S : \rho(a)$ is finite.
- A downward refinement operator ρ is *complete* if $\forall a, b \in S | a \leq b : a \in \rho^*(b)$.
- An upward refinement operator γ is *complete* if $\forall a, b \in S | a \leq b : b \in \gamma^*(a)$.
- A refinement operator ρ is *proper* if $\forall a, b \in S \; b \in \rho(a) \Rightarrow a \not\approx b$.

where ρ^* means the *transitive closure* of a refinement operator. Intuitively, *locally finiteness* means that the refinement operator is computable, *completeness* means we can generate, by refinement of a, any element of S related to a given element a by the order relation $\leq$ (except maybe those which are equivalent to a), and *properness* means that a refinement operator does not generate elements which are equivalent to a given element a. When a refinement operator is locally finite, complete and proper, we say that it is *ideal*. Other interesting properties of refinement operators have been discussed in the literature, such as *minimality* [5], but are not relevant for the purposes of this paper.

Concerning DL, the set of all the possible concept expressions and the subsumption relation between concepts form a quasi-ordered set and, therefore, we can define DL

$$\rho(C) = \rho_1(C) \cup \rho_2(C) \cup \rho_3(C) \cup \rho_4(C)$$

$$\begin{aligned}
\rho_1(C) &= \{C[A_i \rightarrow B] \mid B \in max\{B' \in N_C \mid B' \sqsubset A_i\}\} \\
\rho_2(C) &= \{C \sqcap B \mid B \in max\{B' \in N_C \mid \forall A \in C : A \not\sqsubseteq B' \wedge B' \not\sqsubseteq A\}\} \\
\rho_3(C) &= \{C[\exists R_i.D_j \rightarrow \exists R_i.E] \mid E \in \rho(D_j)\} \\
\rho_4(C) &= \text{see Algorithm 1}
\end{aligned}$$

Fig. 1. Refinement operator

refinement operators to specialize or generalize concepts. In this paper we focus on the $\mathcal{EL}$ logic which has the advantage of having ideal refinement operators to traverse the quasi-ordered set $(\mathcal{C}(\mathcal{EL}), \sqsubseteq)$ [17].

It is also well known that there are no ideal refinement operators for the $\mathcal{ALC}$ logic, nor for any more expressive logic than that [18]. Fortunately, the similarity metric we describe in this paper does not require ideal operators and thus our approach is still valid for more expressive description logics.

2.3 A Refinement Operator for the $\mathcal{EL}$ Logic

Any $\mathcal{EL}$ concept C can be written as a conjunction of concepts $C_1 \sqcap \ldots \sqcap C_n$ where each C_i is either an atomic concept A or an existential restriction $\exists R.D$ which filler D follows the same rules. We say that C_i is *redundant* in C if there is another C_j in C such that $C_j \sqsubseteq C_i$ $(i \neq j)$, that is, if the information contained in C_i is also in C_j. We say that a concept C is *minimal* if it does not contain any redundant subconcept and the fillers of the existential restrictions are minimal as well. And, of course, any concept can be reduced to a minimal concept by removing the redundant information.

The refinement operator we propose, ρ, is shown in Figure 1 and it is proper only if it receives a minimal concept. We defined the operator as the union of four simpler operators (ρ_1, ρ_2, ρ_3 and ρ_4) that specialize the concept C in different ways. The idea behind ρ_1 and ρ_2 is to specialize the original concept adding the most general atomic concepts which provide some new information. Symmetrically, ρ_3 and ρ_4 specialize the original concept adding the most general existential restrictions which provide some new information. Next, we describe each one of these operators in depth.

ρ_1 specializes a concept C replacing any of its atomic concepts A with one of its direct descendants in the conceptual hierarchy. For example, if there is a concept *Car* in the domain ontology, $\rho_1(Car)$ will return different types of car like *ShortCar* or *CloseCar*.

ρ_2 refines a concept C adding the most general atomic concepts which neither subsume nor are subsumed by other atomic concept currently in C. For example, the operator $\rho_1(ShortCar)$ returns formulas like $ShortCar \sqcap CloseCar$. Both ρ_1 and ρ_2 can be easily computed by traversing the hierarchy of atomic concepts.

ρ_3 and ρ_4 follow the same ideas but they operate on existential restrictions rather than on atomic concepts. ρ_3 specializes existential restrictions currently in C applying the refinement operator to their fillers. For example, if $C \equiv Car \sqcap \exists hasLoad.Load$

Algorithm 1. $\rho_4(C)$

```
1: RES = $\emptyset$
2: for $R \in N_R$ do
3:     DS = $\{D \mid \exists R.D \in C\}$
4:     REM = $\{range(R)\}$
5:     while REM $\neq \emptyset$ do
6:         E = pickOne(REM)
7:         if $\exists D \in DS : D \sqsubset E$ then
8:             REM = REM $\cup \rho(E)$
9:         else
10:            if $\nexists D \in DS : E \sqsubseteq D$ then
11:                RES = RES $\cup\{C \sqcap \exists R.E\}$
12:            end if
13:        end if
14:    end while
15: end for
16: return RES
```

and the domain ontology describes different types of loads like *Triangle* or *Circle*, then $\rho_3(C)$ returns concepts like $Car \sqcap \exists hasLoad.Triangle$ and $Car \sqcap \exists hasLoad.Circle$.

Finally, ρ_4 refines a concept C adding the most general existential restrictions which neither subsume nor are subsumed by other existential restrictions currently in C. Algorithm 1 shows how to compute these refinements. The idea is to find the most general fillers for each role which contribute some new information. The algorithm begins with the most general filler for each role, its range, and stores it in the set REM which contains the candidates that have not been processed yet. In each *while* loop the algorithm processes one of these remaining elements, E, according to the following ideas:

- if E subsumes (is more general than) some of the existing fillers in C then E does not provide any new information yet and we need to keep specializing it, so we add all its refinements to REM.
- if E is subsumed by (is more specific than) some of the existing fillers in C then we ignore it because this situation is already covered by ρ_3.
- if E does not subsume any of the existing fillers in C and none of them subsumes E then we have found a new interesting existential restriction and we add the corresponding formula to the solution set.

For example, $\rho_4(Car)$ will return formulas with new existential restrictions like $Car \sqcap \exists load.Shape$ or $Car \sqcap \exists wheels.Number$.

3 Measuring Similarity Using Refinement Operators

In this section we present our *DL refinement* (S_{DL_ρ}) similarity measure for individuals in description logic which is based on the following intuitions:

First, given two concepts C and D such that $C \sqsubseteq D$, it is possible to reach C from D by applying a complete downward refinement operator ρ to D a finite number of times, i.e. $C \in \rho^*(D)$.

Second, the number of times a refinement operator needs to be applied to reach C from D is an indication of how much more specific C is than D. In other words, the length of the refinement chain to reach C from D, which we will note by $\lambda(D \xrightarrow{\rho} C)$, is an indication of how much more information C has that was not contained in D. It is also an indication of their similarity: the smaller the length, the higher their similarity. Additionally, $\lambda(\top \xrightarrow{\rho} C)$ measures the distance from the most general concept, $\top$, to C, which is a measure of the amount of information in C.

Third, given any two concepts, their *least common subsumer* (LCS) is the most specific concept which subsumes both. The LCS of two concepts contains all that is shared between two concepts, and the more they share the more similar they are.

Using the previous three ideas, we can now define similarity between two concepts C and D as:

$$S^C_{DL\rho}(C, D) = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

where

$$\begin{aligned} \lambda_1 &= \lambda(\top \xrightarrow{\rho} LCS(C, D)) \\ \lambda_2 &= \lambda(LCS(C, D) \xrightarrow{\rho} C) \\ \lambda_3 &= \lambda(LCS(C, D) \xrightarrow{\rho} D) \end{aligned}$$

Thus, the similarity between two concepts C and D is assessed as the amount of information contained in their LCS (i.e. the amount of information they share) divided by the total amount of information in C and D (the common information plus the information specific to each one).

Additionally, the same method can be used to assess similarity between individuals by adding an additional idea: that of the *most specific concept*. The most specific concept (MSC) of an individual is the most specific concept we can create in a given DL which contains the given individual, i.e. the concept in the DL which better represents such individual. Given two individuals a and b, their similarity can be assessed in the following way:

$$S_{DL\rho}(a, b) = S^C_{DL\rho}(msc(a), msc(b))$$

The remainder of this section elaborates these ideas.

3.1 Most Specific Concepts

Let us start by briefly describing the idea of most specific concept (MSC). The MSC of a given individual is the most specific concept we can create which contains a given individual. It is well known that depending on the set of constructs allowed in the DL, the MSC might exist or not, and different algorithms have been proposed, like Baader's

Algorithm 2. msc(a,$\mathcal{A}$)

```
1: MSC = $\top$
2: for $C(a) \in \mathcal{A}$ do
3:     MSC = MSC $\sqcap C$
4: end for
5: for $R(a,b) \in \mathcal{A}$ do
6:     MSC = MSC $\sqcap(\exists R.msc(b,\mathcal{A}))$
7: end for
8: return MSC
```

for the $\mathcal{EL}$ logic [3]. This section presents a simple algorithm to approximate the MSC in $\mathcal{EL}$ assuming non-cyclic TBoxes (i.e. non-cyclic concept definitions).

Specifically, a concept C is said to be the MSC of an individual a with respect to an ABox $\mathcal{A}$, $msc_{\mathcal{A}}(a) = C$, if $C(a)$ and for each concept D such that $D(a)$, $C \sqsubseteq D$ holds.

In general, the MSC does not always exist for a given individual in $\mathcal{EL}$. To illustrate why does this happen, let us consider the following example. Let $\mathcal{A} = \{R(a,a)\}$ be an ABox, and $n \geq 0$. It is easy to see that a is an instance of the following concepts:

$$C_n \equiv \underbrace{\exists R. \ldots \exists R}_{n\ times}.\top$$

Notice that C_i is more specific than C_j if $i > j$, and thus, in this case, there is no concept which can satisfy the definition of MSC. This problem arises whenever there is a cycle in the definition of an individual. For simplicity reasons, in the remainder of this paper we will assume that individuals contain no cycles in their definition.

In the $\mathcal{EL}$ logic and under the assumption of no cycles we have used Algorithm 2 to compute the MSC of an individual. For example, given the following ABox which describes a train with one car which contains a triangle and a square:

Train(t1), hasCar(t1,c1), Car(c1), hasLoad(c1,l1), Triangle(l1), hasLoad(c1,l2), Square(l2)

our algorithm computes the following MSC of $t1$ that coincides with what was expected:

$$Train \sqcap \exists hasCar.(Car \sqcap \exists hasLoad.Triangle \sqcap \exists hasLoad.Square)$$

Notice that our acyclicity assumption does not restrict the application of the similarity measure presented in this paper; in case cycles are present, we would only need a different way of computing the MSC, like the one presented in [3].

3.2 Least Common Subsumer

Once we have the MSC of the two individuals we want to compare, the next step is to obtain the most specific concept that subsumes both, that is, their *least common subsumer* (LCS) [21].

Algorithm 3. lcs($C_1, ..., C_n$)

```
1: LCS = $\top$
2: while true do
3:     $N = \{C \in \rho(LSC) | \forall_{i=1...n} C_i \sqsubseteq C\}$
4:     if $N = \emptyset$ then
5:         return LCS
6:     else
7:         LCS = any $C \in N$
8:     end if
9: end while
```

Definition 4. *The* Least common subsumer *(LCS) of a set of given concepts, $C_1, ..., C_n$ is another concept $C = LCS(C_1, ..., C_n)$ such that $\forall_{i=1...n} C_i \sqsubseteq C$, and for any other concept C' such that $\forall_{i=1...n} C_i \sqsubseteq C'$, $C \sqsubseteq C'$ holds.*

Depending on the specific DL being used, computing the LCS is trivial or not. In general, it can be computed by means of a search process, such as the one presented in Algorithm 3. Algorithm 3 works as follows. Initially, the LCS is set to the most general concept, $\top$. Then, the set of refinements of $\top$ that are still more general than all the concepts $C_1, ..., C_n$ is computed and stored in the set N. If N is empty, we know that there are no refinements of the current LCS that are still more general than all of the concepts $C_1, ..., C_n$, and thus, we have already found the LCS. If N is non-empty, we can just select any of the concepts in N, and keep refining. In case the refinement operator is not proper, then which $C \in N$ is selected has to be carefully performed for not entering into an infinite loop.

For example, given two concepts $C_1 = Train \sqcap \exists hasCar.(\exists hasLoad.Triangle)$, and $C_1 = Train \sqcap \exists hasCar.(\exists hasLoad.Square)$, and assuming that the most specific concept that is more general than $Square$ and $Triangle$ in our ABox is $Shape$, we can use the previous algorithm to conclude that $msc(C_1, C_2) = Train \sqcap \exists hasCar.(\exists hasLoad.Shape)$.

3.3 Measuring Distances in the Refinement Graph

The last piece we require for defining $S_{DL\rho}$ is a way to measure the distance in the refinement graph between two concepts C and D, such that $D \sqsubseteq C$, i.e. $\lambda(C \xrightarrow{\rho} D)$. This can be done by measuring the number of refinements required to reach D from C.

Algorithm 4. $\lambda(C \xrightarrow{\rho} D)$

```
1: if $C \equiv D$ then
2:     return 0
3: else
4:     $C' \in \{E \in \rho(C) | D \sqsubseteq E\}$
5:     return $1 + \lambda(C' \xrightarrow{\rho} D)$
6: end if
```

Computing the minimum number of refinements required to reach D from C might be computationally too expensive, so in $S_{DL\rho}$ we just use an estimate computed using Algorithm 4. Algorithm 4 works as follows. If C is already equivalent to D, then their distance in the refinement graph is 0, otherwise, the algorithm takes one refinement C' of C, and recursively computes the distance from C' to D, the distance from C to D is then just 1 plus the distance from C' to D.

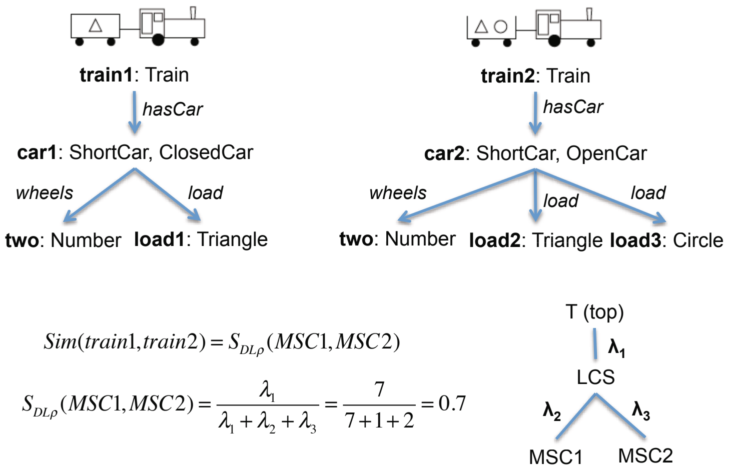
4 Exemplification

This section shows an example of the similarity measure $S_{DL\rho}$ using the domain about trains introduced by Michalsky [16]. The two specific trains we are going to compare are shown in Figure 2. Both of them have just one short car that transports some type of load. The differences are that the car of $train1$ has a closed top and transports a triangle, while the car of $train2$ has an open top and transports a triangle and a circle.

In order to compute the similarity between both trains, we need to compute first the most specific concepts (MSC) that represents them using Algorithm 2:

$$\begin{aligned} msc_1 \equiv\ & Train \sqcap \exists hasCar.(ClosedCar \sqcap ShortCar \sqcap \exists load.Triangle \sqcap \\ & \exists wheels.Two) \\ msc_2 \equiv\ & Train \sqcap \exists hasCar.(OpenCar \sqcap ShortCar \sqcap \exists load.Triangle \sqcap \\ & \exists load.Circle \sqcap \exists wheels.Two) \end{aligned}$$

Next we compute the most specific concept that subsumes the previous ones, that is $LCS(msc_1, msc_2)$, using Algorithm 3. This algorithm produces the following sequence of refinements:



$$Sim(train1, train2) = S_{DL\rho}(MSC1, MSC2)$$

$$S_{DL\rho}(MSC1, MSC2) = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} = \frac{7}{7+1+2} = 0.7$$

Fig. 2. Example of similarity between 2 trains

$$
\begin{aligned}
&0 : \top\\
&1 : Train\\
&2 : Train \sqcap \exists hasCar.Car\\
&3 : Train \sqcap \exists hasCar.ShortCar\\
&4 : Train \sqcap \exists hasCar.(ShortCar \sqcap \exists load.Shape)\\
&5 : Train \sqcap \exists hasCar.(ShortCar \sqcap \exists load.Triangle)\\
&6 : Train \sqcap \exists hasCar.(ShortCar \sqcap \exists load.Triangle \sqcap \exists wheels.Number)\\
&7 : Train \sqcap \exists hasCar.(ShortCar \sqcap \exists load.Triangle \sqcap \exists wheels.Two)
\end{aligned}
$$

The LCS is the last concept in the previous sequence, and describes the information that is common to both trains: they have one short car with two wheels which transports a triangle. The amount of information shared by both trains can be measured as the length of the previous sequence ($\lambda_1 = 7$).

Then, we compute the amount of information that is specific to each train using Algorithm 4. First we search for a sequence of refinements from the LCS to msc_1 ($\lambda_2 = 1$):

$$
\begin{aligned}
1 : &Train \sqcap \exists hasCar.(ClosedCar \sqcap ShortCar \sqcap \exists load.Triangle \sqcap\\
&\exists wheels.Two)
\end{aligned}
$$

Next, we compute the sequence of refinements from the LCS to msc_2 ($\lambda_2 = 2$)):

$$
\begin{aligned}
1 : &Train \sqcap \exists hasCar.(OpenCar \sqcap ShortCar \sqcap \exists load.Triangle \sqcap\\
&\exists wheels.Two)\\
2 : &Train \sqcap \exists hasCar.(OpenCar \sqcap ShortCar \sqcap \exists load.Cricle\\
&\sqcap \exists load.Triangle \sqcap \exists wheels.Two)
\end{aligned}
$$

Finally, the similarity between both trains is computed as follows:

$$
S_{DL\rho}(train1, train2) = S^{C}_{DL\rho}(msc_1, msc_2) = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} = \frac{7}{7 + 1 + 2} = 0.7
$$

5 Empirical Evaluation

In order to evaluate our similarity measure, we used the trains data set shown in Figure 3 as presented by Michalski [16]. We selected this dataset since it is available in many representation formalisms (Horn clauses, feature terms and description logic), and therefore, we can compare our similarity measure with existing similarity measures in the literature. The dataset consists of 10 trains, 5 of them labelled as "west", and 5 of them labelled as "east."

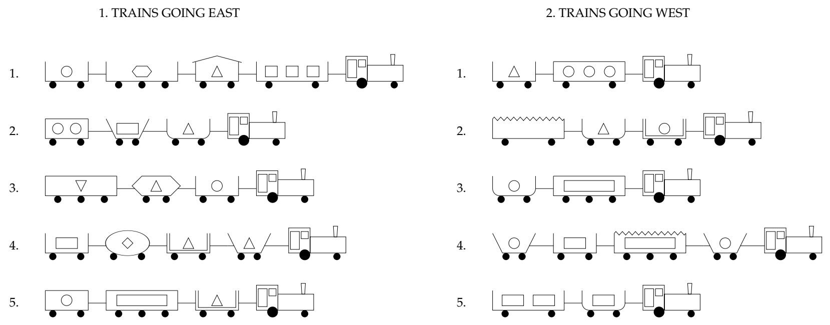


Fig. 3. Trains data set as introduced by Michalski [16]

We compared our similarity measure against 6 others: González et al. [12], a similarity measure for acyclic concepts in description logic, RIBL [11], which is a Horn clause similarity measure, SHAUD [1], which is a similarity measure for feature terms, and S_λ, S_π, and $S_{w\pi}$ [20], which are similarity measures for feature terms but also based on the idea of refinement operators. For RIBL, we used the original version of the trains dataset, for SHAUD, S_λ, S_π, and $S_{w\pi}$, we used the feature term version of the dataset used in [20], which is a direct conversion from the original Horn clause dataset without loss, and for our DL similarity measure (referred to as $S_{DL\rho}$ in Table 4), we used the version created by Lehmann and Hitzler [19].

We compared the similarity measures in 5 different ways:

– Classification accuracy of a nearest-neighbor algorithm.
– Classification accuracy of a 3-nearest neighbor algorithm.
– *Average best rank* of the first correct example: if we take one of the trains, and sort the rest of the trains according to their similarity with the selected train, which is the position in this list (rank) of the first train with the same solution as the selected train (west or east).
– Jaro-Winkler distance: the Jaro-Winkler measure [24] can be used to compare two orderings. We measure the similarity of the rankings generated by our similarity measure with the rankings generated with the other similarity measures.
– Mean-Square Difference (MSD): the mean square difference with respect to our similarity measure, $S_{DL\rho}$.

Table 4 shows the results we obtained by using a leave-one-out evaluation. Concerning classification accuracy, we can see that our similarity measure (labeled $S_{DL\rho}$ in the table) achieves a high classification accuracy, higher than most other similarity measures, except $S_{w\pi}$. The trains data-set is only apparently simple, since the classification criteria is a complex pattern which involves several elements from different cars in a train. The only similarity measure that came close is $S_{w\pi}$, which achieved an 80% accuracy (it misclassified trains west 1 and west 3). Concerning the average best rank, either the first or the second retrieved case using our similarity measure was always of the correct solution class, and thus it is very low, 1.4. Using the Jaro-Winkler similarity, and the

Table 4. Comparison of several similarity metrics in the trains dataset

	$S_{DL\rho}$	González et al.	RIBL	SHAUD	S_λ	S_π	$S_{w\pi}$
Accuracy 1-NN	70%	50%	60%	50%	40%	50%	80%
Accuracy 3-NN	70%	60%	70%	80%	70%	80%	80%
Best Rank	1.4	1.5	2.0	2.0	2.3	2.1	1.7
Jaro-Winkler	-	0.78	0.72	0.76	0.71	0.77	0.72
MSD	-	0.11	0.03	0.05	0.02	0.05	0.17

MSD, we can see that $S_{DL\rho}$ generates an order very similar to González et al.'s similarity, but that in terms of MSD, it is closest to S_λ, which is also based on refinement operators (although for feature terms instead of description logic).

Although a more thorough evaluation of our measure by integrating it into a real CBR system in a more complex task is part of our future work, our empirical evaluation shows promising results and confirms that refinement operators are a viable approach to assess similarity in CBR systems which use description logic as their representation formalism.

6 Conclusions and Future Work

We have presented the similarity measure $S_{DL\rho}$ for the $\mathcal{EL}$ description logic, based on notions of refinement graph and generalization space. Refinement graphs were introduced in a subset of Horn logic for the purpose of modeling inductive learning, until some of the authors of this paper [20] proposed they could be used for the purpose of estimating similarity in knowledge representation formalisms like description logic. Since that previous work presented S_λ, a similarity measure for feature terms, part of the claim was unsubstantiated until now, where $S_{DL\rho}$ is shown to be similarly defined for a given description logic.

Similarity is of great importance to CBR, and similarity for representation formalisms like description logic is important for knowledge-intensive CBR, but also for web-based applications, ontology alignment, and other tasks for AI systems. We consider this work a start into the process of achieving a better understanding of the relationship of case-based reasoning and the other fields of AI, like knowledge representation, logic, and inductive learning. More work need to be done, but this understanding might also be instrumental in greater visibility of CBR in the framework of artificial intelligence community.

Future work will focus on defining refinement-based similarity measures for more expressive description logics (DL) and also for subsets of Horn logics. Much of the work on the family of description logic revolves around finding subsets of DL that are expressive but computationally tractable; OWL, for instance, defines 3 language levels of increasing expressiveness and complexity. Defining refinement operators for high complexity DL may not be practical, so finding a more expressive subset of DL for which a tractable refinement-based similarity measure exists is our next goal.

References

1. Armengol, E., Plaza, E.: Relational case-based reasoning for carcinogenic activity prediction. Artif. Intell. Rev. 20(1-2), 121–141 (2003)
2. Ashburner, M.: Gene ontology: Tool for the unification of biology. Nature Genetics 25, 25–29 (2000)
3. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, pp. 319–324. Morgan Kaufmann Publishers Inc., San Francisco (2003), http://portal.acm.org/citation.cfm?id=1630659.1630706
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, New York (2003)
5. Badea, L., Nienhuys-Cheng, S.H.: A refinement operator for description logics. In: Cussens, J., Frisch, A. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 40–59. Springer, Heidelberg (2000)
6. Bergmann, R., Stahl, A.: Similarity measures for object-oriented case representations. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 8–13. Springer, Heidelberg (1998)
7. Blockeel, H., Ramon, J., Shavlik, J.W., Tadepalli, P.: ILP 2007. LNCS (LNAI), vol. 4894. Springer, Heidelberg (2008)
8. Bodenreider, O., Smith, B., Kumar, A., Burgun, A.: Investigating subsumption in SNOMED CT: An exploration into large description logic-based biomedical terminologies. Artif. Intell. Med. 39, 183–195 (2007), http://portal.acm.org/citation.cfm?id=1240342.1240604
9. Cojan, J., Lieber, J.: An algorithm for adapting cases represented in an expressive description logic. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 51–65. Springer, Heidelberg (2010)
10. d'Amato, C., Staab, S., Fanizzi, N.: On the influence of description logics ontologies on conceptual similarity. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 48–63. Springer, Heidelberg (2008)
11. Emde, W., Wettschereck, D.: Relational instance based learning. In: Saitta, L. (ed.) Machine Learning - Proceedings 13th International Conference on Machine Learning, pp. 122–130. Morgan Kaufmann Publishers, San Francisco (1996)
12. González-Calero, P.A., Díaz-Agudo, B., Gómez-Albarrán, M.: Applying DLs for retrieval in case-based reasoning. In: Proceedings of the 1999 Description Logics Workshop, DL 1999 (1999)
13. van Harmelen, F., McGuinness, D.L.: OWL Web Ontology Language Overview. W3C recommendation, W3C (February 2004), http://www.w3.org/TR/2004/REC-owl-features-20040210/
14. Horváth, T., Wrobel, S., Bohnebeck, U.: Relational instance-based learning with lists and terms. Machine Learning 43(1-2), 53–80 (2001)
15. van der Laag, P.R.J., Nienhuys-Cheng, S.H.: Completeness and properness of refinement operators in inductive logic programming. Journal of Logic Programming 34(3), 201–225 (1998)
16. Larson, J., Michalski, R.S.: Inductive inference of VL decision rules. SIGART. Bull. 63(63), 38–44 (1977)
17. Lehmann, J., Haase, C.: Ideal downward refinement in the EL description logic. In: Raedt, L.D. (ed.) ILP 2009. LNCS, vol. 5989, pp. 73–87. Springer, Heidelberg (2010)

18. Lehmann, J., Hitzler, P.: Foundations of refinement operators for description logics. In: Blockeel, H., et al. (eds.) [7], pp. 161–174
19. Lehmann, J., Hitzler, P.: A refinement operator based learning algorithm for the LC description logic. In: Blockeel, H., et al. (eds.) [7], pp. 147–160
20. Ontañón, S., Plaza, E.: On similarity measures based on a refinement lattice. In: Wilson, D., McGinty, L. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 240–255. Springer, Heidelberg (2009)
21. Plotkin, G.D.: A note on inductive generalization. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. 5, pp. 153–163. Edinburgh University Press, Edinburgh (1970)
22. Salotti, S., Ventos, V.: Study and formalization of a case-based reasoning system using a description logic. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 286–297. Springer, Heidelberg (1998)
23. Sánchez-Ruiz-Granados, A.A., González-Calero, P.A., Díaz-Agudo, B.: Abstraction in knowledge-rich models for case-based planning. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 313–327. Springer, Heidelberg (2009)
24. Winkler, W.E., Thibaudeau, Y.: An application of the Fellegi-Sunter model of record linkage to the 1990 U.S. decennial census. In: U.S. Decennial Census. Technical report, US Bureau of the Census (1987)

Term Similarity and Weighting Framework for Text Representation

Sadiq Sani, Nirmalie Wiratunga, Stewart Massie, and Robert Lothian

School of Computing,
The Robert Gordon University,
Aberdeen AB25 1HG, Scotland, UK
{s.a.sani,n.wiratunga,r.m.lothian,s.massie}@rgu.ac.uk

Abstract. Expressiveness of natural language is a challenge for text representation since the same idea can be expressed in many different ways. Therefore, terms in a document should not be treated independently of one another since together they help to disambiguate and establish meaning. Term-similarity measures are often used to improve representation by capturing semantic relationships between terms. Another consideration for representation involves the importance of terms. Feature selection techniques address this by using statistical measures to quantify term usefulness for retrieval. In this paper we present a framework that combines term-similarity and weighting for text representation. This allows us to comparatively study the impact of term similarity, term weighting and any synergistic effect that may exist between them. Study of term similarity is based on approaches that exploit term co-occurrences within document and sentence contexts whilst term weighting uses the popular Chi-squared test. Our results on text classification tasks show that the combined effect of similarity and weighting is superior to each technique independently and that this synergistic effect is obtained regardless of co-occurrence context granularity.

1 Introduction

While unstructured, natural language text is convenient for human consumption, computers still find it difficult to process such information with satisfactory precision. This is because the lexical content of natural language text can be quite different from its intended meaning due to inherent ambiguities in natural language such as synonymy (different terms having similar meaning) and polysemy (the same term having multiple different meanings). Representation of text documents is of interest to many research fields such as Information Retrieval (IR), Natural Language Processing and Textual CBR. The standard Bag of Words (BOW) representation is a naive approach in that it operates at the lexical level, treating terms as independent features [16]. However in the presence of ambiguities such approaches using lexical features alone for representation will remain ignorant of latent semantics that are needed to disambiguate the text.

To address such limitations the semantic relatedness between terms must be taken into account and one approach to achieve this is through the acquisition of term-term similarity knowledge.

Notion of similarity can be ascertained on the basis of term co-occurrence in a corpus. The context within which co-occurrence is mined is very important. Most approaches for extracting term co-occurrence statistics do so in the context of the whole document [7,19,5]. It can be argued that at the document level, every term can possibly co-occur with every other term and thus, document contexts do not accurately capture semantic relatedness. A sentence however, is a more linguistically justified context as it expresses one complete idea [15]. Thus mining co-occurrence at the sentence level is likely to be better at capturing the semantic relatedness between terms. An alternative approach for term-similarity extraction involves maintaining a profile of the terms (lexical co-occurrents) that co-occur with a given term within a predefined window of text. Accordingly, similarity between a term pair can be determined based on the similarity of their corresponding lexical co-occurrence profiles [17]. In this paper we study co-occurrence based term similarity extracted from document and sentence contexts for text representation. We compare these with a lexical co-occurrence approach with a sentence-based window, where two terms are similar if they have similar term co-occurrence profiles. We present a framework to combine term similarity knowledge and weighting and study the synergistic effect of these on document representation.

The rest of the paper is organised as follows: in Section 2 we provide an overview of different sources of term-similarity knowledge. In particular we look at extrospective (using background knowledge) and introspective approaches to similarity knowledge acquisition. We introduce our term similarity and weighting framework in Section 3 explaining issues related to normalisation and weighting. Section 4 presents three introspective term similarity acquisition approaches based on co-occurrence mining. We present results from five text classification tasks in Section 5, followed by conclusions in Section 6.

2 Term Similarity Extraction

A solution commonly used for overcoming the variation in natural vocabulary is to find measures of semantic relatedness between terms in a corpus. This provides a mapping between different lexical expressions of a similar idea into conceptual groups that capture the inherent meanings of documents. The result of this is the extraction of some high level features that represent the underlying semantic concepts in a document. Achieving this however requires a source of similarity knowledge. Techniques for extracting similarity knowledge range from using extrospective (knowledge rich) sources which include lexical databases and the World Wide Web (WWW) to introspective (knowledge light) techniques that use statistics of term co-occurrences in a corpus.

2.1 Extrospective Sources of Similarity Knowledge

WordNet, a lexical database for the English language [13], has been used extensively for extracting term-similarity knowledge. Words within WordNet are grouped into sets of cognitive synonyms called synsets each expressing a distinct concept. Synsets are further grouped based on their grammatical function into noun, verb, adjective and adverb dictionaries. Synsets within the same dictionary are inter-connected through links representing the semantic and lexical relationships between them. This structure can be viewed as a graph where synsets are nodes and semantic links are edges. Such a graph allows to measure relatedness between terms by means of combining shortest path between term pairs and information about the depth of nodes in the graph [20] and information content [14,10,12]. Despite its popularity, WordNet has recently been criticised for having limited coverage and scope of applications [9]. It also suffers from sparsity in its synset connections [2] with the different dictionaries within WordNet being mostly independent with very limited inter-connections between them.

Unlike WordNet, Wikipedia, a free online encyclopaedia, boasts vast coverage in orders of magnitude greater than that of lexical databases and thesauri. Wikipedia has been used to explicitly represent the meaning of natural language by representing text documents in a high-dimensional space of Wikipedia concepts [8]. Wikipedia is particularly attractive as a source of semantic knowledge because each Wikipedia page provides a comprehensive description of a single topic or concept and can thus be seen as a representation of that concept. Other researchers go beyond this and exploit the entire WWW as a means to extract semantic knowledge e.g. using page counts [6]. Page count of documents returned in response to a search engine query provides useful evidence of relatedness between terms in the query. This can then be quantified as a similarity metric i.e. the higher the proportion of documents that contain both terms, the more related the two terms are. However page count can often be misleading as it does not consider the intended sense of terms and the semantics within which they are used in the result pages. Sophisticated approaches using text snippets[1] can be used to improve on page count by exploiting lexical syntactic patterns in these snippets [1].

The major downside of extrospective techniques is the demand on maintenance and processing power. It is certainly a non-trivial task to develop and maintain a lexical database like WordNet for a new domain or language. While online resources like Wikipedia and the WWW partly address this problem, they also introduce the problems of network availability (and latency), and demand for storage and memory when content is alternatively downloaded and processed locally. Such issues make corpus statistics a popular option for term-similarity extraction.

2.2 Introspective Sources of Similarity

The general idea is that co-occurrence patterns of terms in a corpus can be used to infer semantic relatedness between terms. Co-occurrence is also helpful

[1] Small pieces of text extracted by the search engine around the query term.

for estimating domain specific relationships between terms [5]. Latent Semantic Indexing (LSI) [7] is a popular technique in this category that uses singular-value decomposition (SVD) to exploit co-occurrence patterns of terms and documents to create a semantic concept space where documents that are related are brought closer together. In this way, LSI brings out the underlying latent semantic structure in texts. It has been shown that LSI implicitly exploits higher-order co-occurrences between terms in a collection [11]. When two terms t_1 and t_2 co-occur within the same document they are said to have a first order association. However when t_1 and t_2 do not co-occur but do co-occur with a common term t_3 then t_1 and t_2 are said to share a second order co-occurrence through t_3 and so on for higher orders. Unlike LSI which implicitly exploits higher order associations between terms using SVD, an explicit approach to combining co-occurrence paths up-to the third order between terms has been successfully used to extract similarity arcs for a Case Retrieval Net (CRN) applied to text classification [5].

The standard approach to term similarity mining is to extract term co-occurrence counts within a specific context where the context can range from whole documents to paragraphs, sentences and even word sequences [18]. An alternative approach is to obtain a profile of the co-occurrents of a term within the entire corpus. The co-occurrents of a terms are the other terms that have a first order co-occurrence with it within a predetermined range (or window). A window size can span from about a hundred terms to just a few terms on either side of the focus term. By passing such a window over the entire corpus, we can obtain for each term, a list of its co-occurrents and represent that in a vector space, where the dimensions of this space are the set of unique terms in the corpus. The similarity between any two terms is then obtained by comparing their term vectors. Such an approach has been employed to construct a thesaurus, where synonyms are identified based on vector similarity [17].

3 Text Representation Framework

The first step in our framework is to obtain all pairwise term similarity values to populate a term-term similarity matrix T where the row and column dimensions of T represent the corpus terms. Each entry in T determines similarity of corresponding row and column terms and all entries in T are normalised with the value 1 in any cell corresponding to identical term pairs and 0 to dissimilar. Note that because any term can be at most similar to itself, all entries on the leading diagonal of T are consequently 1.

Next T is used to obtain the new document representation capturing term similarity knowledge. This is done by multiplying a document-term matrix D with T to obtain a new term-document matrix D' .

$$D' = T \times D \tag{1}$$

Documents can be represented as column vectors in a term-document matrix D whilst the row dimensions correspond to terms in the corpus. Here any standard text representation scheme such as binary vectors or tf-idf vectors can be used for

D's columns. Intuitively the impact of equation 1 will be to boost the presence of related terms that were not contained in the original documents, which in turn has the beneficial effect of bringing similar documents closer together. This same effect has also been shown to be equivalent to document retrieval using a Case Retrieval Net where entries in matrix T can represent the weights of the similarity arcs between terms, and entries in matrix D represents the relevance arcs between terms and documents [4].

3.1 Normalisation

In our approach we apply an L_2 normalisation function to the rows of T and columns of D before the matrix multiplication in equation 1. This is done so as to prevent frequent terms and longer documents from dominating representations. The L_2 normalisation function is given in equation 2.

$$L_2(\mathbf{v}) = \frac{v_i}{|v|} \tag{2}$$

where $\mathbf{v}$ is any vector and

$$|v| = \sqrt{\Sigma_{i=i}^{n}|v|^2} \tag{3}$$

A further benefit of enforcing normalisation in this manner is that equation 1 now amounts to taking the cosine similarity between the term vectors in T and the document vectors in D.

3.2 Term Weighting

Typically all terms in a corpus do not have equal importance with some select terms having a higher discerning power than others. Thus, a measure of term importance needs to be introduced into document representations such that more important terms have a higher influence on document similarity. This can be done using feature selection techniques. Feature selection is typically used to obtain a statistical score of term importance which is used to rank all terms in a corpus. Terms that rank below a certain threshold are subsequently eliminated from the term-document space. We observe that this score of term importance can be used instead to weight terms such that more important terms have a greater contribution to document representation. In this way, terms are only eliminated if they have a weight of zero.

Many feature selection techniques have been proposed in the literature which include Document Frequency (DF), Information Gain (IG) and Chi squared (χ^2). A comparative analysis on feature selection techniques for text classification found χ^2 and IG to give the best performance closely followed by DF [21]. We expect any effective feature selection technique to be easily incorporated into our framework to provide useful term weights.

When combining any form of term elimination and term similarity, it seems reasonable to first select a subset of high quality features and subsequently infer term similarity knowledge using only the selected terms [19]. This should however be done with caution because a severely reduced term space is not suited to obtaining T. This is because the subset of terms left after the elimination process may not be sufficiently large to infer useful relationships that otherwise would be discovered with the aid of the eliminated terms. Accordingly we propose to use term weighting only after obtaining T. This allows us to obtain useful term relationships and still end up with highly discriminatory features.

Term weights are used in our framework to populate a diagonal matrix , W, which has the same row and column dimensions as T. W can be used to assign a weight to each term in D corresponding to the significance of that term in the domain. This step is presented in equation 4.

$$D' = W \times (T' \times D) \tag{4}$$

4 Modeling Term Similarity Knowledge

Central to the framework presented in section 3 is the presence of term-term similarity knowledge to populate matrix T. Any of the approaches presented in Section 2 can be utilised for this purpose. Here we present two straight forward introspective approaches for extracting term-similarity knowledge: Context Co-occurrence Approach (CCA) and Lexical Co-occurrence Approach (LCA). CCA measures similarity between terms based on the strength of co-occurrence. Two terms are said to co-occur if they happen to appear within a specified context. Here we consider two possible contexts: document or sentence level. LCA on the other hand measures similarity between terms with respect to their association pattern with other terms. A pair of terms are thus deemed similar if they have similar co-occurrence patterns with other terms within a predetermined context window. Each term thus has a lexical co-occurrence profile in the corpus and the similarity between any two terms is determined by the similarity between their co-occurrence profiles.

4.1 Context Co-occurrence Approach (CCA)

Documents are considered to be similar in the vector space model (VSM) if they contain a similar set of terms. The similarity of two documents can be determined by finding the distance between their vector representations in the term-document space defined by D. In the same way, terms can also be considered similar if they appear in a similar set of documents. If document vectors are defined by the columns in matrix D, then the rows of the matrix define the term vectors. Thus, the similarity between two terms can be determined by finding the similarity of their corresponding term vectors. This model can be extended to a general term-context matrix where the co-occurrence context can range from whole documents to sentences or even word sequences of a predefined length. In this work we consider both the document (CCAdoc) and sentence (CCAsent) contexts for extracting co-occurrence based similarity. We demonstrate the process of creating a first-order term-term similarity matrix using an approach similar to that presented in [5], but without restricting ourselves to document-only context.

Starting with a term-context matrix C where the context can be either at the document or sentence level, we obtain a term-term similarity matrix T by multiplying C with its transpose (C^T).

$$T = C \times C^T \tag{5}$$

We observe that the dot product of term vectors, which is a consequence of equation 5, is not a robust measure of similarity. Firstly, the values produced are not normalised - remember we wish to populate T with similarity values within the range 0 (dissimilar) and 1 (identical). Secondly, dot product disregards the relative distribution of terms in favour of their absolute distribution. We explain this using a document level context representation with three example matrices in Figure 1. Here t_1 has exactly the same document distribution as t_3 yet the similarity of t_1 to t_3 (2.0) in T is the same as it's similarity to t_4 and t_5 simply because t_4 and t_5 are highly frequent terms. Also, the similarity of t_4 to t_5 is higher (3.0) than its similarity to t_3 (2.0) even though t_3 and t_4 fail to co-occur within just a single document (compared with a difference of 2 documents between t_4 and t_5). One way to address this is to use vector normalisation to mitigate the effect of vector length.

$$
\begin{array}{c|ccccc|}
\multicolumn{6}{c}{C} \\
 & d_1 & d_2 & d_3 & d_4 & d_5 \\
t_1 & 0 & 1 & 1 & 0 & 0 \\
t_2 & 1 & 0 & 0 & 1 & 0 \\
t_3 & 0 & 1 & 1 & 0 & 0 \\
t_4 & 1 & 1 & 1 & 0 & 0 \\
t_5 & 1 & 1 & 1 & 1 & 1
\end{array}
\qquad
\begin{array}{c|ccccc|}
\multicolumn{6}{c}{C^T} \\
 & t_1 & t_2 & t_3 & t_4 & t_5 \\
d_1 & 0 & 1 & 0 & 1 & 1 \\
d_2 & 1 & 0 & 1 & 1 & 1 \\
d_3 & 1 & 0 & 1 & 1 & 1 \\
d_4 & 0 & 1 & 0 & 0 & 1 \\
d_5 & 0 & 0 & 0 & 0 & 1
\end{array}
\qquad
\begin{array}{c|ccccc|}
\multicolumn{6}{c}{T} \\
 & t_1 & t_2 & t_3 & t_4 & t_5 \\
t_1 & 2.0 & 0.0 & 2.0 & 2.0 & 2.0 \\
t_2 & 0.0 & 2.0 & 0.0 & 1.0 & 2.0 \\
t_3 & 2.0 & 0.0 & 2.0 & 2.0 & 2.0 \\
t_4 & 2.0 & 1.0 & 2.0 & 3.0 & 3.0 \\
t_5 & 2.0 & 2.0 & 2.0 & 3.0 & 5.0
\end{array}
$$

Fig. 1. Example of context co-occurrence term similarity

We apply the L_2 normalisation (as we did in Section 3.1) on the rows of C and columns of C^T before the matrix multiplication in equation 5. This invariably amounts to calculating the cosine similarity between term vectors. Normalisation in this way contrasts with the absolute frequency count based approach presented in [5]. Importantly normalisation helps to mitigate any undue influences from highly-frequent terms on semantic relatedness.

4.2 Lexical Co-occurrence Approach (LCA)

In LCA, we begin with a term-term co-cooccurrence matrix P which we call the lexical co-occurrence matrix. The dimensions of the rows and columns in P are the unique terms in our corpus. P is thus a square matrix and an entry $p_{i,j}$ in P is an integer value indicating the frequency of first order co-occurrence of term p_i and p_j within a context window. Our window size is a whole sentence. Thus terms that co-occur within two different sentences in the corpus will have the value 2 in their corresponding cell in P while terms that do not co-occur within the same sentence will be 0. The term-term similarity matrix P can be obtained from P by calculating the cosine similarity of all pair-wise combinations of the terms vectors of P. This is achieved by multiplying P with its transpose P^T after the rows of P and columns of P^T have been L_2 normalised (as before).

$$T = P \times P^T \tag{6}$$

Equation 6 appears similar to the approach presented in [5] for creating a second order co-occurrence matrix with the exception that the sentence context is used here instead of whole documents. However, there are important difference between the two approaches. For instance, to obtain second order co-occurrences, the entries of P need to be converted into binary values such that the matrix multiplication (dot product of term vectors) of P and P^T produces integer values that are counts of second order paths between term pairs. On the other hand, the cosine similarity in the lexical approach produces normalised, real values that are a measure of similarity between term pairs based on their first order lexical co-occurrences (see resulting T matrix presented in Figure 2) where term vectors in P need not be binary. Consequently the values produced by the cosine similarity are not counts of second order co-occurrence paths neither can they be interpreted as such.

$$P \quad \begin{array}{c|ccccc|} & t_1 & t_2 & t_3 & t_4 & t_5 \\ t_1 & 0 & 1 & 2 & 0 & 0 \\ t_2 & 1 & 0 & 2 & 0 & 0 \\ t_3 & 2 & 2 & 0 & 0 & 1 \\ t_4 & 0 & 0 & 0 & 0 & 1 \\ t_5 & 0 & 0 & 1 & 1 & 0 \end{array} \qquad P^T \quad \begin{array}{c|ccccc|} & t_1 & t_2 & t_3 & t_4 & t_5 \\ t_1 & 0 & 1 & 2 & 0 & 0 \\ t_2 & 1 & 0 & 2 & 0 & 0 \\ t_3 & 2 & 2 & 0 & 0 & 1 \\ t_4 & 0 & 0 & 0 & 0 & 1 \\ t_5 & 0 & 0 & 1 & 1 & 0 \end{array} \qquad T \quad \begin{array}{c|ccccc|} & t_1 & t_2 & t_3 & t_4 & t_5 \\ t_1 & 1.0 & 0.8 & 0.3 & 0.0 & 0.6 \\ t_2 & 0.8 & 1.0 & 0.3 & 0.0 & 0.6 \\ t_3 & 0.3 & 0.3 & 1.0 & 0.3 & 0.0 \\ t_4 & 0.0 & 0.0 & 0.3 & 1.0 & 0.0 \\ t_5 & 0.6 & 0.6 & 0.0 & 0.0 & 1.0 \end{array}$$

Fig. 2. Example of lexical co-occurrence term similarity

Figure 2 provides an example to illustrate the creation of a term-similarity matrix from a lexical co-occurrence matrix. Notice that the terms t_1 and t_2 both co-occur twice with the term t_3 and thus have a resulting similarity of 0.8 in T. Accordingly t_1 and t_4 fail to co-occur with a common term and thus have 0 similarity. Note that entries on the leading diagonal of P can be non-zero to also include the first order co-occurrence of term pairs. By capturing similarity in this way we avoid the need to have separate matrices for the first and second orders.

5 Evaluation

The aim of our experiments is to examine the effect of combining term similarity knowledge with term weighting for document representation. Two different co-occurrence based approaches for extracting term-term similarity knowledge introspectively were presented in Section 4. Using these we further explore the impact of document versus sentence context on the quality of term similarity knowledge. Accordingly we use the following algorithms to mine term-similarity knowledge for matrix the T:

– CCADOC term co-occurrences at the document context level (see section 4.1);
– CCASENT term co-occurrences at the sentence context level (see section 4.1);
– LCA using lexical co-occurrences within a sentence window (see section 4.2).

Once T is obtained we use the framework in Section 3 to generate the document representations. We also study the influence of feature weighting on these representations using the Chi squared (χ^2) measure[2] to obtain feature weights for the matrix W. Our baseline technique, BASE, is a binary vector representation without similarity knowledge and term weighting. Equally the individual influence of feature weighting alone is further investigated by adding this component into BASE (W+). We applied standard text preprocessing operations such as stop-words removal, stemming and also eliminated rare terms (terms with document frequency less than 3). We chose binary feature vectors to represent documents in D rather than tf-idf vectors after experiments found tf-idf did not improve baseline performance. Combining CCADOC term similarities and term weighting with tf-idf representation performed slightly worse (∼0.3-4%) than with binary representation except on the INTERESTFX datasets where performance remained the same. Tf-idf benefits text representation by implicitly providing term weighting using inverse document frequency which favours rare terms over frequent ones. While the application of idf has been very successful in IR, studies on feature selection have shown text classification to work better with techniques that favour frequent terms e.g MI, χ^2 and DF thresholding [21,3]. Thus the performance achieved by combining χ^2 term weighting with idf in our framework suggests a conflict between the two appraoches.

The different representation schemes were compared on text classification tasks using a weighted kNN as our primary algorithm (with k=3) with the cosine similarity metric to identify the neighbourood. kNN's classification accuracy is also compared to that of SVM classifier. Accordingly we restrict our study to binary classification datasets. Weka was used with the Java wrapper for LibSVM for SVM classification. Results are obtained by averaging over stratified ten fold cross validations.

5.1 Datasets

The first dataset used for the experiments was formed from the 20 newsgroups corpus; a collection of approximately 20,000 documents partitioned into 20 newsgroups. The particular version of the 20 newsgroups dataset we used contains 18846 documents from the original collection that are arranged by date and with duplicates removed. From these groups we created the HARDWARE dataset which contains documents from the comp.sys.ibm.pc.hardware and comp.sys.mac.

[2] The final (χ^2) score of term t_i is obtained as the maximum χ^2 value of t_i with respect to all classes as $\chi^2_{max} = max^n_{i=1}\{\chi^2(t, c_i)\}$.

hardware newsgroups. This dataset contains 1000 documents selected randomly from the respective newsgroups. Documents in the comp.sys.ibm.pc.hardware category contain discussions on PC hardware while those in the comp.sys.mac. hardware contain discussions on Apple hardware.

The second group of data was formed from the OHSUMED corpus. OHSUMED is a subset of MEDLINE, an online database of medical literature, and comprises a collection of 348,566 medical references from medical journals covering a period from 1987 to 1991. We obtained a subset of the OHSUMED collection which consists of all references from the year 1991. From this we created two new, 1000 documents dataset: NervImmuno with 500 documents from each of the Nervous and Immunological diseases categories and BacterialParasitic with 500 documents from each of the Bacterial and Parasitic diseases categories.

The last group of data was formed from the Reuters21578 corpus which is a collection of newswire stories. The corpus contains 5 category sets: EXCHANGES, ORGS, PEOPLE, PLACES and TOPICS. Each category set contains a number of different categories. The TOPICS category set contains economic subject categories e.g. Cocoa, Trade, Grain and Money-fx. From the TOPICS categories we created two, 1000 document datasets: InteresetFX which contains 500 documents from each of the Interest and Money-fx categories and the TradeGrain dataset which contains all 486 documents from the Trade category and 514 documents from the Grain category. The Interest and Money-fx categories contain news wires on interest rates and foreign exchange rates respectively. The Trade and Grain categories on the other had contain reports on miscellaneous international trade and international trade in grain respectively.

Summary of the 5 datasets is listed in Table 1. Vocabulary size and average document length were calculated after documents were processed for stopwords and rare terms. Also, average document length and vocabulary size are counts of unique terms in the documents and corpus respectively. Notice that all datasets have a comparable vocabulary size with Hardware having the largest at 3,824. TradeGrain has the largest average document length of 78 unique terms whilst NervImmuno is smallest with 57 unique terms. We also obtained a measure of inter-class vocabulary overlap by partitioning datasets by class, obtaining the centroid of each partition and calculating the Jaccard similarity of the centroids. Inter-class vocabulary overlap ranges from 0.72 for the Hardware dataset to 0.90 for the InterestFx dataset.

Table 1. Characteristics of datasets

	Vocabulary Size	No of Documents	Ave. Doc. Length	Vocabulary Overlap
Hardware	3824	1000	64	0.72
NervImmuno	3213	1000	57	0.79
BacterialParasitic	3047	1000	58	0.84
InterestFX	3172	1000	66	0.90
TradeGrain	3177	1000	78	0.75

5.2 Results

Overall results show that incorporating term similarity and weighting has generally performed better than Base and Svm on all datasets. When used independently of each other, we can see that weighting on its own achieves higher accuracies than representations with term similarity knowledge. For instance in Table 2, Base (W+), which is the results column for weighting only is consistently better compared to all columns with no weighting (W-). However, using term similarities without weights largely provides gains over baseline performance on 4 out of the 5 datasets. The exception is the Hardware dataset which contains a lot of noisy data including email-style headers and footers which were included in document representations. Such noisy data makes this dataset prone to arbitrary relationships that are likely to lower classification accuracy . But as we can see when term importance is included (all W+ columns) the classification accuracy surpasses both Base (W-) and Base (W+). This indicates that useful relationships are being favoured over arbitrary ones with the introduction of term weights.

Co-occurrence mining within a sentence gives best results in the absence of feature weighting (e.g. compare the W- columns for CCAsent, CCAdoc and LCA). However when weighting is injected to the final representations, CCAsent falls behind CCAdoc and LCA on 4 datasets. Generally, CCAsent tends to extract fewer relationships than CCAdoc and LCA, and thus produces sparser document vectors (average of 11% sparsity compared with 5% for the other two approaches). On the TradeGrain dataset however, CCAsent produced an average document vector sparsity of just 5% and hence achieves comparable performance with CCAdoc and LCA. This suggests our approch of using feature weights works to distinguish between spurious and useful term relationships in document representations and thus works best when a larger number of term relationships are included in document representations.

In Table 2 the best observed results in each row are displayed in bold. Results of paired t-tests show that LCA (W+), CCAsent (W+) and CCAdoc (W+) all performed significantly better than Base on all 5 datasets. In contrast, introducing features weights produces significant gains over Base only on the BacterialParasitic and NervImmuno datasets. It is also on these datasets that CCAsent, performed significantly better than Base. The failure of each

Table 2. Accuracies on the different datasets with (W+) and without (W-) term weights

	Svm	Base		CCAdoc		CCAsent		LCA	
		W-	W+	W-	W+	W-	W+	W-	W+
Hardware	91.4	86.9	87.3	83.4	**94.1**	84.3	90.8	86.3	92.3
NervImmuno	85.6	81.5	86.8	82.8	88.0	84.1	86.1	84.0	**88.5**
BacterialParasitic	78.4	75.0	78.0	76.7	**82.3**	77.9	78.2	73.9	79.7
InterestFX	62.4	67.1	69.2	67.2	69.6	68.5	69.7	67.0	**73.0**
TradeGrain	95.6	93.7	95.5	94.4	95.7	94.4	**96.5**	93.9	95.89

Table 3. Accuracy of SVM when combined with term similarities and weighting

	Svm	CCADoc(W+)	CCASent(W+)	LCA(W+)
Hardware	91.4	**92.1**	89.5	91.5
NervImmuno	85.6	85.8	86.8	**87.1**
BacterialParasitic	78.4	80.6	80.0	**81.9**
InterestFX	62.4	71.4	76.4	**76.6**
TradeGrain	95.6	94.9	**96.2**	94.5

technique to consistently produce significant gains over Base on its own shows that the combined contribution of term similarity and weighting work well to improve text classification accuracy. In comparison with Svm; all three algorithms perform significantly better on the InterestFX dataset; CCAdoc (W+) and LCA (W+) perform significantly better on NervImmuno; and CCAdoc (W+) is significantly better on BacterialParasitic and Hardware while no significant improvement over Svm was observed on the TradeGrain dataset. This indicates that our algorithms are significantly better than Svm on most domains and at least comparable with Svm on all domains.

We also applied SVM on the new document representations generated using CCAdoc (W+) , CCAsent (W+) and LCA (W+) (see Table 3).

The best observed results in each row of table 3 are again displayed in bold. Significant improvement over SVM baseline performance using all three techniques is observed only on the InterestFx dataset, which is also the most difficult classification task. SVM also seems to benefit from LCA (W+) representations whilst there are no obvious gains from using CCAdoc (W+) and CCAsent (W+).

5.3 Discussion

When examining co-occurrence based term similarities a level of non-intuitive term relationships are expected. This is not to say that the non intuitive relationships are useless, especially with text classification, such relationships might be indicative of patterns in the underlying documents that further help to discriminate between classes.

Table 4 summarises the strength of similarity knowledge discovered by our algorithm. For example the value 82 in column $sim < 0.01$ for Hardware means that 82% of the term pairs have a similarity of less than 0.01 (approx. zero similarity) as extracted by CCAdoc. Note that CCAsent is a more restrictive version of CCAdoc and thus discovers fewer term similarities than CCAdoc. LCA on the other hand, because it takes into account second order co-occurrences between term pairs, discovers more term similarities. For example on the InterestFx dataset, CCADoc fails to discover similarity (i.e. $sim < 0.01$) between 74% of the term pairs, this number increases to 93% for CCAsent while LCA fails to discover similarities between just 1.5% of term pairs. However much of the additional term similarities discovered by LCA are weak (most have similarity value < 0.1). The number of term pairs discovered

Table 4. Distribution of Term Pair Similarity values in Percentages

		$sim > 0.99$	$sim >= 0.5$	$sim >= 0.3$	$sim < 0.01$
HARDWARE	CCADOC	0.010	0.10	0.45	82
	CCASENT	0.004	0.03	0.05	97
	LCA	0.002	0.13	0.41	13
NERVIMMUNO	CCADOC	0.0004	0.019	0.16	85
	CCASENT	0.0002	0.004	0.02	95
	LCA	0.0001	0.022	0.27	2.9
BACTERIALPARASITIC	CCADOC	0.0004	0.019	0.16	85
	CCASENT	0.0002	0.004	0.02	95
	LCA	0.0001	0.023	0.37	2.1
INTERESTFX	CCADOC	0.009	0.16	0.87	74
	CCASENT	0.003	0.18	0.06	93
	LCA	0.001	0.16	2.75	1.5
TRADEGRAIN	CCADOC	0.0030	0.07	0.430	78
	CCASENT	0.0012	0.01	0.037	94
	LCA	0.0007	0.18	1.540	1.61

Table 5. Term Similarities for 5 of top ranked terms

Dataset	CCADOC	CCASENT	LCA
BACTPAR	virus ; human = 0.66	virus ; immunodeficiency = 0.52	virus ; immunodeficiency = 0.85
	hiv ; immunodeficiency = 0.66	hiv ; infection = 0.31	hiv ; infection = 0.67
	bacteria ; translocation = 0.33	bacteria ; translocation = 0.2	bacteria ; anaerobic = 0.44
	antibiotics ; therapy = 0.34	antibiotics ; therapy = 0.23	antibiotics ; therapy = 0.56
	viral ; virus = 0.39	viral ; bacterial = 0.15	viral ; infection = 0.47
NRVIMM	cells ; cd = 0.47	cells ; cd = 0.24	cells ; mononuclear = 0.65
	antibody ; monoclonal = 0.49	antibody ; monoclonal = 0.33	antibody ; monoclonal = 0.68
	aids ; hiv = 0.56	aids ; acquired = 0.22	aids ; arc = 0.69
	cd ; lymphocytes = 0.5	cd ; lymphocytes = 0.33	cd ; lymphocytes = 0.67
	pain ; relief = 0.31	pain ; relief = 0.21	pain ; relief = 0.55
TRDGRN	trade ; march = 0.64	trade ; deficit = 0.3	trade ; war = 0.68
	tonnes ; wheat = 0.57	tonnes ; mln = 0.41	tonnes ; mln = 0.73
	wheat ; tonnes = 0.57	wheat ; tonnes = 0.33	wheat ; soft = 0.72
	grain ; agriculture = 0.48	grain ; coarse = 0.26	grain ; coarse = 0.75
	billion ; dlrs = 0.67	billion ; dlrs = 0.59	billion ; dlrs = 0.87
INTFX	dollar ; yen = 0.61	dollar ; yen = 0.39	dollar ; yen = 0.77
	currency ; exchange = 0.62	currency ; stabilize = 0.27	currency ; stabilize = 0.68
	exchange ; foreign = 0.68	exchange ; foreign = 0.47	exchange ; foreign = 0.79
	yen ; japan = 0.69	yen ; dollar = 0.39	yen ; dollar = 0.77
	prime ; effective = 0.45	prime ; raises = 0.3	prime ; raises = 0.76
HARD	windows ; dos = 0.45	windows ; dos = 0.22	windows ; dos = 0.47
	apple ; mac = 0.41	apple ; ergo- = 0.16	apple ; welded = 0.55
	dos ; windows = 0.45	dos ; gosh = 0.22	dos ; aspi = 0.56
	ide ; wlsmith = 0.52	ide ; scsi = 0.27	ide ; scsi = 0.61
	jumper ; settings = 0.35	jumper ; sl = 0.22	jumper ; circuit = 0.51

by LCA with similarity value greater than 0.5 is generaly comparable with that of CCADOC on most datasets.

A closer look at the most similar terms to five top ranking terms from each domain reveals interesting relationships between CCADOC, CCASENT and LCA. (see Table 5). For example all three approaches agree on 'therapy' as the most similar term to 'antibiotics' but disagree on 'viral' with CCAdoc, CCAsent and LCA extracting the terms 'virus', 'bacterial' and 'infection' respectively. Notice also that because CCAsent and LCA use the same context size, they tend to agree on many of the extracted term relationships. However due to the lexical co-occurrence information used by LCA, the term similarity values generally tend to be higher (e.g. hiv and infection have similarity 0.31 using CCAsent and 0.67 using LCA). At the same time LCA is not liberal with assigning maximal similarity because term pairs need to have identical lexical co-occurrents to attain this.

6 Conclusion

The main contribution of this paper is a framework that combines term similarity knowledge with term weighting to represent textual content. We have also discussed how three different approaches for extracting term-similarity knowledge from corpus co-occurrence can be utilised within this framework. We have demonstrated how the common approach of using matrix multiplications for co-occurrence based term similarity is sensitive to document frequency of terms and how that can be addressed through normalisation.

Results from a comparative study on text classification tasks clearly demonstrate the synergistic effect of term-similarity and weighting compared with using either independently of the other. Our results also show kNN with our augmented representations to outperform Svm on a majority of the datasets. Although we have tested our framework on text classification tasks it is worth noting that none of the proposed term-similarity mining approaches were specifically adapted for supervised tasks. Therefore it is quite likely that further benefits are likely to be realised if co-occurrence mining had been biased by class knowledge.

In future work we plan to conduct a more comprehensive study involving both extrospective and introspective sources of knowledge for term-term similarity computation. Our initial findings on document versus sentence context for co-occurrence mining remains inconclusive. Therefore it will be very useful to study this further on many more datasets and also to include syntactic analysis and grammatical dependencies in similarity estimation.

References

1. Bollegala, D., Matsuo, Y., Ishizuka, M.: Measuring semantic similarity between words using web search engines. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 757–766. ACM, New York (2007)
2. Boyd-graber, J., Fellbaum, C., Osherson, D., Schapire, R.: Adding dense, weighted connections to wordnet. In: Proceedings of the Third International WordNet Conference (2006)
3. Brank, J., Milic-Frayling, N.: A framework for characterzing feature weighting and selection methods in text classification. Tech. rep., Microsoft Research (January 2005)

4. Chakraborti, S., Lothian, R., Wiratunga, N., Orecchioni, A., Watt, S.: Fast case retrieval nets for textual data. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 400–414. Springer, Heidelberg (2006)
5. Chakraborti, S., Wiratunga, N., Lothian, R., Watt, S.: Acquiring word similarities with higher order association mining. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 61–76. Springer, Heidelberg (2007)
6. Cilibrasi, R.L., Vitanyi, P.M.B.: The google similarity distance. IEEE Trans. on Knowl. and Data Eng. 19, 370–383 (2007)
7. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990)
8. Gabrilovich, E., Markovitch, S.: Wikipedia-based semantic interpretation for natural language processing. J. Artif. Int. Res. 34, 443–498 (2009)
9. Gracia, J., Mena, E.: Web-based measure of semantic relatedness. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 136–150. Springer, Heidelberg (2008)
10. Jiang, J., Conrath, D.: Semantic similarity based on corpus statistics and lexical taxonomy. In: Proc. of the Int'l. Conf. on Research in Computational Linguistics, pp. 19–33 (1997)
11. Kontostathis, A., Pottenger, W.M.: A framework for understanding latent semantic indexing (lsi) performance. Information Processing and Management 42(1), 56–73 (2006)
12. Lin, D.: An information-theoretic definition of similarity. In: Proc. of the 15th Int'l. Conf. on Machine Learning, pp. 296–304 (1998)
13. Miller, G.A.: Wordnet: A lexical database for english. Communications of the ACM 38, 39–41 (1995)
14. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 1, pp. 448–453 (1995)
15. Sahlgren, M.: An introduction to random indexing. In: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005 (2005)
16. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18, 613–620 (1975)
17. Schütze, H., Pedersen, J.O.: A cooccurrence-based thesaurus and two applications to information retrieval. Inf. Process. Manage. 33, 307–318 (1997)
18. Turney, P.D., Pantel, P.: From frequency to meaning: vector space models of semantics. J. Artif. Int. Res. 37, 141–188 (2010)
19. Wiratunga, N., Koychev, I., Massie, S.: Feature selection and generalisation for retrieval of textual cases. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 806–820. Springer, Heidelberg (2004)
20. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: Proc. of the 32nd Annual Meeting on Association for Computational Linguistics, pp. 133–138 (1994)
21. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proceedings of the Fourteenth International Conference on Machine Learning, ICML 1997, pp. 412–420 (1997)

Fast Subgraph Isomorphism Detection for Graph-Based Retrieval

Markus Weber[1,2], Christoph Langenhan[3], Thomas Roth-Berghofer[4,1], Marcus Liwicki[1], Andreas Dengel[1,2], and Frank Petzold[3]

[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, Germany
[3] Chair of Architectural Informatics, Faculty of Architecture, Technical University of Munich, Arcisstrasse 21, 80333 Munich, Germany
[4] Explanation-aware Computing Systems, Institute of Computer Science, University of Hildesheim, Marienburger Platz 22, 31141 Hildesheim, Germany
firstname.lastname@dfki.de, lastname@tum.de

Abstract. In this paper we present a method for a graph-based retrieval and its application in architectural floor plan retrieval. The proposed method is an extension of a well-known method for subgraph matching. This extension significantly reduces the storage amount and indexing time for graphs where the nodes are labeled with a rather small amount of different classes. In order to reduce the number of possible permutations, a weight function for labeled graphs is introduced and a well-founded total order is defined on the weights of the labels. Inversions which violate the order are not allowed. A computational complexity analysis of the new preprocessing is given and its completeness is proven. Furthermore, in a number of practical experiments with randomly generated graphs the improvement of the new approach is shown. In experiments performed on random sample graphs, the number of permutations has been decreased to a fraction of 10^{-18} in average compared to the original approach by Messmer. This makes indexing of larger graphs feasible, allowing for fast detection of subgraphs.

Keywords: Architecture, graph theory, retrieval.

1 Introduction

A graph is a mathematical representation that consists of vertexes and edges and can be applied to representations that capture relationships between any two elements. Thus they offer a number of advantages over traditional feature vector approaches [1]. Unlabeled graphs have no fixed labels, thus the only applicable similarity methods are ones that search for identical subgraphs. This subgraph isomorphism problem is a well-known problem in literature and is known to be

NP-complete [2]. Similarity assessment for labeled graphs in general is domain-specific. Although being polynomial, graph representations do have a significant computational cost. Fortunately, there are a number of methods and techniques aimed at reducing this problem [1].

However, subgraph isomorphism still is a powerful general similarity measure which also could be applied without any specific domain knowledge. In order to reduce computational cost in subgraph isomorphism, index based approaches have been introduced. Such a method has been proposed by Messmer et al. [3]. It builds an index using the permutated adjacency matrix of the graph. The real-time search is then based on a tree. While the method has shown to be effective for a reference set with small graphs, it is unfeasible for graphs with more than 19 vertices.

In this paper we propose a method to overcome this problem. Assuming that the number of labels for the nodes is relatively small, we introduce a well-founded total order and apply this during index building. This optimization decreases the amount of possible permutations dramatically and allows building indexes of graphs with even more than 30 vertices.

The method has been applied in the architectural floor plan retrieval. In [4] a graph-based structure for representing the spatio-relational content of a floor plan has been introduced. The case-based design approach for floor plan retrieval described in [5] deals with semantic and topological information of spatial configurations not regarding procedural information. The topological and functional inner structure, the orientation of spaces as well as the urban integration, and the relation of buildings to each other is vital for a retrieval of valuable reference to support architects during the early stages of designing a building. Thus a digital fingerprint was proposed in [6] that contains a clear spatial description of an architectural dataset. The proposed system offers a computational approach to extract a few characteristic and prominent features of a floor plan which are then used to generate a digital fingerprint. In the paper, we examine the development of graph-based methods to provide a sketch-based submission and retrieval system for publishing and researching building layouts.

The rest of this paper is organized as follows. First, Section 2 gives an overview over related work. Subsequently, Section 3 introduces definitions and notations which are used and Section 3.1 describes the new preprocessing step and Section 3.2 the modified retrieval algorithm. Next, Section 4 will show that the number of computational steps will be significantly decreased and Section 5 discusses the application in the architectural domain. Finally, Section 6 concludes the work.

2 Related Work

In [7], Goa et al. give a survey of work done in the area of graph matching. The focus in the survey is the calculation of error-tolerant graph-matching; where calculating a graph edit distance (GED) is an important way. Mainly the GED algorithms described are categories into algorithms working on attributed or non-attributed graphs. Ullman's method [8] for subgraph matching is known as one of the fastest methods. The algorithm attains efficiency by inferentially eliminating successor nodes in the tree search.

Bunke [9,1] discussed several approaches in graph-matching. One way to cope with error-tolerant subgraph matching is using the maximum common subgraph as a similarity measure. Another way is by applying graph edit costs, an extension of the well-known string edit distances. A further group of suboptimal methods are approximate methods. They are based on neural networks, such as Hopfield networks, Kohonen maps or Potts MFT neural nets. Finally, genetic algorithms, the usage of Eigenvalues, and linear programming are applied.

Graph matching is challenging in presence of large databases [10,1]. Consequently, methods for preprocessing or indexing are essential. Preprocessing can be performed by graph filtering or concept clustering. The main idea of the graph filtering is to use simple features to reduce to number of feasible candidates. Another concept clustering is used for grouping similar graphs. In principle, given a similarity (or dissimilarity) measure, such as GED [11], any clustering algorithm can be applied. Graph indexing can be performed by the use of decision trees.

Messmer and Bunke [3] proposed a decision tree approach for indexing the graphs. They are using the permutated adjacency matrix of a graph to build a decision tree. This technique is quite efficient during run time, as a decision tree is generated beforehand which contains all model graphs. However, the method has to determine all permutations of the adjacency matrices of the search graphs. Thus, as discussed in their experiments, the method is practically limited to graphs with a maximum of 19 vertices. The main contribution of this paper is to improve the method of Messmer and Bunke for special graphs by modifying the index building process.

As topologies are crucial to describe the relation of spaces, graphs are widely used to store information about buildings. The EsQUIsE project focuses on the the early design stages of buildings to support architects with a pen-based interface to sketch ideas and simulate certain aspects of the design. Juchmes et al. [12] proposes a floor plan like input strategy to use adjacency, dimension and orientation of space to build a 3D model and a graph structure. The PROBADO-framework intends to integrate multimedia content to digital libraries. Apart of indexing methods for music and 3D shape retrieval [13] a room connectivity graph [14] was proposed. To build up the graph structure of the spatial relation of a non semantic 3D-Model the storeys are detected, the room per storey, doors and windows determined. Using a sketch-based retrieval interface 3D-models will be retrieved by inputting schematic topologies of a spatial configuration.

3 Definitions and Notations

Basic definitions that are used throughout the paper are a labeled graph $G = (V, E, L_v, L_e, \mu, v)$ with its common representation as an adjacency matrix M:

Definition 1. *An* ***adjacency matrix*** *is $n \times n$ matrix M.*

$$M = (m_{ij}), i, j = 1, ..., n, \text{ where}$$

$$m_{ii} = \mu(v_i)$$

and

$$m_{ij} = \upsilon((v_i, v_j)) \text{ for } i \neq j.$$

Thus a graph G can also be represented by its adjacency matrix M. But the matrix M is not unique for a graph G. If M represents G, than any permutation of M is also a valid representation of G.

Definition 2. *A $n \times n$ matrix $P = (p_{ij})$ is called a* ***permutation matrix*** *if*

1. $p_{ij} \in 0, 1$ *for i,j = 1, ..., n*
2. $\sum_{i=1}^{n} p_{ij} = 1$ *for j = 1, ..., n*
3. $\sum_{i=1}^{n} p_{ij} = 1$ *for i = 1, ..., n*

Furthermore, a so called row-column representation is given. Each matrix can be represented by its row-column elements a_i, where a_i is a vector of the form

$$a_i = (m_{1i}, m_{2i}, ..., m_{ii}, m_{i(i-1)}, ..., m_{ii}).$$

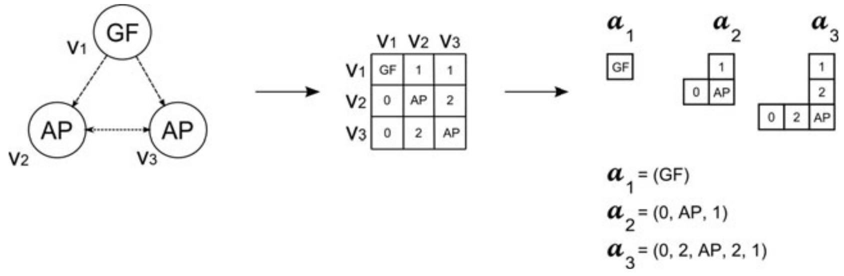


Fig. 1. The row-column representation of an adjacency matrix

In order to compare two adjacency matrices with different dimensions, a notation $S_{k,m}(M)$ is required which reduces the dimension of the matrix with the higher dimension.

Definition 3. *Let $M = (m_{ij})$ be a $n \times n$ matrix. Then $S_{k,m}$ denotes the $k \times m$ matrix that is obtained from M by deleting rows k + 1,..., n and columns j = 1,...,m where $k, m \leq n$. That is, $S_{k,m}(M) = (m_{ij})$; i = 1,...,k and j = 1,...,m.*

Besides, definitions for orders on sets are needed.

Definition 4. *A* ***total order*** *is a binary relation $\leq$ over a set P which is transitive, anti-symmetric, and total, thus for all a, b and c in P, it holds that:*

– *if* $a \leq b$ *and* $b \leq a$ *then* $a = b$ *(anti-symmetry);*
– *if* $a \leq b$ *and* $b \leq c$ *then* $a \leq c$ *(transitivity);*
– $a \leq b$ *or* $b \leq a$ *(totality).*

Definition 5. *A partial or total order* $\leq$ *over a set* X *is* ***well-founded****, iff* $(\forall\ Y \subseteq X\ :\ Y \neq \emptyset \rightarrow (\exists y \in Y\ :\ y$ *minimal in* Y *in respect of* $\leq))$.

Additionally, a weight function is defined, which assigns a weight to a label of a graph.

Definition 6. *The* ***weight function*** σ *is defined as:* $\sigma\ :\ L_v\ \rightarrow\ \mathbb{N}$.

Using the weight function, a well-founded total order is defined on the labels of graph, for example $\sigma(L_1) < \sigma(L_2) < \sigma(L_3) < \sigma(L_4)$. Thus the labeled graph can be extended in its definition.

Definition 7. *A* ***labeled graph*** *consists of a 7-tuple,* $G = (V, E, L_v, L_e, \mu, \upsilon, \sigma)$*, where*

– V *is a set of vertices,*
– $E \subseteq V \times V$ *is a set of edges,*
– L_v *is a set of labels for the vertices,*
– L_e *is a set of labels for the edges,*
– $\mu : V \rightarrow L_v$ *is a function which assigns a label to the vertices,*
– $\upsilon : E \rightarrow L_e$ *is a function which assigns a label to the edges,*
– $\sigma : L_v\ \rightarrow\ \mathbb{N}$ *is a function which assigns a weight to the label of the vertices,*

and a binary relation $\leq$ *which defines a well-founded total order on the weights of the labels:*

$$\forall x, y \in L_v\ :\ \sigma(x) \leq \sigma(y)\ \vee\ \sigma(y) \leq \sigma(x)$$

3.1 Algorithm

The algorithm for subgraph matching is based on the algorithm proposed by Messmer and Bunke [3], which is a decision tree approach. Their basic assumption is that several graphs are known a priori and the query graph is just known during run time. Messmer's method computes all possible permutations of the adjacency matrices and transforms them into a decision tree. At run time, the adjacency matrix of the query graph is used to traverse the decision tree and find a subgraph which is identical.

Let G_1 and G_2 be graphs with their adjacency matrices M_1 and M_2 of dimension $m \times m$ and $n \times n$ and $m \leq n$. The problem of finding a subgraph isomorphism from G_1 to G_2 is equivalent to finding a permutation matrix, so the subgraph isomorphism is given, iff there is a $n \times n$ permutation matrix P such that

$$M_1 = S_{m,m}(PM_2P).$$

Let $G = (V, E, L_v, L_e, \mu, \upsilon)$ be a graph from the graph database and M the corresponding $n \times n$ adjacency matrix and $A(G)$ the set of permuted matrices.

Thus the total number of permutations is $|A(G)| = n!$, where n is the dimension of the permutation matrix, respectively the number of vertices.

Now, let $Q = (V, E, L_v, L_e, \mu, \upsilon)$ be a query graph and M' the corresponding $m \times m$ adjacency matrix, with $m \leq n$. So, if a matrix $M_P \in A(G)$ exists, such that $M' = S_{m,m}(M_P)$, the permutation matrix P which corresponds to M_P represents a subgraph isomorphism from Q to G, i.e

$$M' = S_{m,m}(M_P) = S_{m,m}(PMP^T).$$

Messmer proposed to arrange the set $A(G)$ in a decision tree, such that each matrix in $A(G)$ is classified by the decision tree. However, this approach has one major drawback. For building the decision tree, all permutations of the adjacency matrix have to be considered. Thus, for graphs with more than 19 vertices the number of possible permutations becomes intractable. In order to overcome this issue, the possibilities of permutations have to be reduced. One way is to define constraints for the permutations. Therefore a weight function σ (see Definition 6) is introduced which assigns a weight for each vertex according to its label. Thus each label has a unique weight and a well-founded total order (see Definition 4 and Definition 5) on the set of labels which reduces the number of allowed inversion for the adjacency matrix. Figure 2 illustrates an example for the modified matrices and the corresponding decision tree. Let us consider the following weights for the nodes:

$$\begin{aligned} L_v =& \{L_1, L_2, L_3\} \\ \sigma(L_1) =& 1, \\ \sigma(L_2) =& 2, \\ \sigma(L_3) =& 3. \end{aligned}$$

Each inversion that violates the ordering is not allowed. Thus just the vertices which have the same label, respectively the same weights, have to be permuted and if the labels have a different weight, just the variations are required. Given the graph G, the following labels are assigned to the vertices,

$$\begin{aligned} V =& \{v_1, v_2, v_3\} \\ \mu(v_1) =& L_1, \\ \mu(v_2) =& L_2, \\ \mu(v_3) =& L_2. \end{aligned}$$

Hence, the only valid permutations are:

1. $\sigma(\mu(v_1)) \leq \sigma(\mu(v_2)) \leq \sigma(\mu(v_3))$
2. $\sigma(\mu(v_1)) \leq \sigma(\mu(v_3)) \leq \sigma(\mu(v_2))$
3. $\sigma(\mu(v_2)) \leq \sigma(\mu(v_3))$
4. $\sigma(\mu(v_3)) \leq \sigma(\mu(v_2))$

Let $VA(G)$ be the set of all valid permutations. The decision tree is built according to the row-column elements of the adjacency matrices $M_P \in VA(G)$

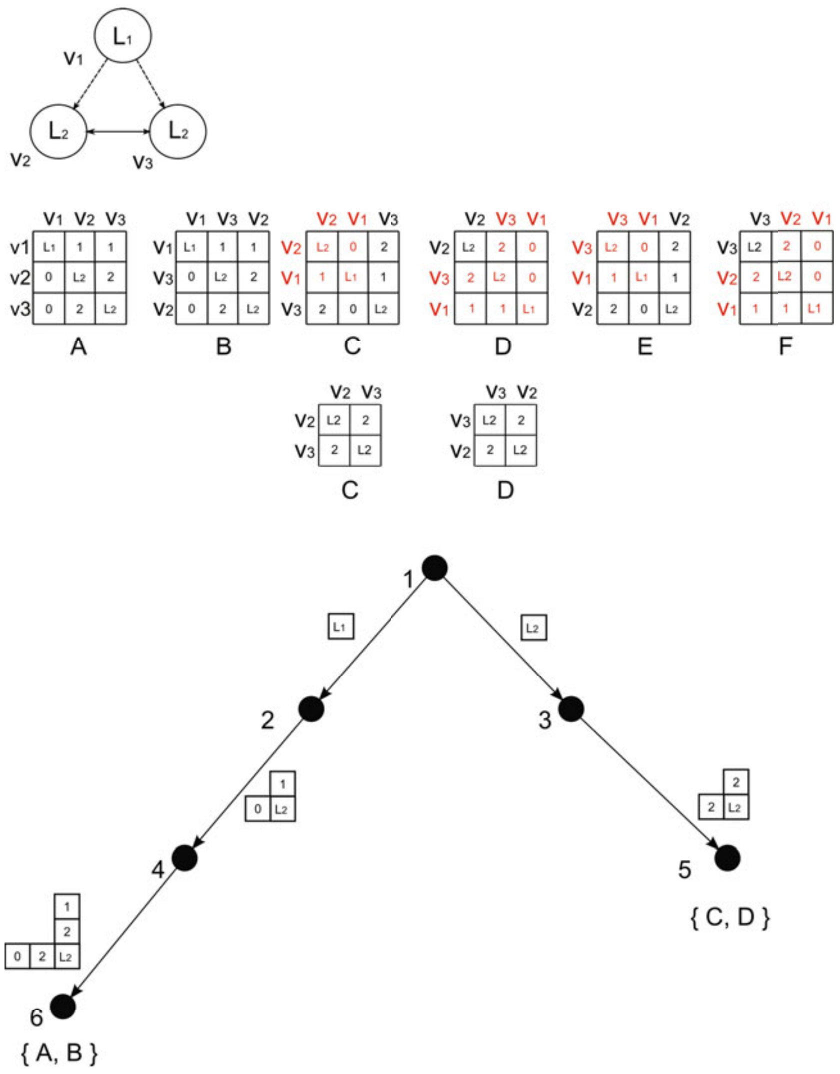


Fig. 2. Modified decision tree for adjacency matrices

and should cover all graphs from the database. So, let R be the set of semantics $R = \{G_1, G_2, ..., G_n\}$, where n is the total number of graphs in the repository, with their sets of corresponding adjacency matrices $VA(G_1)$, $VA(G_2)$, ..., $VA(G_n)$. Now, each set of adjacency matrices has to be added to the decision tree.

3.2 Retrieval Algorithm

An obvious advantage of the method is that the whole process can be done a priori. The decision tree acts as an index for subgraphs. So, during run time the decision tree has been loaded into memory and by traversing the decision tree, the corresponding subgraph matrices are classified. For the query graph Q the adjacency matrix M is determined following the constraints defined by ordering. Afterwards the adjacency matrix is split up into row-column vectors a_i. For each level i the corresponding row-column vector a_i is used to find the next node in the decision tree using an index structure. The pseudo code of Algorithm 1 displays how the results are retrieved by traversing the tree:

Algorithm 1. RETRIEVAL(Q = (V, E, μ, υ, σ, L_v, L_e), Tree)

```
Require: Unsorted set $V$ of vertices, $\mu$ labeling function, $\sigma$ weight function
 1: sort(Q, $L_v$, $\mu$, $\sigma$)
Ensure: Vertices V are sorted according to the defined order.
 2: Let $R$ be an empty sorted set which will contain all results sorted by the similarity
    value $Sim$.
 3: Determine adjacency matrix $M$ from graph $Q$.
 4: Determine row-column list $RCL$ from $M$.
 5: for $i \leftarrow 1$ to $|V| - 1$ do
 6:    for $j \leftarrow 1$ to $|RCL|$ do
 7:       Find best match for row-column vector $a_i \in RCL$ in tree at $level_i$.
 8:       Update $R$ and $Sim$.
 9:    end for
10:    Remove element $V_i$ from $V$.
11: end for
12: return R.
```

3.3 Proof of Completeness

For the proposed modified algorithm it has to be proven that the algorithm finds all solutions. The algorithm elaborated in the previous section reduces the number of valid permutations. So, it has to be shown that by leaving out permutations, no valid solution is lost.

Let $G = (V, E, L_v, L_e, \mu, \upsilon, \sigma)$ be a well-founded total ordered graph and let $A(G)$ be the set which contains all valid permutations of the graph's adjacency matrices. To be complete, the algorithm must find a solution if one exists; otherwise, it correctly reports that no solution is possible. Thus if every possible valid subgraph $S \subseteq G$, where the vertices of S fulfill the order, every corresponding adjacency matrix M has to be an element of the set $A(G)$, $M \in A(G)$.

For this reason to proof that the algorithm is complete it has to be shown that the algorithm generates all valid subgraphs $S \subseteq G$. Therefore the pseudo code of Algorithm 2 shows how the index is built. Algorithm 3 and Algorithm 4 are helping functions for calculating all variations of the set of vertices in an interval. The generation of the index starts with an unsorted set of vertices. By sorting the vertices with their associated labels using the well-founded total order, the set is ordered according to the weights of the labels.

Algorithm 2. BUILD_INDEX($G = (V, E, L_v, L_e, \mu, \upsilon, \sigma)$, Tree)

```
Require: Unsorted set V of vertices, $\mu$ labeling function, $\sigma$ weight function.
 1: sort(V, $L_v$, $\mu$, $\sigma$)
Ensure: Vertices V are sorted according to the defined order.
 2: Let $O$ be an empty list.
 3: for all $l_i \in L_V$ do
 4:    Let $\{v_a, \dots, v_b\}$ contain all $v$ with $\mu(v) = l_i$
 5:    $O_i \leftarrow VARIATIONS(\{v_a, \dots, v_b\})$
 6: end for
 7: Let $AG \leftarrow O_1 \ \times \ \dots \ \times O_{|Lv|}$.
 8: for all $m_i$ in $AG$ do
 9:    Add row column vector for sequence of $m_i$ to Tree.
10: end for
```

Algorithm 3. $PERMUTE(V, begin, end, R)$

```
Require: Sorted set V of vertices and $begin < end$, with $V_{end-1}$ being last the element.
 1: Adding sequence of vertices $V$ to $R$.
 2: for $i \leftarrow end - 2$ to $begin$ do
 3:    for $j \leftarrow i + 1$ to $end - 1$ do
 4:       Swapping position i and j in V.
 5:       Call $PERMUTE(V, i + 1, end, R)$.
 6:    end for
 7:    Call $ROTATE(V, i + 1, end, R)$.
 8: end for
```

Now, the algorithm iterates over all intervals of vertices $\{v_a, ..., v_b\}$ where the labels have the same weights, $\sigma(\mu(v_a)) == \sigma(\mu(v_b))$. For each interval $\{v_a, \dots, v_b\}_i$ all variations with respect to the order have to be determined. These variations are computed in Algorithm 5, by determining all combinations of the interval $\{v_a, \dots, v_b\}_i$ including the empty set and calculating all permutations for these combinations. Algorithm 3 and Algorithm 4 realize the algorithm proposed by Rosen [15] which computes all permutations for a defined interval. It has been proven that Rosen's algorithm computes all permutations. In combinatorial mathematics, a k-variation of a finite set S is a subset of k distinct elements of S. For each chosen variation of k elements, where k is $L_{interval} =$ length of interval; $k = 1 \dots L_{interval}$, again all permutations have to be considered. Now, assuming there would be a valid subgraph $Q = (V', E', L'_v, L'_e, \mu, \upsilon, \sigma)$, respectively the corresponding adjacency matrix A which depends on the alignment of the vertices. To be a valid subgraph, V' has to be a subset of V, $V' \subseteq V$. Furthermore the alignment of the vertices V' according to their labels has to fulfill the defined order, $\sigma(\mu(v_i)) \leq \sigma(\mu(v_{i+1}))$. For the alignment the intervals $\{v'_a, \dots, v'_b\} \in V'$ where the weights of the labels have the same value $\sigma(\mu(v'_a))) == \sigma(\mu(v'_b)$ are important as they can vary. The Algorithm 5 determines all variations for intervals with the same weights for

Algorithm 4. $ROTATE(V, begin, end, R)$

```
1: Let $temp \leftarrow V_{end-1}$.
2: Shift elements in V in from position $begin$ to $end - 1$ one position right
3: Set $V_{begin} \leftarrow temp$.
4: Add sequence of vertices $V$ to $R$.
```

Algorithm 5. VARIATIONS($\{v_a, \ldots, v_b\}$)

```
Require: Sorted set $V = \{v_a, \ldots, v_b\}$ of vertices, $a \leq b$.
1: Let $O$ be an empty list.
2: Determine all combinations $C$ for $\{v_a, \ldots, v_b\}$ including the empty set.
3: for all $c$ in $C$ do
4:     Call $PERMUTE(c, 0, |c|, O)$.
5: end for
6: Return O.
```

labels, thus the alignment $\{v'_a, \ldots, v'_b\}$ is considered. This holds for each interval, thus algorithm produces all valid permutations according to the well-founded total order. As the query graph Q also has to fulfill the order, its adjacency matrix A will be an element of $A(G)$, if Q is a valid subgraph of G. Thus, the solution will be found in the decision tree.

3.4 Complexity Analysis

The computational complexity analysis discussed in this section will be based on the following quantities:

$$\begin{aligned}
N &= \text{the number of graphs in the floor plan graphs in the repository,}\\
M &= \text{the maximum number of vertices of a graph in the floor plan repository,}\\
M_v &= \text{the number of vertex labels for a graph in the floor plan repository,}\\
I &= \text{the number of vertices in the query graph,}\\
I_v &= \text{the number of vertex labels.}
\end{aligned}$$

The original algorithm by Messmer [3] as well as the proposed algorithm need an intensive preprocessing, the compilation of the decision tree. Messmer's method has to compute all permutations of the adjacency matrix of the graph, thus the compilation of the decision tree for a graph $G = (V, E, L_v, L_e, \mu, \upsilon, \sigma)$ has a run time complexity of

$$\mathcal{O}(|V|!).$$

For the size of the decision tree Messmer determined the following bounds. The sum of nodes over all the levels (without the root node) is limited to

$$\mathcal{O}(l_v \sum_{k=0}^{M-1} \binom{M}{k} (l_2^e)^k) = \mathcal{O}(l_v (1 + l_e^2)^M),$$

and as the decision tree becomes linearly dependent on the size of the database N, the space complexity of the decision tree is

$$\mathcal{O}(Nl_v(1+l_e^2)^M).$$

The processing time for the new decision tree compilation algorithm is reduced. Algorithm 2 describes the new way to The basic idea of the algorithm is to take all labels L_v with the same weight which occur in the graph and omit their instances. So, considering the mathematical idea of the algorithm an approximation of the run time complexity would be:

$$\prod_{i=1}^{|L_v|}\left(\underbrace{|\{\forall v\in V|\mu(v)=l_i\}|!}_{n_i}+\sum_{j=1}^{n_i-1}\binom{n_i}{j}\cdot j!\right).$$

The first term considers the permutations for all labels with the same weight, denoted by n_i. The second term describes the k-variations. As we have to consider the variations from n_i-1 to 1, the sum is sufficient.

In order to simplify the equation, we can combine the two terms, as the $n_i!$ is equivalent to $\binom{n_i}{n_i}\cdot n_i! = 1\cdot n_i! = n!$, thus we have:

$$\prod_{i=1}^{|L_v|}\left(\sum_{j=1}^{n_i}\binom{n_i}{j}\cdot j!\right).$$

Now, let n_{max} be the maximum number of vertices with the same weight. Then we have the upper bound of

$$\left(\sum_{j=1}^{n_{max}}\binom{n_{max}}{j}\cdot j!\right)^{|L_v|}.$$

A rather imprecise approximation of the sum $\sum_{j=1}^{n_{max}}\binom{n_{max}}{j}\cdot j!$ would be $(n_{max}+1)!$, so the resulting complexity of the algorithm is

$$\mathcal{O}(((n_{max}+1)!)^{|L_v|}),$$

where n_{max} is the maximum number of vertices with the same weight. Thus for the worst case - where all vertices have the same label - $n_{max}=|V|$, $\mathcal{O}((|V|+1)!)$ which would be worse than the method proposed by Messmer and the best case - where all vertices have different labels ($n_{max}=1$) is

$$\mathcal{O}(2^{|V|}).$$

To find the average case of the algorithm the distribution of the labels in the graph has to be considered. This distribution varies according to the represented data.

Table 1. Results of graph experiments (first 10 graphs)

Graph	**Vertices**	**Permutations**	**Permutations**	**Same lables**
	#	**(modified)**	**(original)**	**(max.)**
1	17	3.26×10^6	3.55×10^{14}	5
2	21	3.59×10^9	5.10×10^{19}	8
3	17	1.08×10^7	3.55×10^{14}	5
4	20	2.50×10^8	2.43×10^{18}	6
5	24	1.64×10^{12}	6.20×10^{23}	10
6	17	1.63×10^6	3.55×10^{14}	3
7	21	2.04×10^7	5.10×10^{19}	3
8	30	1.39×10^{12}	2.65×10^{32}	5
9	22	8.01×10^8	1.12×10^{21}	6
⋮	⋮	⋮	⋮	⋮
100	23	1.00×10^9	2.58×10^{22}	6
$\varnothing$	23.05	1.09×10^{13}	3.73×10^{31}	5.23

4 Evaluation

In order to examine run time efficiency of the modified subgraph matching experiments on randomly generated graphs were performed. The modified decision tree algorithm has been implemented in Java using a Java 6 virtual machine. The experiments ran on a Intel Core Duo P8700 (2.53 GHz) CPU with 4 GByte main memory. For the experiment 100 random graphs were generated with 15 to 30 vertices. It compares Messmers's algorithm with its required permutations to the modified algorithm. The permutations for the modified algorithm were determined according to the algorithm discussed in Section 3.1 and the formula in Section 3.4:

$$\prod_{i=1}^{|L_v|} \left(\sum_{j=1}^{n_i} \binom{n_i}{j} \cdot j! \right)$$

and as the original has to be calculate the permutations for all vertices ($|V|!$ permutations). In the second experiment the time to add a graph to the decision tree was measured and again the number of permutations of the adjacency matrix which were added to the decision tree. As the experiment was quite

Table 2. Run time for compiling the decision tree for each graph

Graph	**Vertices**	**Run time**	**Permutations**	**Same lables**
	#	**(minutes)**	**#**	**(max.)**
1	17	1.47	8.19×10^5	4
2	17	8.90	4.17×10^6	5
3	21	45.67	5.32×10^7	4
4	21	10.06	8.19×10^6	3
5	21	38.01	4.09×10^7	3

time-consuming on a desktop machine, only the performance for five smaller graphs was measured. The results of the experiment are listed in Tab. 2. The experiments show that the algorithm significantly reduces the number of permutations (see Tab. 1). Though, the time needed to compile the decision tree is still quite long even for small problem instance, as shown in Tab. 2. However, as the method is designed for an off-line preprocessing and considered to run on a server machine, it is still reasonable for practical applications.

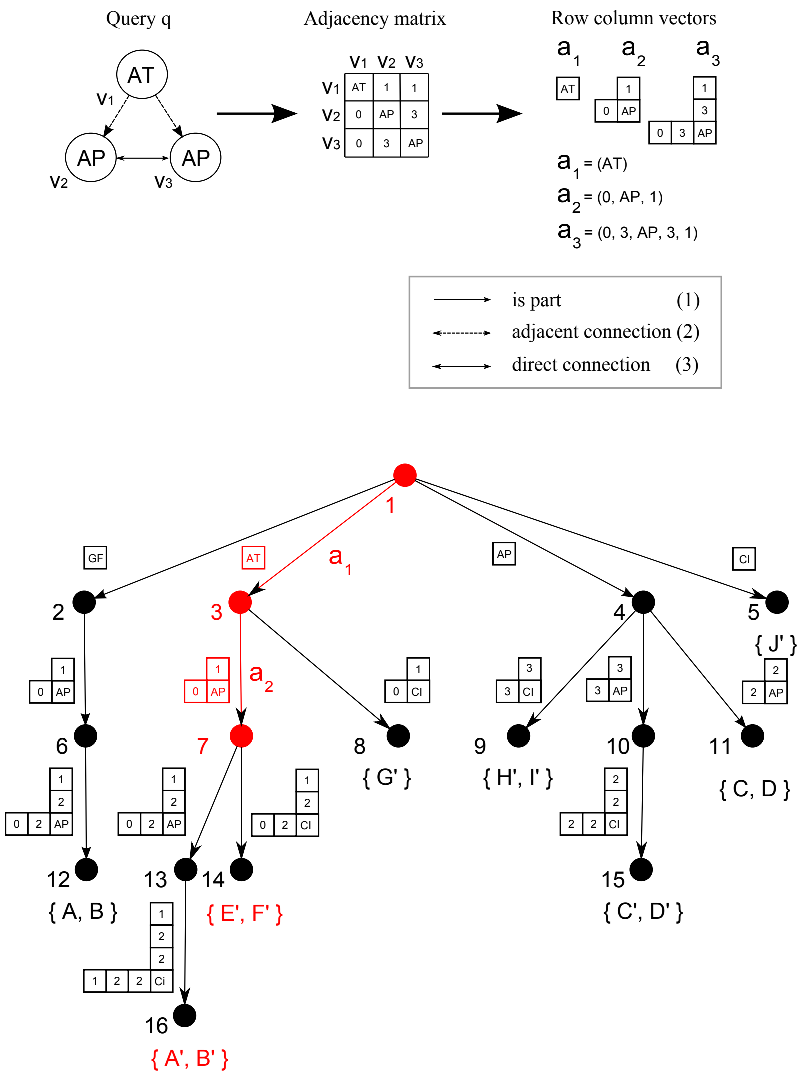


Fig. 3. Retrieval process

5 Application in Architecture

In [4], we already a graph representation for floor plans has been introduced. This representation is used to describe the spatio-relational content of the floor plan. The nodes of the floor plan graph represent functional rooms in a floor plan. Each label is an entity type which has been introduces in [4] and by using the edges the relation between these functional units is described.

So, if the architect is searching for a specific composition of functional rooms, a query is generated which describes this composition. Figure 3 illustrates a query graph Q, where the architect would search for two directly connected apartments (AP) in an attic floor (AT). Each leaf of the decision tree is a possible alignment of the adjacency matrix representing a graph. Thus all leafs beneath the last matching node of the search path, respectively the associated graphs, are added to the result set R. The number of exact row column vector matches N for the query graph Q divided through the number of nodes of the query graph $|V|_q$is used as a simple similarity measure.

$$Sim(q, R) = \frac{N}{|V|_q}$$

In our example the similarity would be $66,\overline{6}\%$ as only two of three comparisons match.

6 Conclusions and Future Work

In this paper an extension for the method of Messmer's subgraph matching has been proposed. The original method is very efficient to perform exact subgraph matching on a large database. However, it has a limitation for the maximum number of vertices. The modification discussed in this paper enables to increase this limit depending on how the vertices are labeled. As the number of permutations in the preprocessing step depends on the vertices with the same labels, an analysis of the data that will be represented in graph is necessary. If there are just a few vertices with the same label, e.g. less than five, even graphs with 30 vertices can be handled. It has been proven that the modification of the method does not affect its completeness.

Noteworthy, the proposed method can be applied in several areas, such as object recognition, matching of 2D or 3D chemical structures, and architectural floor plan retrieval. In the paper we discussed how the method is applied on architectural floor plan retrieval and future work will be to perform experiments on real graph data sets of different domains and research strategies for choosing appropriate weight functions. Furthermore, we plan to extend this method to provide a fast method for error-tolerant graph matching.

References

1. Bunke, H.: Recent developments in graph matching. In: International Conference on Pattern Recognition, vol. 2, p. 2117 (2000)

2. Cook, S.A.: The complexity of theorem-proving procedures. In: STOC 1971: Proceedings of the Third Annual ACM Symposium on Theory of Computing, pp. 151–158. ACM, New York (1971)
3. Messmer, B., Bunke, H.: A decision tree approach to graph and subgraph isomorphism detection. Pattern Recognition 32, 1979–1998 (1999)
4. Weber, M., Langenhan, C., Roth-Berghofer, T., Liwicki, M., Dengel, A., Petzold, F.: a.SCatch: Semantic Structure for Architectural Floor Plan Retrieval. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 510–524. Springer, Heidelberg (2010)
5. Langenhan, C., Weber, M., Liwicki, M., Petzold, F., Dengel, A.: Sketch-based Methods for Researching Building Layouts through the Semantic Fingerprint of Architecture. In: Computer-Aided Architectural Design Futures 2011: Proceedings of the International CAAD Futures Conference (2011)
6. Langenhan, C., Petzold, F.: The Fingerprint of Architecture - Sketch-Based Design Methods for Researching Building Layouts Through the Semantic Fingerprinting of Floor Plans. International Electronic Scientific-Educational Journal: Architecture and Modern Information Technologies (2010)
7. Gao, X., Xiao, B., Tao, D., Li, X.: A survey of graph edit distance. Pattern Analysis and Applications 13(1), 113–129 (2009)
8. Ullmann, J.: An algorithm for subgraph isomorphism. Journal of the ACM (JACM) 23(I), 31–42 (1976)
9. Bunke, H.: Graph matching: Theoretical foundations, algorithms, and applications. In: Proc. Vision Interface (2000)
10. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters 18(8), 689–694 (1997)
11. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters, 255–259 (1998)
12. Juchmes, R., Leclercq, P., Azar, S.: A multi-agent system for the interpretation of architectural sketches. In: EG Workshop on Sketch-Based Interfaces and Modeling, pp. 53–61 (2004)
13. Blümel, I., Berndt, R., Ochmann, S., Vock, R., Wessel, R.: Supporting planning through content-based indexing and 3d shape retrieval. In: 10th Int. Conf. on Design & Decision Support Systems in Architecture and Urban Planning (2010)
14. Wessel, R., Blümel, I., Klein, R.: The room connectivity graph: Shape retrieval in the architectural domain. In: Skala, V. (ed.) The 16th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision 2008, UNION Agency-Science Press (2008)
15. Rosen, K.H.: Discrete mathematics and its applications, 2nd edn. McGraw-Hill, Inc., New York (1991)

Representation, Indexing, and Retrieval of Biological Cases for Biologically Inspired Design

Bryan Wiltgen[1], Ashok K. Goel[1,2], and Swaroop Vattam[1]

[1] Design & Intelligence Laboratory, School of Interactive Computing
[2] Center for Biologically Inspired Design,
Georgia Institute of Technology
Atlanta, GA, USA
{bwiltgen,goel,svattam}@cc.gatech.edu

Abstract. Biologically inspired design is an increasingly popular design paradigm. Biologically inspired design differs from many traditional case-based reasoning tasks because it employs cross-domain analogies. The wide differences in biological source cases and technological target problems present challenges for determining what would make good or useful schemes for case representation, indexing, and adaptation. In this paper, we provide an information-processing analysis of biologically inspired design, a scheme for representing knowledge of designs of biological systems, and a computational technique for automatic indexing and retrieval of biological analogues of engineering problems. Our results highlight some important issues that a case-based reasoning system must overcome to succeed in supporting biologically inspired design.

Keywords: Analogical design, analogical reasoning, biologically inspired design, biomimicry, case-based creativity, case-based design.

1 Introduction

Biologically inspired design, sometimes also called biomimicry or bionics, uses analogies to biological systems to generate ideas for the conceptual (or the preliminary, qualitative) phase of design. Although nature has long inspired designers (e.g., Leonardo da Vinci, the Wright brothers), biologically inspired design recently has become an important and widespread movement, pulled by the growing need for environmentally sustainable designs and pushed by its promise to generate creative designs [1-4]. The design of windmill turbine blades based on the tubercles on a humpback whale's flippers [5-7] illustrates the power of biologically inspired design. By taking inspiration from whale flippers, designers were able to improve the shape of wind turbine blades to improve lift and reduce drag, increasing their efficiency.

Biologically inspired design differs from many traditional case-based reasoning tasks because it employs cross-domain analogies. In classical case-based reasoning (CBR) [8,9], the new, input or target problem is so closely related and similar to familiar, known source cases stored in memory that the case memory supplies an almost correct answer to a given problem and the retrieved case need only be "tweaked" to fit the problem. In contrast, biologically inspired design by definition entails "far" analogies from biology to architecture, engineering, computing, and other design domains, e.g., the design of turbine blades based on the shape of whale flippers or the design of a self-cleaning catheter based on the surface of lotus leaves. Thus, biologically inspired design is an excellent context for studying case-based analogies well as case-based creativity.

In this paper we focus on biologically inspired engineering design that engages transfer of the results of biological evolution to engineering problems. Biology and engineering, however, occur at many different scales in both time and space. Further, biologists and engineers use different languages and methods, and generally share little cultural and disciplinary common ground. This presents an interesting set of challenges for research on case-based reasoning. How does one represent, retrieve, and adapt knowledge from a source domain that is as fundamentally different than the target domain as biology is to engineering?

In this paper, first we briefly describe an information-processing account of biologically inspired design processes. Next, we describe a scheme for representing knowledge of designs of biological systems. Then, we apply the technique of redundant discrimination networks for automatic indexing and retrieval of biological designs. We conclude with a discussion on the results of our application, highlighting identified issues and speculating on how to overcome those difficulties.

2 Case-Based Design

Research on case-based has a long and rich history in which many researchers have not only used CBR theories and techniques in design, but also used insights from design to drive research into CBR. Early examples of development of CBR in design include CYCLOPS [10,11], STRUPLES [12,13] ARGO [14] and KRITIK [15-17]. For example, ARGO used rule-based reasoning to transform design plans for designing VLSI circuits to meet functional specifications of new circuits. In contrast, KRITIK integrated case-based and model-based reasoning to produce conceptual designs for engineering devices such as heat exchange devices and electric circuits. If a designer specified a function, F, KRITIK generated a qualitative specification of a structure S, which could accomplish that function. To do so, it stored an inverse mapping (from structure S, to behavior B, to function F) in the form of a structure–behavior–function (SBF) model for each past case. Thus, SBF model provided a functional vocabulary for indexing past design cases so that they could be stored and later retrieved, adapted, or verified. Maher & Gomez [18] provide a survey of some of the early case-based design systems; Maher & Pu [19] provide an anthology of several early papers.

The last two decades saw an explosion of interest in case-based design. Briefly, we identify four major trends in this period. The first trend was to develop interactive CBR design systems that provided access to libraries of design cases but left the task of design adaptation to the user [20,21]. A second trend was to integrate CBR with a wide variety of reasoning methods such as rule-based reasoning and model-based reasoning [22,23], constraint satisfaction [24,25] and genetic algorithms [26,27] in order to create or evolve emergent new designs from the original case base. A third trend was the development of hierarchical case-based reasoning in which design cases were decomposed into sub cases at storage time and recomposed at problem-solving time [28,29]. A fourth major trend in research on CBR in the nineteen nineties was to develop CBR for a variety of design tasks, such as assembly planning [30], in a wide variety of design domains such as software design [29,31] and design of human-machine interfaces [32]. Goel & Craw [33] survey some of the above developments in case-based design.

More recently, visual CBR has been at the forefront of research. Gross & Do's [34] Electronic Napkin took queries in the form of simple design sketches and retrieved matching design drawings from a design case library. Other work in visual CBR took design drawings generated by vector graphics programs as input and retrieved matching vector graphics design drawings from a diagrammatic case library [35], and used purely visual knowledge to transfer design plans from a known design case to a new design problem [36].

3 Information Processes of Biologically Inspired Design

We study biologically inspired design in the context of an interdisciplinary, senior-level undergraduate course our university offers on biologically inspired design (ME/ISyE/MSE/PTFe/BIOL 4740). Although its contents change slightly every year,

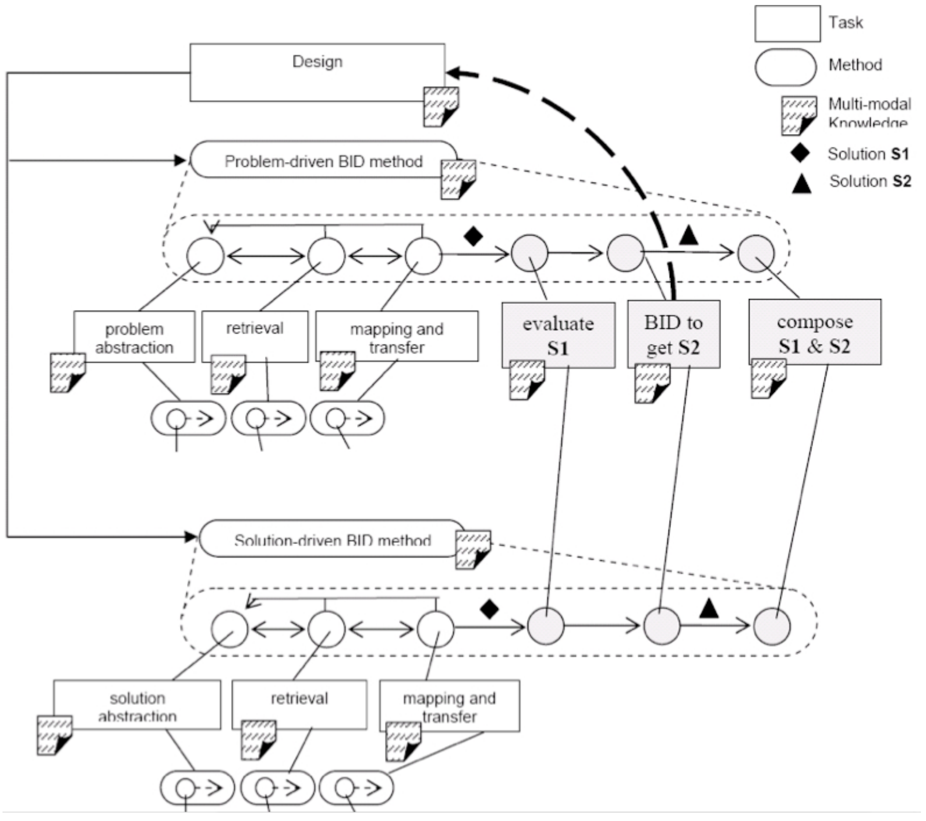


Fig. 1. An Information Processing Model of Biologically Inspired Design (adapted from [37])

it usually consists of three components: lectures, found object exercises, and a semester-long biologically inspired design team project. For the project, teams of 4-6 students are formed such that each team has at least one biology student and students from different schools of engineering. Each team was given a broad problem in the domain of dynamic, adaptable, sustainable housing such as heating or energy use.

Our *in situ* observations have indicated two characteristics of analogical transfer from biology to engineering. First, the process of biologically inspired design can be either problem-driven or solution-driven. The two design processes share many steps but differ in their starting points. As in traditional design, problem-driven biologically inspired design starts with a functional specification of a design problem. In contrast, solution-driven design starts from a biological design looking for a problem to address. The work described here focuses on problem-driven biologically inspired design. Second, biologically inspired design entails compound analogies. In compound analogy, an initial reminding of a biological analog results in a decomposition of the given design problem. The unsolved sub-problems then lead to additional reminders of biological analogues. Figure 1 (adapted from [37]), illustrates our preliminary, high-level information-processing model of biologically inspired design. Note that the ordering of the design subtasks is dependent upon a selection of a particular design method, either problem-driven design (top half of the figure) or solution-driven (bottom half). Note also that due to compound analogies, the process of analogical transfer is iterative and may reoccur for each open sub-problem.

4 Representation of Knowledge of Biological Systems

The promise of biologically inspired design has led to several efforts at developing computational tools for supporting the design process (e.g., [38-40]). In this section, we briefly describe a computational tool called DANE (for Design by Analogy to Nature Engine) that provides a designer with access to a knowledge base of biological systems [41]. To support the problem-driven design process, knowledge of a biological system needs to explicitly specify (1) the functions of the biological system because target design problems typically are specified by the desired functions, (2) the behaviors (or mechanisms) of the biological system that result in the achievement of a specific function because the designer is interested in biologically mechanisms for achieving desired functions, and (3) the structure of the biological system because mechanisms arise from the structure and because the specification of design problems typically contains structural constraints.

Thus, DANE represents designs of biological systems in the SBF knowledge representation language (e.g., [42]). An SBF model of a complex system explicitly represents its structure [S] (i.e., its configuration of components and connections), its functions [F] (i.e., its intended output behaviors), and its behaviors [B] (i.e., its internal causal processes that compose the functions of the components into the functions of the system). The SBF language provides a vocabulary for expressing and organizing knowledge in an F → B → F → B … → F(S) hierarchy, which captures functionality and causality at multiple levels of aggregation and abstraction. Generally speaking, as one moves down the hierarchy, a system is described at lower levels of abstraction. For example, one level of the hierarchy might describe, at a

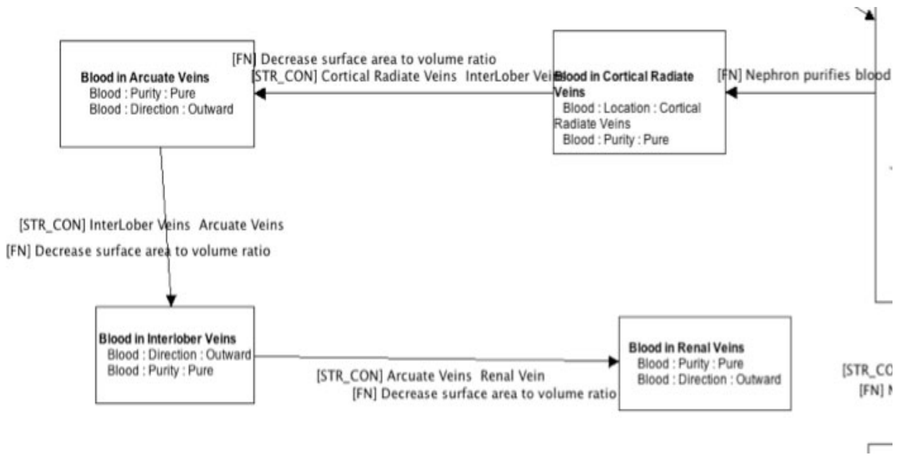

Blood in Arcuate Veins
Blood : Purity : Pure
Blood : Direction : Outward
[FN] Decrease surface area to volume ratio
[STR_CON] Cortical Radiate Veins InterLober Vei
Blood in Cortical Radiate Veins
Blood : Location : Cortical Radiate Veins
Blood : Purity : Pure
[FN] Nephron purifies blood
[STR_CON] InterLober Veins Arcuate Veins
[FN] Decrease surface area to volume ratio
Blood in Interlober Veins
Blood : Direction : Outward
Blood : Purity : Pure
[STR_CON] Arcuate Veins Renal Vein
[FN] Decrease surface area to volume ratio
Blood in Renal Veins
Blood : Purity : Pure
Blood : Direction : Outward

Fig. 2. Partial Behavior Model for the system "Kidney Filters Waste from Blood"

high level of abstraction, the function, structures, and behavior of the entire human digestive system, whereas a deeper level might narrow its focus to describe in detail the function, structures, and behavior of the small intestines.

Biological systems are indexed by function name (e.g., “flamingo filter-feeds self”), by subject (e.g., “flamingo”), and/or by verb (e.g., “filter-feeds”). Upon selecting a system-function pair, users are presented with a multi-modal representation of the paired system-function that combines text descriptions, images, and an SBF model. The function sub-model is given as a structured frame-like representation, and the behavior and structure sub-models are represented as labeled directed graphs. Figure 2 illustrates the behavior (or the mechanism) for the function “Kidney Filters Waste from Blood.”

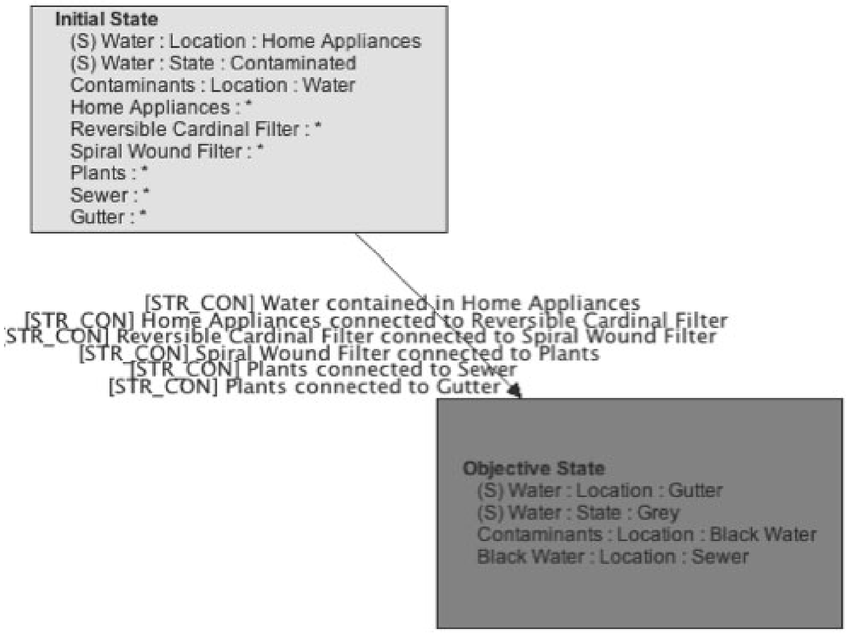
In Fall 2009, we introduced DANE into the class on biologically inspired design mentioned earlier. At the time, the library contained about 40 SBF models, including 22 models of biological systems and subsystems. We discovered that some designers in the class found DANE’s SBF models useful for enhancing their conceptual understanding of biological designs. But we also found the designers did not consistently use DANE for generating design concepts for their problems. This probably was due to several reasons such as learning how to use DANE was not easy enough, DANE’s library of biological systems was not large enough, DANE did not provide enough design support. For example, in reference to the last item, while DANE allowed the user to browse the library of biological systems, it did not have any ability to automatically index or retrieve biological systems relevant to a target design problem. The interested reader may download DANE at http://dilab.cc.gatech.edu/dane.

5 Automatic Indexing and Retrieval

To help resolve this observed need, we are using analogical reasoning techniques for adding automated indexing and retrieval to DANE. AI techniques for analogical retrieval include constraint satisfaction [43], spreading activation [44], and redundant discrimination networks [9,45]. Redundant discrimination networks allow a single case to be indexed in multiple networks. We use this scheme for indexing biological designs because it is not known *a priori* what the most apt indexing method is for any given case. Additionally, the Ideal system [46,47] used this scheme with some success. The open question is whether this scheme works for analogical retrieval of biological designs and if it helps further systemize knowledge of biological designs.

Specification of Design Problems

To automatically retrieve biological analogues relevant to a target design problem, we need to first decide on a vocabulary for specifying design problems. Fortunately, the SBF language already provides such a vocabulary. A desired design is defined by three components: the `initial state`, the `objective state`, and a set of `structural connections`. The `initial state` and `objective state` both contain `state features`, which themselves are composed of `object-property-value` triples. A `state feature` may also be flagged as being a `substance`, which means the `object` in that `feature` flows through the system. Each `structural connection` in the set of `structural connections` (the third primary component of a `specification`) links two `objects` (or optionally one `object` to itself) through a `connection type`, which can be any string.



Initial State
(S) Water : Location : Home Appliances
(S) Water : State : Contaminated
Contaminants : Location : Water
Home Appliances : *
Reversible Cardinal Filter : *
Spiral Wound Filter : *
Plants : *
Sewer : *
Gutter : *
[STR_CON] Water contained in Home Appliances
[STR_CON] Home Appliances connected to Reversible Cardinal Filter
[STR_CON] Reversible Cardinal Filter connected to Spiral Wound Filter
[STR_CON] Spiral Wound Filter connected to Plants
[STR_CON] Plants connected to Sewer
[STR_CON] Plants connected to Gutter
Objective State
(S) Water : Location : Gutter
(S) Water : State : Grey
Contaminants : Location : Black Water
Black Water : Location : Sewer

Fig. 3. Specification of a Filtrating Design Problem

As an example, Figure 3 illustrates a specification of the problem of home water filtration taken from a 2009 class project. The `initial state` of this specification describes the `substance` (denoted with (S)) water being contaminated and located within home appliances. The transition between the `initial` and `objective states` describes a series of `structural connections` that are relevant to the filtration process described by the 2009 design. In the `objective state`, the once-contaminated water has been transformed into grey water and moved to the gutter, and its contaminants are now located in black water in the sewer. In Figure 3, note that a state feature like "Plants : *" is used to denote an `object` that appears in a `connection` but does not otherwise have a `property` or `value` associated with it.

```
Structural Discrimination Network Algorithm
For each behavior model:
    1: Create one node N for the set of all unique, non-substance structures in a by-state
       or by-structural-connection transition.
    2: Attach node N as a child of STRUCTURAL-ROOT.
    3: Create a node M for each by-structural-connection transition that includes at least 1
       non-substance structure.
    4: Attach each node M as child of N.
    5: Create a node C for each non-substance structure in an M-node.
    6: Attach each node C as a child of M where structures are equal.
    7: Create a node P for each structure-property pair in a by-state transition.
    8: Attach each node P as a child of C where structures are equal.
    9: Create a node S to represent the system.
    10: Attach node S as a child of P.
```

```
Functional Discrimination Network Algorithm
For each behavior model:
    1: Create a node N for each substance-property pair in a by-state transition
    2: Attach node N as a child of FUNCTIONAL-ROOT.
    3: Create a node M for each substance-property-value triple in a by-state transition.
    4: Attach each node M as child of N where properties are equal.
    5: Create a node O to represent the system.
    6: Attach node O as a child of all M nodes.
```

Fig. 4. Pseudocode for the Indexing Algorithms

Automatic Indexing of Biological Designs

For automated retrieval to work, models of the biological systems in DANE need to be indexed. We use two algorithms adapted from [45], and named Functional Indexing and Structural Indexing, both of which generate separate discrimination networks. Figure 4 provides the pseudocodes for these indexing algorithms. The Functional Indexing algorithm discriminates on `substance`-flagged `objects` and their `properties`, whereas the Structural Indexing algorithm discriminates on `structural connections` and the `properties` of the `objects` involved in those `connections`. These trees are traversed by matching components in the input specification (e.g., Figure 3) to nodes in the tree. Note that, during the creation of the discrimination trees in both cases, if a node is about to be created but already exists in the network, the pre-existing node is used instead. This step prevents the network from having many duplicate nodes (e.g., multiple nodes representing the `property` of "Location").

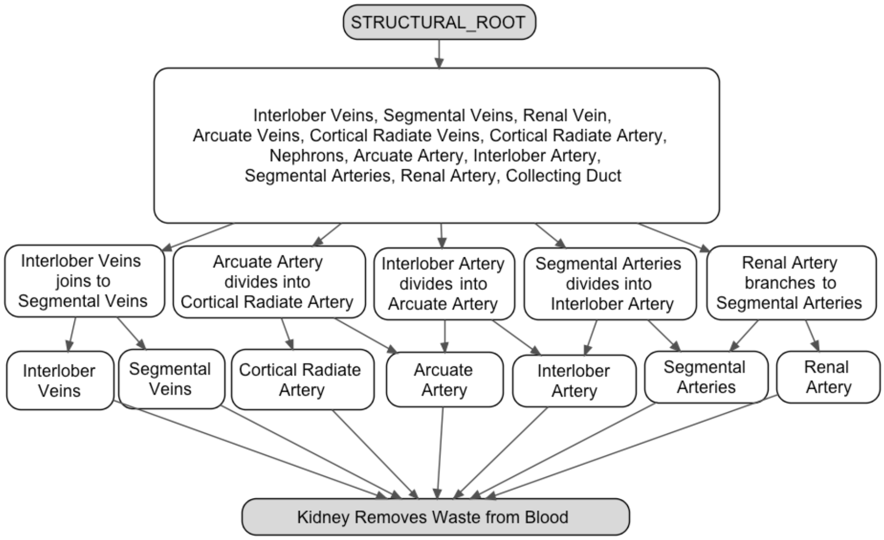

STRUCTURAL_ROOT
Interlober Veins, Segmental Veins, Renal Vein, Arcuate Veins, Cortical Radiate Veins, Cortical Radiate Artery, Nephrons, Arcuate Artery, Interlober Artery, Segmental Arteries, Renal Artery, Collecting Duct
Interlober Veins joins to Segmental Veins
Arcuate Artery divides into Cortical Radiate Artery
Interlober Artery divides into Arcuate Artery
Segmental Arteries divides into Interlober Artery
Renal Artery branches to Segmental Arteries
Interlober Veins
Segmental Veins
Cortical Radiate Artery
Arcuate Artery
Interlober Artery
Segmental Arteries
Renal Artery
Kidney Removes Waste from Blood

Fig. 5. Partial snapshot for the Structural Discrimination Network

Figure 5 shows a partial discrimination tree built by the Structural Indexing algorithm. At the top level, the root connects to a node that indexes a set of `objects`. Underneath that node is a set of nodes that index on `structural connections`, and underneath them is a set of related `objects`. Finally, the `objects` themselves link to a model in the system. Note that we don't have `properties` in this partial network because the `objects` here did not have specified `properties` in their model.

Automatic Retrieval of Biological Designs

Once a user creates a `specification` and attempts retrieval, the system uses the contents of the `specification` as probes into these discrimination networks and retrieves one or more results depending upon if the network allows partial matches. We allow partial matches in the Structural network because it is based on non-state specific information (i.e., it does not index on `value`).

Similarity Measurement

If multiple results are returned, the system must determine how similar each result is to the user's `specification` in order to rank the results and recommend a best match. We use two similarity metrics: Structural and Functional. The Structural similarity measurement looks at how many `structural connections` and `object-property` pairs are the same between the `specification` and the returned result. The Functional similarity measurement compares the `initial` and `objective states` of the `specification` and the returned result by attempting to match `object-property-value` triples. This metric differs from the Structural measurement because it compares `values` and does not compare `structural connections`. Both similarity metrics are calculated using Tversky's ratio model [48] with alpha and beta set to 1, which simplifies to `S(specification, case) = f(specification ∩ case) / f(specification ∪ case)` with f comparing features relevant to the given similarity measurement.

Every result returned, regardless of which discrimination network was used, gets measured by both similarity metrics. The average of both scores is presented to the user as a percentage of total similarity and used by the system to rank the results.

6 Evaluation

We performed two sets of tests to evaluate automatic indexing and retrieval in DANE. In the first test, our system retrieved of a target biological system from the library. In the second tested, our system retrieved a biological system relevant to a design problem. For both experiments, we developed a library of 37 models. 15 of the 37 models in the library are technological; 22 are biological. Technological models were derived from earlier CBR experiments in our lab and are related to acid coolers, pulleys, gyroscopes, crankshafts, and latches. Biological models were developed by members of our lab (including the authors) with the goal of accurately representing their respective topics and are related to three families of systems: transpiration, the human small intestines, and the human kidney.

The functional discrimination network indexes 4 models: 3 technological and 1 biological. The structural discrimination network indexes 19 models: 8 technological and 11 biological. The Structural discrimination network contains all of the models indexed by the Functional discrimination network. We found that some of 37 models were not indexed by either algorithm because they are legacy models developed prior to our building of this automated retrieval technique, and thus do not contain the knowledge needed by the retrieval algorithms.

Method of the First Test

For the first test, we tested our system's ability to retrieve a target biological system. The target was "Kidney Removes Waste from Blood," a model that describes how the human kidneys filter certain materials out of our blood.

The test involved 28 sub-tests across five phases. For each test, we built a `specification` and asked the system to retrieve a set of related results. The results and the similarity scores (average, Functional, and Structural) were then recorded for later analysis. The goal of each sub-test was to systematically add something to the `specification` and see how the results changed.

In the first phase, we added an `object-property` pair (e.g., `Blood-Purity`) to the Initial State, beginning with a single `object-property` pair and stopping when the `initial state` of the Specification matched that of the model in the database, excluding any `value`'s.

The second phase was the same as the first, except `object-property` pairs were now added to the Objective State. Note that each phase and each sub-test are additive, so the `specification` is being built up over time.

In the third phase, we incrementally added a `value` to each `object-property` pair, creating an `object-property-value` triple (e.g., `Blood-Purity-Impure`) that matched the target model.

The fourth phase was the same as the third except that we added `value`'s to the `objective state.`

In the fifth and final phase, `structural connections` were incrementally added to the `specification` (e.g., `Renal Artery branches to Segmental Arteries).` Note that these `structural connections` were added in the order that they appear in the `behavior` of the target model so as to mimic how a user of our system might add things as he or she is thinking about the target process.

Method of the Second Test

In the second test, our goal was to evaluate our system's ability to retrieve designs of biological systems based on the problem `specification` illustrated in Figure 3. This `specification` is based on a design project in the Fall 2009 edition of the biologically inspired design course described above. This sustainable design project had the goal of conserving home water use by recycling greywater for use in toilets. The design has contaminated home water going through an elaborate filtration process, eventually being divided into blackwater, which contains the contaminants and must be sent for treatment, and greywater, which can be reused.

Next, we deployed the same `specification`-building, five-phase experiment discussed in the first test. In this second test, we start from scratch and eventually the design `specification`, given in Figure 3, instead of assembling something that already exists in the database. For example, the first `object-property` pair added was `Water-Location`; the first `object-property-value` triple added was `Water-Location-Home`; and the first `structural connection` added was `Water Contained in Home Appliances.`

Note that there was a model already in the database called "Recycle Greywater," but it used neither the same terminology nor the same structures as the design `specification` used for this test.

Results

For the first test, our system returned the "Intestinal Villi Absorb Water and Nutrients into Blood" model as a best match for the first four phases. This model was not our target model. Despite being the best match, our system only gave this model on average a 10% similarity score. In the fifth phase where `structural connections` were included in the `specification`, the system returned "Kidney Removes Waste from Blood," which was our target model, and gave it from 68% similarity (in the beginning of phase 5) to 98% similarity (at the end of phase 5). The reason we didn't get 100% similarity is because the Structural similarity metric considers `objects` from all the behavioral states in the model and we've only included information in our `specification` from the `initial` and `objective states.`

For the second test, our system returned the "Transpiration" model as the best match for all phases, ranging from a 15% similarity score in the beginning of Phase 1 to a 28% similarity score at the end of Phase 5. Additionally, the "Osmosis" model was returned as the second best match for the first four phases and first step of the fifth phase (ranging from 14% similarity to 20% similarity), and the "Root Absorbs Water" model was returned as the second best match for the rest of the fifth phase (20-23% similarity).

7 Conclusions

Since biologically inspired design is a promising paradigm for creative and sustainable solutions, there is a race to develop computational techniques and build computational tools to support the design process. However, at present both the practice of biologically inspired design and the development of supporting computational tools are *ad hoc*. In this paper, we presented an information-processing model of biologically inspired design as well as a knowledge representation scheme for organizing knowledge of biological systems from a design perspective. We then applied a case retrieval technique that utilized redundant discrimination trees for automatic indexing and retrieval of biological cases in the DANE system.

Based on the experiments described in this paper, We can conclude two things about automated retrieval in DANE. The first is that the system works as advertised when given a model structured appropriate to the indexing and retrieval algorithms. The "Kidney Removes Waste from Blood" model, for example, was indexed correctly, was retrieved after the relevant information was added to the problem specification, and was ranked by our system as the best match with a very high similarity score. Similarly, our second test showed that relevant models could be retrieved given a realistic design problem specification. All three of cases returned as best match and second-best match were related to the movement of water through a system, and although they weren't about filtration *per se*, one could speculate on how learning the process by which plants move water across long distances with little energy (i.e., "Transpiration") might inspire a designer to create a low-energy system to move water throughout the filtration process.

However, our tests also highlight some areas for improvement. For one, our indexing algorithms failed to index about half of the models in our database. Clearly, if an automated retrieval system is to be successful, it must also be comprehensive. Second, our indexing algorithm in the first test only retrieved the correct model when `structural connections` were used in the specification. Both of these shortcomings highlight the need for additional systemization of biological knowledge for use in automated retrieval. Because our indexing scheme was designed separately from many of the models, the model-builders did not include aspects in their models that would allow them to be indexed into discrimination networks.

In addition to improving DANE's automatic indexing and retrieval scheme, we plan to expand it to the full spectrum of the biologically inspired design process described here.. Specifically, we will enable adaptation of retrieved biologically cases to fit the input specification of engineering design problems. We are also investigating mechanisms for the discovery, abstraction, and application of design patterns to further enhance resolution of the input problem specification.

Acknowledgments. This research has benefited from many discussions with Michael Helms and other members of the Design & Intelligence Laboratory (http://www.dilab.gatech.edu/). We are also grateful to Professor Jeannette Yen, Director of Georgia Tech's Center for Biologically Inspired Design (www.cbid.gatech.edu) and coordinator of the ME/ISyE/MSE/PTFe/BIOL 4740 class for her strong support and encouragement for this work. We also thank the US National Science Foundation that has generously supported this research through an NSF CreativeIT Grant (#0855916) entitled "Computational Tools for Enhancing Creativity in Biologically Inspired Engineering Design."

References

1. Bar-Cohen, Y. (ed.): Biomimetics: Biologically Inspired Technologies. Taylor & Francis, Abington (2006)
2. Benyus, J.: Biomimicry: Innovation Inspired by Nature. William Morrow (1997)
3. Vincent, J., Mann, D.: Systematic Transfer from Biology to Engineering. Philosophical Transactions of the Royal Society of London 360, 159–173 (2002)
4. Yen, J., Weissburg, M.: Perspectives on biologically inspired design: Introduction to the collected contributions. Journal of Bioinspiration and Biomimetics 2 (2007)
5. Ashley, S.: Bumpy flying. Scalloped flippers of whales could reshape wings. Scientific American 291(2), 18–20 (2004)
6. Fish, F.: Imaginative solutions by marine organisms for drag reduction. In: Meng, J. (ed.) Procs. International Symposium on Seawater Drag Reduction, pp. 443–450. Office of Naval Research, Newport (1998)
7. Fish, F., Battle, J.: Hydrodynamic design of the humpback whale flipper. Journal of Morphology 225, 51–60 (1995)
8. Reisbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Hillsdale (1989)

9. Kolodner, J.: Case-Based Reasoning. Morgan Kauffman, San Francisco (1993)
10. Navinchandra, D.: Case-Based Reasoning in CYCLOPS: A Design Problem Solver. In: First DARPA Workshop on Case-Based Reasoning. Morgan Kaufman Publishers, San Francisco (1988)
11. Navinchandra, D.: Exploration and Innovation in Design. Springer, New York (1991)
12. Maher, M., Zhao, F.: Using Experiences to Plan the Synthesis of New Designs. In: Gero, J. (ed.) Expert Systems in Computer-Aided Design, pp. 349–369. North-Holland, Amsterdam (1986)
13. Zhao, F., Maher, M.: Using Analogical Reasoning to Design Buildings. Engineering with Computers 4, 107–122 (1988)
14. Huhns, M., Acosta, E.: Argo: A System for Design by Analogy. IEEE Expert 3(3), 53–68 (1988)
15. Goel, A., Chandrasekaran, B.: Integrating Case-Based and Model-Based Reasoning for Design Problem Solving. In: AAAI 1988 Workshop on Design, St. Paul, Minnesota (1988)
16. Goel, A., Chandrasekaran, B.: Case-Based Design: A Task Analysis. In: Tong, C., Sriram, D. (eds.) Artificial Intelligence Approaches to Engineering Design. Innovative Design, vol. II, pp. 165–184. Academic Press, San Diego (1992)
17. Goel, A.: Representation of Design Functions in Experience-Based Design. In: Brown, D., Waldron, M., Yoshikawa, H. (eds.) Intelligent Computer Aided Design, pp. 283–308. North-Holland, Amsterdam (1992)
18. Maher, M.L., Gomez, A.: Case-Based Reasoning in Design. IEEE Expert 12(2), 34–41 (1997)
19. Maher, M., Pu, P. (eds.): Issues and Applications of Case-Based Reasoning in Design. Lawrence Erlbaum Associates, Mahwah (1997)
20. Pearce, M., Goel, A., Kolodner, J., Zimring, C., Sentosa, L., Billington, R.: Case-based decision support: a case study in architectural design. IEEE Expert 7(5), 14–20 (1992)
21. Sycara, K., Guttal, R., Koning, J., Narasimhan, S., Navinchandra, D.: CADET: a case-based synthesis tool for engineering design. International Journal of Expert Systems 4(2), 157–188 (1991)
22. Gebhardt, F., Voß, A., Gräther, W., Schmidt-Belz, B.: Reasoning with Complex Cases. Kluwer, Norwell (1997)
23. Börner, K., Coulon, C., Pippig, G., Tammer, E.C.: Structural similarity and adaptation. In: Smith, I., Faltings, B. (eds.) EWCBR 1996. LNCS, vol. 1168. Springer, Heidelberg (1996)
24. Smith, I., Lottaz, C., Faltings, I.: Spatial Composition Using Cases: IDIOM. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 88–97. Springer, Heidelberg (1995)
25. Hua, K., Faltings, B., Smith, I.: CADRE: Case-Based Geometric Design. Artificial Intelligence in Engineering 10(2), 171–183 (1996)
26. Louis, S.: Working from Blueprints: Evolutionary Learning in Design. Artificial Intelligence in Engineering 11(3), 335–341 (1997)
27. Gómez de Silva Garza, A., Maher, M.L.: An evolutionary approach to case adaption. In: Third International Conference on Case-Based Reasoning, pp. 162–172. Springer, Berlin (1999)
28. Maher, M.L., Balachandran, B., Zhang, D.M.: Case-based Reasoning in Design. Lawrence Erlbaum Associates, Hillsdale (1995)
29. Smyth, B., Keane, M.: Design à la Déjà Vu: reducing the adaptation overhead. In: Leake, D.B. (ed.) Case-Based Reasoning: Experiences, Lessons and Future Directions, pp. 151–166. AAAI Press/The MIT Press, Menlo Park, CA (1996)

30. Pu, P., Reschberger, M.: Assembly Sequence Planning Using Case-Based Reasoning Techniques. Knowledge-Based Systems 4(3), 123–130 (1991)
31. Bhansali, S., Harandi, M.: Synthesis of UNIX Programs Using Derivational Analogy. Machine Learning 10, 7–55 (1993)
32. Barber, J., Bhatta, S., Goel, A., Jacobson, M., Pearce, M., Penberthy, L., Shankar, M., Simpson, R., Stroulia, E.: AskJef: integration of case-based and multimedia technologies for interface design support. In: Gero, J.S. (ed.) Artificial Intelligence in Design 1992, pp. 457–474. Kluwer, Dordrecht (1992)
33. Goel, A., Craw, S.: Design, Innovation and Case-Based Reasoning. Knowledge Engineering Review 20(3), 271–276 (2005)
34. Gross, M., Do, E.: Drawing on the Back of an Envelope: a framework for interacting with application programs by freehand drawing. Computers & Graphics 24(6), 835–849 (2000)
35. Yaner, P., Goel, A.: Visual Analogy: Viewing Retrieval and Mapping as Constraint Satisfaction. Journal of Applied Intelligence 25(1), 91–105 (2006)
36. Davies, J., Goel, A., Nersessian, N.: A Computational Model of Visual Analogies in Design. Journal of Cognitive Systems Research, Special Issue on Analogies - Integrating Cognitive Abilities 10, 204–215 (2009)
37. Vattam, S., Helms, M., Goel, A.: Biologically Inspired Design: A Macrocognitive Account. In: 2010 ASME IDETC/CIE Conference on Design Theory and Methods, Montreal, Canada (August 2010)
38. Fisher, Maher (eds.): Proceedings of AAAI Spring Symposium on AI and Sustainable Design. Stanford University, Stanford (March 2011)
39. Biomimicry Institute. Ask Nature – The Biomimicty Design Portal, `http://www.asknature.org/`
40. Chakrabarti, A., Sarkar, P., Leelavathamma, B., Nataraju, B.: A functional representation for aiding biomimetic and artificial inspiration of new ideas. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 19, 113–132 (2005)
41. Vattam, S., Wiltgen, B., Helms, M., Goel, A., Yen, J.: DANE: Fostering Creativity in and through Biologically Inspired Design. In: First International Conference on Design Creativity, Kobe, Japan (November 2010)
42. Goel, A., Rugaber, S., Vattam, S.: Structure, Behavior & Function of Complex Systems: The Structure-Behavior-Function Modeling Language. AI for Engineering Design, Analysis and Manufacturing 23, 23–35 (2009)
43. Thagard, P., Holyoak, K.J., Nelson, G., Gochfeld, D.: Analog retrieval by constraint satisfaction. Artificial Intelligence 46, 259–310 (1990)
44. Forbus, K., Gentner, D., Law, K.: MAC/FAC: A model of Similarity-based Retrieval. Cognitive Science 19(2), 141–205 (1995)
45. Feigenbaum, E., Simon, H.: EPAM-like models of recognition and learning. Cognitive Science 8, 305–336 (1994)
46. Bhatta, S., Goel, A.: Model-Based Indexing and Index Learning in Engineering Design. International Journal of Engineering Applications of Artificial Intelligence 9(6), 601–610 (1996)
47. Bhatta, S., Goel, A.: Learning Generic Mechanisms for Innovative Strategies in Adaptive Design. Journal of Learning Sciences 6(4), 367–396 (1997)
48. Tversky, A.: Features of Similarity. Psychological Review 84(4), 327–352 (1997)

Ontology-Aided Product Classification: A Nearest Neighbour Approach

Alastair A. Abbott and Ian Watson

Department of Computer Science,
University of Auckland,
Auckland, New Zealand
aabb009@aucklanduni.ac.nz, ian@cs.auckland.ac.nz

Abstract. In this paper we present a k-Nearest Neighbour case-based reasoning system for classifying products into an ontology of classes. Such a classifier is of particular use in the business-to-business electronic commerce industry, where maintaining accurate products catalogues is critical for accurate spend-analysis and effective trading. Universal classification schemas, such as the United Nations Standard Products and Services Code hierarchy, have been created to aid this process, but classifying items into such a hierarchical schema is a critical and costly task. While (semi)-automated classifiers have previously been explored, items not initially classified still have to be classified by hand in a costly process. To help overcome this issue, we develop a conversational approach which utilises the known relationship between classes to allow the user to come to a correct classification much more often with minimal effort.

1 Introduction

In business-to-business (B2B) e-commerce, the problem of classifying products for the purpose of maintaining electronic product catalogues (EPCs) can be particularly difficult and costly. Since different suppliers often use different product classification standards, products from different suppliers need to be reclassified into the standard required [8]. This is a problem not only affecting businesses which need to maintain catalogues of products available from various suppliers, but also in the greater e-procurement industry. Well maintained and classified EPCs are of particular importance given the movement to electronic business practices and allow for accurate spend-analysis and increased efficiency in both finding and selling products [14].

The gold-standard solution would be to have a standardised classification system valid across all domains; this is the primary goal of the UNSPSC (United Nations Standard Products and Services Code) schema.[1] However, due to differing requirements of businesses the UNSPSC standard might not be suitable for everyone, and other classification schemes such as eCl@ss[2] are also widely used.

[1] http://www.unspsc.org
[2] http://www.eclass-online.com

Further, the process of transferring existing catalogues to a different schema might simply be too costly to be justifiable. As such, even for those businesses using the UNSPSC standard, products from other suppliers must be re-classified into this standard making this a perpetual issue.

The manual classification of products into a standard such as the UNSPSC is a time-consuming and costly process: some studies have indicated it can take 5–10 minutes to classify an item into a complex standard such as the UNSPSC, primarily because of the large number of similar sounding classes [6]. Automated or semi-automated systems to perform this task are a promising (and perhaps inevitable) solution since this task fits well within the Machine Learning domain, and have significant potential to cut cost and time [2]. Some existing automatic classification systems have been developed, primarily using statistical methods [5,8]. In this paper we develop a k-Nearest Neighbour (kNN) case-based reasoner to aid this classification task. Since the accuracy of such systems (particularly on difficult products) is not sufficient to allow fully automated classification, we use the ontological structure of the classification hierarchy to aid a conversational approach to classification. This dramatically reduces the cost to classify new items, and allows an accurate classification to be achieved with minimal user interaction.

2 Background

2.1 The UNSPSC Schema

The UNSPSC is a four-level hierarchical classification schema [14]. It was designed by the United Nations and Dun & Bradstreet as a universal system to classify goods and services across all domains and ease spend analysis, the finding and purchasing of products, and the maintenance of product catalogues. Each class, or "commodity," is assigned a unique 8-digit code and the schema is designed so that every product or service should fit uniquely into a class.

The levels of the schema are shown in Fig. 1, and two of the eight digits of the commodity code come from each level [14]. For example, in Fig. 2 the commodity "Staplers" would have a commodity code of 44121615.

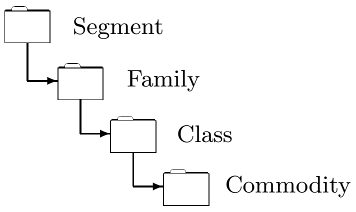


Fig. 1. The UNSPSC class hierarchy structure

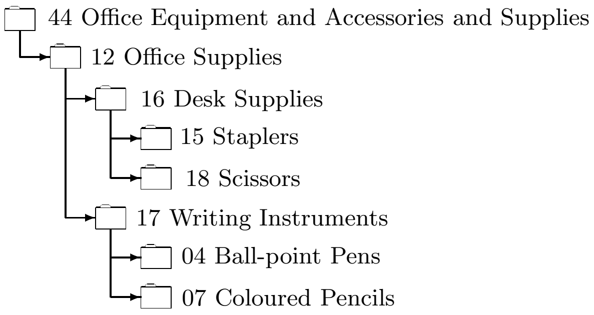


Fig. 2. An example (partial) snapshot of the UNSPSC schema

2.2 Previous Work

Ding et al. developed a system for automatic product classification into the UNSPSC schema called GoldenBullet [5]. Using a Naïve-Bayes approach, Ding et al.'s classifier correctly classified products with an accuracy of 78% when using the top result returned by the classifier. The correct classification was in the top 10 results a slightly higher 88% of the time. Ding et al. also explored using a kNN classifier as well as a vector-space model classifier, but found these were much less accurate. Unfortunately, the nature of the product catalogue they used is not specified other than that it consisted of 40,000 items primarily in the electronics domain. Crucial factors such as the quality of the data or spread of classes encompassed were not discussed.

Another classifier is described in [8], which also uses the Naïve-Bayes method, and obtained a similar accuracy of 86% based on the top result returned. Their data was obtained from an e-procurement database maintained by the Public Procurement Services of Korea, a government run organisation. This uses a classification schema based on the UNSPSC hierarchy, and the database that was used contained almost 500,000 items spread over 7960 classes.

Since the accuracy of a classifier is heavily reliant on the dataset used, the lack of information or availability of datasets used for these classifiers makes comparison of results extremely difficult. Certain types of classifier may be inclined to work better on particular types of dataset. Further, as was noted in [8], the quality of the dataset used will significantly affect the accuracy.

3 Classifier Development

In this paper we detail a project carried out in collaboration with with Unimarket,[3] a locally based e-procurement business. In order to provide a good e-procurement service it is necessary to maintain accurate catalogue data, something which necessarily entails working with a range of classification systems from various suppliers. This is crucial for the facilitation of B2B trading and

[3] http://www.unimarket.co.nz

accurate spend analysis services where expenses across different classes of commodities and services must be readily compileable.

No suitable datasets using the UNSPSC schema were available from Unimarket or elsewhere, and instead data from Amazon using their classification hierarchy were used for testing and development. While we could have looked at transferring the data into the UNSPSC hierarchy, this migration process is primarily a mapping rather than a classification task and is beyond the scope of this project—it cannot be done completely automatically from scratch. Either an initial mapping between classes across schemata is required, or an initial subset of data would need to be manually classified before a (semi)-automated system could complete the task. Rather, the task of the project was to develop a tool to classify new products into the classification schema. This could not only be used to complete the migration to the UNSPSC hierarchy, but is necessary for catalogue maintenance where it would be of use in the ongoing task of adding new products from suppliers correctly and efficiently into the hierarchy.

3.1 Solution Outline

In this paper a kNN Instance Based Learner [16] is explored as a proof-of-concept for a new classification system. In existing systems such as GoldenBullet the significant 10–20% of misclassified items require a costly manual classification procedure to be applied. As a result, a considerable degree of user interaction is inevitable and the system cannot reasonably run in a completely autonomous fashion; in practice the manual classification of the 10–20% of products the automated system struggles with is costly and a limiting factor. To improve on this and handle more constructively the difficult cases, our proposed learner uses a conversational approach to classification. This allows much greater accuracy to be achieved at the cost of some minor user interaction for each classification—an overall gain when compared to classifying difficult items manually from scratch.

To achieve this aim, an ontology of the hierarchical relationship of classes is used, allowing intelligent questions to be asked based on the inferred relationships of candidate classification solutions, in turn aiding the classification of classes and improving the accuracy in a "mostly-automated" approach. The goal of the automated classifier is hence to classify correctly with a minimal number of questions; inevitably some difficult cases will be as bad as the worst case possibility, but this can be drastically reduced when compared to existing approaches. This technique is similar to the method of critiquing [9] in Case Based Reasoning (CBR) recommender systems.

The approach taken in developing our classifier was to build a CBR classifier [1]. In this setup, products in the catalogue are stored as *cases* in the *case-base*. An item to be classified is regarded as a query, and a global similarity metric is used to find the most similar cases in the case-base, and classify the query based on these. We opted to use the kNN algorithm for the global similarity metric, which retrieves the k cases (products) which are most similar to the query based on the similarity of individual attributes. These k results will then be used to guide the conversational procedure, and with care can be made

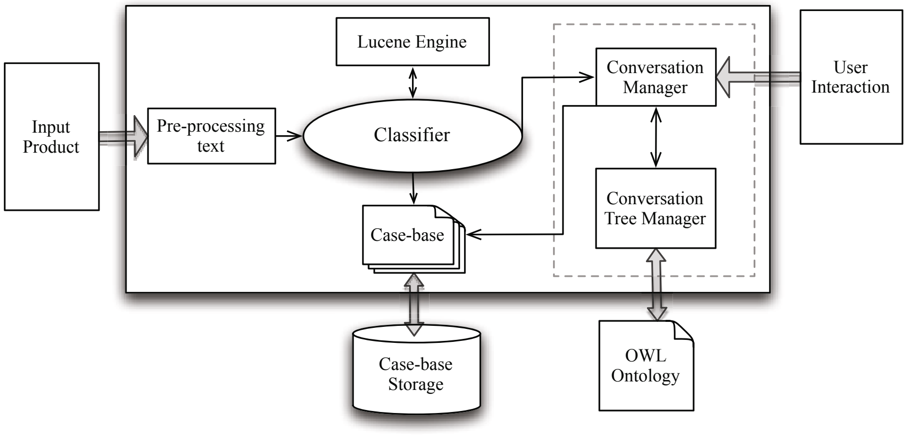


Fig. 3. Architecture diagram of the CBR system

to return an optimal range of queries allowing the conversational procedure to be both accurate and efficient (in terms of number of questions asked).

An OWL (Web Ontology Language) ontology[4] storing the hierarchy of product classes was created from Amazon's existing classification hierarchy. For the UNSPSC hierarchy, OWL format ontologies are already available for use. The ontology stores the hierarchical relationship between different classes which will be utilised by the classifier in the conversational stage.

In order to implement the CBR classifier, the open-source Java CBR development framework jCOLIBRI[5] [4,10] was used. jCOLIBRI allows for flexible development of various types of CBR systems. It includes functionality for pre-defined and user-defined global and local similarity metrics, has built-in (and extendable) functionality for testing the accuracy of the CBR systems, and functionality to integrate with OWL ontoligies via the OntoBridge[6] framework [12]. This allowed the class associated with a product in the case-base in our case representation to be easily and dynamically linked into the hierarchy of classes stored in the ontology file. The overall design of the system is shown in Fig. 3.

3.2 Data

Data were collected from Amazon[7] through the *Amazon Product Advertising API*.[8] These data are hierarchically structured and representative of a large number of classes. Since Amazon allows products to be classified under more than one category, care was taken to only sample a product once so the situation

[4] http://www.w3.org/TR/owl-ref/
[5] http://gaia.fdi.ucm.es/projects/jcolibri/
[6] http://gaia.fdi.ucm.es/grupo/projects/ontobridge/
[7] http://www.amazon.com
[8] http://affiliate-program.amazon.com/gp/advertising/api/detail/main.html

could be treated as a single-label scenario. This hints at some of the difficulties of classification as a product may reasonably fit into more than one category even in a schema such as the UNSPSC, but our aim was to develop a classifier suitable for such a single-label hierarchy. The Amazon class hierarchy is not as consistent as the UNSPSC hierarchy: classes at the same depth in the hierarchy may not be of comparable generality. However, given the size of the case-base and the reasonably good quality of the data, an automated system should be able to function fairly well, and the conversation stage should help deal with some of the issues with the class structure. This reiterates the benefits of adopting a unified schema such as the UNSPSC, and the classifier is only expected to function better on such an organised hierarchy.

An important step was to determine which attributes should be used for classification. The Amazon data is attribute-rich, but most attributes were deemed irrelevant for classification—indeed, too many attributes can be detrimental if they are not relevant enough to the classification. The majority of attributes were deemed to be irrelevant either by mere consideration, or were found to contribute little, if at all, to the accuracy of the classifier during initial testing. This selection was performed manually with the aim of developing a general model, and care was also taken to choose attributes which one would expect to be present in most kind of catalogues to increase the portability of the system. The attributes which were eventually determined to be useful for classification were:

Store (supplier). Often the supplier will give a good indication of the type of product supplied.
Product Name. Clearly this contains information about the class a product should belong to.
Product Description. As for *Product Name.*

A primary case base consisting of over 23,000 items was obtained, with items classified into approximately 140 classes (over 3 top-level classes). Sample data is shown in Table 1. Around a third of the items were identified to be near-identical to other items (varying perhaps, for example, only in the colour of the item), an effect which would skew the results making the accuracy appear falsely high. Since such items are likely to be added to the catalogue (and classified) all at once, fair testing of the system should work with only one of each such item in the case-base. Further, this can have a strong detrimental impact on the diversity of results obtained by the kNN search and thus affect the accuracy of the conversational stage. This is likely to be a problem in many real-world systems, so a reasonable approach is to only use one of each group of items as a representative in the case-base for the classification procedure. Indeed, this approach is motivated by footprint-based retrieval [13], and although differs in its goal it is similarly based on the assumption that the joint competence of such groups of cases is no different from that of the representative chosen.

So that testing would more accurately mimic the real world situation where such similar items would be classified together, the dataset was pre-processed. The pre-processing removed all but one of these nearly identical items—i.e. those

Table 1. A snapshot of the Amazon data

ID	Store	Name	Description	Class
B002EA1EZE	Ultimark	ultimark pre inked stock message stamp faced...	ensure efficient communication and provide clear instructions...	ID490540011
B000V27TCO	Sato	sato industrial labels...	by sato industrial labels per roll	ID490540011
B002ZHQ3KW	Discount Rubber...	self inking christmas rubber stamp...	spread a little holiday cheer with christmas rubber stamps...	ID490540011
B0006HX78Y	Pendaflex	pendaflex pressboard end tab guides...	pt pressboard includes a xyz and mc tabs	ID490540011
B00007LP9W	Avery	avery clip style rigid laserink jet badges...	each kit includes clear plastic badge holders and sheets of...	ID490540011

from the same supplier with similarity above an empirically determined threshold. This similarity was determined using the same similarity metrics as were used for classification, and a conservative approach was taken as it is better to accidentally remove unique items from the case-base than to leave similar items in the case base, potentially skewing results.

All textual data (which is the primary focus in classification) was stripped of punctuation, numbers and product dimensions (e.g. width, height, weight etc.) automatically, as descriptions often contained these technical details of a product that are of little value in determining the class a particular item belongs to.

3.3 Classification Details

The kNN algorithm was used as the global similarity metric of choice, assigning a value in $[0, 1]$ to a pair of cases based on their similarity. This is calculated as the weighted sum over the local similarity metrics of the supplied query attributes. If i indexes the n attributes of the cases and f_i is the local similarity metric for the ith attribute, the kNN similarity between two cases Q and T is calculated as

$$\text{Similarity}(Q, T) = \frac{\sum_{i=1}^{n} f_i(Q_i, T_i) \times w_i}{\sum_{i=1}^{n} w_i} .$$

The k items with the highest similarity to the query are selected, and the conversational stage of the classification procedure is entered. In our setup we chose to use a value of $k = 10$, which seemed to strike a good balance between the number of choices the user was given based on these results, and the accuracy of the classification.

The weighting used placed relative weights of *Store*: 2, *Product Name* and *Product Description*: 5 each, determined to give the best results in testing. In cases where data was missing for a particular attribute, the attribute is given a weighting of 0 so that it is not taken into consideration. This was empirically determined to produce better results than treating this as a local similarity value of 0, which would instead indicate that the cases were actually dissimilar with respect to this attribute.

As for the global similarity metric, the local similarity metrics f_i assign a value in $[0, 1]$ to a pair of cases based on the similarity of the ith attribute. The local similarity metrics used for each attribute were as follows:

Store (supplier). A similarity of 1 is assigned if and only if the suppliers of the cases are identical. In all other cases a score of 0 is used, as the similarity between suppliers cannot otherwise be quantified without auxiliary knowledge on the type of products various suppliers supply.

Product Name. Since this is a textual attribute, textual similarity methods must be used. For this application it is necessary to employ domain independent textual retrieval similarity methods. We opted to use the Lucene textual comparison engine,[9] based on information retrieval techniques, for which jCOLIBRI includes built-in functionality [11]. The Lucene textual comparison is detailed more fully in the following paragraphs.

Product Description. As for *Product Name*, the Lucene textual comparison engine was used.

Since two out of three attributes consist entirely of textual data (and these are the most heavily weighted attributes, accounting for 10/12ths of the weighting), the textual comparison methods are critical and hence deserve further discussion. While textual CBR usually refers to the situation in which the cases themselves are stored as blocks of text [15], techniques from textual CBR are equally applicable for use in classifying products when combined with a kNN global similarity function [11]. As such, textual methods suitable for the type of data which product descriptions consist of should be chosen appropriately.

In particular, textual similarity functions must be domain independent, as the product data is not confined to a particular domain. Further, suppliers do not necessarily provide structured paragraphs of information—the data may consist of lists, technical data and other unstructured text. Hence, semantic similarity functions would also be inappropriate. This limits the ability to compare cases, but is an inherent feature a classifier for this problem must cope with. As such, the obvious option is to use a statistical approach.

The Lucene engine used is based on a combination of the Vector Space Model from the Information Retrieval domain and a Boolean model [7,11]. It is a keyword based search method, which analyses the relative frequencies of the terms in the query in the cases in the case-base. In order to perform this efficiently, the case-base is pre-processed on startup to extract the statistical information required, and then the search query is analysed against this [7,11].

Advanced techniques and modifications were also investigated, such as checking if nodes in the ontology were present as words in the text, as this is a strong indicator for narrowing the search. However, the effect of this was not significant, especially given the extra cost incurred in the search procedure. Given the large weighting the textual attributes hold extra effort spent implementing more advanced techniques would pay off in accuracy in real world deployments of such a classifier. In this paper, however, we are focusing on using the ontology

[9] `http://lucene.apache.org`

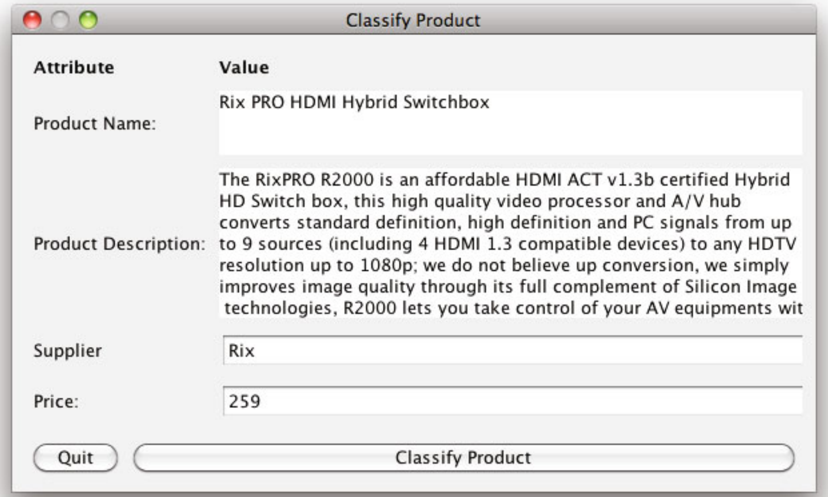
to guide the conversational stage, and for this the Lucene engine, which is built into the jCOLIBRI framework, is sufficient.

The final, important part of the classification process is, of course, the conversational stage of the process. As can be seen in Fig. 3, this acts as a sub-unit of the system mediating the input from the user with the ontology, case base and classification engine. It is this stage which differentiates this proposal most strongly from existing proposals (although also based on different classification algorithms), and has the most significant impact on the most costly phase of classification: when the top or most common result is not the correct classification. The ontology which represents the hierarchy of relationships between classes can be regarded as a rooted tree where the depth of each leaf is identical, leaf nodes correspond to individual classes (commodities in the UNSPSC hierarchy), and internal nodes correspond to higher level classes. The root itself can be regarded as a "Thing" class, although presumably nothing is classified under this category directly. Given the k cases returned by the kNN algorithm, the $m \leq k$ classes present in these results are extracted as potential classes.

The nodes corresponding to these classes are identified within the ontology tree, and the subtree that these induce is extracted. In other words, all classes which are not an ancestor of one of the m candidate classes are removed from the tree. In general, however, this still leaves many classes which are not one of the m candidate classes in the extracted subtree. To reduce this to a minimal structure, all edges between nodes (except the root) of degree two are contracted, i.e. non-candidate nodes which are not "important" to the subtree structure are removed. This leaves a tree which contains only the ontological relationship between the m candidate classes, and no others. In general, this is no longer a complete tree, although the average depth of a node is much lower than before. All candidate classes have maximal depth in the original ontology, but in the extracted tree, they can be direct children of the root "Thing" class.

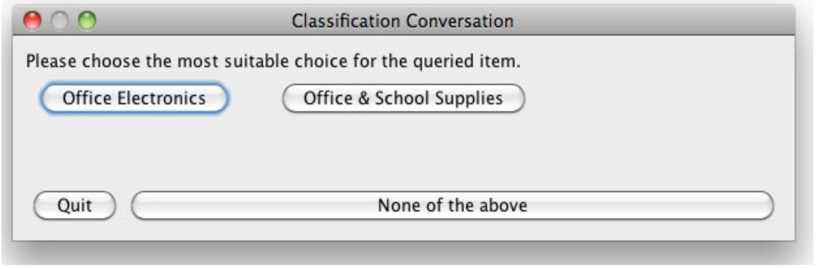
Once the candidate subtree has been extracted, the conversational stage proceeds as follows: Starting at the root, the user is presented, as options, all the children of the root. If the user selects an option which is a leaf class, the classification is complete and the item can be stored in the case-base with the chosen class. If the chosen option is not a leaf class, the user is then presented the children of the chosen class as options. This proceeds until either a classification is made, or the user chooses an option specifying that none of the presented options are correct.

In this case that a classification cannot be made based on the top k results, the general approach is to resort to manual classification. The idea of the conversational approach utilising all k of the top k results (rather than the most common or top one) significantly reduces how often this occurs. However, if the user makes some choices before they determine that none of the sub-classes presented are correct, this means that the top k results contained items which, while they were not in the correct class, were somewhat similar to the query in the fact that they belong to the same more general classes. This information can be used to speed up manual classification by only requiring the user to classify



Attribute	Value
Product Name:	Rix PRO HDMI Hybrid Switchbox
Product Description:	The RixPRO R2000 is an affordable HDMI ACT v1.3b certified Hybrid HD Switch box, this high quality video processor and A/V hub converts standard definition, high definition and PC signals from up to 9 sources (including 4 HDMI 1.3 compatible devices) to any HDTV resolution up to 1080p; we do not believe up conversion, we simply improves image quality through its full complement of Silicon Image technologies, R2000 lets you take control of your AV equipments wit
Supplier	Rix
Price:	259

Fig. 4. A sample query for a product to be classified



Please choose the most suitable choice for the queried item.

Fig. 5. A view of the conversational stage of classification

within this subtree, aiding the manual classification somewhat. The benefit of this is, unfortunately, a little harder to quantify as it needs to be measured in time taken by a user—something harder to approximate from the stats alone.

In Figs. 4 and 5 some stages of the classification procedure for the developed classifier are shown. In Fig. 4, the query can be seen for the product to be classified—in this case a switchbox, an item of office hardware. Once the kNN classifier determines the top k classes, the conversational stage begins. Figure 5 shows the question asked at a particular point in the conversational stage for this particular item.

4 Results

As previously discussed, the Amazon classification hierarchy is not ideal and is inconsistent in some instances. Specifically, in some categories the classes are

Table 2. Datasets used for classification

Dataset	Description
DS1	Full dataset, not pre-processed
DS2	Full dataset, pre-processed
DS3	Revised dataset, not pre-processed
DS4	Revised dataset, pre-processed

much more specific than others, and categories with large amounts of overlap (e.g. "Office Lighting" and "Lamps"). The UNSPSC hierarchy is much more consistent and less ambiguous, so to try and eliminate the difficulty this would cause for the classifier a second slightly smaller dataset of 13,000 items with some manual cleaning was also used. Specifically, for testing purposes some over-specific or ambiguous classes were removed, and the generality of the classes remaining match more closely (although still only roughly) that of the UNSPSC hierarchy.

Datasets both with and without these classes removed were tested, and a summary of the datasets used is given in Table 2. We also tested the effect of the pre-processing to remove near-identical items, so a total of four datasets were tested.[10]

Testing was performed using 10-fold cross-validation: the case-base is randomly split into 10 folds, and in turn each fold is used for querying the case-base consisting of the other 9 folds. The complete case base was used in all results, although randomly chosen subsets of each fold were used for preliminary testing and are an necessity if testing is to be performed on larger cases bases since classification of each item took approximately 1 second. Indeed, the classification time is $O(n)$ for kNN retrieval [3], so this will only further increase the testing time for larger case bases.

Four test metrics were used:

"kNN" accuracy. This is the standard metric for kNN classifiers, and records the percentage of queries for which the most common class among the k returned was the correct classification. Since the purpose of the project is to use and assess a conversational approach where all k results are utilised, this is probably not the most accurate metric, but gives a useful idea of the quality of the classifier.

"top-one" accuracy. The percentage of queries for which the top result was the correct classification. This metric is largely included to aid comparison to the previous product classifiers which used this metric.

"top-k" accuracy. This records the percentage of queries for which one of the top k results was the correct classification. This indicates the item would have been successfully classified in the conversational stage by the user, and is a more important metric given the application. A value of $k = 10$ was used in our testing.

[10] All four datasets and the class ontology file are available for download from `http://www.cs.auckland.ac.nz/~aabb009/resources/AmazonProductData.zip`. We invite others to test their classifiers and try and improve upon our results.

"average-depth". This is the average depth of the conversation tree. In other words, it records the average number of questions the user would have to be asked to successfully classify the product based on the returned results. If none of the top k results are the correct classification, the maximum possible depth is recorded to represent unsuccessful completion of the conversation stage.

Note that the *average-depth* metric needs to be interpreted with the relevant value of k in mind. Since the maximum depth is equal to the number of levels in the schema (3 for the Amazon schema used, 4 for the UNSPSC schema), even if the whole ontology was used for the conversational stage (as opposed to the extracted subtree), the depth would still be very small. However, the number of options (answers for the questions) presented to the user would by overwhelming, and would not be practical. Thus, the *average-depth* scores need to be interpreted with it in mind that no more that k (10 in our case) options will ever be presented to the user, and usually much less than this.

The results of these tests are presented in Table 3, and are discussed in detail in the following section.

Table 3. Results of the evaluation of the classifier

Dataset	*kNN*	*top-one*	*top-k*	*average-depth*
DS1	80%	82%	95%	1.31
DS2	68%	69%	92%	1.51
DS3	77%	77%	94%	1.38
DS4	71%	70%	94%	1.46

5 Discussion

As can be seen from the evaluation, the *top-one* metric shows that top result returned is the correct classification around 70% of the time on the dataset DS4, which we consider to be the dataset for which the results will best reflect the real world accuracy. This is less than the *top-one* accuracy obtained by the Naïve classifiers GoldenBullet [5] and the one described in [8] by around 15%. However, since the datasets are different it is rather difficult, if not impossible, to compare results meaningfully. It is likely that the fact that items can reasonably belong to more than one class in the Amazon hierarchy, as well as the similarity and lack of consistency between class definitions limits the *top-one* accuracy from being better than we obtained, and a more structured classification schema such as the UNSPSC should work better with the classifier.

The results for the *top-k* metric are very promising. In DS4, for 94% of items, one of the top 10 results was a correct classification, indicating that the conversation stage would have resulted in a correct classification. This result is higher than obtained by the GoldenBullet [5] classifier for *top-10* accuracy, although the other Naïve Bayes classifier [8] does not quote any results for *top-10* accuracy. Because of the nature of the intended functionality of this classifier and the fact that this represents the percentage of instances for which classification is successful in the conversational stage, we believe this is the more informative accuracy measure for our classifier.

The *average-depth* results are fairly stable. These show that even though 10 results were returned by the kNN search, the user only had to answer 1.4 questions on average to classify the item. This is a significant improvement over presenting the user with the top classes and asking them to pick the correct one. This should result not only in a saving of time, but by asking simple questions and having the user look over less options this should reduce the rate of human error in classification, resulting in a more accurate product catalogue.

As we can see, the *top-10* accuracy of the classifier increases slightly for DS4 compared to DS2. This is consistent with our prediction that the accuracy of the classifier is negatively affected by the inconsistencies within the Amazon class system, and while there will always be ambiguities and difficult classifications in such a dataset—this is part of the problem the classifier should solve—but it is unlikely to be made worse on the UNSPSC dataset, and if anything we would expect the more logical structure of the hierarchy to be beneficial.

Also note that the accuracy decreases when pre-processing is applied. This is not a negative effect, but rather the "false" positives of nearly identical items being classified correctly are removed, meaning the accuracy metric reflects the actual accuracy of the system in a more realistic manner. This effects the *top-one* accuracy much more than the *top-10* accuracy, since the top result should generally remained unchanged after pre-processing (recall one of the multiple similar cases are retained), whereas the top ten results cover a wider range of classes rather than being populated with cases similar to the query.

An interesting point is the suitability of various values of k for use in the classifier. We chose $k = 10$ because it seemed to be a good balance between accuracy and number of questions to answer. The value of k used will impact the *top-k* accuracy and thus the average number of questions that must be answered by the user, so should be considered carefully in a commercial application. A larger value of k will result in slightly more user interaction (on average) for each classification, but would increase the accuracy and thus decrease the instances in which costly complete manual classification is required. Thus, the value of k should be chosen to balance the time required to classify an item from scratch and the time spent on each individual item to be classified. This will depend on the size and nature of the dataset, and classification schema used.

Taking the results and relevant considerations into account, we believe the classifier is a successful proof-of-concept, and shows much promise to be used in a commercial setting. The primary issue in a company such as Unimarket wishing to adopt such a classifier is that it already requires their data to be classified into a useful classification schema, at least in part. If this is the case, then the classifier described can aid in significantly reducing the cost of classifying new items and maintaining the quality of the electronic product catalogue for efficient business practices.

Acknowledgements. We thank Unimarket for providing us with data and helping fund this project, as well as S. Datt and the anonymous referees for comments which helped improve this paper.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications 7(1), 39–59 (1994)
2. Abels, S., Hahn, A.: Reclassification of electronic product catalogs: The "apricot" approach and its evaluation results. Informing Science Journal 9, 31–47 (2006)
3. De Mántaras, R.L., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, resuse, revision and retention in case-based reasoning. The Knowledge Engineering Review 20(3), 215–240 (2005)
4. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J.A., Sánchez-Ruiz-Granados, A.A.: Building CBR systems with jCOLIBRI. Science of Computer Programming 69, 68–75 (2007)
5. Ding, Y., Korotkiy, M., Omelayenko, B., Kartseva, V., Zykov, V., Klein, M., Schulten, E., Fensel, D.: GoldenBullet in a nutshell. In: Proceedings of FLAIRS 2002 (2002)
6. Grabowski, H., Lossack, R., Weißkopf, J.: Datenmanagement in der Produktentwicklung (Data management in product development). Hanser-Verlag (2002)
7. Hatcher, E., Gospodnetić, O., McCandless, M.: Lucene in Action, 2nd edn. Action Series. Manning Publications Co. (2004)
8. Kim, Y., Lee, T., Chun, J., Lee, S.: Modified Naïve Bayes classifier for e-catalog classification. In: Lee, J., Shim, J., Lee, S.-g., Bussler, C., Shim, S. (eds.) DEECS 2006. LNCS, vol. 4055, pp. 246–257. Springer, Heidelberg (2006)
9. McGinty, L., Smyth, B.: Improving the performance of recommender systems that use critquing. In: Mobasher, B., Anand, S.S. (eds.) ITWP 2003. LNCS (LNAI), vol. 3169, pp. 114–132. Springer, Heidelberg (2005)
10. Recio-García, J.A., Sánchez-Ruiz, A.A., Díaz-Agudo, B., González-Calero, P.A.: jCOLIBRI 1.0 in a nutshell: A software tool for designing CBR systems. In: Petridis, M. (ed.) Proccedings of the 10th UK Workshop on Case Based Reasoning, pp. 20–28. CMS Press, University of Greenwich (2005)
11. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Textual CBR in jCOLIBRI: From retrieval to reuse. In: Wilson, D.C., Khemani, D. (eds.) Proceedings of the ICCBR 2007 Workshop on Textual Case-Based Reasoning: Beyond Retrieval, pp. 217–226 (August 2007)
12. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A., Sánchez-Ruiz, A.A.: Ontology based CBR with jCOLIBRI. In: Ellis, R., Allen, T., Tuson, A. (eds.) Proceedings of AI 2006, The Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Applications and Innovations in Intelligent Systems XIV, pp. 149–162. Springer, Cambridge (2006)
13. Smyth, B., McKenna, E.: Footprint-based retrieval. In: Althoff, K.D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 343–357. Springer, Heidelberg (1999)

14. UNSPSC: Better supply managment with UNSPSC. Electronic, http://www.unspsc.org/AdminFolder/Documents/adopting-unspsc.pdf (retrieved January 14, 2011)
15. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual case-based reasoning. The Knowledge Engineering Review 20(3), 255–260 (2005)
16. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review 11, 273–314 (1997)

A Case-Based Reasoning Approach for Providing Machine Diagnosis from Service Reports

Kerstin Bach[1,2], Klaus-Dieter Althoff[1,2], Régis Newo[1,2], and Armin Stahl[1]

[1] Competence Center Case-Based Reasoning
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Strasse 122, 67663 Kaiserslautern, Germany
firstname.surname@dfki.de
[2] University of Hildesheim
Institute of Computer Science - Intelligent Information Systems Lab
Marienburger Platz 22, 31141 Hildesheim, Germany
lastname@iis.uni-hildesheim.de
http://www.dfki.de/web/competence/cccbr/

Abstract. This paper presents a case-based reasoning system that has been applied in a machine diagnosis customer support scenario. Complex machine problems are solved by sharing machine engineers' experiences among technicians. Within our approach we made use of existing service reports, extracted machine diagnosis information and created a case base out it that provides solutions faster and more efficient than the traditional approach. The problem solving knowledge base is a data set that has been collected over about five years for quality assurance purposes and we explain how existing data can be used to build a case-based reasoning system by creating a vocabulary, developing similarity measures and populating cases using information extraction techniques.

Keywords: Case-based reasoning, machine diagnosis.

1 Introduction

Making use of available data in companies is one of today's major challenges. We collect a lot of data, have various information systems, and are able to track and record many actions people or machines perform. However, getting the best value out of our data might be easy for an expert as long as there are not too many data sets. Transferring this expertise into machine logic is still a challenging task. Within this paper, we describe how we pick up this challenge and explain how case-based reasoning can be applied to existing data within a company in order to create a knowledge-based system that makes work more effective.

Case-Based Reasoning (CBR) is an approach to solve new problems by adapting solutions of similar past problems [1]. CBR is also viewed as a methodology for building knowledge-based systems. In its core functioning CBR is based on experience, often called case-specific knowledge (cases), usually in the form of problem-solution pairs that are stored in a database of cases (case base) along with continuative general knowledge. The latter is used for defining a similarity notion between problem descriptions (similarity measures), which supports partial (non-exact) matching in the case base, and case adaptation (adaptation knowledge).

To develop a CBR system its basic knowledge containers vocabulary, case base, similarity measures, and adaptation knowledge have to be filled [14]. Based on the defined vocabulary the cases have to be described as well as the general knowledge underlying similarity assessment and case adaptation. It is an important degree of freedom of CBR systems that a lack of knowledge in one of its containers can be compensated by using additional knowledge in the other knowledge containers. For instance, having a large number of cases available can compensate for a less precise vocabulary, rather rough similarity measures, and/or a lack of adaptation knowledge.

Especially when dealing with existing data and given representations one usually has a huge amount of cases available. When developing CBR systems in this context, the challenge is to transfer knowledge from the case base to the vocabulary, similarity measures, and adaptation knowledge container.

Within this paper we will explain how CBR systems can be developed using existing data and creating an added value out of it. We explain how applying CBR as methodology can be integrated in existing processes and in the end will provide decision support capabilities and therewith reduce the effort of finding problem's solutions.

This paper is structured as follows: First we will describe the application domain of service reports for vehicles and discuss how existing data containing experiences can be used to create new applications. Section 3 explains first the underlying methodology followed by the processes executed in order to create the application. The following section 4 presents the evaluation based on first experiments followed by a related work discussion (section 5). The last section gives a short summary and an outlook on further activities in this area.

2 Service Reports of Vehicle Problems

Most vehicle companies provide service after delivering their machines to customers [10]. During the warranty period they are able to collect data about how and when problems occurred. They take this data for improving vehicles in different ways: collected data can go back in the product development process, it can be used for improving diagnostic systems to repair them at dealerships or in the factory and also educating service technicians repairing vehicles abroad. This is extremely important if vehicles cannot easily be taken back to factory, e.g. services for aircrafts or trucks.

Such machine manufacturers collect information about machine problems that are submitted by technicians containing machine data, observations and sometimes further correspondence between an engineer or analyst with the technician at the machine. In the end, these discussions usually come up with a solution - however, the solution is normally neither highlighted nor formalized and the topics and details discussed highly depend on the technician and what the Customer

Support asks for. That is the reason why cases that are stored for collecting Customer Support information can not directly be used for CBR. Therefore we will differentiate between Customer Support Cases (CS Cases) and CBR Cases. CS Cases contain original information collected during a discussion between Customer Support and the field technician, while a CBR Case contains only relevant information to execute a similarity based search on the cases. The CBR cases content represents each CS Case, but contains machine understandable information.

For the creation of CBR Cases, there were the following sources available:

– CS Cases created when a problem on a particular machine occurred
– Expertise that is in the field technician's and analyst's head, which they provide during a problem solving discussion.
– CS Solutions are textual descriptions of workflows created by analysts based on frequently occurring problems.

The CS Solutions are not solutions in terms of CBR, because they are not directly connected to a problem. The CS Solutions are FAQ-like descriptions how to solve a technical problem on a machine and are often created for an automatic diagnostic system. They provide a walk-through guidance for a technician. The quality of the CS Solution is very high because analysts looked through each CS Solutions and prepared diagnostic procedures. Reliable information is given and the diagnostic procedures are well structured. However, the CS Solution deal with one issue at a time and are only available for a certain number of problems. To increase the number of CS Solutions in order to have a higher coverage for occurring problems a high effort to create solutions is necessary. Another point why we decided to not use CS Solutions is the long time until the CS Solution is available for the technicians as well as the facts that solutions are not verified in the field and technicians might get stuck during a diagnostic procedure, because they contain many steps and not all events that might occur during a diagnostic procedure are covered.

CS Cases describe problems of a machine that cannot be solved by the technician. The cases describe a realistic situation in the field and contain information the field technician has access to while trying to solve the problem in the field. Further, the cases contain information about what was done in the end and what solved the problem. Those facts indicate that these cases should be reused, because the same problems might occur at the same type of machines and the work of analysts for solving the problem can be applied many times. Further, if the analyst would save time providing a solution that has already been applied, he can use his effort creating new solutions covering new problems. The disadvantages using CS cases are the facts that the cases describe a problem and their solution is hidden in unstructured text, the case content differs depending on the technician and cases that solve a problem right away with the first visit in the field are not available at all. Eventually, we decided to work with such CS cases, because they deal with experiential knowledge which perfectly matches the nature of CBR systems.

3 CBR Approach on Service Reports

3.1 Methodology

For making CBR system development as efficient and effective as possible the DISER methodology for developing CBR systems introduced by Tautz, Althoff and Nick [17,18,11] has been applied. It consists of four main development phases:

1. The creation of a *vision* that completely describes how to develop a CBR system on a highly abstract level. The main aspects in this phase are the identified *topics* and *goals* of the system. This phase also addresses whether *existing knowledge and infrastructure* can be accessed and reused. Further, the usage, creation and population of knowledge is defined in order to start a goal-oriented development process.
2. *Version 1* revisits the aspects of the previous phase and results in more detailed definitions that leads into a model and implementation of a first running system.
3. The *pilot* phase starts immediately after finishing the Version 1 phase. All features have been developed and can now be tested by users and according to the given feedback, they are refined by the developers. In this phase, change and maintenance management is established.
4. The *operation* phase starts with the system going online and providing its service to users. In this phase change and maintenance management starts.

For each CBR system to be developed the respective goals, based on the existing knowledge and infrastructure, have to be defined. Starting from these goals the subject areas have to be identified with which these goals can be achieved. For each subject area the scenarios for using the CBR system as well as the recording scenarios for filling the knowledge containers have to be defined. Based on this information the general and the case-specific knowledge is formally modeled, implemented, tested, and maintained.

Further, the research on mining textual, community based data has been carried out as a part of the further development of SEASALT, a domain independent architecture for creating knowledge-based systems based on experiences provided in web communities. SEASALT follows the idea of collaborating expert agents that join their expertise to solve complex problems. SEASALT describes on the one hand, how complex tasks can be modularized by creating so called topic agents and a knowledge line (a topic agent is a software agent that provides information on only one subtask) and also how to represent and extract knowledge out of an expert community and provide it for the overall knowledge-based system [13]. The work described in this paper focuses on the second part, the knowledge provision. The collection of machine data enhanced with the discussion between an analyst and a technician can be described as a community of experts that deals as knowledge source. In the following sections, we describe processing of the raw data in order to build a CBR system.

3.2 Source Data Analysis

After analyzing the available data with customer support and machine experts, we have created a case representation for the CBR system. Table 1 contains the structure of each case.

We have organized the attributes in various classes, however, each case in a complete instance of the given case representation. Basically each case can be divided in three parts: general case information, machine characterization and problem characterization.

Table 1. Service Report Case Representation

Concept: Meta				
	Class	Symbolic	17 values	
	Date Created		Date	
	Status	Symbolic	3 values	
	Season	Symbolic	7 values	
	Origin	Symbolic	3 values	
	Solution	String		
Concept: Problem				
	Software	Symbolic	6 values	
	Control Unit 1	Symbolic	68 values	
	Control Unit 2	Symbolic	98 values	
	Control Unit 3	Symbolic	63 values	
	Control Unit 4	Symbolic	75 values	
	Functional Code	Integer	min: 0, max: 10000	
	Functional Area	String		
	Brand Names	Symbolic	8 values	
	Mechanisms	Symbolic	16 values	
	Vehicle Parts	Symbolic	39 values	
	Problem Description		String	
Concept: Location				
	City	String		
	State	String		
	Topography	min: 0, max: 2000		
Concept: Vehicle				
	ID	String		
	Year	Integer	min: 1950, max: 2050	
	Manufacturer	String		
	Model	Symbolic	7 values	
	Series	String		
	Concept: Usage			
		Acres	Integer	min: 0, max: 20000
		Hours	Integer	min: 0, max: 5000
Concept: Discussion				
	Key Part No	String		
	Affected Control Unit	String		
	Symptom	String		
	Concept: Operating Conditions			
		Applicable	Boolean	
		Field	Symbolic	7 values
		Terrain	Symbolic	9 values
		Machine State	Symbolic	6 values
		Machine Loc	Symbolic	3 values
		Weather	Symbolic	3 values
		Activity	Symbolic	7 values
		Op Cond	String	

General case information (class Meta) contains general information about the case like the case *Class*, the *Date Created*, the case's *Origin* and whether one or more CS *Solution* is available or not. The Class is set by Customer Support analysts when the case is viewed the first time and describes the quality of the given information. The origin contains which way was used to submit the case: either using a software tool or calling the customer support by phone. The few cases submitted via phone were created during a phone call between the field technician and the Customer Support where the content of the call has been submitted later by the analyst. The *Season* is computed based on the date a case has been initially submitted. In the normal use case cases are submitted right after a problem has occurred and because of this fact we compute the season based on this date. The *Location* of a machine is described by the *City* and *State* where the machine is located as well as the *Topography*, which is the elevation of the City. This information is used to enhance the collected data with environmental information.

The second part describes the affected machine's characterization focusing on two aspects: On the one hand what kind of machine we are dealing with based on manufacturing information and on the other hand the current state of usage. The *Machine ID* is given for each machine and is usually a part of the case descriptions. Further, we use the Machine ID to extract the *model*, the *series*, the *manufacturing* factory as well as the *year* of production. Another characteristic feature of a machine is the usage until the problem appeared. The usage can be captured either as hours or acres. However, hours is the most common usage information among the CS cases.

The third part describes the problem description of the affected machine. Within this table the case-specific information is represented. The machine's problem description representation is divided in two parts: the problem description itself and the so called case text. The problem description contains a short description of the vehicle's fault. This information is reviewed by Customer Support and therewith the system can rely on that the given information describes the problem a particular case deals with. Since the problem description is a free text field it can contain all kinds of information, any of the following attributes can be populated: *Brand Names* contains names that can be used to describe a part of the machine. Further, we have different types of trouble codes which are modeled according the control unit they belong to. Like brand name, our system also scans the problem description for mechanisms, software names, and vehicle parts. We have modeled those information in different attributes rather than in one attribute to be able to assign more domain-specific similarity measures. The two attributes functional area and functional code are classifying the type of problem. This classification is assessed by Customer Support analysts and the CBR system uses this information to narrow down the potential matching cases to the relevant ones.

The Case Description attributes contain detailed information on how a machine's problem can be described. It contains observation information as well as trouble codes and information gathered during the machine's problem solving process done by the technician. The population of this attribute varies depending on who is entering data as well as how data is entered. The *Symptoms* usually pick up on the problem description and describe briefly the detected failure. The *Operating Conditions* contain observations of the environment of the machine (like weather conditions or machine status). The *Affected Control Units* contain controller codes along with more observations. *Key Part Numbers* attributes contain information of the affected or to be replaced part of the machine. The attribute Discussion Text contains the whole discussion between Customer Support and the technician.

The starting point of this development is the analysis of the data available (historic experience items), data that is possible to be collected (potential experience items) and data that should be tracked/collected in the future (future experience items).

3.3 Vocabulary

The vocabulary is one of the knowledge containers and contains the definition of information entities and structures used to express knowledge, specifies the language within an application and contains a collection of terms that cover the relevant terms within an application domain.

The vocabulary of a CBR system covers the terms a CBR system can deal with. When we are referring to the vocabulary, we look at those terms that are modeled using symbolic value ranges and are based on a list of terms. Because of this fact, we will only cover those attributes that are represented as symbols.

The selection of terms that belong to an attribute are based on the occurrence of terms in the raw data. The goal during the modeling process is to cover the subject of an attribute with as much terms as necessary. However, the development of a CBR system is an incremental process and every time when new cases are inserted it should be ensured that the existing vocabulary is able to represent the newly inserted information. If the prerequisite is not fulfilled, the vocabulary has to be extended.

Depending on the attribute, the vocabulary can be created either by using common knowledge, value ranges that are given according the application domain, or based on specific domain dependent terms and abbreviations used within a company to describe products or processes. An attribute created based on common knowledge is for example Season: there are six different seasons (spring, summer, fall, winter, rainy and sunny season) available and we have created a list containing those terms. Value ranges or a given set of symbolic values are for example the usage of a machine or the series. Companies do have specifications for those kinds of attributes and they can be directly included in the vocabulary. Both of those types have in common that the value ranges are complete from the very beginning of the project and will only change after a major changes within the company or production (i.e. introducing a new machine series). However, obtaining the domain dependent vocabulary is more challenging, because there is no specific controlled vocabulary used. As described in Bach [2] it is essential, that the vocabulary covers as much relevant terms as possible.

Therefore we have used the following approach to create symbolic attributes that contain terms describing the information provided in the cases. Since we are dealing with unstructured text and each field technician can use the terms he would like to use, we choose a different way to first eliminate the terms to be modeled and second to create relations between two or more terms.

For the identification of the relevant terms, we executed this procedure on the raw data:

1. **Extraction of occurring terms:** We first executed a string tokenization algorithm so each term can be reviewed individually. Afterwards we removed stop words in order to only review descriptive terms. On these terms we applied a basic stemming algorithm. The result is a sorted list of terms that can be further processed.
2. **Classification of occurring terms:** We review the automatically created list of terms and created classes of terms.
3. **Discussing terms with experts:** Since we deal with a specialized vocabulary we then discussed the proposed classes and the contained terms regarding completeness, correct classification, known synonyms for the terms and relations between terms (for obtaining the initial similarity measures). The results were attribute values organized in tables and taxonomies with assigned similarity measures.
4. **Create the Knowledge Model:** Based on the classification attributes that contain the according terms were created.
5. **Refine the Knowledge Model:** We further reviewed the created models with experts and discussed examples ensuring that all relevant parts are covered.

At this point the vocabulary also contains some similarity information retrieved from the discussion with the domain experts.

3.4 Similarity Measures

Similarity measures are highly domain dependent and describe how cases are related to each other. The assessment of the similarity between two cases is calculated by an amalgamation function. We differentiate two types of similarity measures: Local Similarity Measures and Global Similarity Measures. Local Similarity Measures describe the relation between two symbols while the Global Similarity Measure represents the relation among attributes. The amalgamation function uses both similarity measures to compute the similarity between two cases. The next sections will discuss first the applied Local Similarity Measures, followed by the Global Similarity Measure.

Local Similarity Measures. Depending on the given raw data and available information, we picked a similarity measure. For attributes of the data type float and integer we used distance functions, for strings, we used string match or trigram matching and for symbolic value ranges we have used either similarity tables or taxonomies as they are described in [3].

Value ranges for numeric attributes are usually limited by the lowest and highest occurring value and initially we decided on a strictly monotonicly decreasing function. However, during discussions we have occasionally adjusted these measures.

Assigning similarities for symbolic attributes is more challenging, since you have to discuss the relation between the symbolic values. We usually started out creating a taxonomy and organizing terms within it incrementally. At the point when there are relations between different nodes, we moved on and created a similarity table to represent some kind of more dimensional attributes.

Global Similarity Measure. The Global Similarity Measure defines the relations between attributes. The weight of each attribute indicates its relevance within the case. We decided to use the weighted euclidean distance for the calculation of the global similarity. The development of each attribute's weight has been carried out by discussing the importance of each attribute when trying to find a similar case for solving a certain problem.

We decided to use a weight range between 1 and 5. The most important values are weighted with 5.0 and 4.0 depending on the experts opinion on which value they would focus on. The operating condition has also been named as an important attribute, but since there are seven different attributes which can be set while also a combination of up to seven attributes is possible (e.g. $activity = driving$, $applicable = true$, $terrain = hilly$ and $weather = dry$), we decided to decrease the individual weight. Usually one to four operating conditions attributes are set.

Further, one might point out that the symptom of a problem is weighted quite low, however, there is no clear distinction between the use of the symptoms text field and the problem text field. For the population of the attributes Brand Names, Control Units, Mechanisms, Software and Vehicle Parts the algorithm looks up for matching terms in the problem text field, the symptom text field as well as in the discussion text field.

3.5 Case Population

The CBR methodology supports us representing the available experience. In terms of CBR, each case can be seen as an experience item. A CBR case contains specific knowledge applied in a specific context in order to achieve a given goal. Transferring this idea to the project, each CS Case is subject to be transferred in a CBR Case since every case contains experience how to solve a problem for a certain machine. During the discussions with the Customer Support analysts we found out how knowledge contained in cases can be generalized and transferred from one case to another. This knowledge has been modeled in the vocabulary and similarity measure container.

Once the cases are created, they are organized within the case base. Each case can be described as a specific set of experiential knowledge that contains a problem and a solution description. The content of a case is based on the information that can be provided by a technician. Since we are only inserting case information the CBR system can deal with, we match the information given with the available knowledge models, so the resulting cases are a subset of CS cases. The creation of the CBR cases can be seen as a mapping of raw data to the target cases. This mapping has to be done once the case base is created as well as for each query before the query can be submitted to the CBR system. We reuse the information extraction and mapping each time a query case is created. The mapping makes use of the vocabulary to extract relevant information and transforms the source data into CBR Cases. In this particular project we had to deal with different levels of data quality, because in some cases we had to deal with arbitrary text and other attributes contain trouble codes or company-intern terms that are very easy to extract. We have developed four different ways of how data is transferred from source to target data:

Copy Content. In this case we directly used the given data within the CS Case. There are only two different data types used: either string or date. Using this form of transformation, the case model has not been used.

Computation. In this case, the source data is used to compute an attribute value that is represented in the case model. We use this approach when we generalize attribute values in order to make them easier comparable. This has been applied to create two attributes: Based on the date a case was created we computed the season. The second attribute that we created based on the source attributes city and state is topography. Based on the city and state we used the GeoNames Web Service[1]. For the computation we used the city, state, country and the language the previous information is given in and retrieved a whole set of information about this place. From this information we extracted the elevation and added this to our case.

Model-Based IE. The Model-Based Information Extraction makes use of the vocabulary modeled previously. We use those terms and look them up in the according text field and store them in the case representation. An example for this type of extraction is the following problem description of a case:

gear box does not respond - **rb7** software download

The case model contains the terms *gear box* in the attribute vehicle parts and *rb7* in software. The CBR system recognizes both terms, extracts those and assigns them to the according attribute ($software = rb7$, $vehicleparts = gearbox$).

New Composition and Model-Based IE. The most comprehensive extraction type uses the source attributes and computes/extracts a new set of information and then provides this result to do further *copying* or *model-based information extraction*. This has been done with the Case Description

[1] http://www.geonames.org/export/ws-overview.html, GeoNames is a geographical database accessible via web services that contains over 2.8 millions populated places.

attribute, because there are more structured information included in one large text attribute. We first re-created the structure and then applied the information extraction.

The case problem description in general contains information that can be initially provided by the technician and the solution is suggested in the discussion between the Customer Support analyst with the technician. The case representation does not contain each source attribute, however, when the cases are presented to the user, they are included to provide the most comprehensive information.

3.6 Rapid Prototyping Tool myCBR

The CBR tool myCBR has been mainly developed to provide an easy-to-use interface for rapid-prototyping of CBR applications [16]. Its main focus is providing a easily manageable tool for creating CBR prototypes, especially similarity measures. Further, myCBR supports the import of cases stored in csv files and testing the created vocabulary and similarity measures.

We have used the myCBR interface to develop the knowledge model and talk with the domain experts about the definitions and relations of value ranges. myCBR provides the most common similarity measures and as an open source tool additional ones can be included as well. After creating the vocabulary and similarity measures we imported the cases from a csv file. In advance, we created cases as described in section 3.5 and loaded them into myCBR. Afterwards first retrieval tests are possible and we started first refinements on the similarity measures. The included retrieval mask and the handling of multiple case bases allows the knowledge engineer to carry out various tests.

For demonstrating the capabilities of the CBR system, we used the myCBR SDK to access the CBR system and built a JSP interface. Therewith, we were able to install the application on a web server in the company and possible users can easily access the application via a web browser. For the presentation of the results we decided to use the source data the users are familiar with and extended this with a case representation that comprises the CBR cases.

4 Experimental Evaluation

The cases and processes we deal with are rather complex so we decided to do a qualitative evaluation using experts to measure the effort of developing such a kind of CBR system. Today the analysts are not able to search within all attributes. They are only able to search within certain attributes. The problem that we have used for the detailed result analysis, for example, could not be solved with the currently available sources. In this case, the analyst or an engineer has to create a solution for this problem.

For the first test, we included 953 cases and each case can be represented using up to 40 attributes depending on the detailedness of each case. About the half of the attributes are represented as symbols and have an individual local similarity measure. We have evaluated three different aspects of our approach that came up during carrying out the project.

4.1 Extending the Case Base

The initial development has been based on a set of 953 cases covering one certain class of cases. Those cases dealt with a particular area of a machine. Extending the case base does also require an extension of the vocabulary (of symbolic attributes), value ranges (for numeric attributes), and possibly the similarity measures. After a first evaluation of the created case base, the customer requested an extension of the case base using 145 new cases that are slightly different in the problem type and extended the scope of the application. This requested extension was carried out very quickly and without any problems.

First of all, we had to analyze the data as described in section 3.2, before the extension of the vocabulary could be conducted. The term extraction is completely automated and all we had do to was inserting about 20 new terms in our taxonomies. Afterwards the csv file using the knowledge models is created and loaded into the system.

4.2 Comparing with Utility

For evaluating the quality of the CBR system, we had a customer support expert who created an example question that describes a problem, which occurred in the past, but its solution is not covered in manuals, nor the exact same case is available in the training case base.

The task for the CBR system was to come up with the best solution. The comparison of the results can only be measured by rating the quality and for this purpose the customer support experts developed their own measure to rate the utility of a case.

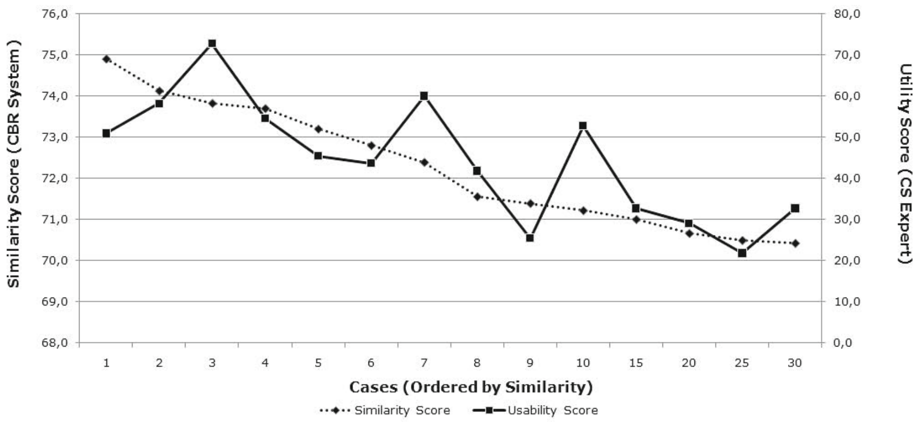


Fig. 1. CBR Similarity vs. Utility

They created 11 criteria describing the utility and rated them on a scale from 0 (this criterion is not covered in the case) to 5 (this criteria is correctly covered). The criteria used are the relevancy to the problem, how explicit the problem is described and a solution is presented. Further, the understandability of the cases and references to further readings are scored. The expert reviewed the top 10, the 15th, 20th, 25th and 30th case and rated them.

Figure 1 shows the results of the evaluation. The y-axis contains the relevance scores. On the left part you can see the similarity score of the CBR system and on the right part you have the utility score based on the rating of the expert. The x-axis contains the case rank according to the CBR system. The tendency of both, the CBR system and the expert, is alike, however the expert would have come up with a different order. Further, within the top 10 results of the CBR system were the most relevant cases to the given query. This shows that the CBR system containing semi-automatically extracted knowledge, which was slightly refined during discussions with experts produces results that match an expert's opinion. Even if the best match did not show up in first place, our tests showed that the first ten results always contained the best matching case.

4.3 Effort of Including Experts

The approach presented in this paper is aiming at as many automation during the build-up of the CBR system as possible. The process included discussion with experts, but did not require that experts had to create the knowledge models themselves. We carried out six discussions as telephone conferences or tele-presence meeting along with two two-day workshops. During the discussion there were usually two experts present. This shows that the approach makes use of the experience provided in the source cases and technical manuals from the very beginning throughout the entire project.

5 Related Work

Case-based diagnosis is a traditional topic in CBR research. The task of our approach can be compared to the diagnosis tasks described by Lenz et. al. in [7,8]. However, we decided to use a structural CBR approach, because the overall goal was to create diagnosis automatically by further developing the acquired knowledge. A Textual CBR approach could not be extended this way so easily.

The number of features and the associated values of our system are comparable to Koton's CASEY [6], a medical diagnosis system. However, our approach uses existing service reports and builds a CBR system on them while all knowledge in CASEY was created to be applied in the CBR system. Besides supporting medical diagnosis, help-desk support systems like HOMER [4] or CASSIOPÉE [5] can also be related to our work. Both approaches are Structural CBR system.

The raw data which has been created as a collaboration between technicians and analysts can be described as community experiences captured during expert discussions. Also, the type of experience we deal with according to Plaza and

Baccigalupo [12] is how-to experience, however we did not focus on the process aspects like Minor et. al. [9] do, we focused more on the extraction and provision of the whole items. Given the fact of the fast and broad dissemination of web communities and therewith the availability of huge amounts of experience knowledge, it is very promising integrating experiences in CBR systems, because the underlying methodology of CBR relies on previously made experiences. It is important to use, combine and further develop technologies that have already been applied on the Web (2.0) together with standard technologies. Therefore software engineers for CBR systems should create web-based systems, because that might be the only way to receive feedback [15]. Our approach uses web technologies to disseminate the application within a company and we have tried to be as less restrictive as possible when searching for a case. For example, the most criteria can be described in free text and the result list contains the original data. This ensures that the whole content of a case is provided to the user, if necessary.

6 Summary

This paper presented the outcomes of a research project that applies CBR as a methodology to make use of existing data. We first introduced the application domain and described the characteristics of service reports for machines before we explained how we transferred existing data to cases and filled the knowledge containers during the development of the CBR system. The evaluation of our approach shows that it can definitely be applied in this scenario. Further, during the development process, the experts confirmed that they learned new aspects of their own data and how it can be looked at in different ways depending on the goal of a system. The CBR system we have developed also demonstrated the capabilities of the technology and at the same time identified new potentials. For example, collecting more structured data can lead to more automation towards an automatic machine diagnosis and higher confidence in a proposed solution. Another aspect is collecting feedback and start to learn or improve similarity measures, which will also lead into more precise retrieval results. Of course, the more knowledge we put into the similarity measures the higher the retrieval precision will be.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 1(7) (March 1994)
2. Bach, K., Hanft, A.: Domain modeling in tcbr systems: How to understand a new application domain. In: Wilson, D.C., Khemani, D. (eds.) ICCBR 2007 Workshop Proceedings, WS on Knowledge Discovery and Similarity, pp. 95–103 (2007)
3. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. LNCS (LNAI), vol. 2432, p. 25. Springer, Heidelberg (2002)

4. Göker, M.H., Roth-Berghofer, T.: The development and utilization of the case-based help-desk support system homer. Engineering Applications of Artificial Intelligence 12(6), 665–680 (1999)
5. Heider, R.: Troubleshooting cfm 56-3 engines for the boeing 737 using cbr and data-mining. In: Smith, I., Faltings, B. (eds.) EWCBR 1996. LNCS, vol. 1168, pp. 512–518. Springer, Heidelberg (1996)
6. Koton, P.: Reasoning about evidence in causal explanations. In: 7th National Conference on Artificial Intelligence, pp. 256–263. AAAI Press, Menlo Park (1988)
7. Lenz, M.: Defining knowledge layers for textual case-based reasoning. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 298–309. Springer, Heidelberg (1998)
8. Lenz, M., Busch, K.H., Hübner, A., Wess, S.: The simatic knowledge manager. In: Aha, D., Becerra-Fernandez, I., Maurer, F., Muñoz-Avila, H. (eds.) AAAI 1999 KM/CBR Workshop, pp. 40–45. AAAI Press, Menlo Park (1999)
9. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards case-based adaptation of workflows. In: Montani, S., Bichindaritz, I. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 421–435. Springer, Heidelberg (2010)
10. Müller, T., Lange, K., Breuer, A., Krieger, O., Form, T.: Automatic and expierence-based diagnostics using distributed data and neural networks. In: 12. Internationaler Kongress Elektronik im Kraftfahrzeug, pp. 593–605. VDI Verlag (2008)
11. Nick, M.: Experience Maintenance through Closed-Loop Feedback. Ph.D. thesis, TU Kaiserslautern (October 2005)
12. Plaza, E., Baccigalupo, C.: Principle and praxis in the experience web: A case study in social music. In: Delany, S.J. (ed.) ICCBR 2009 Workshop Proc. Workshop Reasoning from Experiences on the Web, pp. 55–63 (July 2009)
13. Reichle, M., Bach, K., Althoff, K.D.: The SEASALT Architecture and its Realization within the docquery project. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 556–563. Springer, Heidelberg (2009)
14. Richter, M.M.: Introduction. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 1–16. Springer, Heidelberg (1998)
15. Smyth, B., Champin, P.A., Briggs, P., Coyle, M.: The case-based experience web. In: Delany, S.J. (ed.) ICCBR 2009 Workshop Proc. Workshop Reasoning from Experiences on the Web, pp. 74–82 (July 2009)
16. Stahl, A., Roth-Berghofer, T.R.: Rapid prototyping of CBR applications with the open source tool myCBR. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 615–629. Springer, Heidelberg (2008)
17. Tautz, C., Althoff, K.D.: A case study on engineering ontologies and related processes for sharing software engineering experience. In: Proc. of the 12th Internal Conference on Software and Knowledge Engineering, Chicago, USA (July 2000)
18. Tautz, C., Althoff, K.D., Nick, M.: A case-based reasoning approach for managing qualitative experience. In: Proc. 17th National Conference on Artificial Intelligence (AAAI 2000), Workshop on Intelligent Lessons Learned Systems, pp. 74–81 (2000)

CBR with Commonsense Reasoning and Structure Mapping: An Application to Mediation

Atılım Güneş Baydin[1,2], Ramon López de Mántaras[1], Simeon Simoff[3], and Carles Sierra[1]

[1] Artificial Intelligence Research Institute, IIIA - CSIC, Campus Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain
[2] Escola d'Enginyeria, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain
[3] School of Computing and Mathematics, University of Western Sydney, NSW 1797, Australia
{gunesbaydin,mantaras,sierra}@iiia.csic.es, s.simoff@uws.edu.au

Abstract. Mediation is an important method in dispute resolution. We implement a case based reasoning approach to mediation integrating analogical and commonsense reasoning components that allow an *artificial mediation agent* to satisfy requirements expected from a human mediator, in particular: utilizing experience with cases in different domains; and structurally transforming the set of issues for a better solution. We utilize a case structure based on ontologies reflecting the perceptions of the parties in dispute. The analogical reasoning component, employing the Structure Mapping Theory from psychology, provides a flexibility to respond innovatively in unusual circumstances, in contrast with conventional approaches confined into specialized problem domains. We aim to build a mediation case base incorporating real world instances ranging from interpersonal or intergroup disputes to international conflicts.

1 Introduction

Mediation is a process of dispute resolution where an intermediary—called a mediator—assists two or more negotiating parties to reach an agreement in a conflict, who have failed to do so on their own. In the field of law, it is defined as a form of *alternative dispute resolution* (ADR), i.e. a collection of techniques the parties might resort to instead of a judicial process, including, besides mediation, other types such as facilitation[1] and arbitration[2] [27].

Two defining aspects of a mediation process are:

– that the mediators have special training that allows them to identify issues and explore options for solutions based on their experience, often by drawing parallels with similar past cases

[1] The intermediary constructively organizes a discussion.
[2] The intermediary has the power to impose a resolution.

– and that the mediators handle the discussion with total impartiality, without having a personal stance on the discussed issues, and instead, offering to expand the discussion beyond the original dispute for allowing creative new solutions [28].

Within the field of artificial intelligence (AI), there is an active effort of research for studying negotiation processes using agent based modeling [20] and developing support tools for mediation [5]. These, together with the recent formulation of a mediation framework by Simoff et al. [32], provide a theoretical basis for a computational approach to mediation, which can promisingly address the aspects mentioned above.

In this paper, we describe the implementation of a case based reasoning (CBR) approach for an autonomous mediation system that can satisfy, to a sufficient degree, the requirements expected from a human mediator; and that can eventually tackle non-trivial disputes in a variety of problem domains. Our approach stems from, and is an improvement upon, the early case based problem solver nicknamed MEDIATOR by Simpson [33,18]. To this end, we introduce a CBR model that uses a case structure based on *ontologies* and that incorporates: (1) a structure-matching *analogical reasoning* component, which allows it to recall its experience with past cases in different domains; and (2) a *commonsense reasoning* component, which emulates, to some extent, human-like innovation in reshaping the set of issues of conflict.

The role of analogical reasoning in the CBR algorithm that we present is twofold: it forms the basis of the retrieval stage with scores based on structural evaluation of possible analogies between case ontologies; and it is used in the adaptation stage for the inference of new knowledge about the current case by means of analogical mappings from retrieved cases. The commonsense reasoning component provides modifications of existing ontologies via commonsense knowledge, aiding in the uncovering of extensive analogies in all stages of the CBR algorithm.

After providing background information on analogical and commonsense reasoning in Section 2, we present the ideas underlying our approach by means of a structure mapping example in Section 3. Details of the implementation of our model are given in Section 4, illustrated by a sample run of the CBR algorithm and the mediation process. This is followed by a discussion of building a mediation case base in Section 5. The paper ends with our conclusions and plans for future research in Section 6.

2 Background

CBR is a well-studied problem solving model in AI [17,1,23,29], utilizing past experience in the form of a case base. By its nature, the CBR model can be viewed as a type of lazy, or instance-based, learning, i.e. without making any generalizations or deriving rules on how to solve the problems in the domain, in contrast with other models such as artificial neural networks and decision trees [21,6]. Even if this lack of generalization can at times be seen as a disadvantage in particular application domains, the inherent instance-based approach of CBR renders it highly suitable as the backbone of a mediator. Due to the nature of the mediation process, instances of dispute can be conveniently represented as cases. Being kept intact for future reasoning by the CBR system, these cases also provide any decisions by the mediator with valuable explanation and backing as supporting precedents.

The case representation structure that we use here is based on ontologies describing the perceptions of the negotiating parties on the set of issues forming the dispute; and we augment the conventional CBR cycle with analogical and commonsense reasoning modules operating on these ontologies, details of which are presented further below.

2.1 Analogies and Structure Mapping

Analogy is a cognitive process where information on an already known subject (the *analogue* or *base domain*) is transferred onto a newly encountered subject (the *target domain*), through inference based on similarity. Analogical reasoning plays a crucial role in many defining aspects human cognitive capacity, such as problem solving, memory, and creativity [30,13,14]. There are several cognitive science approaches for the modeling of analogical reasoning [12], such as the *Structure Mapping Theory* (SMT) [14] and a collection of other models inspired by it, for instance the work by Ferguson [11] and Turney [35].

The very technique of CBR that we employ is occasionally emphasized as an analogy-based method, with an inherent facility of recalling instances from a case base that are similar to a given new situation. Nevertheless, in practice, CBR systems have been almost universally restricted to indexing and matching strategies as cases in a strictly defined single domain [1]. This is arguably because of the difficulty in developing implementations capable of case retrieval by inter-domain analogies, and especially, adapting the solutions of past cases into the target domain.

A pivotal part of our research is the integration of the computational implementation of SMT, the seminal *Structure Mapping Engine* (SME) [9,14] into a CBR framework. By doing so, we achieve a degree of analogical reasoning ability for recalling analogous cases of past mediations in different base domains (CBR retrieval stage), together with the ability to bring new knowledge into the target domain by analogical inference (CBR adaptation stage). This is crucial for addressing the first requirement expected from a competent mediator mentioned before.

2.2 Commonsense Reasoning

Within AI, since the pioneering work by McCarthy [24], commonsense reasoning has been commonly regarded as a key ability that a system must possess in order to be considered truly intelligent [26]. There is an active effort to assemble and classify commonsense information involved in everyday human thinking into ontologies and present these to the use of scientific community in the form of commonsense knowledge bases, of which *Cyc*[3] maintained by the Cycorp company and the *ConceptNet* project[4] of Massachusetts Institute of Technology (MIT) are the most prominent examples. The lexical database WordNet[5] maintained by the Cognitive Science Laboratory at Princeton University also has characteristics (via synonym/hypernym/hyponym relations) of a commonsense knowledge base.

The artificial mediator model implemented in this study interfaces with MIT ConceptNet and WordNet in the process of discovering middle-ground ontologies between the disputing parties and considering expansions or contractions of the involved ontologies to facilitate the analogical reasoning process, thereby allowing for solutions unforeseen before mediation.

3 Approach: A Classical Example

The orange dispute and Sinai Peninsula: To illustrate our approach to mediation and analogical reasoning, let us briefly describe a classical mediation example that embodies the essence of our approach, which was introduced by Simpson [33] and later used by Kolodner [18,17] in her seminal work on CBR. Considering a resource dispute where two sisters want the same `orange` (Figure 1(a)), a mediator first assumes that a simple division of the `orange` into two would solve the dispute, but this is unacceptable for the parties. After a point in the mediation process, it is revealed that one sister wants the `orange` for the reason of cooking a `cake` (for which its `peel` is sufficient) and the other for making a `drink` (for which its `pulp` is sufficient). The solution is then to redefine the disputed resource as an entity composed of a `pulp` and a `peel` and to assign these to the parties (dashed edges).

This simple mediation case is strikingly similar to a real world crisis in international relations, where the countries of `Egypt` and `Israel` had a dispute over the control of the `Sinai` peninsula following the Yom Kippur War in 1973 (Figure 1(b)). Starting the mediation with the initial ontology (lacking the dashed edges), through analogical reasoning by SME, we can consider a structural correspondence between concept pairs in these domains (e.g. `orange`–`Sinai`). Moreover, by this analogical mapping we can *infer* that, corresponding to `pulp` and `peel` in the base domain, there may exist two more concepts in the target domain that we can base a solution upon (linked by dashed edges in Figure 1(c)), which incidentally corresponds to a simplistic view of how the dispute was successfully mediated by the US president Jimmy Carter in 1979. The uncovering of these postulated concepts—namely, that the control of a territory has `military` and `civilian` aspects—is addressed by the commonsense reasoning module of our approach using ConceptNet. This works by generating expansions of the ontology at hand by a given factor—by attaching more concepts to existing concepts through identified commonsense relations—and making use of the robustness of

[3] http://www.cyc.com
[4] http://csc.media.mit.edu
[5] http://wordnet.princeton.edu

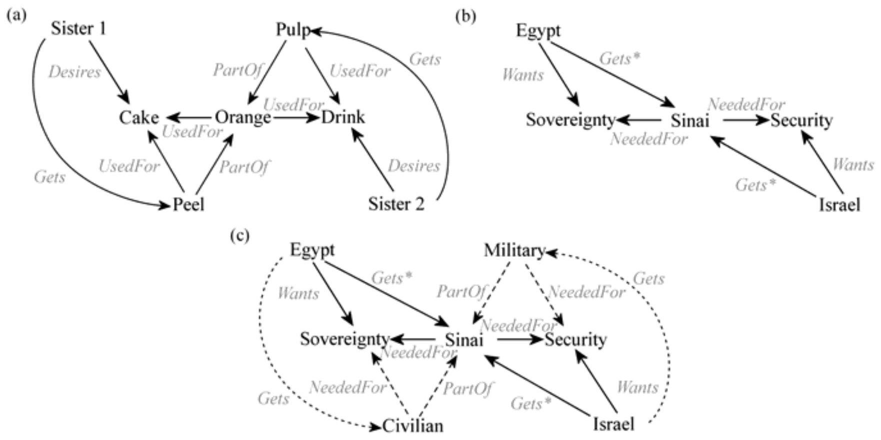


Fig. 1. Ontologies of mediation cases in the orange dispute domain (a) and the Sinai peninsula dispute domain (b and c). Vertices and edges respectively represent concepts and relations; dashed edges represent subsequently discovered relations leading to a solution; relations causing conflict are marked with "*".

SME to capture relations and concepts relevant to the considered analogy. For overcoming possibly different semantics used in naming the relations in the base and target domains (note the `desires`–`wants`, `usedFor`–`neededFor` relations in Figure 1(a) and (b)), we make use of WordNet by considering the synsets[6] in the matching of relation structures by SME.

This solution by "agreeable division based on the real goals of the disputants" can form a basis for solving many future cases of mediation involving resource allocation. Note that, by using an analogical reasoning approach, this ability is maintained regardless of the semantics of problem domains, because we reach solutions only through similarities in the ontological structures of disputes.

4 Implementation of the Mediator

The fundamental part underlying our approach is a CBR cycle integrated with analogical and commonsense reasoning components, capable of: (1) creating a middle-ground ontology representing the views of all agents in dispute; (2) using this ontology for the retrieval of cases through analogical reasoning from a case base of previous successful mediations in various domains; and (3) adapting a solution for the current case, again by utilizing the middle-ground ontology and the ontology of the retrieved previous case, taking the goals and reservations of the parties into account.

This system is to act as a mediator in-between several negotiating agents in a multiagent environment, in similar fashion to the "curious negotiator" model

[6] A *synset* or *synonym ring* is a set of synonyms that are interchangeable without changing the truth value of any propositions in which they are embedded.

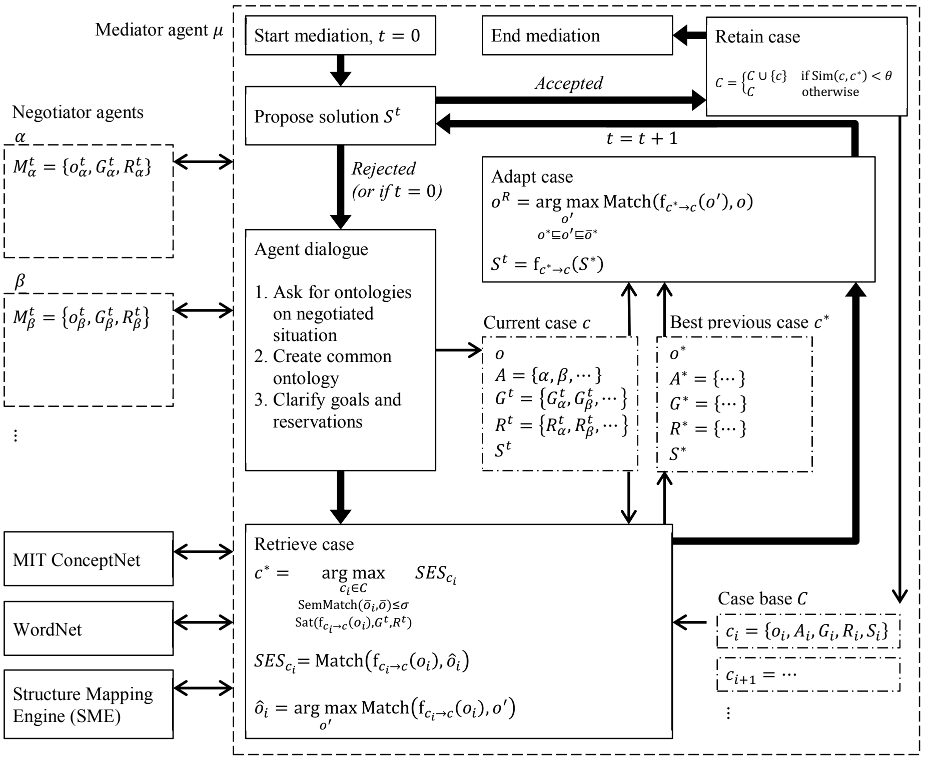

Mediator agent μ
Start mediation, $t = 0$
End mediation
Retain case
$C = \begin{cases} C \cup \{c\} & \text{if } \mathrm{Sim}(c,c^*) < \theta \\ C & \text{otherwise} \end{cases}$
Accepted
Propose solution S^t
$t = t + 1$
Negotiator agents
α
$M_\alpha^t = \{o_\alpha^t, G_\alpha^t, R_\alpha^t\}$
β
$M_\beta^t = \{o_\beta^t, G_\beta^t, R_\beta^t\}$
Rejected (or if $t = 0$)
Adapt case
$o^R = \arg\max_{o', o^* \sqsubseteq o' \sqsubseteq \hat{o}^*} \mathrm{Match}(f_{c^* \to c}(o'), o)$
$S^t = f_{c^* \to c}(S^*)$
Agent dialogue
1. Ask for ontologies on negotiated situation
2. Create common ontology
3. Clarify goals and reservations
Current case c
o
$A = \{\alpha, \beta, \cdots\}$
$G^t = \{G_\alpha^t, G_\beta^t, \cdots\}$
$R^t = \{R_\alpha^t, R_\beta^t, \cdots\}$
S^t
Best previous case c^*
o^*
$A^* = \{\cdots\}$
$G^* = \{\cdots\}$
$R^* = \{\cdots\}$
S^*
MIT ConceptNet
WordNet
Structure Mapping Engine (SME)
Retrieve case
$c^* = \arg\max_{c_i \in C, \mathrm{SemMatch}(\bar{o}_i, \bar{o}) \le \sigma, \mathrm{Sat}(f_{c_i \to c}(o_i), G^t, R^t)} SES_{c_i}$
$SES_{c_i} = \mathrm{Match}(f_{c_i \to c}(o_i), \hat{o}_i)$
$\hat{o}_i = \arg\max_{o'} \mathrm{Match}(f_{c_i \to c}(o_i), o')$
Case base C
$c_i = \{o_i, A_i, G_i, R_i, S_i\}$
$c_{i+1} = \cdots$

Fig. 2. Architecture of the implemented CBR model and the flow of the mediation process. Thick arrows represent the flow of CBR algorithm while thinner arrows represent the utilization of case data. Bidirectional arrows represent communication between components.

by Simoff et al. [31]. The following sections give a description of the model and its implementation together with the flow of the mediation process in Figure 2 and an example run in Table 1.

4.1 Case Representation

After initial dialogue between the mediator and the agents in dispute, a newly acquired dispute c is represented in the same manner as the cases in the CBR case base, except that the dispute at hand will lack a solution. The case base C holds instances of successfully ended past mediations. Each case c_i in the case base is fully described by the set

$$c_i = \{o_i, A_i, G_i, R_i, S_i\} \ , \tag{1}$$

denoting respectively the associated ontology of the dispute, the agents, their goals, their reservations, and the solution (Figure 2). Even if A_i, G_i, R_i, and S_i

Table 1. Example mediation process with the implementation. Relations between concepts are presented in LISP notation; relations causing conflict are marked with "*".

1. CBR new case	**1.b. Ontology expansion**

New case c
5 concepts, 6 relations
`(Wants Egypt Sovereignty)`
`(Wants Israel Security)`
`(NeededFor Sinai Sovereignty)`
`(NeededFor Sinai Security)`
`(Gets* Egypt Sinai)`
`(Gets* Israel Sinai)`
	Ontology expansion factor: 6.0
	30 concepts, 38 relations
	`(Wants Egypt Sovereignty)`
	`(Wants Israel Security)`
	`(NeededFor Sinai Sovereignty)`
	`(NeededFor Sinai Security)`
	`(Gets* Egypt Sinai)`
	`(Gets* Israel Sinai)`
	`(PartOf Civilian Sinai)`
	`(PartOf Military Sinai)`
	`...`

2. CBR retrieval	

Best case c^* from case base C (Total SME score: 67.28)
26 possible analogies, average SME score: 2.58, maximum SME score: 3.90
`(Desires Sister1 Cake)`
`(Desires Sister2 Drink)`
`(UsedFor Orange Cake)`
`(UsedFor Orange Drink)`
`(PartOf Peel Orange)`
`(PartOf Pulp Orange)`
`(UsedFor Peel Cake)`
`(UsedFor Pulp Drink)`
`(Gets Sister1 Peel)`
`(Gets Sister2 Pulp)` } *Solution S^**
	Best analogy for case c^*
	Analogy 1 (SME score: 3.90)
	`Sister1 -> Egypt`
	`Sister2 -> Israel`
	`...`
	25 other (rejected) analogies
	Analogy 2 (SME score: 2.70)
	`Pulp -> Israel`
	`Orange -> Middle East`
	`...`
	Analogy 3 (SME score: 2.46)
	`Glass -> Sinai`
	`Drink -> Sovereignty`
	`...`
	`...`

3. CBR adaptation	**3.b. Inferences and solution**

Best analogy $f_{c^* \to c}$ (SME score: 3.90)
`Sister1 -> Egypt`
`Sister2 -> Israel`
`Orange -> Sinai`
`Cake -> Sovereignty`
`Drink -> Security`
`Peel -> Civilian`
`Pulp -> Military`
`...`
	`(Gets Sister1 Peel)`
	`->` *`(Gets Egypt Civilian)`*
	`(Gets Sister2 Pulp)`
	`->` *`(Gets Israel Military)`*
	`...`

4. CBR retaining	

New case c solved
`(Wants Egypt Sovereignty)`
`(Wants Israel Security)`
`(NeededFor Sinai Sovereignty)`
`(NeededFor Sinai Security)`
`(PartOf Civilian Sinai)`
`(PartOf Military Sinai)`
	`(NeededFor Civilian Sovereignty)`
	`(NeededFor Military Security)`
	`(Gets Egypt Civilian)`
	`(Gets Israel Military)` } *Solution S*

already exist as subgraphs of concepts and relations embedded into the ontology o_i (Figure 1), they are also explicitly listed as features of the case c_i for indicating which concepts and relations correspond to the agents together with their goals and reservations, and also for case indexing purposes. We permit the possibility that the parties modify their stances (e.g. $M_\alpha^t = \{o_\alpha^t, G_\alpha^t, R_\alpha^t\}$ in Figure 2) after successive solution proposals S^t, where t is the time index of the number of CBR cycles in the current run.

Commonsense Reasoning Module. For enabling the discovery of extensive analogies between different domains, we treat every given ontology o as a partial view of a more general ontology $\bar{o}$, denoted $o \sqsubseteq \bar{o}$. We produce expansions of a given ontology (e.g. $o \sqsubseteq o' \sqsubseteq \bar{o}$ in Figure 2) by inserting into it new concepts and relations involving existing concepts, until the total number of concepts equals its previous value multiplied by a given expansion factor $\eta \geq 1$ (Algorithm 1). Figure 3 gives an example, for the case of orange–Sinai peninsula analogy, of how the number of discovered analogies and the average and maximum structural evaluation scores are affected by the ontology expansion factor. As illustrated by this example, it is generally observed that there exists an asymptotic upper bound for the quality of attainable analogies between two domains (Figure 3(b)). We therefore reason that, while the number of analogies keeps monotonically increasing with the expansion factor η (Figure 3(a)), the best analogy does not improve further after its maximum possible extent is uncovered by the expansions up to that point. Based on this observation, we limit the expansion factor η in our implementation by a maximum $\eta_{max} = 6$.

Algorithm 1. Ontology expansion

```
procedure EXPAND($o, \eta$)                          ▷ Ontology $o$, expansion factor $\eta$
    Set $n = \lfloor(\eta - 1)\,\mathrm{NumConcepts}(o)\rfloor$                ▷ Number of new concepts
    while $n > 0$ do
        Select $con = Random(o)$                            ▷ Random concept in $o$
        Create ontology $r$ of all concepts in relation with $con$
          from commonsense knowledge bases
        if $r \neq \emptyset$ then
            Select $new = Random(r)$                        ▷ Random concept in $r$
            Append($o, new$)       ▷ Append $new$ to $o$ with corresponding relation
            $n = n - 1$                                     ▷ New concept appended
        end if
    end while
end procedure
```

As the commonsense knowledge base, our implementation depends mainly on ConceptNet [16], part of the "Common Sense Computing Initiative" framework of the MIT Media Lab.[7] This commonsense knowledge base is being built by the contributions of a few thousand people across the world and is maintained as a simple graph of concepts and binary relations. We also make use of WordNet [10] as a commonsense knowledge base, given that, in addition to grouping words into synsets (e.g. "object, thing, article, item, entity"), it also provides a taxonomical structure defined by hyponym and hypernym relations (e.g. "dog"–"canine"–"mammal"–"vertebrate"–"animal") that are highly useful and noise-free as compared to ConceptNet.

[7] http://csc.media.mit.edu/

The implementation accesses MIT ConceptNet 4.0 as an XML service through a REST application interface[8] and WordNet 2.1 from local database files. In further stages of research, we plan to add to these the proprietary knowledge base Cyc [22], a portion of which has been recently released as an open source project called OpenCyc,[9] and information mining agents, which would crawl online textual information for specific pieces of ad hoc knowledge and deliver these to the mediation agent in a structured manner. This could be of help especially in the adaptation stage of CBR.

4.2 Retrieval

The majority of CBR systems have case specifications consisting of preselected attributes and values; and use techniques such as nearest neighbor computations or decision trees for retrieval [7]. In contrast to this, here, case retrieval is a complex task based on the structural and semantic composition of ontologies associated with cases, using SME to find cases analogous to the current dispute and not necessarily in the same problem domain.

During the retrieval process, presented in Algorithm 2, the best case c^* is selected as the case maximizing the structural evaluation score

$$SES_{c_i} = Match(f_{c_i \to c}(o_i), \hat{o}_i) \ , \tag{2}$$

between the current case c and cases c_i, where $f_{c_i \to c}(o_i)$ is the analogical mapping of o_i from the domain of c_i to that of c, and $\hat{o}_i$ is an expansion of ontology o for comparison with o_i. The expansion $\hat{o}_i$, in turn, is found by expanding o by the commonsense reasoning module

$$\hat{o}_i = \underset{\substack{o' \\ o \sqsubseteq o' \sqsubseteq \bar{o}}}{\arg\max} \, \mathrm{Match}(\mathrm{f}_{c_i \to c}(o_i), o') \ , \tag{3}$$

as to maximize its match with $f_{c_i \to c}(o_i)$ (Algorithm 1). The two constraints on the considered cases c_i,

$$\begin{gathered} \mathrm{Sat}(\mathrm{f}_{c_i \to c}(o_i), G^t, R^t) \\ \mathrm{SemMatch}(\bar{o}_i, \bar{o}) \leq \sigma \ , \end{gathered} \tag{4}$$

ensure that the selected case permits a mapping satisfying the goals G^t and reservations R^t in the current case and that the general ontologies $\bar{o}_i$ and $\bar{o}$ lie in sufficiently different domains (i.e. the concepts they include are semantically dissimilar compared with a treshold σ).

SME is very robust and quick with the computation of analogical matchings between ontologies in real time. For instance, the discovery of the 26 possible analogies between the query case and the retrieved best case given in Table 1 takes only a fraction of a second on a currently average laptop computer. This is

[8] http://csc.media.mit.edu/docs/conceptnet/webapi.html

[9] http://www.cyc.com/cyc

Algorithm 2. Case retrieval

```
procedure RETRIEVE($c, C$)                                          ▷ Current case $c$, case base $C$
    for each case $c_i$ in the case base $C$ do
        Compute $\hat{o}_i = \arg\max_{o \sqsubseteq o' \sqsubseteq \bar{o}}^{o'} \mathrm{Match}(\mathrm{f}_{c_i \to c}(o_i), o')$    ▷ Expansions $\hat{o}_i$ of ontology $o$
        Compute $SES_{c_i} = \mathrm{Match}(\mathrm{f}_{c_i \to c}(o_i), \hat{o}_i)$        ▷ Structural evaluation scores
    end for
    Select $c^* = \underset{\substack{c_i \in C \\ \mathrm{SemMatch}(\bar{o}_i, \bar{o}) \leq \sigma \\ \mathrm{Sat}(\mathrm{f}_{c_i \to c}(o_i), G^t, R^t)}}{\arg\max} SES_{ci}$
    return $c^*$                                                    ▷ Case with best matching
end procedure
```

achieved by—instead of computing every possible mapping between two ontology graphs—using an incremental procedure for combining local matches into global match hypotheses under heuristic rules warranting structural consistency [9]. Still, in the event that the mediation case base becomes prohibitively large for the computation of structural evaluation scores for each retrieval phase, a *base filtering* approach for retrieval [34] can also be employed, in effect running the analogical reasoning process on a smaller subgroup for each retrieval.

Analogy Module. For analogical reasoning between different domains, we employ our own implementation of SME as described by Falkenhainer et al.[9], a very fast analogical matching algorithm derived from SMT with a firm basis in cognitive science and often cited as the most influential work on computational analogy-making [35]. In addition to computing the match score between two ontologies (e.g. $Match(f_{c_i \to c}(o_i), o'))$, SME also provides the mapping function f between the domains, through which one can infer previously unknown information in the target domain ontology (e.g. $f_{c^* \to c}(S^*)$) (Figure 2).

Given two ontologies in different domains, SME gives a set of all structurally meaningful analogical mappings between these, each with its attached structural evaluation score. While we pick the analogy with the highest score as the basis for the analogical mapping function f, in our implementation of the $Match$ function, we sum up the scores from all possible analogies between the given two ontologies as a measure of the susceptibility of these two to analogies (Table 1, retrieval).

The discovery of analogies between ontologies by SME is guided by the structure of relations between concepts and this is dependent upon a consistent labeling of the types of relations across these ontologies. As it is highly probable that different semantics for the naming of the same relations would be used when the ontologies belong to different domains (Figure 1), our model makes use of WordNet to attach a tag of the synset of each relation within the ontologies. The SME matching then operates between these instead of the particular name of each relation.

4.3 Adaptation

In principle, the adaptation stage of our implementation falls under *substitutional adaptation*, where the substitutions are made by the analogical mapping function f from c^* to c. Hence, we get a candidate solution to the current case by the mapping

$$f_{c^* \to c}(S^*) \ , \tag{5}$$

where S^* is the solution of the retrieved case c^* (Table 1, adaptation). We use the mapping $f_{c^* \to c}$ corresponding to an analogical match established between o, the ontology of the current case, and

$$o^R = \underset{\substack{o' \\ o^* \sqsubseteq o' \sqsubseteq \bar{o}^*}}{\arg\max} \, \mathrm{Match}(\mathrm{f}_{c^* \to c}(o'), o) \ , \tag{6}$$

an expanded ontology of the retrieved case (Figure 2).

A possibility that we consider for a more powerful adaptation stage is to use a *generative adaptation* technique, where each of the derivational steps of solution from the domain of the retrieved case would be mapped into the domain of the current case, and the solution would be reached by reusing and modifying these steps for the current case. For this, the case structure has to be modified to include the solution steps in addition to the solution arrived at.

4.4 Retaining

At the point in the CBR cycle where the proposed solution is accepted by the parties in dispute, the case base C is updated to include the case c now with an accepted solution S^t. This new solution is retained whenever the newly solved case c is sufficiently different from the retrieved case c^*, compared with a similarity threshold parameter θ, in order to prevent overpopulation of the case base with instances of essentially the same dispute (Figure 2).

5 Building a Mediation Case Base

Instances of conflict cited in mediation literature range from familial disputes about inheritance or divorce to workplace disputes between coworkers, and from tenant–landlord disputes about the rent of a property to full-fledged armed conflicts between countries [8]. Even if these pose an apparent diversity, we argue that there should be a limited number of supercategories of conflicts subject to mediation—where each given conflict will be analogous to all other conflicts within its category—such as resource allocation, compensation, or scheduling. In fact, our analogical reasoning approach can be extended to discover these categories in a supplied case base in an unsupervised manner, via clustering, using the SME structural match scores as a distance metric.

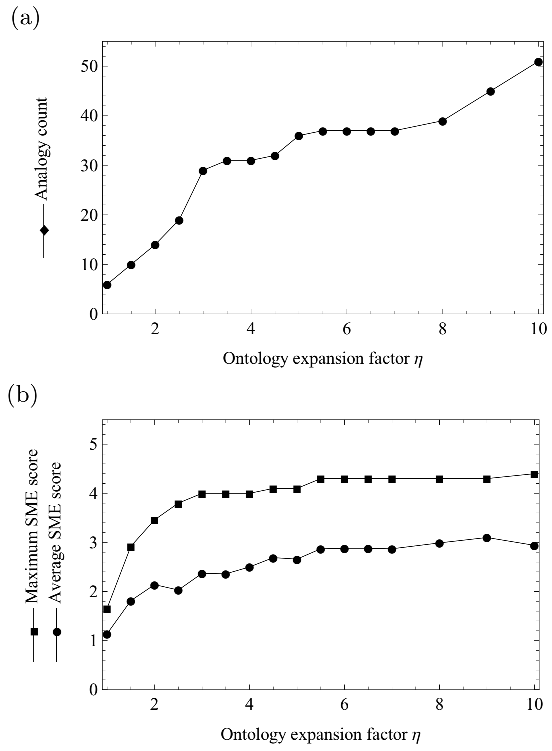


Fig. 3. Plot of the number of analogies (a) and the maximum and average SME structural evaluation scores (b) corresponding to a given ontology expansion factor η, computed for the orange dispute domain

5.1 International Conflict Databases

An important topic within mediation studies is international conflict resolution. As already exemplified in Figure 1, it is reasonable that there exists enough structural similarity between seemingly unrelated domains such as familial disputes and the resolution of international conflicts, which would allow our approach to uncover meaningful analogies. Incorporating international conflicts into the case base is desirable for benefiting from experience with non-trivial real world disputes and also rendering our research interesting from the perspective of social scientists in international relations and related fields.

There are several efforts for cataloging international conflicts, such as the *Confman* database [2], the *International Crisis Behavior* (ICB) project [4], and the *Uppsala Conflict Data Program* (UCDP) maintained by Uppsala University in Sweden and the International Peace Research Institute (PRIO) in Oslo, Norway. In particular, the UCDP Peace Agreement Dataset v. 1.0, 1989–2005 [15], which provides information on third party involvement in peace negotiations, together with the Confman database of conflict management attempts during 1945–1989, provide the best resources for our purpose.

On the other hand, as the main aim of these datasets is to index and classify conflicts according to a chosen set of features, they do not wholly submit to our approach due to a lack of descriptions of the steps in the mediation process.

5.2 Case Generation

Recognizing (1) the near absence of mediation knowledge bases that include the exact steps of mediation in each case and that contain sufficiently detailed information enabling the formulation of ontologies, and (2) that mediation-prone disputes fall into an arguably limited number of categories, we consider an approach for generating mediation cases with metaheuristic optimization techniques. We propose for future work using *genetic programming* (GP)[10] [19] for creating large numbers of ontologies that are analogous to a given ontology, guided by a fitness function based on SME structural evaluation scores. The tree-based based nature of GP makes it highly suitable for this purpose, with appropriate modifications taking the unconnected nature of ontology graphs into account.

6 Conclusion

We have presented a case based artificial mediator implementation integrating analogical and commonsense reasoning. The components of the model work together to create a system with the ability to back any of its solutions with supporting explanations, in terms of analogies with prior cases in its case base. This feature is highly advantageous within the context of law, where reference to precedent cases are deemed highly important.

In terms of practical value, the line of research following from this study has potential to find real life applications in diverse domains involving negotiation, among which law, dispute resolution, international conflict management, and commerce are foremost. These can be in the form of a support system augmenting the abilities of a human mediator, as well as in some situations replacing the human component altogether. For the case of law, this study can form a meaningful connection with several existing research efforts in the field of AI and law, such as the using of analogies in legal problem solving [3] and ethical reasoning [25].

In future work we plan to address further development of the mediation case base, largely by the case generation technique we mentioned; and improving the adaptation stage of our model, by generative adaptation. Another issue that we will concentrate on is the process of dialogue between the agents and the mediator (represented by the three steps before retrieval in Figure 2). Rather than taking the ontological perceptions of agents as given, this dialogue should be implemented from an AI argumentation perspective. Lastly, it would be certainly interesting to see our integration of structure mapping and commonsense reasoning in a CBR framework applied to problem domains other than mediation.

[10] An evolutionary optimization method working on tree structures.

Acknowledgments. This work was supported by a JAE-Predoc fellowship from CSIC, and the research grants: 2009-SGR-1434 from the Generalitat de Catalunya, CSD2007-0022 from MICINN, and Next-CBR TIN2009-13692-C03-01 from MICINN. We thank the reviewers for their constructive input.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
2. Bercovitch, J., Langley, J.: The nature of dispute and the effectiveness of international mediation. Journal of Conflict Resolution 37(4), 679–691 (1993)
3. Branting, L.K.: Reasoning with Rules and Precedents: A Computational Model of Legal Analysis. Kluwer, Dordrecht (2000)
4. Brecher, M., Wilkenfeld, J.: A Study of Crisis. University of Michigan Press, Ann Arbor (2000)
5. Chalamish, M., Kraus, S.: Automed: An automated mediator for bilateral negotiations under time constraints. In: Proc. AAMAS 2007, pp. 1–3. ACM, New York (2007)
6. Chen, D., Burrell, P.: Case-based reasoning system and artificial neural networks: A review. Neural Computing & Applications 10(3), 264–276 (2001)
7. Cunningham, P.: CBR: Strengths and weaknesses. In: del Pobil, A.P., Mira, J., Ali, M. (eds.) Proc. IEA/AIE 2011. LNCS (LNAI), vol. 2, pp. 517–523. Springer, Heidelberg (1998)
8. Domenici, K., Littlejohn, S.W.: Mediation: Empowerment in Conflict Management. Waveland, Prospect Heights (2001)
9. Falkenhainer, B., Forbus, K.D., Gentner, D.: The Structure-Mapping Engine: Algorithm and examples. Artificial Intelligence 41, 1–63 (1989)
10. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
11. Ferguson, R.W.: MAGI: Analogy-based encoding using symmetry and regularity. In: Proc. of the Sixteenth Annual Conference of the Cognitive Science Society, pp. 283–288. LEA (1994)
12. French, R.M.: The computational modeling of analogy-making. Trends in Cognitive Sciences 6(5), 200–205 (2002)
13. Gentner, D.: Structure-mapping: A theoretical framework for analogy. Cognitive Science 7, 155–170 (1983)
14. Gentner, D., Markman, A.B.: Structure mapping in analogy and similarity. American Psychologist 52, 45–56 (1997)
15. Harbom, L., Högbladh, S., Wallensteen, P.: Armed conflict and peace agreements. Journal of Peace Research 43(5) (2006)
16. Havasi, C., Speer, R., Alonso, J.: ConceptNet 3: A flexible, multilingual semantic network for common sense knowledge. In: Proc. RANLP 2007 (2007)
17. Kolodner, J.L.: Case-Based Reasoning. Morgan Kaufmann, San Mateo (1993)
18. Kolodner, J.L., Simpson, R.L.: The MEDIATOR: Analysis of an early case-based problem solver. Cognitive Science 13, 507–549 (1989)
19. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer, Dordrecht (2003)

20. Kraus, S.: Strategic Negotiation in Multiagent Environments. MIT Press, Cambridge (2001)
21. Leake, D.B.: Case Based Reasoning: Experiences, Lessons, and Future Directions. MIT Press, Cambridge (1996)
22. Lenat, D.B.: Cyc: A large-scale investment in knowledge infrastructure. Communications of the ACM 38, 33–38 (1995)
23. Lopez de Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision, and retention in case-based reasoning. Knowledge Engineering Review 20(3), 215–240 (2005)
24. McCarthy, J.: Programs with common sense. In: Symposium on Mechanization of Thought Processes. National Physical Laboratory, Teddington, England (1958)
25. McLaren, B.M.: Assessing the Relevance of Cases and Principles using Operationalization Techniques. Ph.D. thesis, University of Pittsburgh, Pittsburgh, PA (1999)
26. Minsky, M.: Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind. Simon & Schuster, New York (2006)
27. Nabatchi, T., Bingham, L.B.: Alternative Dispute Resolution Processes. In: Encyclopedia of Public Administration and Public Policy, Taylor & Francis, Abington (2004)
28. Pace University School of Law: A Guide to Careers in Alternative Dispute Resolution. Pace University School of Law (2008)
29. Rissland, E.L.: AI and similarity. IEEE Intelligent Systems 21(3), 39–49 (2006)
30. Rumelhart, D.E., Norman, D.A.: Analogical Processes in Learning. In: Cognitive Skills and Their Acquisition. Psychology Press, San Diego (1981)
31. Simoff, S.J., Debenham, J.: Curious negotiator. In: Klusch, M., Ossowski, S., Shehory, O. (eds.) CIA 2002. LNCS (LNAI), vol. 2446, pp. 104–111. Springer, Heidelberg (2002)
32. Simoff, S.J., Sierra, C., Lopez de Mantaras, R.: Mediation = Information revelation + Analogical reasoning. In: Meyer, J.-J.C., Broersen, J. (eds.) KRAMAS 2008. LNCS, vol. 5605, pp. 145–160. Springer, Heidelberg (2009)
33. Simpson, R.L.: A Computer Model of Case-based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation. Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA (1985)
34. Smyth, B., Cunningham, P.: Adaptation in real-world case-based reasoning systems. Irish Journal of Psychology 14(3) (1993)
35. Turney, P.D.: The latent relation mapping engine: Algorithm and experiments. Journal of Artificial Intelligence Research 33, 615–655 (2008)

Comparison of Reuse Strategies for Case-Based Classification in Bioinformatics

Isabelle Bichindaritz

University of Washington Tacoma, Institute of Technology,
1900 Commerce Street,
Tacoma, Washington 98402, USA
ibichind@u.washington.edu

Abstract. Bioinformatics offers an interesting challenge for data mining algorithms given the high dimensionality of its data and the comparatively small set of samples. Case-based classification algorithms have been successfully applied to classify bioinformatics data and often serve as a reference for other algorithms. Therefore this paper proposes to study, on some of the most benchmarked datasets in bioinformatics, the performance of different reuse strategies in case-based classification in order to make methodological recommendations for applying these algorithms to this domain. In conclusion, k-nearest-neighbor (kNN) classifiers coupled with between-group to within-group sum of squares (BSS/WSS) feature selection can perform as well and even better than the best benchmarked algorithms to date. However the reuse strategy chosen played a major role to optimize the algorithms. In particular, the optimization of both the number k of neighbors and the number of features accounted was key to improving classification accuracy.

Keywords: bioinformatics, feature selection, reuse, classification, k nearest neighbor.

1 Introduction

Bioinformatics research has become more and more important for the field of computer science as a whole due to the rapid expansion of the field. Often, these papers have been highlighted as the most cited in the computer science literature. The most common applications relate to data mining, for example to determine the optimal gene signature of a disease, or to classify samples into diagnostic categories. Bioinformatics applications often involve the analysis of genetic information about patients for purposes of diagnosis assessment, severity and risk assessment, treatment, and follow-up of many diseases (Cohen 2004). As a matter of fact, the range of diseases better known with these data is growing every day. Beyond the typical oncology realm, emergency medicine and primary practice are next in line for benefitting from its advances.

One of the classical tasks in bioinformatics is to analyze microarray data (see Fig. 1). These data provide information about the genetic characteristics of patients in terms of which genes are expressed at a certain point in time, and repeated measures also allow to evaluate evolution of diseases as well as response to treatment. In terms of data mining, the data are known to be highly dimensional, with a number of features ranging between thousands and several tens of thousands of features, and a number of samples being comparatively scarce, ranging from tens of samples to one hundred or a few hundreds of samples. This is due to both the cost of the studies and the small size of the populations studied. Although the availability of data is increasing, publications about algorithmic methods often compare themselves on benchmarked datasets.

Microarray data are often visualized through heat maps (see Fig.1) where rows represent individual genes and columns represent samples (Wilkinson and Friendly 2009). A cell in the heat map represents the level of expression of a particular gene in a particular sample. The color green usually represents high expression level, while the color red represents low expression level.

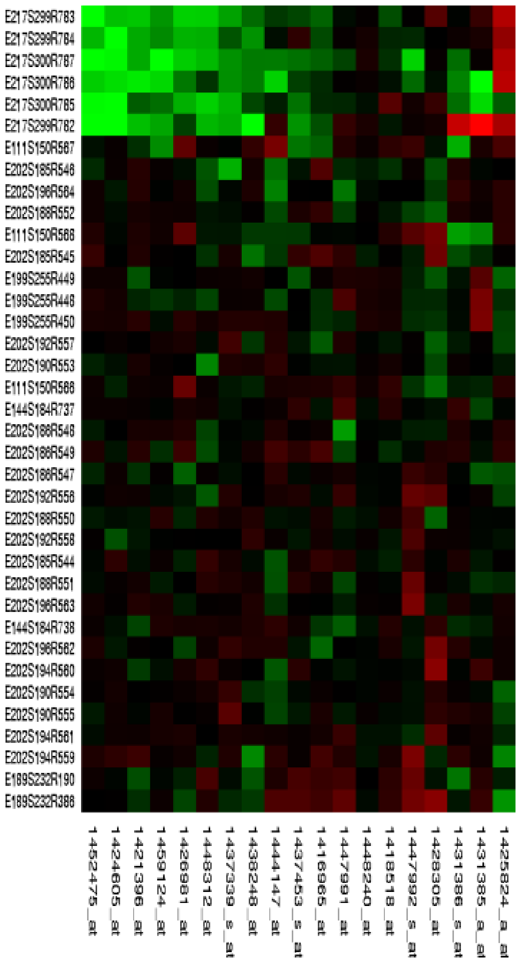


Fig. 1. A heatmap of microarray data

In practice, a microarray data row looks like a series of numbers representing the expression levels of a particular sample, with the last column representing the class, for example: (3.116, 2.004, 2.22, 2.144, 1.665, 4.161, 3.112, 2.076, … , 1). Therefore the file representation is a transposed representation of the heatmap matrix, since the rows represent samples and the columns represent genes.

This article proposes to evaluate major methods related to similarity-based classification on some of the most studied datasets in microarray classification and to answer several important methodological questions when applying in particular case-based classification to these types of data. More specifically, the importance of feature selection methods to preprocess the data has been shown in many publications on highly dimensional data (Jurisica and Glasgow 2004). The first question addressed concerns the reuse strategies relative performance of k nearest neighbor (kNN) and other case-based classification methods. The second question addressed concerns the advantage of using a feature selection algorithm called between-group to within-group sum of squares (BSS/WSS) to preprocess the features before applying a kNN algorithm.

The results presented in this article confirm that combining feature selection methods with kNN – as was previously shown with other supervised classifiers – provides the best strategy for classifying highly dimensional data, and that in most cases gene signatures of less than 100 genes yield better accuracy than classifying on thousands of genes – although the results on these were quite encouraging. The comparison of several reuse strategies shows that some perform better than others in some cases – particularly highly dimensional data. The best methods in the experiments presented in this article have been kNN with voting and with best *k* determined by leave-one-out cross-validation on the training set, and class-based classification. This latter reuse strategy does not require knowledge of a specific *k*, which is advantageous from an efficiency standpoint. The results obtained can serve the reader when conducting analyses of data involving microarray.

This article is organized as follows. The second section presents the methods used, namely the different reuse strategies for case-based classifiers and the feature selection algorithm. The third section details the experimental setting and results, including the datasets used, comparative performance results with and without feature selection, and evaluation set-up choices. A conclusion follows the discussion.

2 Methods

The main focus of this article is on instance-based classification, referred to here as case-based classification. Four of these case-based classification algorithms are presented in this section. Each can be in turn applied to samples pre-processed by a feature selection algorithm (see Fig. 2). Based on prior experiments (Bichindaritz 2010), we selected one feature selection algorithm which performed better than the other ones in our tests. In turn, the feature selection method chosen provides weights, in the form of probabilities, which can be included in the distance measure. This yields another set of four algorithms with feature weighting.

Reuse Strategies

The algorithms chosen are the kNN algorithm and its weighted variations and the class-based algorithm (Bichindaritz 2011). The reuse strategies include the majority voting strategy, the averaging strategy, and the weighted averaging strategy. In addition, we present a class-based strategy which combines similarities from all the samples in memory.

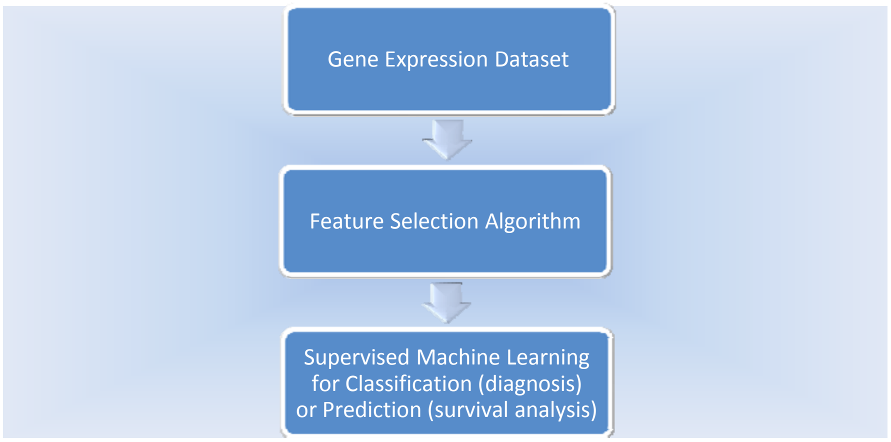

Gene Expression Dataset
Feature Selection Algorithm
Supervised Machine Learning for Classification (diagnosis) or Prediction (survival analysis)

Fig. 2. Process-flow diagram illustrating the use of feature selection and supervised machine learning on gene expression data

k Nearest Neighbor with majority voting. The kNN algorithm, underlying case-based classification, bases its classification recommendations on similar examples or cases in memory. Based on a defined distance measure – or its corresponding similarity measure – the algorithm selects the *k* samples x_i from the training set being closest, according to a distance measure, such as the Euclidian distance *dist* (1), to an example to classify and associates to this new sample a class based on the majority vote of the similar examples (2). Given a training set of *n* examples, $TrainSet = \{1 \leq i \leq n, (x_i, c_i)\}$, where x_i is a vector of *N* features and c_i its corresponding class, and a test set of *m* examples, $TestSet = \{1 \leq j \leq m, y_j\}$, a new example y_j is attributed a class c_j through equation (2).

$$dist(x_i, y_j) = \frac{\sum_{k=1}^{N}(x_{i,k} - y_{j,k})^2}{N} \tag{1}$$

$$c_j = majority\ vote\ \{x_i\ /\ arg\ \ min_K\ dist(x_i, y_j)\} \tag{2}$$

k Nearest Neighbor with averaging. When the classes are relabeled in an ordinal manner, another variant is to sum all the distances *dist* from these nearest neighbors and to calculate the class by rounding the sum of distances (3).

$$c_j = round(\ mean(\ \{c_i\ /\ arg\ \ min_K\ dist(x_i, y_j)\})) \tag{3}$$

k Nearest Neighbor with weighted averaging. The weighted *k* nearest neighbor or weighted kNN introduces a notion of weight for neighbors in the predicted class calculation.

For neighbor weighting, the algorithm, known as weight adjusted kNN, learns weights for each neighbor to advantage closest neighbors and penalize farthest neighbors. For example, one possible formula, used in this article, is provided in equation (4) for the weight $w_{i,j}$ associated to the i^{th} nearest neighbor in the calculation of the average in equation (3). This weight multiplies the class value c_i in the average formula (3). This version of the nearest neighbor is close to a kernel based classification, since the weight function is equivalent to a kernel, except that not all training examples are used in the calculation of the class but only the nearest neighbors – which amounts to setting the kernel function to zero outside of the nearest neighbors.

$$w_{i,j} = \frac{1}{dist(x_i, y_j) + 0.01} \tag{4}$$

Class-based. The class-based algorithm is a variation of the kNN. The distance measure provided in equation (1) affords the measure of the distance between each training example and each test example. The class for a given test example is predicted by averaging all distances within each class then determining which of these averages is the smallest. The test sample is classified in class z, of cardinality n_z in the training set, among C classes, with the smallest average (5), corresponding to the greatest similarity. The advantage of this method is that it is independent from the value of k in kNN since it calculates a global similarity score per class. Therefore, it will be interesting to verify in the experimental section whether this algorithm performs as well as the kNN algorithms presented above.

$$c_j = arg\ min_C\ mean_{n_z}\ dist(x_i, y_j) \tag{5}$$

Feature weighting. Feature weighting can also be introduced to all the strategies above. This method simply modifies the similarity calculation by introducing weights for each feature. Therefore, this method can be added to the four strategies above.

For feature weighting, weights can be either imported from another algorithm, such as posterior probabilities from a Bayesian algorithm or as a ratio such as the one presented in the next sub-section, or from an expert. The distance formula would be the same and describes the calculation of a weighted sum in which w_k represents the weight associated with the k^{th} feature (6). The mechanisms for calculating the classes on the test set would continue to be schemes (2), (3), (4), or (5).

$$dist(x_i, y_j) = \frac{\sum_{k=1}^{N} w_k (x_{i,k} - y_{j,k})^2}{\sum_{k=1}^{N} w_k} \tag{6}$$

Feature Selection Algorithm

The feature selection algorithm (Liu and Motoda 2008) selected is the between-group to within-group sum of squares (BSS/WSS) algorithm.

BSS/WSS. This feature selection method, developed by Dudoit et al. (2002), ranks features according to a ratio such that features with large variation between classes and small variations within classes are given higher ratings. This univariate feature selection algorithm determines features having higher discriminating power between classes. For feature k, $x_{i,k}$denotes the value of feature k for training example i, $\overline{x_{z,k}}$ the average value of feature k over the examples of class z, and $\overline{x_k}$ the average value of feature k over all the examples. The BSS/WSS ratio of gene k is provided by equation (7) where $\delta_{i,z}$ is equal to 1 if example i belongs to class z, and 0 otherwise.

$$\frac{BSS(k)}{WSS(k)} = \frac{\sum_i \sum_z \delta_{i,z} (\overline{x_{z,k}} - \overline{x_k})^2}{\sum_i \sum_z \delta_{i,z} (x_{i,k} - \overline{x_{z,k}})^2} \tag{7}$$

Accordingly, features can be ranked by decreasing order of BSS/WSS ratio. This ratio can be used as a weight associated with a feature, since the larger it is, the more important is the discriminating power of the feature.

One interesting question for this kind of feature selection, which simply provides a ranking of genes, consists in knowing which subset of features to select. The method generally used to address this question is to study the curve of the accuracy from the number of features and to find a local, or best, a global maximum after which the accuracy decreases as features are added, for example using leave-one-out cross-validation on the training set. More simply, it is also possible to select a fixed number of features, or to cut the set of selected features at a natural break in the complete list. This is the method we have chosen here.

Evaluation Methods

The evaluation methods are the independent training and test sets and the cross validation methods.

Independent training and test sets. This method is favored in bioinformatics where benchmarked datasets are often provided in the form of independent training and test sets.

Cross validation. Cross validation can be used when independent training and test sets are not available. It consists in dividing a single dataset in a certain number *k* of folds, often 10 (Witten 2005). Each fold is a random partition of the dataset and the algorithm is run *k* times, each time consisting in choosing one subset as the test set, and the other *k*-1 subsets as the training set. The results of the algorithm are obtained by averaging or combining the results from each fold. *k*-fold cross-validation can be stratified, which means that each class is equally represented in each fold. Another variant is the leave-one-out cross-validation (LOOCV), in which the test set is reduced to a single example during each fold, and is equivalent to a *k*-fold cross-validation where *k* is equal to the size of the dataset. This method is used in particular for small datasets.

k Selection Algorithm

In kNN algorithms, one important question is how to select the optimal number of neighbors to take into account. One promising method is to calculate the optimal *k* by running a leave-one-out cross-validation on the training set for a range of values of *k*, for example [1..10], and selecting the *k* yielding best results on the training set. This is the method chosen in this article.

3 Results

The results presented in this section need to be taken in the context of the datasets, hardware, and software chosen. We performed two different sets of experiments. The first set, on moderate (less than 5,000 genes) dimensionality datasets, is meant to evaluate the reuse strategies comparatively on all features and on selected features. The second set, on large (more than 5,000 genes) dimensionality datasets, is meant to evaluate the reuse strategies on subsets of features since the evaluation on all features was not an option with the experimental infrastructure.

Table 1. Summary of Moderate Dimensionality Datasets

Dataset	Total Number of Samples	# Training Samples	# Test Samples	Number of Genes
Leukemia 2 classes	72	38	34	3051
Leukemia 3 classes	72	38	34	3008
Breast cancer	72	22	0	3226

Datasets

For comparison purposes, three datasets of moderate dimensionality among the most benchmarked have been selected: the Leukemia dataset with 2 classes, the Leukemia dataset with 3 classes, and the Hereditary Breast Cancer dataset with 3 classes (see Table 1). These datasets are already preprocessed so that their dimensionality is moderate. Some results on these datasets have been published in another paper (Bichindaritz 2011) however the focus on this paper was to compare case-based classification with other classification methods, while the focus of this paper is to compare reuse strategies.

The Leukemia dataset originally consisted of 7129 genes, 38 samples in the training set, and 34 in the test set, and exists in two formats – 2 classes or 3 classes. Golub et al. describe the process they applied to filter out the genes not exhibiting significant variation across the training samples, leaving a dataset with 3051 genes (Golub et al. 1999). The samples belong to either Acute lymphoblastic leukemia (ALL), or Acute myeloid leukemia (AML) (see Table 1). In the 3 classes dataset, the ALL class was further divided into two subtypes of ALL: B-cell and T-cell (see Tables 3).

Table 2. Classes of Leukemia dataset with 2 classes

Class		Training Set	Test set
ALL	0	27	20
AML	1	11	14
		-------	-------
Total		38	34

Table 3. Classes of Leukemia dataset with 3 classes

Class		Training Set	Test set
AML	0	11	14
ALL-B cell	1	19	19
ALL-T cell	2	8	1
		-------	-------
Total		38	34

The Hereditary Breast Cancer dataset consisted of 3226 genes and 22 samples. There is no test set. The sample comprises 15 samples of hereditary breast cancer, 7 with the BRCA1 mutation and 8 with the BRCA2 mutation, and 7 samples of primary breast cancer (see Table 4).

Table 4. Classes of Hereditary Breast Cancer dataset with 3 classes

Class		Training Set	Test set
BRCA1	0	7	0
BRCA2	1	8	0
Primary	2	7	0
		-------	-------
Total		22	0

Table 5. Summary of Large Dimensionality Datasets

Dataset	Total Number of Samples	# Training Samples	# Test Samples	Number of Genes	Number of Classes
E-GEOD-10334 (Dataset 1)	247	123	124	54674	2
E-GEOD-5406 (Dataset 2)	210	105	105	22282	3
E-GEOD-13425 (Dataset 3)	190	95	95	22276	5
E-GEOD-13904 (Dataset 4)	247	123	124	54674	5
E-GEOD-4290 (Dataset 5)	180	90	90	54612	4
E-GEOD-9635 (Dataset 6)	186	93	93	59003	5

Another set of large datasets, un-preprocessed, was chosen from the microarray discovery challenge of RSCTC'10 conference (Wojnarski et al. 2010) in order to compare results with the winning algorithms at this competition. These datasets are referred to as Dataset 1 through 6 in future sub-sections (see Table 5).

Software and Hardware

The experiments were conducted on an Intel Pentium P6100 CPU at 2.00 GHz with 4GB of RAM.

Software used under Windows 7 home premium 64-bit operating system was R version 2.12.2. The case-based and feature selection algorithms were all developed under R for this article.

Table 6. Algorithms evaluated

Abbreviation	Description
kNNV	**kNN algorithm with voting**
kNNA	**kNN algorithm with averaging**
kNNWA	**kNN algorithm with averaging and neighbor weighting adjusted**
CNNA	**Class based algorithm with averaging**

Algorithms

The different algorithms evaluated and their combinations are summarized in Table 6. The first four correspond to the case-based reuse strategies.

For each of these, tests were performed on the complete set of features, for the moderate size datasets, and on a subset of features. In each case, weights learned from the feature selection algorithms were also injected in the algorithms that allowed for it

– namely the four case-based methods. In addition, tests were performed either on independent training and test sets, or with cross validation – whenever applicable.

Performance on Moderate Dimensionality on All Features

The evaluation methods are the independent training and test sets and the cross validation methods. Performance is measured with accuracy, which is the percentage of correctly classified instances. This performance measure was used to compare with the articles presenting results on these datasets. All best results were obtained for k = 1. The best k was not calculated by LOOCV due to the computing time required.

For independent training and test sets, the hereditary breast cancer could not be evaluated since this dataset only has one training set. LOOCV was used instead.

Table 7. Summary of performance *on all 3051 features* of the Leukemia 2 dataset and *on all 3008 features* of the Leukemia 3 dataset with independent training and test sets

Algorithm	#errors Leukemia2 (/34)	Average accuracy	#errors Leukemia3 (/34)	Average accuracy
kNNV	**2**	**94%**	**2**	**94%**
kNNA	**2**	**94%**	**2**	**94%**
kNNWA	**2**	**94%**	**2**	**94%**
CNNA	**3**	**91%**	**2**	**94%**
kNNV+feature weights	**2**	**94%**	**2**	**94%**
kNNA+ feature weights	**2**	**94%**	**2**	**94%**
kNNWA+feature weights	**2**	**94%**	**2**	**94%**
CNNA+ feature weights	**3**	**91%**	**2**	**94%**

Table 8. Summary of performance *on all 3226 features* of the hereditary breast cancer dataset with LOOCV cross validation

Algorithm	# errors Breast cancer (/22)	Average accuracy
kNNV	5	77%
kNNA	5	77%
kNNWA	5	77%
CNNA	9	59%
kNNV+feature weights	5	77%
kNNA+feature weights	5	77%
kNNWA+feature weights	5	77%
CNNA+feature weights	9	59%

Table 7 presents the performance of the four algorithms from Table 6. The three kNN strategies perform the best on the Leukemia 2 dataset with the same performance. On the leukemia 3 dataset, the results of the four algorithms are identical. . Class-based performed slightly less well on Leukemia 2 but as well on leukemia 3. It is notable that the best published results on these datasets are 2 classification errors, with feature selection. We obtain the same accuracy without any feature selection in these experiments. However, in a sense the number of features has been reduced already by Golub et al. (Golub et al. 1999).

Table 8 presents the results on hereditary breast cancer on all 3226 features and with LOOCV since there is no independent test set for this dataset. The best results obtained were 77% with the kNN algorithms.

Table 9. Summary of performance *on 16-20 selected features* with independent training and test sets on Leukemia 2 and Leukemia 3 datasets

Algorithm	#errors Leukemia2 (/34)	# errors Leukemia3 (/34)	Average accuracy
BSS/WSS+kNNV	1	1	97%
BSS/WSS +kNNA	1	1	97%
BSS/WSS +kNNWA	1	1	97%
BSS/WSS +CNNA	1	1	97%
BSS/WSS+kNNV+feature weights	1	1	97%
BSS/WSS +kNNA+feature weights	1	1	97%
BSS/WSS +kNNWA+feature weights	1	1	97%
BSS/WSS +CNNA+feature weights	1	1	97%

Performance on Moderate Dimensionality on Selected Features

The BSS/WSS feature selection algorithm was used to select features before running the same four algorithms on the reduced datasets. The number of selected features was taken from the articles having the best results so far on these datasets. The 20 features for Leukemia 2 and 16 features for Leukemia 3 were selected by BMA with nbest=20 and p=1000 (Yeung et al. 2005). For hereditary breast cancer, 18 genes were selected with nbest=50 and p=3226 (Yeung et al. 2005).

Table 10. Summary of performance *on 18 selected features* with leave-one-out cross-validation on Hereditary breast cancer (Hbcr) dataset

Algorithm	#errors Hbcr (/22)	Average accuracy
BSS/WSS+kNNV	0	100%
BSS/WSS +kNNA	0	100%
BSS/WSS +kNNWA	0	100%
BSS/WSS +CNNA	0	100%
BSS/WSS+kNNV+feature weights	0	100%
BSS/WSS +kNNA+feature weights	0	100%
BSS/WSS +kNNWA+feature weights	0	100%
BSS/WSS +CNNA+feature weights	0	100%

Yeung et al. report best results of 3 classification errors on the Leukemia 2 dataset and 1 error on the Leukemia 3 dataset with Bayesian Model Averaging (BMA) for feature selection and averaging regression on all the models selected.

With BSS/WSS, which ranks all the features, we selected the top 16-20 genes, depending on the algorithm. All the BSS/WSS and case-based classification algorithms provided the best results with 97% classification accuracy and 1 error (Table 9), which is the same result as the BMA + regression method for Leukemia 3, but much better than that method for Leukemia 2 – 1 error instead of 3.

Table 11. Summary of balanced accuracy in the discovery challenge datasets *on selected features* with independent training and test sets with balanced accuracy. *fw* stand for feature weighting.

Algorithm / Dataset	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Average
BSS/WSS+kNNV	**92.4%**	**68.0%**	**93.4%**	**52.2%**	**74.4%**	**64.9%**	74.2%
BSS/WSS +kNNA	**92.4%**	**68.0%**	**93.4%**	**52.2%**	**74.4%**	**64.9%**	74.2%
BSS/WSS +kNNWA	**92.4%**	**68.0%**	**93.4%**	**52.2%**	**74.4%**	**64.9%**	74.2%
BSS/WSS +CNNA	**89.7%**	**63.0%**	**84.0%**	**53.3%**	**67.1%**	**59.5%**	69.4%
BSS/WSS+kNNV+fw	**92.4%**	**68.0%**	**93.4%**	**52.2%**	**74.4%**	**64.9%**	74.2%
BSS/WSS +kNNA+fw	**92.4%**	**68.0%**	**93.4%**	**52.2%**	**74.4%**	**64.9%**	74.2%
BSS/WSS +kNNWA+fw	**92.4%**	**68.0%**	**93.4%**	**52.2%**	**74.4%**	**64.9%**	74.2%
BSS/WSS +CNNA+fw	**89.7%**	**63.0%**	**84.0%**	**53.3%**	**67.1%**	**59.5%**	69.4%

All best results were obtained for $k = 1$ – which was also the optimal value of k obtained by LOOCV on the training set. Other values of k often produced optimal results, but not regularly so, depending upon the algorithm and the value of k.

For hereditary breast cancer, we selected 18 genes with BSS/WSS, the same number selected by BMA (Yeung et al. 2005). Table 10 indicates that all the case-based classification strategies yielded the same accuracy of 100% since no error was detected. Yeung et al. report 6 errors out-of 22 on the same dataset.

Overall, the combination of the feature selection algorithm and the case-based classification algorithms consistently provided the best results in comparison with those published in the literature (Yeung et al. 2005), which is a very encouraging result. These datasets did not permit to separate the case-based classification reuse strategies, which all performed as well.

Performance on High Dimensionality on Selected Features

Due to the size of these datasets, the experiments deal only with a subset of features selected by BSS/WSS. The evaluation methods are the independent training and test sets and balanced accuracy as described in the discovery challenge (Wojnarski et al. 2010). Balanced accuracy is defined as calculating the regular accuracy independently for each class, then averaging over all the classes. This yields one accuracy figure per dataset. These accuracies are then averaged on the 6 datasets. This performance measure was used to compare with the article presenting results on these datasets. The best result in the competition for this task was 73.9% among 93 evaluated submissions – and 226 participants overall.

Table 11 shows the results of our experiments, comparing all the algorithms. Results show that all the case-based classification methods, at the exception of CNNA, combined with BSS/WSS feature selection, provide better balanced accuracy than the best result in the competition, even if it is not statistically significant. Moreover, the case-based methods show exactly the same results on this task. For kNNV, kNNA, and kNNWA, k was selected with LOOCV on the training set. For example, for dataset 1, the best k for kNNV was obtained for $k = 5$ (see Table 12). As can be seen in Table 12, results were very different depending upon the value of k, and generally not the best for $k = 1$ – by contrast with results on the moderate dimensionality datasets. The optimal value of k ($k = 5$ for Dataset 1 for example) was successfully found by LOOCV on the training set.

Table 12. Comparison of balanced accuracy for different values of *k* for kNNV on Dataset 1. Best *k* selected by LOOCV on the training set is indicated in bold (*k* = 5).

k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
78.67 %	82.88 %	86.65 %	84.87 %	**92.44 %**	84.87 %	92.44 %	85.65 %	92.44 %	84.87 %

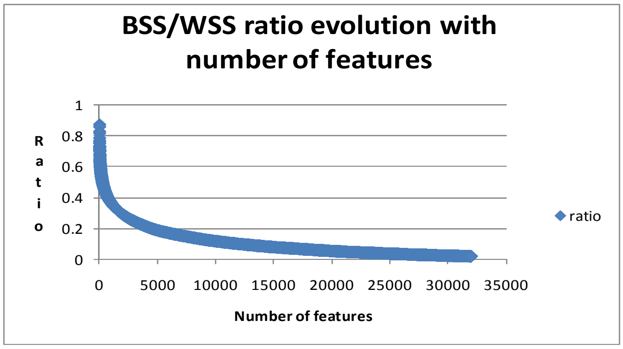


Fig. 3. Evolution curve of BSS/WSS ratio as a function of the number of features

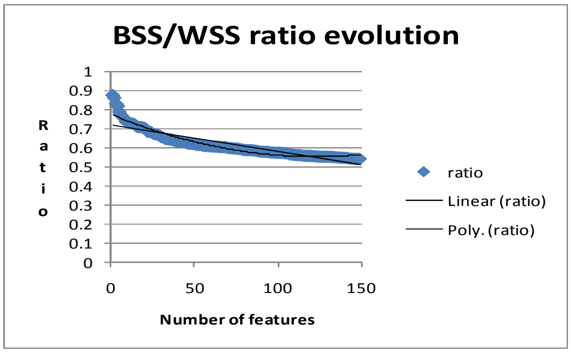


Fig. 4. Evolution curve of BSS/WSS ratio as a function of the features 1 through 150 showing an inflexion point around feature N=34

The best overall results were obtained for kNNV, kNNA, and KNNWA. kNNV results were obtained with *k* obtained with LOOCV. CNNA does not need a value for *k*, which ended up in significant time saving. The best average balanced accuracy was 74.2%, which is higher than the 73.9% result at the competition.

It is also interesting to note that feature weighting did not provide added value – nor worsening value. This points toward a need to explore improved feature weighting methods as explained in the discussion.

Finally feature selection was instrumental to improving the classification performance. First, we tried to take a fixed number of genes – say 20, based on the moderate dimensionality datasets results. We reached best average balanced accuracy of 64.99% with class-based NN (CNNA). We then opted to select the optimal number of features based on studying the curve of decrease in BSS/WSS ratios. The detection of an inflexion point marked the value at which to limit the number of features retained. Fig.3 shows the curve of decrease of BSS/WSS ratios as a function of the number of features selected on a dataset. Fig. 4 shows the inflexion points and Fig. 5 the curve of balanced accuracy as a decreasing function of the number of features selected on this dataset.

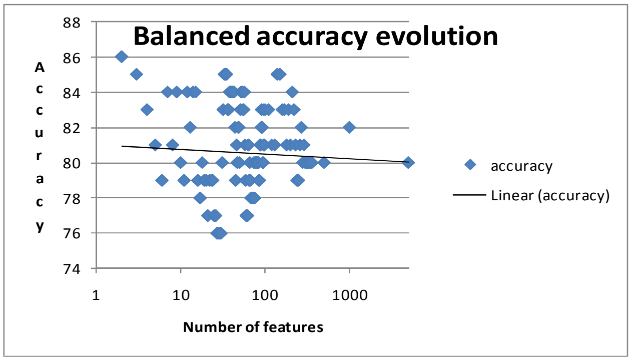


Fig. 5. Evolution curve of balanced accuracy as a function of the number of features

Table 13. Optimal number of features *n* and number of neighbors *k* selected for each high dimensionality dataset

Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
n=20 / k=5	n=11 / k=1	n=1000 / k=1	n=13 / k=7	n=81 / k=5	n=500 / k=2

Resulting from this selection method, Table 13 shows the optimal number of features and the optimal value of *k* selected for each algorithm. One can see from these results that the size of the gene signature can vary greatly between datasets.

4 Discussion

In conclusion these experiments show the usefulness of feature selection to both improve the efficiency and effectiveness of classification on highly dimensional data. However, the case-based classification algorithms performed quite well on all the features – at the same level as the best results published so far for some of the moderate dimensionality datasets – Leukemia 2 and 3 – but not for the Hereditary breast cancer dataset. BSS/WSS based feature selection improved the classification accuracy in all cases and led to better classification accuracy than published results.

The best system on these same datasets, which constituted the basic track at RSCTC'2010 discovery challenge, describes a feature selection method with multi-class Wilcoxon criterion. After feature selection, a classification is applied that averages on all features the distance between the value of this feature for a sample to the average of the feature per class on the dataset (Wojnarski et al. 2010). Therefore this method is close to the class-based method described in this paper. The competition also had an advanced track, which describes the best paper method as a feature selection algorithm based on cut point importance and dynamic clustering (Wojnarski et al. 2010). Our results on the importance of feature selection are in line with this research.

In terms of comparison of reuse strategies, in all experiments, the case-based methods provided the same accuracy as the best published results, or better, for moderate dimensionality datasets. The kNN variants yielded exactly the same results, surprisingly, which did not permit to differentiate between them on these datasets.

However, for highly dimensional datasets, the case-based strategies provided either comparable or higher accuracy than the best published results. The best average balanced accuracy was obtained for kNN with voting, kNN with averaging, and weight adjusted kNN with the optimal *k* selected automatically through LOOCV on the training set. This method chose the optimal value for *k* in all our tests. It was important to have the possibility to select *k* automatically from the training set because results varied a lot depending on the value of *k*. The class based classification performed less well but had the advantage of not requiring to choose a given number of neighbors – the *k* value.

In terms of feature selection, our results with BSS/WSS outperform significantly the baseline methods of the competition – namely Baseline-relief-1NN (65.1%), Baseline-corTest-1NN (64.1%), and Baseline-tTest-1NN (63.5%). These figures indicate that the feature selection method and strategy plays a major role in the performance of the case-based classifier.

The experimental setting of this article is based on comparing average classification accuracy – or error rate - on datasets benchmarked in recent publications in prominent bioinformatics journals or at the RSCTC'10 discovery challenge. This experimental choice respects the experimental settings chosen by the authors of these publications (Golub et al. 1999, Yeung et al. 2005, Annest et al. 2009) or by the competition (Wojnarski et al. 2010). However we would like to apply different experimental settings such as the ones presented by Demsar (2006) in machine learning and in bioinformatics by Truntzer et al. (2007). In addition, we plan to expand our tests in particular for cost-sensitivity since in biomedical domains, the cost of a false-negative is higher than the cost of a false-positive.

The experiments conducted point toward two areas of study. One is changing the feature weighting method since feature weighting did not modify the results significantly in this article. We need to calculate the weights differently – for example by adding 1 to all the weights, instead of taking the raw weights from the BSS/WSS algorithm. The second area to study is the design of a novel reuse strategy that will locally select the optimal number of neighbors based on their distribution around the new case to classify. Such a method would have the advantage of avoiding to have to run LOOCV on the training set to determine the optimal value of *k*, but instead would tailor on the fly the optimal value of *k* on a case by case basis.

5 Conclusion

In conclusion, case-based classifiers combined with feature selection performed either as the best or among the best classifiers in comparison with the literature on bioinformatics benchmarked datasets. These results are encouraging for the future integration of genetic data and medical data in case-based decision support systems in translational informatics. Optimizing the value of *k* in kNN and the number *n* of features in feature selection are key to improving the classifier's performance.

References

1. Aha, D.W., Kibler, D., Albert, M.K.: Instance-Based Learning Algorithms. Machine Learning 6, 37–66 (1991)
2. Annest, A., Bumgarner, R.E., Raftery, A.E., Yeung, K.Y.: Iterative Bayesian Model Averaging: a method for the application of survival analysis to high-dimensional microarray data. BMC Bioinformatics 10, 10–72 (2009)
3. Bichindaritz, I., Annest, A.: Case-Based Reasoning with Bayesian Model Averaging: An Improved Method for Survival Analysis on Microarray Data. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 346–359. Springer, Heidelberg (2010)
4. Bichindaritz, I.: Methods in Case-Based Classification in Bioinformatics: Lessons Learned. In: Perner, P. (ed.) ICDM 2011. LNCS (LNAI), vol. 6870, pp. 300–313. Springer, Heidelberg (2011)
5. Cohen, J.: Bioinformatics – An Introduction for Computer Scientists. ACM Computing Surveys 36(2), 122–158 (2004)
6. Demsar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)
7. Dudoit, S., et al.: Comparison of discrimination methods for the classification of tumors using gene expression data. J. Am. Stat. Assoc. 97, 77–87 (2002)
8. Furnival, G., Wilson, R.: Regression by Leaps and Bounds. Technometrics 16, 499–511 (1974)
9. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiui, M.A., Bloomfield, C.D.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286, 531–537 (1999)
10. Jurisica, I., Glasgow, J.: Applications of Case-Based Reasoning in Molecular Biology. AI Magazine 25(1), 85–95 (2004)
11. Liu, H., Motoda, H. (eds.): Computational Methods of Feature Selection. Data Mining and Knowledge Discovery Series. Chapman & Hall/Crc, Boca Raton (2008)
12. Trunzter, C., Mercier, C., Esteve, J., Gautier, C., Roy, P.: Importance of data structure in comparing two dimension reduction methods for classification of microarray gene expression data. BMC Bioinformatics, 8–90 (March 13, 2007)
13. Wilkinson, L., Friendly, M.: The History of the Cluster Heat Map. The American Statistician 63(2), 179–184 (2009)
14. Witten, I., Frank, R.: Data mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufman Series in Data Management Systems. Elsevier, Inc., San Francisco (2005)
15. Wojnarski, M., Janusz, A., Nguyen, H.S., Bazan, J., Luo, C., Chen, Z., Hu, F., Wang, G., Guan, L., Luo, H., Gao, J., Shen, Y., Nikulin, V., Huang, T.-H., McLachlan, G.J., Bošnjak, M., Gamberger, D.: RSTC 2010 Discovery Challenge: Mining DNA Microarray Data for Medical Diagnosis and Treatment. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS, vol. 6086, pp. 4–19. Springer, Heidelberg (2010)
16. Yeung, K., Bumgarner, R., Raftery, A.: Bayesian Model Averaging: Development of an Improved Multi-Class, Gene Selection and Classification Tool for Microarray Data. Bioinformatics 21(10), 2394–2402 (2005)

Integration of Sequence Learning and CBR for Complex Equipment Failure Prediction

Marc Compta and Beatriz López

University of Girona,
Campus Montilivi, edifice P4, Girona, Spain
mcompta@eia.udg.edu, beatriz.lopez@udg.edu
http://exit.udg.edu

Abstract. In this paper we present a methodology based on combining sequence learning and case-based reasoning. This methodology has been applied in the analysis, mining and recognition of sequential data provided by complex systems with the aim of anticipating failures. Our objective is to extract valuable sequences from log data and integrate them on a case-based reasoning system in order to make predictions based on past experiences. We have used an *Apriori–style* algorithm (CloSpan) to extract patterns from original data. Afterwards, we have extended our own tool (eXiT*CBR) to deal with sequences in a case-based reasoning environment. The results have shown that our methodology anticipated correctly the failures in most of the cases.

Keywords: Sequence Learning, Data Mining, Sequence Pattern Matching, Complex Failure Prediction, Medical Application.

1 Introduction

The use of complex medical equipment has been crucial for patient diagnosis [1,2]. For instance, Magnetic Resonace Imaging (MRI) devices use large magnets and radiofrequency waves to produce high-quality images; Computer Tomography (CT) Scans provide different representations of the body structures from X Ray CT process; Positron Emission Tomography - CT (PET-CT) combines positron emission and CT into a single medical image device, allowing several sequential images from the same session of a patient to be superposed (co-registered).

Such equipment is known to be complex, composed of several pieces and subsystems, all of them subjected to high safety constraints, since they are using sensible technology regarding human health. Thus, if there is a minimal deviation of a given configuration parameter of the equipment component, the equipment sets up an alarm and stops. Nevertheless, when a machine like that fails, it causes a lot of problems in the clinical service. Thus a desired situation is to monitor the medical equipment so that failures can be predicted. Our research concerns the development of methods that help on the evaluation of the machine state, and then predicting failures before they occur.

Our starting point is the massive volumes of log data that such complex machines generate. They are unmanageable from an human point of view and poses us the challenge of looking for an automatic process with the aim of anticipating unexpected behaviors. The raw data from these systems needs to be efficiently managed and transformed to usable information in the same way that transcendental data (typically experience) is encapsulated into cases for future use. In this paper we present a methodology to achieve such goals that combines sequence learning and case-based reasoning (CBR) in order to catch meaningful information over system logs and reuse it in order to avoid future failures.

On the one hand, sequence learning algorithms allow us to relate events in the log file which happen in a time window and take place frequently before failures. On the other hand, case-based reasoning facilitates combination of sequences in a case base so that new partial sequences of events from equipment can be labeled as candidates to failure.

This paper is organized as follows. First we introduce in Section 2 a review of sequence learning techniques that are basic to understand what follows next. Afterwards, we describe in Section 3 the proposed methodology. We continue by providing the information about the application domain we are working on and the results obtained in Section 4. Finally, in Section 5 some related work is highlighted and some conclusions and discussion are provided in Section 6.

2 Background on Sequence Learning

The goal of sequence learning methods is to find frequent patterns of sequences. Among a large number of mining algorithms over sequential data, the Apriori algorithm is a classic [3].

Given a set of sequences, Apriori selects patterns over them with a minimum support. For that purpose, it builds a lattice of patterns. At the bottom of the lattice sequences of length one are considered; at the top, the longest sequence that can be generated is provided; each element of the lattice is annotated with a counter that represents the number of times the sequence is present in the input set. The ones that goes over a given threshold (support) are considered as frequent. For example, given the input sequences provided in Table 1, Apriori builds the lattice provided in Figure 1.

Apriori has three main difficulties: it generates huge sets of candidate sequences, scans the data multiple times, and has problems when mining long sequential patterns. PrefixSpan [4] improves Apriori but still generates a big

Table 1. A collection of sequences

Sequence id	**Sequence data**
S_1	<A,B,C,D>
S_2	<A,C,D>
S_3	<A,B,D>
S_4	<A,C>

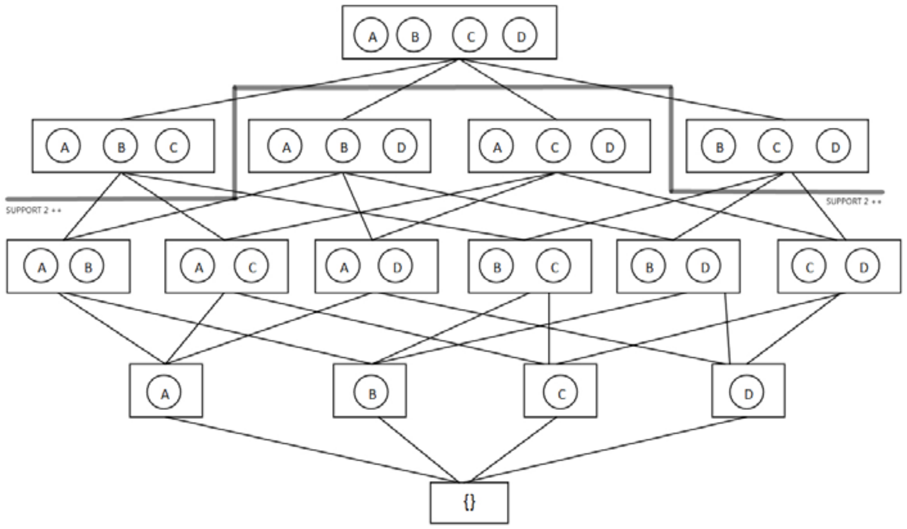


Fig. 1. Resulting lattice from sequences of Table 1. Grey line separates sets with a support higher than 2.

search space of sequence candidates. To reduce candidates, the concept of closed patterns that represents a reduced subset of interesting frequent patterns is introduced in the LCM (linear-time closed itemset miner) [5] and CloSpan (closed sequential pattern mining) algorithms [6].

Particularly, CloSpan assumes that mining the full set of frequent subsequences generates an explosive number of long patterns, which is prohibitively expensive in both time and space. As authors say, the purpose of this alternative approach is to mine frequent closed subsequences only, i.e., those containing no super-sequence with the same support. This method produces a significantly less number of discovered sequences while preserving the same expressive power since the whole set of frequent subsequences, together with their supports, can be derived easily from mining results. This allows mining of really long sequences that are un-minable by other algorithms.

Finally, the GSP (generalized sequential pattern) algorithm allows for time constraints such as maximum gap and minimum gap among the sequence elements [7]. A comparison of the different outputs of the algorithms described in this section is provided in Table 2.

Table 2. Algorithm classification according to their output models

Model	**Algorithm**
$event_j$ and $event_k$ and $event_l$ –> $event_m$ with supp y	Apriori PrefixSpan CloSpan
$event_j$ and $event_k$ within 2 seconds –> $event_m$	GSP

Other interesting algorithms to deal with sequences, but not based on lattices, are Hidden-Markov methods as [8], ELPH (Entropy Learning Pruned Hypothesis space) [9], an alternative proposal based on entropy measures, and series predictors, as ARIMA [10]. They are out of the scope of this work, but can be taken into account for further research.

In addition to learning algorithms, there are other interesting works related to primitives for handling sequences, as sequence features (like gaps between elements of a sequence, position of elements in a sequence), distances between sequences (Hamming, Levenshtein (edit), Kullback-Leibler-based), and generative models (as motifs, short distinctive sequence patterns shared by a number of related sequences).

3 Methodology

Given a set of log files of complex medical machines, our goal is to obtain prediction information. For that purpose, we have defined a methodology consisting on the following steps:

1. **Generation of sequences:** Transformation of log records into sequential data.
2. **Find sequential patterns:** Obtention of closed frequent patterns from sequential data.
3. **Build cases:** Association of sequential data to machine states in a case structure.
4. **Use case-based reasoning to predict failures.**

Each log file we are given corresponds to the information gathered from a given machine during a day (see Figure 2). The length of the file is not fixed, but depends on the machine behavior. There are several log files corresponding to fault machines and some log files that come from operational machines (that do not fail). For every machine, $M^1, \ldots, M^n$, there are several log files chronologically ordered, namely $LogFile(M^i, d^i_1), \ldots, LogFile(M^i, d^i_{k_i})$, one per day d^i_j. In case that the machine breaks, the last log file contains the last information gathered from the fault. Step 1 and 2 of the methodology is repeated for each log file (i.e. $k_1 + \ldots + k_n$ times); then in step 3, all the previous results are joined to configure out a case base that is then used in step 4 (see the flow diagram of Figure 3). All the steps are detailed in the remaining of the section.

3.1 Sequence Generation

Given a log file $LogFile(M^i, d^i_j)$, which contains log records, we generate sequence data into a new file $LogFile^{sq}(M^i, d^i_j)$.

The structure of the input log file is the one shown in Figure 2. There are four fields per record, which represents an event. They are the following:

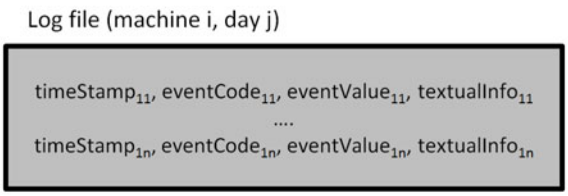

Log file (machine i, day j)
timeStamp$_{11}$, eventCode$_{11}$, eventValue$_{11}$, textualInfo$_{11}$
....
timeStamp$_{1n}$, eventCode$_{1n}$, eventValue$_{1n}$, textualInfo$_{1n}$

Fig. 2. Log file example

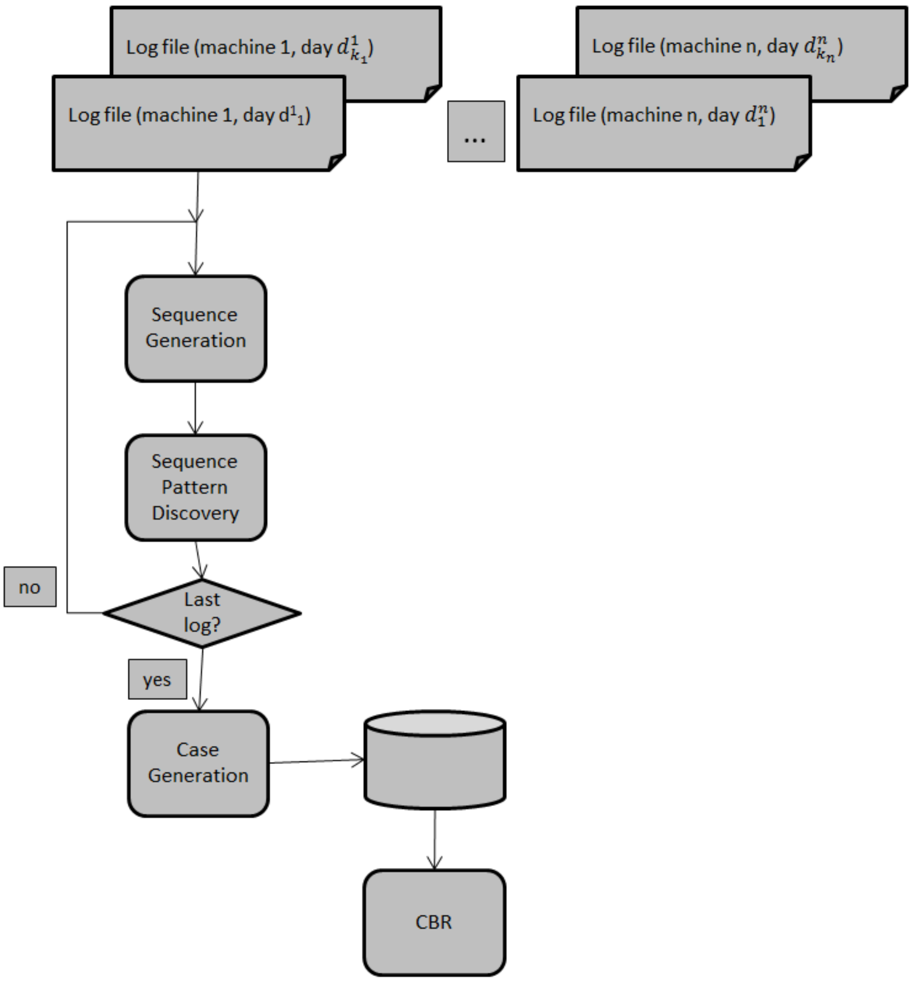
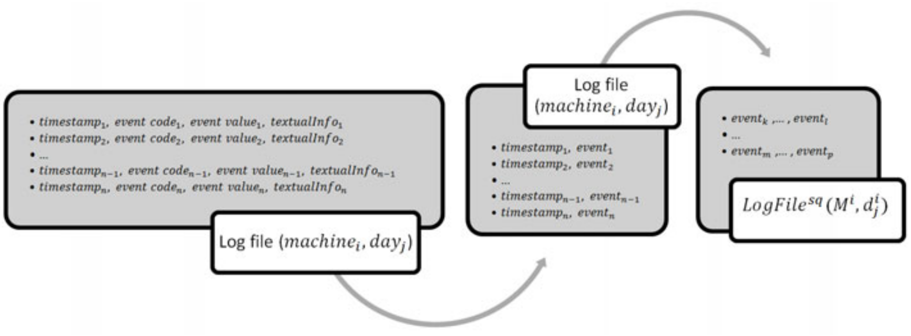

Log file (machine 1, day $d^1_{k_1}$)
Log file (machine 1, day d^1_1)
...
Log file (machine n, day $d^n_{k_n}$)
Log file (machine n, day d^n_1)
Sequence Generation
Sequence Pattern Discovery
no
Last log?
yes
Case Generation
CBR

Fig. 3. Flow diagram



• timestamp₁, event code₁, event value₁, textualInfo₁
• timestamp₂, event code₂, event value₂, textualInfo₂
• ...
• timestampₙ₋₁, event codeₙ₋₁, event valueₙ₋₁, textualInfoₙ₋₁
• timestampₙ, event codeₙ, event valueₙ, textualInfoₙ
Log file (machineᵢ, dayⱼ)
Log file (machineᵢ, dayⱼ)
• timestamp₁, event₁
• timestamp₂, event₂
• ...
• timestampₙ₋₁, eventₙ₋₁
• timestampₙ, eventₙ
• eventₖ ,..., eventₗ
• ...
• eventₘ ,..., eventₚ
LogFileˢᑫ (Mⁱ, dⱼⁱ)

Fig. 4. Sequence data generation

– **Timestamp**, which records the time when the event happens.
– **Event code**, which identifies the event.
– **Event value**.
– **Textual information**, i.e., annotations about the event.

As a first step, we reduce event codes and values to a single event by concatenating them (see Figure 4), and then, we generate sequences. In our context, we define a *sequence* as a list of events, in a specific order, that occurs close in time. We are aware that *close* is a subjective term, therefore we model it using a time window of length MaximumEventTime (MET). Thus,

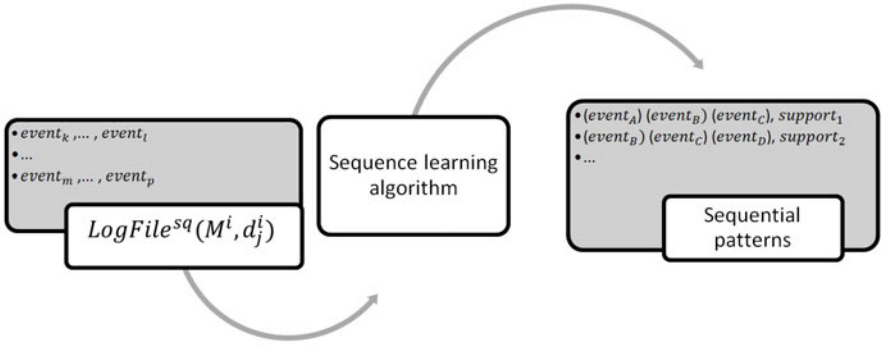
– If an event $event_i$ occurs in time t and an event $event_j$ occur in time $t' < t + MET$ we consider that they are belonging to the same sequence. Thus the sequence ($event_i$, $event_j$) is generated.
– If an event $event_i$ occurs in time t, and a second event $event_j$ in time $t+MET$ we consider that they are not related; they belong to different sequences. Then, there are two sequences, one finishing by $event_i$, and the next one starting by $event_j$: $(\ldots event_i)$ $(event_j \ldots)$.

Although other decisions could have been taken, we decided to fix this interval experimentally, in the domain we are involved, as 1 second. The sequences grow as does the time interval, becoming unmanageable in subsequent steps. As a future work, a scalability analysis based on this value should be performed. But it depends on the domain, and for our current problem the 1 second time interval seems to be adequate.

After this step we get a new file, $LogFile^{sq}(M^i, d^i_j)$, in which each row correspond to a different sequence $event_{i_1}, \ldots event_{i_j}$.

3.2 Sequential Patterns Mining

Given a file of sequences $LogFile^{sq}(M^i, d^i_j)$, we use a sequential learning algorithm to learn frequent patterns $LogFile^{fp}(M^i, d^i_j)$. As a result, we obtain



$\bullet\ event_k, \ldots, event_l$
$\bullet\ \ldots$
$\bullet\ event_m, \ldots, event_p$
$LogFile^{sq}(M^i, d^i_j)$
Sequence learning algorithm
$\bullet\ (event_A)\ (event_B)\ (event_C),\ support_1$
$\bullet\ (event_B)\ (event_C)\ (event_D),\ support_2$
$\bullet\ \ldots$
Sequential patterns

Fig. 5. Sequential patterns mining

several sequence patterns with its corresponding support (see Figure 5). A patterns has the following form:

$$< (event_1, \ldots, event_i)(event_{i+1}, \ldots, event_j) \ldots (\ldots event_n) >$$

There are events grouped between parentheses, and then groups ordered between angle brackets. Thus, a sequence represented by the pattern can be any of the one that selects one event from each group, while maintaining the order. For example, the sequence pattern $< (event_1)(event_2, event_3)(event_4) >$ represents the following sequences: $(event_1, event_2, event_4)$ and $(event_1, event_3, event_4)$.

Due to the fact that most of the sequence learning algorithms could return us a big amount of resulting sequences we only take into account patterns with a support higher than 8, meaning that we deal with a much smaller number of sequences.

3.3 Case Generation

This step comes after we have repeated the sequence generation and sequential pattern mining steps for every log file and for every machine; so the inputs of this step are $k_1 + \ldots + k_n$ files $(LogFile^{fp}(M^1, d^1_1), \ldots LogFile^{fp}(M^1, d^1_{k_1}),$ $\ldots, LogFile^{sq}(M^n, d^n_{k_n}))$. All such sequences are used in this step to compound cases. A case has three main elements:

– **Sequence pattern (SP)**.
– **Support**: the number of times that a pattern appears in the log file.
– **Class** (failure or operational).

Each row of an input file produces a case appropriately, but we only consider sequences that are exclusive of every class. That is, if a sequence belongs to a broken machine, and it does not appear into a file of a operational machine, then we generate a (positive, failure) case. Conversely, if a sequence belongs to an operational machine register and it does not appear into a file of a broken machine, then we generate a (negative, operational) case. Any other situation in the middle is ignored.[1]

As a result of this step we get a single case base.

3.4 Case Based Reasoning for Sequential Data

CBR is used here as a classification tool in order to distinguish between failures and normal cases. Regarding the particularities of our data, we have focused on the definition of a distance measure between sequences that allow us to compare the similarities between cases (retrieve phase).

Given a new case C in which a sequence of events S from a machine has to be analyzed, the retrieve phase recovers the set of most similar cases $C^1, \ldots, C^r$, where $C^i =< SP^i, support^i, class^i >$. For that purpose we need to define a distance between the sequence S and the sequence pattern SP, which takes into account that sequence patterns can be partially fulfilled. That is, a sequence $S = (event_a, event_c)$ partially fulfills the pattern $SP =< (event_a)(event_b)(event_c) >$.

Thus, the distance between a given sequence $S = event_{i_1}, \ldots event_{i_j}$ and a sequential pattern SP is defined as follows:

$$dist(S, SP) = 1 - \frac{maxss(S, SP)}{length(SP)} \ . \tag{1}$$

where $maxss(S, SP)$ is the length of the maximum subsequence of S following partially or completely the constraints of SP. That is, SP defines a collection of ordering constrains among its events. For example, a pattern defined by 3 events such as $event_1$, $event_2$ and $event_3$ defines the 3 following constraints: *$event_1$ goes before $event_2$*, *$event_1$ goes before $event_3$* and *$event_2$ goes before $event_3$*; and it does not allow, for example, $event_2$ goes before $event_1$.

To determine the maximum sequence to compute $maxss(S, SP)$, the following pattern matching algorithm is defined:

1. Select the events common to S and SP into I.
2. Select the events of I that satisfy the ordering constraints of SP.

The output of this algorithm could be more than one longest sequence. But due to the fact that the found sequences have the same length, it does not matter which of them is taken to compute the $maxss$ result.

Moreover, with this algorithm we are given the same result to sequences that exists in the pattern without gaps, as well as subsumption of sequences in the pattern. We can explain that with the following examples:

- Without gaps: $S = (A, A, B, D, A)$ and $SP =< (A)(A)(B) >$; the resulting longest sequence is (A, A, B).
- Subsumption without repetition: $S = (A, D, B, A, B)$ and $SP =< (A)(A)(B) >$; the resulting longest sequence is (A, A, B).
- Subsumption with repetition: $S = (A, D, B, A, B, A, A, B)$ and $SP =< (A)(A)(B) >$; the resulting longest sequence is (A, A, B).

[1] In fact, we are xor-ing the sequences.

According to Equation 1, k cases are selected (k-m method). Next, in the reuse phase, the final solution is obtained using the Pous method [11]. This method assigns the final class to a case depending on the ratio DV_{pous} between accumulated similarities of positive (failures) and recovered cases. If DV_{pous} goes over a threshold τ, cases are considered positive (failures).

Finally, revise and retain methods are left for future work.

4 Experimentation and Results

We have implemented the methodology by combining CloSpan [6] with eXiT*CBR [12]. On the one hand, CloSpan is a very well known sequence learning algorithm, GNU available, in which several researchers are continuously improving its efficiency. On the other hand, eXiT*CBR is a case-based reasoning tool developed with modularity on mind, so that every step of the basic four-r CBR cycle can be easily re-implemented in order to adapt it to a particular problem context. Particularly, we have added the sequence distance defined in Section 3.4 as a new method, so that the retrieve module is able to deal with sequences. Moreover, this tool provides valuable validation facilities as provides cross-validation techniques, automatized data set generation and an advanced experiment navigation framework based on plot images that assists the interpretation of the results.

4.1 Experimental Set Up

Our experimental data consists on log files from 14 machines, 10 of them that stops due to some deviation (failure), and 4 than works operationally. There are 21 log files for every fault machine, one per day before it stops working; the information collected from operational machines is not homogeneous: 24, 17, 20 and 18 log files[2].

Of course the experimental data, as it is, is biased towards failures. So we followed a stratified cross validation approach to obtain up to 10 folders for training and testing the methodology.

We have used ROC (*Receiver Operator Characteristics*) curves [13] to analyse the results. ROC curves depict the tradeoff between true positives and false positives regarding to a changing parameter. In our case, the parameter analyzed affects the reuse method, varying the τ threshold from 0 to 1 to generate the ROC curves.

4.2 Results on Sequence Learning

We started working with CloSpan [6]. This resulted in an efficient algorithm, based on mining frequent closed subsequences (i.e., those containing no

[2] Data is not public available because it is subject to a non-disclosure agreement with the provider.

super-sequence with the same support). We started testing with a concrete implementation [14] and then we developed a tool to parse and convert our original log data to the correct CloSpan input files.

From each log file, we run the algorithm and we obtain sequences as the following one (outcomes of step 3):

$$< (E37)(E37)(E42)(E37)(E43)(E37)(E46)(E320) >$$

$$Num = 1435976, Support = 85, ItemsetNumber = 8$$

where Num is the internal number of CloSpan assigned to the sequence, $Support$ the number of times this pattern occurs, and $ItemsetNumber$ the length (number of items) of the sequences.In Table 3 an output example of this step is provided, in which CloSpan has been found patterns up to 11 event length. Regarding the required parameters of CloSpan, we have used a minimum support of 1% due to the high volume of data available.

Average results differ significantly between normal and failure situations. About 1,940 sequences per log file have been obtained in the first case, while 122,626 in the second case. In this latter case, the deviation among the number of frequent sequences found in every machine is high (from 200 to 20,000). So there is a lot of events in normal functioning that could be related to information events (and maybe related to the patient conditioning), but not necessarily to malfunctioning. Such results, then, reinforces our strategy of xor-ing the sequences between normal and operational cases as suggested in the step 3 of the methodology.

4.3 Results on Case-Based Prediction

As explained before, there was a high bias towards abnormal cases: among 174,542 cases compiled finally in step 3 of the methodology, 16,155 correspond to normal situations, while the remainder (158,387) to failures. Thus, stratified 10-fold cross-validation has been used with a 3% of cases for testing purposes. That means that in a single experiment:

– There are 6,148 cases for training, half failures, half normal
– There are 190 cases for testing, half failures, half normal.

The results are very satisfactory. They are shown in the left ROC plot of Figure 6. The area under the curve is surprisingly high 0.97 so probably we are in an overfitting situation.

There are some reasons that can explain this behavior: the fact that test cases have been randomly taken from the big amount of sequences, some of them really similar and partially subsuming others. Taking a subset of them with the aim of generating the set of test cases may not be representative of a real situation.

Further experiments should be performed under the guidance of human experts who should interpret the obtained sequences, deal with the specificity of them, and provide additional knowledge that allow the definition of additional steps in the methodology that produce higher abstract patterns.

Table 3. Results from CloSpan

```
< (E37)(E37)(E42)(E37)(E43)(E37)(E46)(E320) > Support = 85
< (E37)(E37)(E42)(E320)(E37)(E37)(E46)(E320) > Support = 87
< (E37)(E37)(E42)(E320)(E37)(E43)(E37)(E46)(E320) > Support = 84
< (E37)(E37)(E42)(E320)(E37)(E43)(E37)(E46) > Support = 88
< (E37)(E37)(E42)(E320)(E37)(E43)(E37)(E320) > Support = 86
< (E37)(E37)(E42)(E45)(E320)(E37)(E43)(E37)(E46)(E320) > Support = 81
< (E37)(E37)(E42)(E45)(E320)(E37)(E43)(E37)(E46)(E405)(E320) > Support = 78
< (E37)(E37)(E42)(E45)(E320)(E37)(E43)(E37)(E46) > Support = 84
< (E37)(E37)(E42)(E45)(E320)(E37)(E43)(E37) > Support = 85
< (E37)(E42)(E45)(E320)(E37)(E37)(E46)(E320) > Support = 82
< (E37)(E45)(E320)(E37)(E43)(E37)(E46)(E320) > Support = 84
< (E320)(E37)(E37)(E42)(E37)(E43)(E37)(E46) > Support = 60
< (E320)(E37)(E37)(E42)(E320)(E37)(E43)(E37) > Support = 59
< (E320)(E37)(E37)(E42)(E320)(E37)(E46)(E320) > Support = 59
< (E320)(E37)(E37)(E42)(E45)(E320)(E37)(E320) > Support = 59
< (E320)(E37)(E42)(E37)(E43)(E37)(E46)(E320) > Support = 59
< (E320)(E320)(E37)(E43)(E37)(E46)(E405)(E320) > Support = 72
< (E493)(E493)(E493)(E493)(E493)(E493)(E493)(E493) > Support = 76
Total # of TreeNode: 1337
Closed/Max 1 : 53
Closed/Max 2 : 113
Closed/Max 3 : 170
Closed/Max 4 : 267
Closed/Max 5 : 327
Closed/Max 6 : 190
Closed/Max 7 : 49
Closed/Max 8 : 14
Closed/Max 9 : 2
Closed/Max 10 : 1
Closed/Max 11 : 1
```

5 Related Work

One of the first attempts to use sequences in a CBR system is [15] for solving a intrusion detection problem. There, the authors define a case structured in parts, in what they called sequence cases; moreover, cases follow an organizational structure that allows a dynamic selection of cases according to the information available. The authors approach is useful particularly in their problem domain, when sequences are unsegmented, that is, there is no clear separation from one sequence to another. This work differs from us since we are integrating sequence learning with case-based reasoning, in different steps of a methodology, and then, we feed CBR with segmented sequences.

Pioneer work on intrusion detection is [16] which also deals with sequences but following an instance based learning approach. In this work, the authors propose a method to learn sequences from a stream based on defining a similarity measure that requires equal fix length sequences. As a consequence, a complementary segmenting process is carried out to obtain sequences from event streams based on

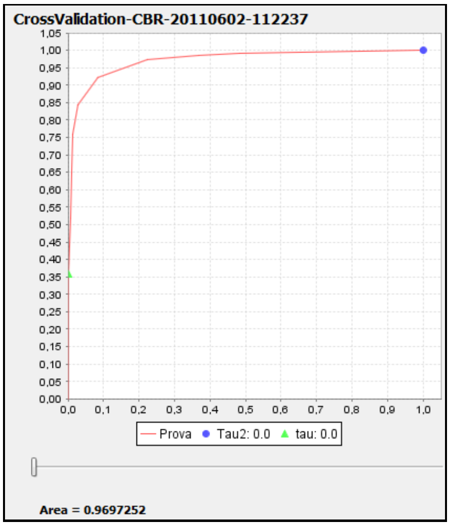


Fig. 6. ROC obtained with eXiT*CBR. Average over 10 sets

sliding windows of a given length. Each fix length generated sequence becomes an instance, and so the amount of instances becomes significantly increased. Therefore, reduction methods are further required to control the case growth. Conversely, we are taking advantage of existing sequence learning algorithms, so sequences are obtained with a given support and confidence, becoming generalized patterns of cases, and no further reduction of the case base is required. On the other hand, our similarity measure takes into account subsumption relations between the sequences to be compared independently of their length.

There have been other previous synergies between sequence learning and CBR, as [17], in which the authors propose learning methods to select sequences of repairs in a CBR system, when more than one repair exists.

Other works deal with sequences as data to be handled by case-based reasoning systems although not necessarily hybridizing leaning methods. For example, in [18] the authors propose to code RSA sequences into numbers (0 normal, 1-9 dysfunctional), and then use the obtained time series as cases in order to find risk of suffering illnesses to the patient the RSA belongs. In this case, sequences have been translated to time series. Conversely, the work of [19] starts from time series which are qualified so that they become sequences, that feed a CBR for voltage sags detection.

There are also domains in which the use of sequences, as data structures to be handled by CBR, are naturally used. This is the case of product design [20,21] and assembly sequence planning [22,23]. In most of these approaches, optimization techniques, as constraint satisfaction problem or genetic algorithms are integrated in the reuse phase of a CBR system. Thus, CBR shows its versatility as general problem solving methodology in which several techniques can be integrated in the different phases.

Finally, it is important to highlight other related works to sequence learning in complex equipment prediction but which not follow a CBR approach. For example, in [24] the authors propose to use the GSP algorithm (see Section 2) to learn frequent patterns, and then use statistic methods to correlate the learnt patterns. That is, frequent sequences could end in information, warning or serious errors. Next, statistical methods are used to establish relationships between different pattern sequences (warning and then error). If we can extend our case-base system to consider different classification situations, then we think that we can also improve our system with the insights of [24].

6 Conclusions

In this paper we have presented an original methodology that combines data mining over sequences and case-based reasoning to predict failures on complex medical equipment. From raw data coming from log files, frequent sequences are learnt, that thereafter are used to predict future equipment behaviours (failures). The methodology has been implemented and tested obtaining high successful results.

As a future work, we will continue experimenting with different methods of knowledge extraction in order to improve case definition and, consecutively, anticipate prediction. For example, we should be able to provide as a solution the interval time in which the failure is expected to occur. Other deeper research should include the use of the supporting values obtained in sequences in the distance measure defined to compare sequences. Finally, the combination of different errors should also be explored, as in [24].

Acknowledgments. This research project has been partially funded by the EU Eureka ITEA 2 project with funds of the Spanish Government (labels 09039 and TSI-020400-2010-84 correspondingly), and by the Spanish Government funds through the project labelled TIN2008-04547.

References

1. Smith-Bindman, R., Lipson, J., Marcus, R., et al.: Radiation dose associated with common computed tomography examinations and the associated lifetime attributable risk of cancer. Arch. Intern. Med. 169(22), 2078–2086 (2009), doi:10.1001/archinternmed.2009.427
2. Berrington de González, A., Mahesh, M., Kim, K.P., et al.: Projected cancer risks from computed tomographic scans performed in the United States in 2007. Arch. Intern. Med. 169(22), 2071–2077 (2009)
3. Agrawal, R., Srikant, R.: Fast algorithm for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data bases, VLDB, Santiago, Chile, pp. 487–499 (September 1994)

4. Pei, J., Han, J., et al.: PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th International Conference on Data Engineering, ICDE, Heidelberg, Germany, pp. 215–224 (2001)
5. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining Sequential Patterns by Pattern-growth: The PrefixSpan Approach. IEEE Transactions on Knowledge and Data Engineering 16(11), 1424–1440 (2004)
6. Yan, X., Han, J., Afshar, R.: CloSpan: Mining Closed Sequential Patterns in Large Datasets. In: Proceedings of 2003 SIAM International Conf. Data Mining, SDM 2003 (2003)
7. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
8. McCallum, A., Freitag, D., Pereira, F.: Maximum Entropy Markov Models for Information Extraction and Segmentation. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML), pp. 591–598 (2000)
9. Jensen, S., Boley, D., Gini, M., Shrater, P.: Rapid On-line Temporal Sequence Prediction by an Adaptive Agent. In: Proc. of the Fourth International Conference on Autonomous Agents and Multi-Agent Systems, pp. 67–73 (2005)
10. Box, G., Jenkins, G.:Time Series Analysis: forecasting and control. Holden Day (1976)
11. Pous, C., Gay, P., Pla, A., Brunet, J., Sanz, J., Ramon y Cajal, T., López, B.: Modeling Reuse on Case-Based Reasoning with Application to Breast Cancer Diagnosis. In: Dochev, D., Pistore, M., Traverso, P. (eds.) AIMSA 2008. LNCS (LNAI), vol. 5253, pp. 322–332. Springer, Heidelberg (2008)
12. López, B., Pous, C., Pla, A., Gay, P., Sanz, J., Brunet, J.: eXiT*CBR: A framework for case-based medical diagnosis development and experimentation. Artificial Intelligence in Medicine 51, 81–91 (2011)
13. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27, 861–874 (2006)
14. Yan, X.: Clospan: Closed Sequential Pattern Mining Package, `http://www.cs.ucsb.edu/~xyan/software/Clospan.htm` (accessed March 30, 2011)
15. Martin, F.J.: Case-Based Sequence Analysis in Dynamic, Imprecise, and Adversarial Domains. PhDThesis, Universitat Politènica de Catalunya (2004)
16. Lane, T., Brodley, C.E.: Temporal Sequence Learning and Data Reduction for Anomaly Detection. ACM Transactions on Information and System Security 2(3), 295–331 (1999)
17. Cox, M.T.: Loose Coupling of Failure Explanation and Repair: Using Learning Goals to Sequence Learning Methods. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 425–434. Springer, Heidelberg (1997)
18. Funk, P., Xiong, N.: Case-based reasoning and knowledge discovery in medical applications with time series. Computational Intelligence 22(3/4), 238–253 (2006)
19. Gamero, F.I., Meléndez, J., Colomer, J.: QSSI: A new similarity index for qualitative time series. Applciation to classify voltage sags. Applied Artificial Intelligence 25(2), 141–162 (2011)
20. Purvis, L., Pu, P.: COMPOSER: A case-based reasoning system for engineering design. Robotica 16(3), 285–295 (1998)

21. Pajula, E., Seuranen, T., Hurme, M.: Selection of separation sequences by case-based reasoning. Computer-Aided Chemical Engineering 9, 469–474 (2001)
22. Su, Q.: Applying case-based reasoning in assembly sequence planning. International Journal of Production Research 45(1), 29–47 (2007)
23. Pu, P., Reschberger, M.: Assembly sequence planning using case-based reasoning techniques. Knowledge-Based Systems 4(3), 1123–1130 (1991)
24. Zirkel, W., Wirtz, G.: A Process for Identifying Predictive Correlations Patterns. In: 7th International Conference on Service Management Systems Service Systems and Service Management (ICSSSM), pp. 1–6 (2010)

Time Series Case Based Reasoning for Image Categorisation

Ashraf Elsayed[1], Mohd Hanafi Ahmad Hijazi[1,5], Frans Coenen[1], Marta García-Fiñana[2], Vanessa Sluming[3], and Yalin Zheng[4]

[1] Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool L69 3BX, UK
[2] Centre for Medical Statistics and Health Evaluation, University of Liverpool, Shelley's Cottage, Brownlow Street, Liverpool L69 3GS, UK
[3] School of Health Sciences, University of Liverpool, Thompson Yates Building, The Quadrangle, Brownlow Hill, Liverpool L69 3GB, UK
[4] Department of Eye and Vision Science, Institute of Ageing and Chronic Disease, University of Liverpool, UCD Building, Liverpool L69 3GA, UK
[5] School of Engineering and Information Technology, Universiti Malaysia Sabah, Locked Bag 2073, 88999 Kota Kinabalu, Sabah, Malaysia
{a.el-sayed,m.ahmad-hijazi,coenen,martaf,slumingv, yalin.zheng}@liverpool.ac.uk

Abstract. This paper describes an approach to Case Based Reasoning (CBR) for image categorisation. The technique is founded on a time series analysis mechanism whereby images are represented as time series (curves) and compared using time series similarity techniques. There are a number of ways in which images can be represented as time series, this paper explores two. The first considers the entire image whereby the image is represented as a sequence of histograms. The second considers a particular feature (region of interest) contained across an image collection, which can then be represented as a time series. The proposed techniques then use dynamic time warping to compare image curves contained in a case base with that representing a new image example. The focus for the work described is two medical applications: (i) retinal image screening for Age-related Macular Degeneration (AMD) and (ii) the classification of Magnetic Resonance Imaging (MRI) brain scans according to the nature of the corpus callosum, a particular tissue feature that appears in such images. The proposed technique is described in detail together with a full evaluation in terms of the two applications.

Keywords: Case Based Reasoning, Image Analysis, Time Series Analysis, Dynamic Time warping.

1 Introduction

In its traditional form Case Based Reasoning (CBR) is typically directed at tabular data. Current research within the domain of CBR seeks to widen the scope of the technology by, amongst other initiatives, applying it to alternative forms of data such as images, sound, video, etc. There are two principal issues to be considered when applying CBR to non-standard data. The first is how to best represent the input so as to facilitate CBR. The second is the nature of the similarity checking mechanism to be applied. The two issues are closely related. One potential solution is to translate the input data format into an appropriate tabular format so that traditional approaches to CBR can be applied. In the case of image datasets this involves the application of segmentation and registration techniques. This paper proposes the adoption of an alternative approach to image representation whereby salient images features are encapsulated using time series.

Referring back to the two issues identified above the questions to be addressed are: (i) how can images best be translated into time series; and (ii) given an example time series represented image, how can we identify the most similar image within a Case Base (CB). With respect to the time series representation of images, we may consider images in their entirety or in terms of some sub region that features across the image set. The first takes into account the entire image while the second is directed at some specific feature within the image. Which is the most appropriate depends in part on the nature of the application. If the content of the entire image is important or if there is no single defining feature, then the first should be adopted. The second approach is only applicable if there is some feature that exists across the image set that is significant with respect to a particular application. Once a time series representation has been generated a similarity checking mechanism is required. Essentially this entails some form of *curve comparison.* The technique promoted in this paper is Dynamic Time Warping (DTW). This was selected because it is well understood and operates on curves that are not necessarily of the same unit length.

The intention of the paper is to provide an insight into the operation of time series analysis CBR with respect to the above. To act as a focus for the analysis, two specific applications are considered: (i) the screening of retinal images for Age-related Macular Degeneration (AMD), and (ii) the categorisation of Magnetic Resonance Imaging (MRI) brain scans. AMD is an eye condition that affects the macula, the central portion of the retina. It is the leading cause of irreversible blindness in the elderly and is a growing global healthcare challenge due to our ageing population; early detection may offer an opportunity for the application of timely treatment to inhibit the progress of the condition. A good way of identifying the early onset of AMD is through the identification of "fatty deposits" (called *drusen*) and pigment abnormality in the retina. Currently this is achieved by visual inspection, by clinicians, of the "fundus" photograph. This is a time consuming process and subject to human factors such as skills and tiredness; automated screening is therefore seen as beneficial even if only a coarse grading can be achieved.

The screening of retinal images for AMD requires the entire image to be taken into consideration. The second application, the categorisation of MRI brain scans is directed at a particular region within such scans, namely the *corpus callosum.* The corpus callosum connects the two hemispheres of the brain. It is conjectured that the size and shape of the corpus callosum dictates certain human abilities (such as mathematical or musical abilities), human characterises (such as "handedness") and certain medical conditions (such as epilepsy).

The rest of this paper is organised as follows. Section 2 provides some background with respect to time series analysis, DTW and CBR. The two applications, that form the focus of the work reported in this paper, are then described in Sections 3 and 4 respectively. The sections also include a full evaluation of the proposed approaches using real data. A summary and some conclusions are then presented in Section 5.

2 Background

Time Series Analysis (TSA) is concerned with the study of data that can be represented as one or more curves with a view to extracting knowledge using this representation. Note that the data dimensions do not necessarily need to include time, TSA may be applied to any form of data that can be represented as a sequence of one or more curves. The fundamental issues of TSA are: (i) how to measure similarity between time series and (ii) how to compress the time series while maintaining discriminatory power. Similarity can be measured in terms of ([2]): (i) similarity in time, (ii) similarity in shape and (iii) similarity in change. Related areas are Time Series Data Mining (TSDM), forecasting and the analysis of moving object data. An example of the last can be found in [24]. Keogh and Kasetty [18] give a survey of TSDM. For a discussion of time series forecasting see [1].

Dynamic Time Warping (DTW) is a technique whereby two time series can be compared. The technique does not require the two curves to be of the same length and takes into account a certain amount of "skew" to obtain a best fit. DTW was first proposed by Sakoe and Chiba [27] and was originally applied to speech recognition problems, but subsequently has been applied in much wider areas of application [19]. Given two time series: $Q = \{q_1, q_2, \ldots, q_i, \ldots, q_n\}$ and $C = \{c_1, c_2, \ldots, c_j, \ldots, c_m\}$, these can be aligned using DTW by constructing a n by m grid (matrix) such that the value for element (i, j) is the *squared Euclidean distance* from point c_j on curve C, a comparator sequence, to point q_i on curve Q, the query sequence, i.e. a sequence we wish to compare to C with the aim (say) of categorising Q. The best match between the two sequences Q and C is the *warping path* that minimises the total cumulative distance from grid element $(0, 0)$ to (n, m). A warping path is any contiguous set of matrix elements from $(0, 0)$ to (n, m). The warping cost associated with a particular path is its cumulative distance. DTW tends to produce better results than using a straight forward point-to-point comparison, however it tends to be computationally expensive. A number of tricks can be used to speed up the process. For example we can coarsen the data to produce an approximate path (i.e. do not use every sample point). Alternatively, from the observation that we can expect the best path to approximate to the line from $(0, 0)$ to (n, m), we can omit many calculations. Work has been done on the nature of the *warping window* (for example use of the "Sakoe-Chiba band" or the "Itakura parallelogram".)

Case Based Reasoning (CBR) has a well established body of literature associated with it. Recommended reference works include [21] and [20]. For a review of the application of CBR in medical domains see [16] or [3]. For a discussion on the crossover between data mining and CBR interested readers are referred to [25].

3 AMD Screening

The motivation for AMD screening was introduced in Section 1. The objective is to detect the presence of drussen (a primary indicator of AMD) in retina images collected as part of a screening programme; thus we wish to categorise/classify retina images as positive (evidence of AMD detected) or negative (normal). Three example images are presented in Fig. 1. The image on the right (Fig. 1[c]) is from a normal eye, while the other two (Figures 1[a] and [b]) are images of eyes that contain drusen (light coloured flecks scattered across the image) and other pathological features. Inspection of the images indicates the difficulty in detecting the drusen. This is thus one of our motivations for employing CBR techniques, rather than to attempt to detect these features using other techniques such as image segmentation. The optic disc (OD), a bright coloured disc featured in the retinal images in Fig. 1 from which all blood vessels emanate, connects the retina to the "optic nerve". The macula, clearly visible in the center of the image given in Fig. 1[c] as a dark coloured region, acts as a light detector and provides humans with the central vision essential for seeing fine details and with the colour vision (the macula is obscured by drusen in Figures 1[a] and [b]).

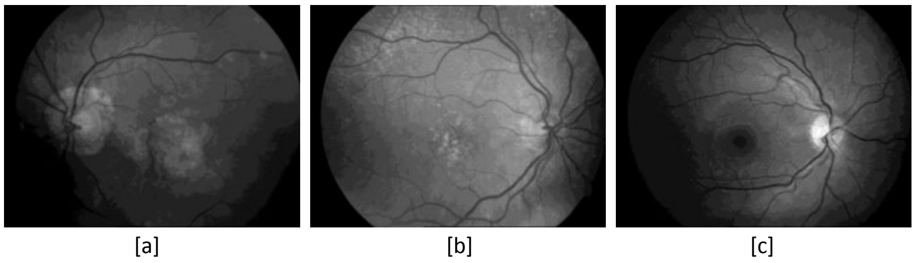


Fig. 1. Example of Retinal Images, [a] and [b] feature AMD, while [c] does not

3.1 Time Series CBR for AMD Screening

To represent images of the form shown in Fig. 1 the approach advocated in this paper is to consider the images in terms of pixel values using the Red-Green-Blue (RGB) colour model and the Hue-Saturation-Intensity (HSI) representation of the RGB model. As such each image can be represented as a sequence of histograms, with length M, to which a curve can easily be fitted. Each histogram is represented as a curve h_i, such that each point along the curve, $h_i(m)$ takes some value β (where $0 \leq m < M$, and β is the number of occurrences of intensity value m in image i).

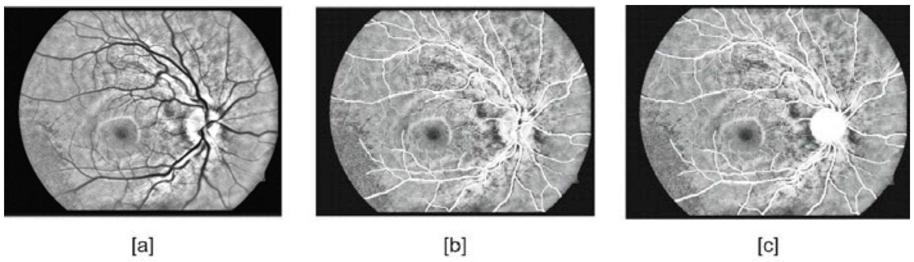


Fig. 2. Retinal images after [a] enhancement, [b] blood vessels removed and [c] optic disc removed

Prior to translating the images into time series some enhancement to the histograms was undertaken. The enhancement was done by applying a Contrast Limited Adaptive Histogram Equalisation (CLAHE) [33] technique. CLAHE computes histogram for different parts of an image and equalises each histogram separately. This image enhancement process increased the visibility of edges in the retinal images, as can be seen by comparing the enhanced image given in Fig. 2[a] with the image given in Fig. 1[c]. Initial experiments [13], indicated that the green and saturation channels produced the best results. The green channel was thus selected as the most appropriate for retina representation using the RGB model because of its ability to show the greatest contrast compared to other colour channels, seen as essential for retinal object identification [4,32]. The saturation component was selected as this has also been shown to produce good performance in identifying AMD featured retinal images [13]. The technique described here thus considers only the green and saturation channels. The length of the histograms, M, was set to 256 (number of RGB colour space cells) for the green channel histograms and 101 (with values ranging from 0 to 100) for the saturation histograms.

It was also found that the removal of pixels representing blood vessels enhanced the categorisation process. This was achieved by applying a retinal blood vessels segmentation algorithm [29] to segment the blood vessels. The identified blood vessels pixels were replaced by null values and consequently omitted from the histogram generation process. Fig. 2[b] gives an example of a retinal image with blood vessel pixels removed (indicated in white) by applying this process to the image given in Fig. 2[a].

Further experiments [14], indicated that the optic disc can obscure the presence of drusen. It is technically possible to remove the pixels representing the optic disc in the same way that blood vessel pixels were removed. To achieve this, a variation of optic disc detection using the horizontal and vertical axis of retinal images as proposed in [23] was applied. The retinal blood vessels binary image and the enhanced green channel image was utilised to generate both the horizontal and vertical signals, instead of using the original green channel image [23]. The optic disc pixel values were then replaced with null values.

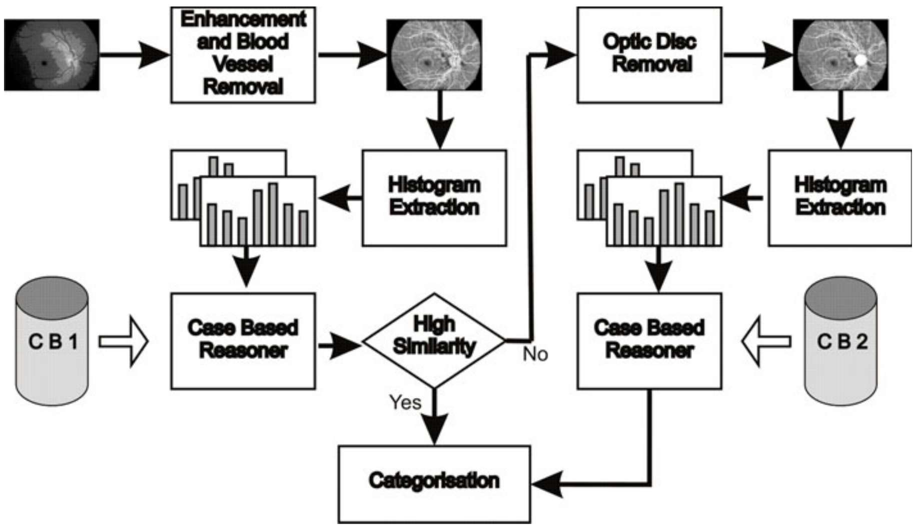


Fig. 3. Retina Image Categorisation Using Time Series CBR

Fig. 2[c] shows the retinal image given in Fig. 2[b] with the optic disc removed. It is also worth noting that the back coloured pixels (around the retina image) were excluded as well when the histograms were generated.

However, the routine removal of the optic disc can result in the removal of pixels representing drusen; especially where the drusen are close to, or superimposed over it. A two stage CBR approach is thus proposed consisting of two Case Bases (CBs), the primary CB and a secondary CB. The primary CB comprised the green and saturation histograms of labelled retina images (positive and negative) that included the optic disc but with blood vessels pixels removed, and the secondary CB comprised similar histograms but with the optic disc removed also.

A block diagram indicating the proposed CBR process is presented in Fig. 3. Given a new image we attempt to categorise this with reference to the primary CB first (CB1 in Fig. 3). The green channel histogram of the new image is compared to each of the green channel histograms in CB1 by means of computing the similarity measure between two histograms using DTW. A similar approach is also applied to the saturation histograms. These processes will generate the *preliminary results* comprising distance values between the green and saturation histograms of the new image, with the green and saturation histograms of each image in CB1. The *similarity* between the new image and each case in CB1 is then calculated by taking the average of each case's green and saturation histograms similarity values. If there exists only one "most similar" case, or there exist a number of most similar cases but all with the same label, the preliminary results will be taken as the final categorisation result and consequently the new image will be labelled as AMD or normal according to the label of the most similar image in CB1. If no clear result is obtained (i.e. there are two or more most similar cases with contradicting labels) the pixels representing the optic disc in the new image are removed and the CBR process is repeated but this time with the secondary CB (CB2 in Fig. 3). For a more complete description interested readers are referred to [14].

3.2 AMD Screening Evaluation

To evaluate the time series CBR approach as applied to retina image screening a data set comprising 161 images, of which 101 were AMD featured images, were utilised. All of the images were acquired as part of ARIA[1] project, which aims to provide a platform that is capable of predicting eye disease risk on individuals at the point of image acquisition process.

Table 1. Results from AMD Screening Experiments

TCV	Specificity (%)		Sensitivity (%)		Accuracy (%)	
run	*CBs*	*SH*	*CBs*	*SH*	*CBs*	*SH*
1	67	67	82	91	77	82
2	50	33	60	80	56	63
3	84	83	100	90	94	88
4	84	67	90	80	88	75
5	67	50	100	90	88	75
6	83	50	80	70	81	63
7	50	67	90	100	75	88
8	67	50	90	90	81	75
9	67	33	70	70	69	56
10	67	33	100	100	88	75
Average	*68*	56	86	86	*80*	74

The results of the evaluation, using Ten-fold Cross Validation (TCV) are given in Table 1. The table gives values for the *specificity*, *sensitivity*, and *accuracy* recorded for each TCV. Results obtain using the above approach (columns marked CBs) were compared with results obtained using a spatial-histogram approach [15]. From the table it can be seen that the proposed approach, that is advocated in this paper, provides the best results with 80% accuracy, and an average increase of 5% over all evaluation metrics.

4 MRI Scan Categorisation

The second application considered in this paper is the categorisation of MRI brain scans according to a single feature within those scans, namely the corpus callosum. The objective of the study was to investigate the application of time series CBR in the context of a Region of Interest (ROI) contextualisation. As noted in Section 1 the study of the nature (shape and size) of the corpus callosum in MRI brain scans is of interest to the medical community with respect to

[1] `http://www.eyecharity.com/aria$_$online/`

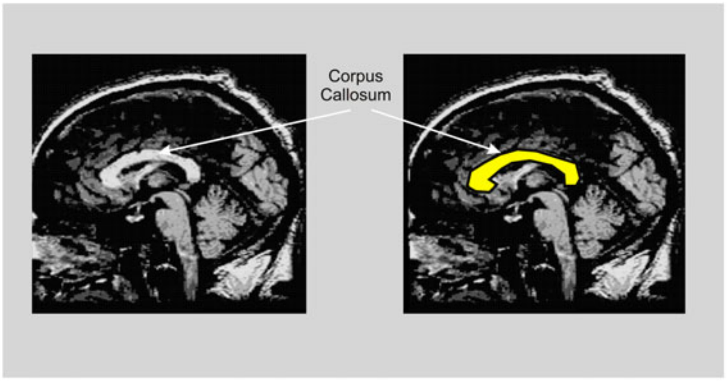


Fig. 4. Midsagital MRI brain scan slice showing the corpus callosum (highlighted in the right-hand image)

certain medical conditions that affect the function of the brain and particular skills. The size and shape of the corpus callosum has been shown to be correlated to gender, age, neurodegenerative diseases and various lateralised behaviour in people such as "handedness". It is also conjectured that the size and shape of the corpus callosum reflects certain human characteristics (such as a mathematical or musical ability). Several studies indicate that the size and shape of the corpus callosum, in humans, is correlated to gender [7,28], age [28,31], brain growth and degeneration [12,22], handedness [6], epilepsy [5,26,30] and brain dysfunction [8,17].

Fig. 4 gives an example (midsagital slice) of a MRI brain scan. The corpus callosum is located at the center of the image (highlighted in the image on the right). Fig. 4 also highlights a related structure called the fornix the significance of which is that image segmentation techniques frequently find it difficult to determine where the corpus callosum ends and the fornix starts. The focus of the study described here is directed at the categorising of MRI brain scan images according to corpus callosum, but could equally be applied to the characteristaion of other types of image that feature a given object.

4.1 Time Series CBR for MRI Scan Categorisation

When attempting to categorise images according the nature of a particular feature, regardless of whether a CBR technique or some other techniques is to be used, the first issue is to identify and isolate the feature of interest. In the case of the corpus callosum we know, approximately, where it is located with respect to the boundaries of an MRI brain scan. Thus we can apply a segmentation algorithm to identify the corpus callosum pixels. For the work described here the *efficient graph-based segmentation* algorithm [11] was used. This method is based on Minimum Spanning Trees (MST). All pixels of the original image are viewed

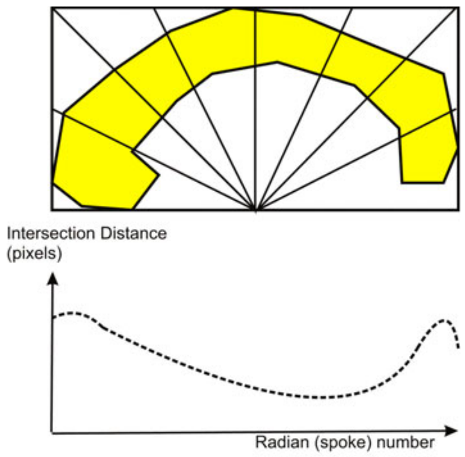


Fig. 5. Corpus callosum time series generation

as separate components. Two components are merged if the external variation between the components is small compared to the internal variation. Note that the segmentation can be problematic as a related tissue structure, the Fornix (also shown in the example given in Fig. 4) is often included together with some other spurious pixel clusters. Some data cleaning must therefore be undertaken. A smoothing technique was first applied to the MRI scans before the application of segmentation so as to preserves the boundaries between regions. This smoothing operation had the overall effect of bringing points in a cluster closer together.

Once the corpus callosum was identified we wish to represent it as a time series so that our proposed time series CBR technique could be applied. The adopted time series generation approach is illustrated in Fig. 5. A series of “spokes” were radiated out from the mid-point of the base of the Minimum Bounding Rectangle (MBR) surrounding a detected corpus callosum. The interval between spokes was one pixel measured along the edge of the MBR. Consequently, the number of spokes used to encode a corpus callosum varied from image to image. For each spoke the distance D_i (where i is the spoke identification number) over which the spoke intersects with a sequence of corpus callosum pixels was recorded. The mid point along the base of the MBR was chosen as this would ensure that there was only one intersection per spoke. The result is a time series with the spoke number i representing time and the value D_i, for each spoke, the magnitude. By plotting the D_i against i a time series may be derived (as shown in Fig. 5).

To categorise “unseen” MRI brain scans, according to the nature of the corpus callosum, an appropriate Case Base (CB) was constructed comprising labelled curves generated in the manner described above. A new case could then be compared, using DTW, to identify the most similar curve(s) in the CB. Further information on the categorisation of MRI brain scans according to the nature of the corpus callosum can be found in [9] and [10].

4.2 MRI Categorisation Evaluation

This section describes the evaluation of the proposed technique using "real life" MRI image sets. The evaluation was undertaken in terms of classification accuracy, sensitivity and specificity. Three studies are reported here: (i) a comparison between musician and non-musician MRI scans, (ii) a comparison between MRI scans belonging to right handed and left handed people, and (iii) an epilepsy screening process. The studies are discussed in more detail below.

Musicians v. Non-Musicians. For the musicians study a data set comprising 106 MRI scans was used, 53 representing musicians and 53 non-musicians, the data set was thus divided into two equal classes. The study was of interest because of the conjecture that the size and shape of the corpus callosum reflects certain human abilities (such as a mathematical or musical ability). Table 2 shows the TCV results obtained using the proposed technique. Inspection of Table 2 demonstrates that the overall classification accuracy of the time series CBR approach is significantly high. In many TCV cases the time series based approach obtained 100% accuracy, even though visual inspection of the corpus callosums in the image set does not lead to any clear identification of any defining feature.

Table 2. TCV Classification Results from Musicians Study

Test set ID	Accuracy(%)	Sensitivity(%)	Specificity (%)
1	91	100	85.71
2	100	100	100
3	91	100	85.71
4	100	100	100
5	100	100	100
6	100	100	100
7	100	100	100
8	100	100	100
9	100	100	100
10	100	100	100
Average	98.2	100	97.14
SD	3.8	0.0	6.03

Right handed v. Left handed. For the handedness study a data set comprising 82 MRI scans was used, 42 representing right handed people and 40 left handed people. The study was of interest because of the conjecture that the size and shape of the corpus callosum reflects certain human characteristics (such as handedness). Table 3 shows the TCV results obtained using the proposed technique. Inspection of Table 2 demonstrates that the overall classification accuracy of the time series CBR approach was again significantly high.

Epilepsy Screening. For the epilepsy study three datasets were used:

1. The first comprised the control group from the above musicians study together with 53 MRI scans from epilepsy patients.
2. The second data set used all 106 MRI scans from the musicians study as the control group, and the 53 epilepsy scans.
3. The third comprised all 106 MRI scans from the musicians study and a further 106 epilepsy cases so that the control and epilepsy groups were of equal size.

The aim of the study was to seek support for the conjecture that the shape and size of the corpus callosum is influenced by conditions such as epilepsy ([26,30]). Table 4 shows the TCV classification results for the three epilepsy data sets. Inspection of Table 4 indicates that the time series CBR approach performed significantly well in the context of distinguishing epilepsy MRI scans from the control group.

Table 3. TCV Classification Results from Handedness Study

Test set ID	Accuracy(%)	Sensitivity(%)	Specificity (%)
1	88.89	80	100
2	100	100	100
3	100	100	100
4	88.89	100	80
5	88.89	75.0	100
6	100	100	100
7	100	100	100
8	100	100	100
9	100	100	100
10	100	100	100
Average	96.67	95.5	98
SD	5.37	9.56	6.32

Table 4. TCV Classification Results for Epilepsy Study

Test	106 MR scans			159 MR scans			212 MR scans		
set ID	Acc.	Sens.	Spec.	Acc.	Sens.	Spec.	Acc.	Sens.	Spec.
1	72.73	80.00	66.67	75.00	70.00	83.33	81.82	88.89	76.92
2	81.82	83.33	80.00	81.25	85.71	77.78	77.27	80.00	75.00
3	72.73	80.00	66.67	75.00	70.00	83.33	81.82	88.89	76.92
4	81.82	83.33	80.00	81.25	85.71	77.78	77.27	80.00	75.00
5	81,82	83.33	80.00	81.25	85.71	77.78	68.18	70.00	66.67
6	81.82	83.33	80.00	75.00	70.00	83.33	72.73	77.78	69.23
7	63.64	66.67	60.00	81.25	85.71	77.78	77.27	80.00	75.00
8	81.82	83.33	80.00	68.75	66.67	71.43	81.82	88.89	76.92
9	72.73	80.00	66.67	68.75	66.67	71.43	72.73	77.78	69.23
10	63.64	66.67	60.00	81.25	85.71	77.78	81.82	88.89	76.92
Average	75.46	79.0	72.0	76.88	77.19	78.18	77.27	82.11	73.78
SD	7.48	6.67	8.78	5.15	9.06	4.37	4.79	6.51	3.89

4.3 Discussion of Results

With respect to classification accuracy the time series CBR approach performed well. Although the time series approach produced good results there was no obvious reason why this might be the case; visual inspection of the MRI scans did not indicate any obvious distinguishing attributes with respect to the size and shape of the corpus callosum. Further investigation is therefore deemed to be appropriate. With respect to computational complexity, image segmentation and the application of DTW for categorisation of images are both computationally expensive processes. The time complexity for the image segmentation was about 30 seconds per image. For the given data sets the application of DTW required, on average, 90 seconds to complete the classification of the entire test set.

5 Summary and Conclusion

In this paper an approach to the categorisation of images using a time series based CBR approach has been described. Two variations of the approach were considered. An investigation of its application using entire images, and an investigation of its application with respect to a specific region of interest in common across a set of images. The first was illustrated using an AMD screening programme applied to retina images. The second was demonstrated by considering the categorisation of MRI brain scans according to a particular feature common across such images, namely the corpus callosum. Both techniques used the "tried and tested" technique of Dynamic Time Warping to comparing images contained in a case base with a new image to be categorised. Different time series generation processes were demonstrated, and although these were specific to the applications under consideration it is argued that they have general utility. Evaluation of the approach, using "real life" data produced excellent results. Best results were produced in the context of the MRI brain scan data, although the reason for these excellent results requires further investigation.

References

1. Aburto, L., Weber, R.: A Sequential Hybrid Forecasting System for Demand Prediction. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 518–532. Springer, Heidelberg (2007)
2. Bagnall, A., Janacek, G.: Clustering Time Series with Clipped Data. Machine Learning 58, 151–178 (2005)
3. Bichindaritz, I., Marling, C.: Case-based reasoning in the health sciences: What's next? Artificial Intelligence in Medicine 36(2), 127–135 (2006)
4. Chaudhuri, S., Chatterjee, S., Katz, N., Nelson, M., Goldbaum, M.: Detection of Blood Vessels in Retinal Images using Two-Dimensional Matched Filters. IEEE Transactions on Medical Imaging 8(3), 263–269 (1989)
5. Conlon, P., Trimble, M.: A Study of the Corpus Callosum in Epilepsy using Magnetic Resonance Imaging. Epilepsy Res. 2, 122–126 (1988)
6. Cowell, P., Kertesz, A., Denenberg, V.: Multiple Dimensions of Handedness and the Human Corpus Callosum. Neurology 43, 2353–2357 (1993)

7. Davatzikos, C., Vaillant, M., Resnick, S., Prince, J., Letovsky, S., Bryan, R.: A Computerized Approach for Morphological Analysis of the Corpus Callosum. Journal of Computer Assisted Tomography 20, 88–97 (1996)
8. Duara, R., Kushch, A., Gross-Glenn, K., Barker, W., Jallad, B., Pascal, S., Loewenstein, D., Sheldon, J., Rabin, M., Levin, B., Lubs, H.: Neuroanatomic Differences Between Dyslexic and Normal Readers on Magnetic resonance Imaging Scans. Archives of Neurology 48, 410–416 (1991)
9. Elsayed, A., Coenen, F., Jiang, C., García-Fiñana, M., Sluming, V.: Region Of Interest Based Image Classification Using Time Series Analysis. In: IEEE International Joint Conference on Neural Networks, pp. 3465–3470 (2010)
10. Elsayed, A., Coenen, F., Jiang, C., García-Fiñana, M., Sluming, V.: Corpus Callosum MR Image Classification. Knowledge Based Systems 23(4), 330–336 (2010)
11. Felzenszwalb, P., Huttenlocher, D.: Efficient Graph-based Image Segmentation. Int. Journal of Computer Vision 59(2), 167–181 (2004)
12. Hampel, H., Teipel, S., Alexander, G., Horwitz, B., Teichberg, D., Schapiro, M., Rapoport, S.: Corpus Callosum Atrophy is a Possible Indicator of Region and Cell Type-Specific Neuronal Degeneration in Alzheimer Disease. Archives of Neurology 55, 193–198 (1998)
13. Hijazi, M.H.A., Coenen, F., Zheng, Y.: A Histogram Based Approach to Screening of Age-related Macular Degeneration. In: Proc. of Medical Image Understanding and Analysis (MIUA 2009), pp. 154–158 (2009)
14. Hijazi, M.H.A., Coenen, F., Zheng, Y.: Retinal Image Classification using a Histogram Based Approach. In: IEEE International Joint Conference on Neural Networks, pp. 3501–3507 (2010)
15. Hijazi, M.H.A., Coenen, F., Zheng, Y.: Retinal Image Classification for the Screening of Age-related Macular Degeneration. In: Proceedings of SGAI Conference, pp. 325–338 (2010)
16. Holt, A., Bichindaritz, I., Schmidt, R., Perner, P.: Medical Applications in Case-Based Reasoning. The Knowledge Engineering Review 20, 289–292 (2005)
17. Hynd, G., Hall, J., Novey, E., Eliopulos, D., Black, K., Gonzalez, J., Edmonds, J., Riccio, C., Cohen, M.: Dyslexia and Corpus Callosum Morphology. Archives of Neurology 52, 32–38 (1995)
18. Keogh, E., Kasetty, S.: On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. Data Mining and Knowledge Discovery 7(4), 349–371 (2003)
19. Keogh, E., Pazzani, M.: Scaling up dynamic time warping to massive datasets. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 1–11. Springer, Heidelberg (1999)
20. Kolodner, J.L.: Case-based Reasoning. Morgan Kaufmann Series in Representation and Reasoning (1993)
21. Leake, D.B.: Case-based Reasoning: Experiences, Lessons and Future Directions. AAAI Press Co-Publications (1996)
22. Lyoo, I., Satlin, A., Lee, C.K., Renshaw, P.: Regional Atrophy of the Corpus Callosum in Subjects with Alzheimer's Disease and Multi-infarct Dementia. Psychiatry Research 74, 63–72 (1997)
23. Mahfouz, A.E., Fahmy, A.S.: Ultrafast Localization of the Optic Disc using Dimensionality Reduction of the Search Space. In: Medical Image Computing and Computer Assisted Intervention, pp. 985–992 (2009)
24. Morzy, M.: Mining Frequent Trajectories of Moving Objects for Location Prediction. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 667–680. Springer, Heidelberg (2007)

25. Pal, S., Aha, D., Gupta, K.: Case-Based Reasoning in Knowledge Discovery and Data Mining. Wiley-Blackwell (in Press, 2011)
26. Riley, J.D., Franklin, D.L., Choi, V., Kim, R.C., Binder, D.K., Cramer, S.C., Lin, J.J.: Altered White Matter Integrity in Temporal Lobe Epilepsy: Association with Cognitive and Clinical Profiles. Epilepsia 42(4), 536–545 (2010)
27. Sakoe, H., Chiba, S.: Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Transactions on Acoustics, Speech and Signal Processing 26(1), 43–49 (1978)
28. Salat, D., Ward, A., Kaye, J., Janowsky, J.: Sex Differences in the Corpus Callosum with Aging. Journal of Neurobiology of Aging 18, 191–197 (1997)
29. Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R.M., Jelinek, H.F., Cree, M.J.: Retinal Vessel Segmentation using the 2-D Gabor Wavelet and Supervised Classification. IEEE Transactions on Medical Imaging 25, 1214–1222 (2006)
30. Weber, B., Luders, E., Faber, J., Richter, S., Quesada, C.M., Urbach, H., Thompson, P.M., Toga, A.W., Elger, C.E., Helmstaedter, C.: Distinct Regional Atrophy in the Corpus Callosum of Patients with temporal Lobe Epilepsy. Brain 130, 3149–3154 (2007)
31. Weis, S., Kimbacher, M., Wenger, E., Neuhold, A.: Morphometric Analysis of the Corpus Callosum using MRI: Correlation of Measurements with Aging in Healthy Individuals. American Journal of Neuroradiology 14, 637–645 (1993)
32. Youssif, A.A.-H., Ghalwash, A.Z., Ghoneim, A.A.A.A.-R.: Optic Disc Detection from Normalized Digital Fundus Images by Means of A Vessel's Direction matched Filter. IEEE Transactions on Medical Imaging 27, 11–18 (2008)
33. Zuiderveld, K.: Contrast Limited Adaptive Histogram Equalization. Academic Press Graphics Gems Series, pp. 474–485 (2001)

CBRSHM – A Case-Based Decision Support System for Semi-Automated Assessment of Structures in Terms of Structural Health Monitoring

Bernhard Freudenthaler

Software Competence Center Hagenberg GmbH, Softwarepark 21,
4232 Hagenberg, Austria
bernhard.freudenthaler@scch.at

Abstract. This contribution describes a case-based decision support system which is intended for being used in the field of Structural Health Monitoring to support the assessment of structures (by using the example of lamp posts). Interpreting measuring data manually is a very complex task, time-consuming and influenced by the subjectivity of civil engineers. Therefore, the engineers shall be supported in assessing measuring data by using a case-based decision support system. A measurement of a structure and a manual assessment by an engineer represent a case in a case base. Similar cases/structures shall be retrieved and made available for assessing new measurements. For supporting the assessment of simple structures (lamp posts), a case-based system shall be provided for interpreting measuring data semi-automatically to make suggestions about lamp posts' condition. Thereby, time and costs can be reduced more than 90% by the use of computer-aided assessment in comparison with the manual interpretation.

Keywords: Case-Based Reasoning, Decision Support System, Structural Health Monitoring, Semi-Automated Assessment.

1 Introduction

The overall objective of this contribution is to check the possibility whether the method Case-Based Reasoning (CBR) (see more at [1], [3], [9], [13]) can contribute successfully to support Structural Health Monitoring (SHM) to monitor, analyze and assess structures like lamp posts. Thereby, important information of structures shall be provided but also suggestions about the condition assessment of structures shall be made [4].

SHM defines the implementation of a damage identification strategy to identify, localize and, in best case, even to prevent possible damages of structures like buildings, bridges, pipes, lamp posts or wind engines. A pending damage should be detected preferably before damage occurs to initiate countermeasures.

The traditional condition assessment is done manually by civil engineers. Thereby, measured data (from different sensors) and optical parameters of the structure have to be interpreted. The analysis and interpretation of measuring data done by engineers represent a very complex and time-consuming issue whereby this way of condition assessment is very cost-intensive. In addition, the results are affected by the subjectivity and experience of the civil engineers.

For assessing complex measuring data numerous parameters have to be considered to detect the dynamic behavior of structures. These parameters are e.g., eigenfrequencies, mode shapes, damping or vibration intensities and they must be interpreted in conjunction. To analyze and assess these complex parameters correctly the engineer needs a wide experience in SHM and condition assessment. But the problem of the manual assessment is the subjectivity of engineers in interpreting measuring data. Each engineer assesses measuring data differently than a colleague and thereby can cause misinterpretations. Thus, possible damages of structures can be overlooked or not localized in time. This problem will be pointed up with the following statement:

„Each administration uses different condition rating techniques. Such a situation may derive from the fact that the same bridge, assessed by two engineers from different countries, can be rated with different grades." [17]

For these reasons a computer-aided, prototypical system was developed to support the engineers in assessing structures. A short introduction of the case-based system shall be presented to understand the idea of using CBR for SHM. The idea was to use CBR to compare structures and its assessments. The cases stored in a case base consist of parameters (e.g., eigenfrequencies, optical condition) and a manual assessment of the structure. If a new structure shall be assessed the parameters of this new structure have to be imported into the case-based system. The system searches for the most similar cases in the case base and presents the results to the user. Due to the fact that the cases stored in the case base are already assessed from an engineer, the system can reuse this assessment for the new structure. The engineer now has a suggestion of similar structures together with assessments. This supports the engineer in his/her decision making process. After adapting the suggested solution by the engineer s/he can store the new case (parameters of the new structure and reused/adapted solution) in the case base and finally, the case is available for future reuse.

In conclusion, the engineers shall be supported in their decision making process to get cost-saving, rapid and mainly objective assessments of structures. Tests and experiments with real measuring data shall show empirically the possibility of using CBR for SHM. Therefore, a prototypical case-based decision support system for the semi-automated assessment of structures in terms of SHM was developed [4].

Finally, the contribution is structured as follows:

Section 2 gives a motivation for the case-based decision support system for SHM and points out the advantages of semi-automated assessment of structures. Section 3 describes the most important fundamentals of SHM to understand the domain even better. Section 4 gives an overview about related research of CBR for SHM. Section 5 describes the case-based decision support system for SHM and in section 6 an example about the assessment of lamp posts is presented. Thereby, tests and experiments with real measuring data as well as results are shown. Finally, the conclusion and intended future work is discussed in section 7.

2 Motivation

SHM is gaining more and more in importance whereupon the assessment and life-time prediction of structures shall be supported with computer-aided systems. The numerous structures produce massive amount of measuring data. For this reason the manual assessment is very time-consuming, cost-intensive and moreover, subjective and it is impossible to assess all measuring data manually [6]. Therefore, SHM engineers have a strong need to semi-automate the assessing process to increase the quality of interpreting measuring data and to decrease costs and subjectivity.

Life-time prediction is another interesting and important task in SHM. Each structure has a theoretical life cycle and with increasing age the structural resistance decreases. Figure 1 shows the planned and the real decrease of structural resistance over the years. As one can see a warning level can occur earlier than the planned point. With measuring and assessing the structure repair actions can be organized at the right time and thereby (theoretically), damages can be avoided and the structural resistance can be increased again which can extend the life-time of the structure.

If possible damages can be detected before damage occurs, a well-timed SHM campaign can optimize service intervals and therefore reduce costs immensely. For example, a monitoring campaign of a typical 3-span bridge with 150 meters length in good condition produces average costs about 10.000 EUR. If the bridge is in bad condition the costs can increase over 100.000 EUR [17]. With a case-based decision support system the engineers have the opportunity to monitor and assess more structures in shorter time and thereby, in best case, the condition of the structures can be hold in stable condition which means massive cost reduction.

The current monitoring situation of bridges in the US shall be considered to underline the importance of SHM even more: 10.000 of 595.000 bridges in the US have to be replaced every year [17], [19], [2]. This causes yearly costs of 7 billion US$. With permanent or even periodic SHM these costs could be reduced tremendously. For this reason, a NSF-FHWA Joint Research Initiative started a 20-year project with a budget of 100 million US$ to capture measuring data of hundreds of bridges [17]. The collected data shall be used for researching and developing intelligent decision support systems to support SHM campaigns. Finally, the case-based decision support system described in this contribution focuses on similar issues: supporting engineers by semi-automated assessment of structures.

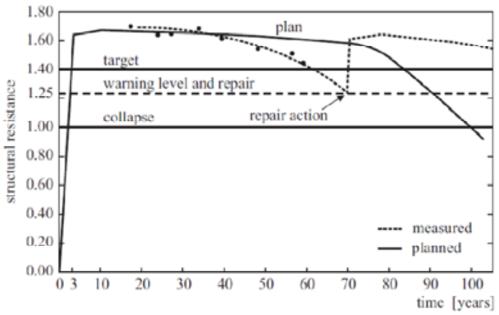


Fig. 1. Diagram of the theoretical life-time of a structure [17]

3 Structural Health Monitoring

The damage identification process in terms of SHM consists of four steps: detection, location, type and dimension of damage. Each of these steps has to be treated to guarantee a successful monitoring campaign. To achieve this one possible monitoring method shall be introduced; the so-called "Ambient Vibration Monitoring" method (AVM). AVM deals with the assessment and analysis of measured signals which represent the dynamic behavior of structures. For this purpose, the most important parameters in this context shall be described in the following section.

3.1 Eigenfrequency

"The frequency at which a structure is most easily excited." [17]

According to [19], [17], [18] eigenfrequencies are essential parameters to describe the vibration behavior in the linear elastic field. They are influenced by structural parameters like geometry (dimension, shape...), material properties (weight, damping factor...) and boundary conditions (loading...). To assess a structure successfully not only the 1st eigenfrequency shall be considered but also higher eigenfrequencies.

Figure 2 shows two frequency spectra of bridges. The left one is in good condition, in contrast the right one is a damaged bridge.

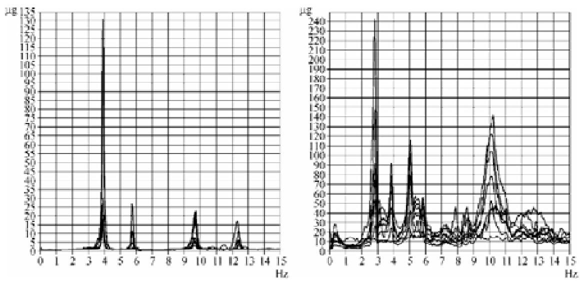


Fig. 2. Frequency spectra of a bridge in good (left) and bad (right) condition [19]

During the identification process of eigenfrequencies external influences shall be avoided (e.g. temperature fluctuation). The transformation of acceleration signals from the time domain into the frequency range can be done for example with a Fast-Fourier-Transformation (FFT).

3.2 Mode Shape

"*Characteristic shape of amplitude distribution when a structure is freely vibrating at one of its natural frequencies."* [17]

According to [17], [19], a mode shape represents a vibration mode in which a structure is vibrating in the particular eigenfrequency. For each eigenfrequency a mode shape can be provided. The combination of all mode shapes defines the vibration of a structure. In addition to eigenfrequencies, mode shapes are the second important parameter to describe the dynamic behavior of structures.

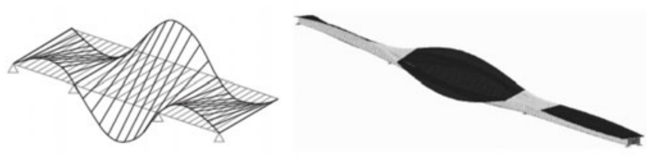


Fig. 3. 4th mode shape 2.09 Hz (measurement) and 2.29 Hz (calculation) – 1TL main span [17]

Figure 3 shows an example of a mode shape which could be derived from the frequency spectrum. Thereby, it is described which eigenfrequency was measured and which value was calculated. 1TL main span means that one torsion exists in longitudinal direction of the main span.

3.3 Damping

Each structure has a damping. With unique vibration stimulation the damping causes a continuous reduction of vibration until a static equilibrium is reached [17], [19], [18]. The damping properties are frequency dependent and are the third important parameter for system identification. Figure 4 shows an example of a damping pattern.

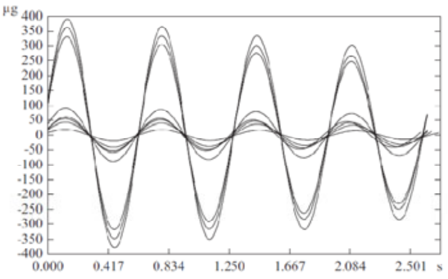


Fig. 4. Example of a damping pattern [17]

Furthermore, damping is an indicator for the current load factor of the load-bearing capacity of structures because with increasing utilization of the limit load-bearing capacity the damping coefficients increase seriously.

Beyond that, other parameters like vibration intensity can also be mentioned as important factors for assessing structures successfully.

4 Related Research

Due to the fact that SHM was introduced in the previous section the related research of CBR for SHM can be shown.

CBR is used in [8] to store information of damaged structures as cases in different case bases. Wavelet transformation for signal processing and neural networks for maintaining the case bases are utilized. The prototype was tested by using an aluminum bar with 98 cm length, 2 cm width and 0.5 cm height. The tests have shown that the case bases have to contain numerous critical cases to have the possibility of retrieving similar cases. Due to the fact that this approach uses more than one case base the quality of solutions depends strongly on the selection of the most suitable case base. The prototype only achieves a vague result. But for very simple materials like an aluminum bar this approach can start with damage detection. Experiments for not as simple materials/structures could not be shown.

An approach to monitor satellites is described in [12]. Data of different parameters are sent periodically to a ground monitoring station and are compared with cases in a case base. If anomalies occur an alarm is given. But solutions for problem solving are not supported by this CBR system. The approach can be used to find similarities between actual satellite data and data stored in form of cases. But the knowledge about monitoring satellites cannot be reused to assess other satellites.

ICARUS is a case-based reasoning system for locomotive diagnostics [15]. Thereby, on-board fault messages indicating the presence of anomalous conditions are generated. It is difficult and time-consuming to interpret these messages manually. Therefore, a case-based reasoning system for diagnosing locomotive faults using such fault messages as input was developed. Candidate cases were built from historical fault logs and repair records with a two year data window. This approach is on some points similar to our approach but with the difference that a CBR system for locomotive diagnostics matches not exactly with the idea of SHM (monitoring structures and structural elements). Finding faults on machines (like locomotives) matches more with the idea of condition monitoring.

Finally, a forth approach of CBR for SHM shall be shown. A CBR prototype is used in [10], [11] to monitor wing flaps of aircrafts. Therefore, impacts shall be detected by using nine piezoceramic sensors which shall record the dynamic behavior of these wing flaps. A case base was created for each sensor and filled with cases which represent measuring data. In an adaptation step it will be decided which sensor respectively which case base and how many cases have to be used to detect and localize impacts on the wing flaps. Thereby, the case bases are organized with a SOM (Self Organizing Map). Possible impacts can be detected by using nine sensors which are well-positioned on the wing flaps. But solutions for anomalies are also not given in this approach.

5 Case-Based Decision Support for Structural Health Monitoring

Each structure has its unique properties whereby it is nearly impossible to describe the structure with facts and rules. The engineers use their experience when interpreting measuring data. For these reasons CBR shall be used for developing a decision support system to interpret measuring data in terms of SHM. By that, the most important advantages of CBR for SHM are [4]: 1) Using previous experience of engineers. 2) Concentration on experienced important aspects of certain problems. 3) Avoiding the repetition of failures in interpreting measuring data.

Basically, three possible opportunities exist to use CBR for a decision support system for SHM [4]:

- *Periodic Monitoring*: This kind of monitoring bases either on the comparison of similar measurements of similar structures or on the comparison of past measurements of the structure which shall be analyzed.

- *Supporting the Design of Structures*: The provision of information and knowledge from known cases can be used for the design of new structures. If similar structures from the case base have unusual structural behavior, these abnormalities shall be considered by the design of new structures.
- *Permanent Monitoring*: Structures can also be monitored permanently. If structures are at higher risk this kind of monitoring is mainly used. Thereby, an integrated alarm system shall give an alarm if a critical condition is reached. Basically, permanent monitoring compares past with new measurements from the structure which shall be analyzed and not with similar structures. If an actual measurement is similar to a past measurement of the same structure then the actual measurement has also a similar assessment as the past measurement.

In the following the data preparation and the calculation of similarities are presented.

5.1 Data Preparation

After engineers have made measurements of a structure the measuring data has to be analyzed and assessed. Thereby, the condition assessment can be done in form of risk levels or as a classification. Measurement results like eigenfrequencies or damping coefficients and, in most cases, also additional data are considered for the analysis. These additional data can be the optical inspection and assessment of a structure (e.g., for lamp posts: the assessment of the coat or the inside condition of the post (e.g. rust)).

Consequently, each case consists of plenty of attributes with different relevance and data types. The goal is to represent the cases as points in an n-dimensional, metric space (each dimension is between an interval of 0 and 1). Thereby, the basis is the Euclidean distance. Based on this a generalized distance can be created which does not act on the assumption of two numerical attributes but on a function f_i for each kind of distance between two attributes. Formula (1) shows this generalized distance calculation [4], [7].

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} f_i(x_i, y_i)^2} \tag{1}$$

Formula (1) calculates a value between 0 and 1 whereby this represents the distance between two attribute values of a certain dimension. Conducting experiments with the decision support system for lamp posts (see chapter 6) it emerged that the calculation of similarities in a metric space has been proved by using numerical attribute values (e.g. eigenfrequencies) and predefined distances or ordinal values for non-numerical attribute values (e.g. rust). These predefined distances or ordinal values have to be defined by civil engineers with high experience.

Thereby, the distance function $d(o_x,o_y)$ of two objects o has to be symmetric (2), reflexive (3), non-negative (4) and transitive (5) [14].

$$d(o_x, o_y) = d(o_y, o_x) \tag{2}$$

$$d(o_x, o_x) = 0 \quad (3)$$

$$d(o_x, o_y) \geq 0 \quad (4)$$

$$d(o_x, o_y) \leq d(o_x, o_z) + d(o_z, o_y) \quad (5)$$

Predefined distances can only be used if all possible attribute values are known. The number of predefined distances can become very huge if the number of attribute values n is not small. The following function (6) shows the calculation of the number of predefined distances.

$$\frac{n^2 - n}{2} \quad (6)$$

For example, if the number of attribute values is 20 the number of predefined distances is 190. Thus, the definition of predefined distances by civil engineers is unmanageable. For this reason, ordinal values shall be used if the attribute values can be sequenced.

5.2 Calculation of Similarities

According to [14] similar objects can be described as a set of objects which are "near" to each other. The need for representations and abstractions can be satisfied with a metric space notation. Thereby, different types of similarity queries exist. Two of the most popular are "similarity range" and "nearest neighbour search" and shall be introduced briefly in this contribution. The similarity search in the metric space, in respect to a query object and a certain similarity, can be seen as a kind of ranking of objects.

The two mentioned similarity queries shall now be described. The first example is the range query R. It bases on a query object o and a radius r. All objects from the set of objects O which are within the range o_{res} are returned [14].

The second type of a similarity query which shall be introduced in this contribution is the nearest neighbour query. Thereby, a certain number (k) of the most similar objects to a certain query object are returned. Especially for CBR the nearest neighbour query can be an important query type and provides a set of the most similar cases to a new case [14].

In general, the similarity between two cases X and Y is the inverse, normalized distance $1\text{-}d(X,Y)$. For calculating the distance a lot of methods exist but generally, two kinds of distance functions can be mentioned: the discrete (set of possible values) and the continuous (mostly normalized). Also for calculating the similarity between cases a lot of approaches exist. Thereby, for each kind of attribute (quantitative, qualitative...) a method has to be defined which can calculate the distance respectively the similarity between attributes and finally the distance of all different attributes as numerical value. A very general similarity measure is the similarity measure by Hamming. The weighted similarity measure by Hamming uses weights w_i to emphasize some attributes (w_i=1 means that the attribute is very important for assessment; w_i=0 that the attribute is completely unimportant). Due to the fact that not only two-valued attributes but also attributes with arbitrary attribute values can be evaluated, the weighted similarity measure by Hamming can be generalized.

Therefore, the generalized similarity measure by Hamming (7) seems to be adequate for a case-based decision support system to assess lamp posts (see section 6) [14].

$$d(X,Y) = \frac{\sum_{i=1}^{n} w_i \times f_i(x_i, y_i)}{\sum_{i=1}^{n} w_i} \tag{7}$$

The function f_i calculates a value between 0 and 1 which represents the (normalized) distance between two attribute values of a certain attribute type (dimension). It is adequate for the condition assessment of lamp posts to use similarity representations in the metric space with numerical attributes (e.g. eigenfrequencies) and predefined distances or ordinal values between non-numerical attribute values (e.g. optical assessment of lamp post's condition inside).

6 Example: Assessment of Lamp Posts

In the following section, an example of a case-based decision support system for the semi-automated assessment of structures in terms of SHM shall be described. Thereby, the assessment of lamp posts (simple structure) is used exemplarily. At first, lamp posts are introduced (description, attributes, manual assessment). Secondly, the case-based prototype is shown. In a next step, tests and experiments with real measuring data are presented. Finally, the results of using CBR for the assessment of lamp posts are summarized.

6.1 Description of Lamp Posts

Lamp posts have as all other civil engineering structures limited life-time. Especially lamp posts made of steel are affected from numerous negative environmental influences like rain, humidity, wind or even aggressive ground conditions. These influences over the years cause corrosion inducted reduction of cross section and load-bearing capacity as well as material fatigue. Therefore, lamp posts have to be inspected in periodic intervals like all other structures of the traffic infrastructure. The manual assessment is done by experts who have to analyze the measuring data and combine the measurement results with parameters of the optical inspection. Subjectivity and time pressure influence the results negatively. Thus, increasing the objectivity and decreasing expenditure of time shall be achieved with a case-based system [7], [5].

Discussions with civil engineering experts about all possible attributes for lamp post assessment have shown that the number of attributes could be reduced to 20 which are the most important ones for semi-automated lamp post assessment. In general, three different attribute types can be distinguished:

- *Geometric Attribute Types*: Cover the geometric classification of lamp posts (e.g., height, number of arms...).
- *Numerical Attribute Types*: Include the eigenfrequencies which are detected from the measured signals. Thereby, 12 different eigenfrequencies are considered. Eigenfrequencies 1-3, each with "longitudinal stimulated", "longitudinal ambient", "transverse stimulated" and "transverse ambient". "Longitudinal" means the vibration in longitudinal direction and "transverse" the vibration in transverse direction. "Stimulated" stand for stimulating the

structure e.g. with hammer strokes and finally, “ambient” means a measurement under natural environmental influences.

- *Descriptive Attribute Types*: Attributes which describe the optical inspection (e.g., condition inside, condition outside, condition lower region, condition upper region...).

Beside these 20 attributes also other attributes could be considered but this would only be useful for fine-tuning of the case-based system.

The manual assessment of lamp posts by engineers is done in the following steps:

1. *Optical Assessment*: E.g. the inspection of the lamp post and stand if there are obvious damages.
2. *Measurement*: The vibration behavior is recorded with high sensitive accelerometers under ambient and stimulated influences.
3. *Evaluation*: The engineer detects a raw and smoothed spectrum, an averaged normalized power spectral density, all relevant eigenfrequencies, mode shapes of the basic frequencies, critical damping for the dominant eigenfrequencies and vibration intensity of selected frequencies.
4. *Classification*: All parameters have to be considered in combination to assess the condition of the lamp post successfully. Thereby, the civil engineers use six classes of classification: A (post is at least twelve years stable) until F (post has to be replaced immediately).

6.2 Case-Based Prototype

In this section the prototype of the case-based decision support system for the semi-automated assessment of lamp posts to support the engineers shall be introduced. The analysis of measuring data is based on real measuring data of lamp posts (provided from [16]). The case base contains 799 cases and each case represents a single measurement of a certain lamp post. These 799 cases consist of parameters (eigenfrequencies, optical inspection...) and a classification (classes A-F) from a civil engineering expert. The assessment done by the case-based system also provides solutions in form of classes A-F [4].

Evaluation. The user can load or enter measuring data of a lamp post into the CBR system. These measuring data can be compared now with the cases in the case base and the similarity respectively the distance to the most similar cases are presented. The most similar case to the current measuring data is used as reference case to suggest the condition of the new case. It is also possible that not only one reference case is considered but also several reference cases which are the most similar ones. The reason for this is that it is possible that the most similar case is for example a lamp post with condition A. The next four most similar cases can have a classification of B. Therefore, normally it is more likely that the current case has also condition B.

Weighting. Some attributes can be more important for assessment than others. For this reason, weights can be used for each attribute. Also the engineer considers some attributes as more important than others during the manual assessment. But again the problem is the subjectivity of the engineers. By using a computer-aided assessment always the same weighting can be used. A board of civil engineering experts can define the weighting for each attribute which guarantees most possible objectivity.

The tests and experiments were done with different weightings to analyze their influence. Weightings from civil engineering experts, with the method PCAWeighting and a self-made optimized weighting were tested.

Result. After loading or entering measuring data the evaluation/comparison can be finished. The case-based system presents the most similar cases with already known classifications (done by the engineer). According to the number of reference cases the solution/classification of one similar case from the case base is selected. The user has now the possibility to correct the suggested classification. Maybe s/he has further information about the lamp post which is not recorded but influences the engineer's decision. Finally, the user has the possibility to store the new case in the case base for further reuse (current measurement together with the reused (adapted) solution/classification) or to discard the evaluation.

6.3 Tests and Experiments

This section shall summarize numerous tests and experiments which were made with real measuring data of lamp posts using the case-based decision support system. The 799 cases in the case base are classified by a civil engineering expert (class A: 523, B: 170, C: 66, D: 31, E: 6, F: 3) whereat each case is compared with all cases in the case base. All tests and experiments were done with one, three, five, ten, 15 and 20 reference cases to detect the influence of the number of reference cases for the classification. By using cluster centers (one for each class A-F) only six reference cases are considered for classification [4].

Outlier Detection. For detecting outliers in the set of measurements the so-called Density-based-Outlier-Detection method is used. Outlier detection is an important task to avoid the case that an outlier is used as reference case. An outlier is detected if at least a part p of all objects is further away than distance D. In this case-based system the distance D is defined with 1.5 and part p with 80%. 59 outliers of the 799 cases could be detected by using the Density-based-Outlier-Detection method with these parameters. The outlier detection caused that nearly in all test cases the results were better than without outlier detection (up to 8.54% of quality improvement). The detected outliers were reviewed by civil engineering experts who corroborated the incorrectness of the manual assessment which was the reason for the detection as outlier.

Cluster Centers. For correct classification of critical cases (classes E or F) the case base has to be reduced. These critical cases occur only rarely (class E: 6, F: 3). But the correct detection of these cases is a very interesting part of the case-based system. Without nearly the same number of cases for each class this challenge cannot be achieved with our current approach. Therefore, the case base has to be reduced to e.g., only six cases which represent the cluster centers of each class A-F. A new measurement/case can now only be compared with these six cluster centers and the most similar representative is used for suggesting the solution/classification. Thereby, tests with a different number of reference cases must not be done. The goal for detecting also critical cases correctly could be achieved. All cases of class F (3) and up to two-thirds of class E (6) could be assessed correctly.

6.4 Results

Finally, the results of the tests and experiments with the case-based decision support system for the semi-automated assessment of lamp posts and the differences between the manual assessment done by civil engineers and the case-based system shall be summarized.

Results of Tests and Experiments. Regarding all executed tests and experiments the correctness of evaluation tends to be better if more reference cases are considered. In general, the more reference cases are used for classification the more the correctness of classification increases (under the assumption of using one, three, five, ten, 15 and 20 reference cases). Another indicator is the averaged correctness of evaluation per class A-F. This indicator is interesting if considering also critical cases from classes E and F. Thereby, the averaged correctness of evaluation per class tends to be better if less reference cases are considered. The best result by far was achieved with cluster centers (only one representative per class).

Considering the tests and experiments with different weightings the best results could be achieved with the self-made optimized weighting and the PCAWeighting. The weighting of the civil engineer could not really improve the results in comparison with no weighting and shall be reconsidered.

The correctness of evaluation in comparison with the civil engineering experts achieved 85.86-89.05% by considering reference cases and 78.22-86.76% by considering cluster centers. The correctness of evaluation per class achieved 43.31-49.22% by considering reference cases and 58.72- 72.14% by considering cluster centers.

Furthermore, the runtime of assessing 799 respectively 740 cases could be reduced from about 30 seconds (considering reference cases) to about three seconds (considering cluster centers) [4]. A better performance was not necessary and not in focus because the experts need a one day review anymore.

Differences between Manual and Semi-Automated Assessment. The case-based system enables the semi-automated assessment of measuring and optical inspection data of lamp posts. The considered parameters can be reduced to 20 attributes (twelve eigenfrequencies and eight attributes of the visual inspection) to ensure a correctness of evaluation up to 89.05% and a correctness of evaluation per class up to 72.14%. On condition that this quality of assessment is sufficient enough the following advantages could be summarized by using computer-aided system identification:

- By reducing all possible parameters to 20 attributes it is not essential anymore to determine mode shapes, damping coefficients and vibration intensities. Furthermore, it is sufficient to consider only the first three eigenfrequencies (each with longitudinal stimulated, longitudinal ambient, transverse stimulated and transverse ambient).
- The assessment of lamp posts is more objective.
- No more complex interpretations of all possible parameters of lamp posts are necessary. This yields to saving time and therefore costs for the assessments. Thus, more than 90% of costs can be reduced.

7 Conclusion and Future Work

The successful analysis and assessment of lamp posts by using CBR in terms of SHM could be shown. The civil engineering experts can be supported in assessing lamp posts and relieved of routine work. The complexity of interpretation and assessment of measuring data can be reduced by using a case-based decision support system because only 20 attributes (with up to 89.05% correctness of evaluation and 72.14% correctness of evaluation per class) have to be considered. Class A could be assessed correctly up to 99.58% in comparison with the engineers. Parameters like mode shapes, damping coefficients or vibration intensities must not be identified. The subjectivity of engineers does not influence the results of assessment as much as without the computer-aided system and future assessments base on the same assessment scheme each time. Furthermore, the case-based system can reduce time and costs for the assessment immensely. Therefore, the assessment of about 800 lamp posts can be done within a few seconds and about one day of expert's review. On the other hand the manual assessment needs about 100-130 working hours [4]. The feedback from civil engineering experts shows that our approach is a tremendous improvement and support in comparison with the manual assessment.

The case-based system can be adapted to different requirements by using an adequate number of reference cases or the reduction of the case base to cluster centers of the six classes:

- If the system shall execute routine work to relieve engineers (cases of the class with the highest number of cases (condition A)) more than one reference cases should be considered to guarantee a correct classification of nearly 100%.
- If the system shall also assess critical cases (classes E and F which are only available in a small number) the case base should be reduced to the six cluster centers. An almost same number of cases per class shall be ensured.

The correct classification of cases in classes with a small number of cases is also a main goal of case-based decision support systems because critical conditions of structures, in this case lamp posts, have to be detected to avoid pending damages.

The tests and experiments also have shown that the previous weighting of civil engineers have to be reconsidered and optimized because without weighting the results are nearly at the same quality level. But with corresponding tests the weightings can be optimized.

At the end of this contribution future work shall be described to optimize the case-based decision support system. Thereby, the consideration of environmental influences, a kind of plausibility check and a key performance indicator for the level of correctness shall be explained [4]:

- *Consideration of Environmental Influences*: The current prototype does not consider environmental influences which can impact the dynamic behavior of structures. The temperature (air temperature as well as the material temperature) can have influence on the eigenfrequencies. If the temperature increases the eigenfrequencies decrease. If the temperature decreases the eigenfrequencies increase. Also the humidity, wind or the traffic volume could be considered. For this purpose, rules could be defined and integrated into the case-based system which e.g., defines the influence of the temperature on the eigenfrequencies.

- *Plausibility Check*: Measuring data shall be checked for plausibility before they are analyzed by the case-based system. Only valid measuring data shall be stored in the case base for further reuse. It is possible that failures during the measurement process occur (e.g., if the lamp post has contact with the local electricity supply). In this case these measuring data shall not be stored as new cases in the case base because this would influence future assessments negatively. These asymmetries maybe could be detected with correlation coefficients.
- *Key Performance Indicator for the Level of Correctness*: A key performance indicator should identify the level of correctness. Thereby, the civil engineer can be supported even more if s/he knows the probability of correctness of the suggested condition assessment. The engineer can be relieved of routine work even more because s/he has to control the computer-aided assessment only marginally. An obvious way would be the consideration of the distance of the new measurement to the most similar reference cases. If the similarity is near 100% it is probable that the condition assessment could be correct.

Furthermore, the semi-automated assessment of more complex structures like bridges will be further researched in future.

Acknowledgement. The research was developed in collaboration with the Johannes Kepler University of Linz (Institute for Applied Knowledge Processing) and is funded under EU-projects SafePipes (Safety Assessment and Lifetime Management of Industrial Piping Systems, [FP6-NMP]) and IRIS (Integrated European Industrial Risk Reduction System, [FP7-NMP]). Parts of the contribution were supported within the program Regionale Wettbewerbsfähigkeit OÖ 2007-2013 by the European Fund for Regional Development as well as the State of Upper Austria.

References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variation, and System Approaches. AI Communications 7, 39–59 (1994)
2. Federal Highway Administration, Official Website, Bridge Technology, `http://www.fhwa.dot.gov`
3. Freudenthaler, B.: Case-based Reasoning (CBR): Grundlagen und ausgewählte Anwendungsgebiete des fallbasierten Schließens. VDM Verlag, Saarbrücken (2008)
4. Freudenthaler, B.: Case-based Decision Support for Structural Health Monitoring. PhD thesis, Johannes Kepler University of Linz (2009)
5. Freudenthaler, B., Fritz, M., Stumptner, R., Küng, J.: Case-based Reasoning for the Automated Assessment of Simple Structures in Terms of Structural Health Monitoring. In: 7th International Workshop on Structural Health Monitoring 2009. DEStech Publications, Lancaster (2009)

6. Freudenthaler, B., Stumptner, R., Forstner, E., Küng, J.: Case-based Reasoning for Structural Health Monitoring. In: Uhl, T., Ostachowicz, W., Holnicki-Szulc, J. (eds.) 4th European Workshop on Structural Health Monitoring 2008. DEStech Publications, Lancaster (2008)
7. Fritz, M.: Standsicherheitsprüfung von Lichtmasten: Visuelle Inspektion und messtechnische Prüfung mit BRIMOS®. Technical Report, Vienna Consulting Engineers (2009)
8. Kolakowski, P., Mujica, L.E., Vehí, J.: Two Approaches to Structural Damage Identification: Model Updating versus Soft Computing. Intelligent Material Systems and Structures 17(1), 63–79 (2006)
9. Kolodner, J.: Case-based reasoning. Kaufmann, San Mateo (1993)
10. Mujica, L.E., Vehí, J., Staszewski, W., Worden, K.: Impact Damage Detection in Aircraft Composites Using Knowledge-based Reasoning. In: Chang, F.K. (ed.) 5th International Workshop on Structural Health Monitoring. DEStech Publications, Lancaster (2005)
11. Mujica, L.E., Vehí, J., Staszewski, W., Worden, K.: Impact Damage Detection in Aircraft Composites Using Knowledge-based Reasoning. Structural Health Monitoring 7, 3 (2008)
12. Penta, K.K., Khemani, D.: Satellite Health Monitoring Using CBR Framework. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 732–747. Springer, Heidelberg (2004)
13. Richter, M.M.: Introduction. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 1–15. Springer, Heidelberg (1998)
14. Stumptner, R., Freudenthaler, B., Küng, J.: On Similarity in Case-based Reasoning for Structural Health Monitoring. In: Quesada-Arencibia, A., Carlos Rodriguez, J., Moreno-Diaz Jr., R., Moreno-Diaz, R. (eds.) EUROCAST 2009. LNCS, vol. 5717, pp. 231–238. Springer, Heidelberg (2009)
15. Varma, A.: ICARUS: Design and Deployment of a Case-Based Reasoning System for Locomotive Diagnostics. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 581–596. Springer, Heidelberg (1999)
16. Vienna Consulting Engineers, `http://www.vce.at`
17. Wenzel, H.: Health Monitoring of Bridges. John Wiley & Sons Ltd., Chichester (2009)
18. Wenzel, H., Geier, R., Eichinger, E.M.: Endbericht Brückentests: Untersuchungen anlässlich des Abbruchs ausgewählter Tragwerke. Technical Report, Vienna Consulting Engineers und Technische Universität Wien (2001)
19. Wenzel, H., Pichler, D.: Ambient Vibration Monitoring. John Wiley & Sons Ltd., Chichester (2005)

A Case Base Planning Approach for Dialogue Generation in Digital Movie Design

Sanjeet Hajarnis, Christina Leber, Hua Ai,
Mark Riedl, and Ashwin Ram

College of Computing, Georgia Institute of Technology,
Atlanta, Georgia, U.S.
{sanjeet,cleber3}@gatech.edu,
{hua.ai,riedl,ashwin}@cc.gatech.edu

Abstract. We apply case based reasoning techniques to build an intelligent authoring tool that can assist nontechnical users with authoring their own digital movies. In this paper, we focus on generating dialogue lines between two characters in a movie story. We use Darmok2, a case based planner, extended with a hierarchical plan adaptation module to generate movie characters' dialogue acts with regard to their emotion changes. Then, we use an information state update approach to generate the actual content of each dialogue utterance. Our preliminary study shows that the extended planner can generate coherent dialogue lines which are consistent with user designed movie stories using a small case base authored by novice users. A preliminary user study shows that users like the overall quality of our system generated movie dialogue lines.

Keywords: case based planning, story generation, dialogue generation.

1 Introduction

Recent advances in artificial intelligence (AI) techniques radically change the way users interact with computer entertainment systems. Users are no longer passive consumers of built content, but have become involved in adding value to computer entertainment systems by providing their own content. User generated AI has been deployed in computer games to integrate user-designed strategies into real-time strategy games [17, 28], as well as to train virtual avatars to play in user-defined styles in virtual reality games [14]. Although game designers define the games, user generated game behaviors can tailor character specificities while taking advantage of professionally crafted game space. Moreover, a new trend in the computer entertainment industry is to assist users in creating entertaining contents themselves. Users can now create their own plots in games using intelligent authoring tools [7]. Furthermore, intelligent systems with AI techniques are designed to assist users to build highly dynamic game plots [13]. It is generally believed that users are more engaged in games partially or fully created by themselves or their peers.

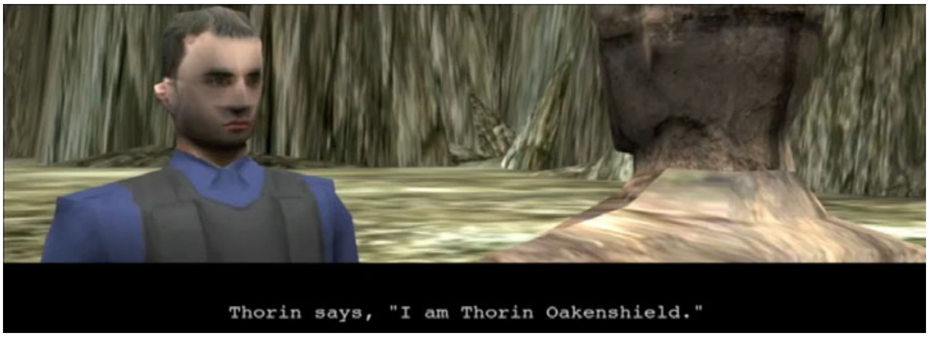


Fig. 1. A screen shot of a movie scene generated by Cambot

Besides generating game plots, we are also witnessing new interests in user-generated digital media, which serves as the sole content source for popular social websites such as YouTube^{TM}, Flickr^{TM}, and many others. YouTube recently launched a new portal where users can design their own video clips using animation tools such as GoAnimate^{TM} or Xtranormal^{TM} to build custom videos featuring their own story plots, dialogues, and virtual avatars[1].

While such tools make it possible for non-technical users to design their own media contents, it is still hard for novice users to manage story writing and animation design at the same time. In this paper, we describe our recent effort in using AI techniques, i.e., planning and conversation generation, to build an intelligent system for non-technical users to design their own digital media contents in the format of animated digital movies. We identify two key challenges in this task. One is to design a compelling and interesting story plot, and the other is to render animation. We use Cambot [18] (see Fig. 1.), a virtual movie director for 3D worlds to shoot and edit animation scenes into a movie. In this paper, we describe our work on applying case based planning to assist users in generating story plots that are expressed through dialogue lines between two movie characters. More specifically, we have designed and implemented a case based planning system to fill in gaps in user-generated story contents. We assume that when a user wants to design her own movie, she has a general idea about what story she wants to deliver, e.g., several key movie scenes. However, the user may find it tedious to author the full details in the story or alternatively, novice users would find it difficult to author stories from scratch. Our system is designed as an intelligent authoring tool, which can automatically fill in story contents to free users from the authoring burden, or suggest story content so that users can revise and edit them as per their will.

We consider applying Case Based Planning techniques in story generation because such techniques can utilize interesting story contents generated by previous users to enrich new users' stories. Case Based Planning has been applied to a variety of tasks, including computer game AI design, robotics and story generation [6, 9, 22]. It is planning as remembering [8], which involves reusing previous plans and adapting them to suit new situations. However, applying case based reasoning for story

[1] Can be created at YouTube.com/create.

generation is different from the well-studied applications in the game domain [9]. In a game, there is a finite set of actions that a user or an avatar can take. But in story generation, there is not a defined set of actions that can happen in the story. Instead, the characters in the story can perform any actions that are suitable given a certain story context. There have been query based approaches to restrict the content for using Case Based Reasoning in story generation [6]. Another case based reasoning solution is used for story representation and adaptation in the fairy tale domain where the CBR uses cases extracted from a multi-move story scripts given by Propp [19]. In this study, we code character behaviors at an abstract level in order to utilize similarities among different action sequences in case based planning. We define character behaviors in regards to the behaviors' effects on characters' emotion changes because we believe emotion expression is an important factor in movie story generation.

The rest of the paper is organized as follows. Section 2 surveys different applications of case based planning as well as story and dialogue generation. Section 3 defines our task domain. Section 4 introduces our case based planning system. Section 5 describes a preliminary evaluation. We conclude in Section 6 and point out future directions.

2 Related Work

Case-based reasoning (CBR) is a known problem solving technique based on reutilizing specific knowledge of previously experienced problems stored as cases [1], [11]. The CBR cycle consists of four major stages: Retrieve, Reuse, Revise and Retain [1] (See Fig. 2). In the Retrieve stage, the system selects a subset of cases from the case base that are relevant to the current problem. The Reuse stage adapts the solution of the cases selected in the retrieve stage to the current problem. In the Revise stage, the obtained solution is verified (either by testing it in the real world or by examination by an expert), which provides feedback about the correctness of the predicted solution. Finally, in the Retain stage, the system decides whether or not to store the new solved case into the case base [16].

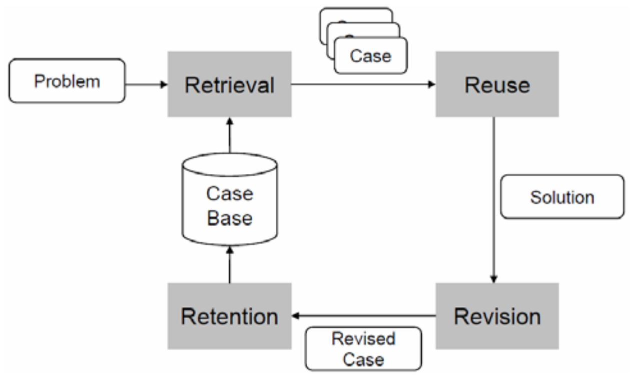


Fig. 2. The CBR cycle

We further explore varied case-base reasoning and case-base planning applications based on different underlying domains and purposes. Case-base reasoning and planning are generally used when generating plans from scratch can be computationally expensive or when many new problems can be solved using solutions from previous experiences. There have been numerous such case-based planning techniques that adhere to the above philosophy [9].

Prodigy/Analogy [27] is such an architecture which stores the reasoning trace while planning instead of storing the plans. Thus, when planning for a new problem, the system just replays the stored decision traces. While Prodigy/Analogy is an example of the derivational plan adaptation approach, CHEF is an example of the transformational plan adaptation approach [15]. PRIAR is yet another planner system that focuses on reusing previously annotated plans with a validation structure that contains an explanation [10]. This system requires annotated plans to ensure appropriate reusing of the stored plans. The MAYOR [5] system contains a concept net of all the factors in the game domain and how they are related to each other. These factors include money, crime, pollution etc. Each plan in the plan library has some expectations on the aforementioned factors. While planning, if the expectations on the factors are not complete, the concept net allows the system to manipulate the expectation failure. The above CBR systems are domain dependent and function on different domains from dialogues or conversations. HICAP [2] is a general-purpose planner that integrates hierarchical plan base structure with constraint satisfaction based on domain-dependent doctrines.

Story Generation can be viewed as a planning problem. It is a problem of crafting a structured sequence of events that can be told to a recipient [23]. A particular planning problem can be thought of as follows: Given an initial state, an underlying domain and a goal state, find a sequence of operations that transform the world from the initial state to the goal state where the operators adhere to the requirements in the domain. Such a planning problem can be solved by different types of planners. One particular class of planners that can be used to solve the above problem is called partial-order planners (POP) [20]. A partially-ordered plan consists of a set of actions that are ordered according to a set of temporal constraints forming a partial ordering such that some actions may be unordered relative to each other. Moreover, there exists causal links between two actions which imply that the first action creates changes in the world that enable the second action to be performed. POP planners encapsulate many features that appear in the cognitive models of narrative [29] and hence prove useful in narrative generation. As we mentioned in Section 1, case based planning systems have also been used in story generation [6]. Moreover, there have been attempts to combine case-base reasoning and planning algorithms to develop a better system. MEXICA uses elements of both previous story retrieval and means-ends planning [21]. Here, we perform a similar task by extending a case based planning system with hierarchical planning strategies.

In this study, we handle a subtask of story generation, which is to generate dialogue lines for the characters in a narrative story. This is a very important part in generating stories for movies. Research on dialogue generation has been mainly on information providing dialogue systems [24, 25], which are used to provide information to users in order to complete user defined tasks, such as booking flight tickets, or querying weather conditions. In these task-oriented dialogues, dialogue contents can be represented by a set of information slots. For example, in a flight booking conversation, the dialogue system needs to know information about departure and arrival cities to recommend flight options. Dialogue generation in this case is centered on updating states of these two basic information slots. The information state update approach is a simple way to keep track of conversation history [26]. We use a similar approach in managing dialogue content in our system. However, this simple approach alone is not enough to generate interesting and entertaining dialogues required in our task. Therefore, we deploy a case based planning system that decides how the dialogue content will be delivered with different emotions. More details are described in Section 4.

3 Task Domain

Our ultimate goal is to build an intelligent system which can help users generate narrative movie stories by filling in gaps between user generated contents. In this study, we focus on generating dialogues between two movie characters, because dialogues are an important way to create engaging experience in movies. Dialogue can not only deliver movie plots, but also represent the personalities of movie characters and their emotions.

In this study, our task is to generate dialogues between two main characters namely Great Goblin and Thorin, given the background story:

> *When crossing the Misty Mountains, Thorin and his company run in to a storm and take shelter in a cave. The cave actually serves as an entrance to the lair of the Goblins of the Misty Mountains who sneak into the cave while the company is sleeping and capture them. The captives are brought before the Great Goblin, who queries about why they were in his cave. Thorin speaks for his party and tries to negotiate with the Great Goblin and get released.*

When a user specifies the emotion states for the two characters, our system can generate a dialogue based on character emotions and the background story. We currently support 3 emotion states: calm, fear and anger.

4 Case-Based Reasoning for Dialogue Generation

We choose to use a case-based reasoning approach for dialogue generation because we want to make use of user generated dialogues. However, since user generated dialogues may happen in different background contexts, we cannot directly store these dialogues in the case base and retrieve them for another story. Therefore, we represent the dialogues in their abstract format using dialogue acts (Section 4.1). Then we generate dialogues in two steps. First, we use a case-based planner to generate a dialogue act sequence (4.2). Then, we apply an information state update approach for content generation (4.3). Finally, we use a simple natural language generator to deliver the full dialogue in natural language (4.4).

4.1 Dialogue Act Representation

Dialogue acts are generally used to model and automatically detect discourse structure in dialogue systems. A dialogue act represents the meaning of an utterance at the level of illocutionary force [3]. Thus, a dialogue act is approximately the equivalent of the actions in computer games. Dialogue acts are usually defined to be relevant to a particular application, although there have been efforts to develop a domain-independent dialogue act labeling systems [4].

In our task, we define a set of dialogue acts for a negotiation scene. Table 1 shows the dialogue acts and their definitions. We can see that some dialogue acts are likely to trigger the change of emotions. For example, when one character decline to answer a question, the asker is likely be angry. Similarly, threatening is likely to trigger fear. In Section 4.2, we use a case based reasoning system to generate traces of dialogue acts based on user defined character emotion states.

Table 1. List of Dialogue Acts used by the Natural Language Generation module

Dialogue Acts	Definitions
Question	Ask a question
Answer	Directly answer a question
Dodge	Give an indirect answer to a question
Decline	Refuse to answer a question
Follow-up	Ask a question based on the previous answer/question
Validate	Ask a question to further verify the previous answer
Propose	Propose a solution
Agree	Agree to a proposal
Reject	Reject a proposal
Counter	Follow-up on a rejection with a new proposal
Inform	Provide new information
Provoke	Provide new information intended to anger the listener
Persuade	Statement intended to persuade the listener of something
Threaten	Issue a threat
Warn	Issue a warning
Decide	End of conversation

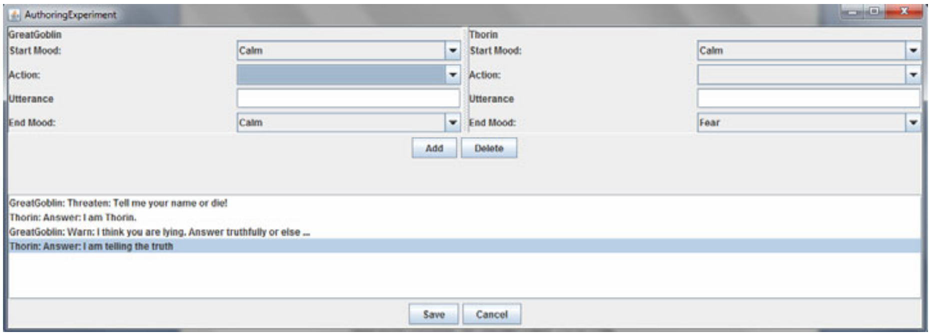
4.2 Case Based Reasoning System

We use Darmok2 [16] for case-based planning to generate dialogue act traces. In our system, since the case base consists of different plans learned from user generated cases, we also refer to it as the plan base. A plan consists of a start state of the characters in the dialogue, a sequence of dialogue actions that occurred between the two characters and an end state of the characters. A state in this dialogue generation domain consists of the characters involved in the dialogue and their emotional conditions. Currently, the domain supports three main emotional conditions namely Calm, Angry, and Fear, but the system is generic to incorporate more emotional conditions. An action comes from the list of 15 actions in the domain.

In order to use Darmok2, we need to provide the system with a case base to support the system's planning mechanisms. In Section 4.2.1, we show how cases are authored and represented in Darmok2. In Section 4.2.2, we describe how Darmok2 can be used to learn plan cases from the traces authored using our authoring interface. In Section 4.2.3, we describe how Darmok2 uses these cases for planning.

4.2.1 Case Base Authoring

The system acts as an intelligent tool to assist story authors. An author can choose to use the system for intelligent recommendations or author the entire story from scratch. When asking for recommendations, the case based planning system is called to fill in story scenes based on the current story. However, before the case based planning system can function, we need to build a case base to support its planning. Therefore, when users are authoring their own stories, story traces are stored in the system and represented as cases to be used in planning later. Fig. 3. shows the story authoring interface.



GreatGoblin		Thorin	
Start Mood:	Calm	Start Mood:	Calm
Action:		Action:	
Utterance		Utterance	
End Mood:	Calm	End Mood:	Fear

GreatGoblin: Threaten: Tell me your name or die!
Thorin: Answer: I am Thorin.
GreatGoblin: Warn: I think you are lying. Answer truthfully or else ...
Thorin: Answer: I am telling the truth

Fig. 3. Authoring interface for generating stories

The top left of the interface contains Great Goblin's state information (which includes initial mood, dialogue actions and utterances corresponding to the dialogue act and final mood). Similarly, the top right Section contains information about Thorin's state information. The bottom part contains a log of the dialogues so far, which includes both dialogue acts and the actual utterances between Great Goblin and Thorin. To add new dialogues, a user must pick a dialogue action from the list in Table 1. Then, the author can enter the dialogue in the utterance field. Once an action is picked and the utterance written, author can press the add button to include the newly created dialogue in the log.

4.2.2 Case Base Learning

Once stories are generated after the authoring phase, we can enrich our plan base by learning from these traces. Learning from a given trace involves three major steps: (i) detecting goals (ii) building plans (iii) updating the plan base. The learning algorithm detects changes in the consequent states based on the changes in the character's behavior or emotional condition. Once the changes are detected, the learning algorithm must build appropriate plans. There are two main types of plans in the plan base: (i) Goal plan – a plan that satisfies a particular emotional goal. A goal plan is created by directly calling the planner in Darmok2; (ii) Combined plan – a plan which makes use of the hierarchical goal structure to make new plans from goal plans. A combined plan is created by first decomposing the final goal state into several intermediate goal states which can eventually lead to the final state, and then calling the Darmok2 planning on each intermediate goal states. After building appropriate plans from the changes, the learning algorithm enriches the plan base with the newly created plans.

Consider a simple example:

State:
Great Goblin: Calm
Thorin: Calm
Dialogue Acts:
Great Goblin [Threaten]: Tell me your name or die!
Thorin [Answer]: I am Thorin.
State:
Great Goblin: Calm
Thorin: Fear
Dialogue Acts:
Great Goblin [Question]: Are you sure? I think you are lying.
Thorin [Answer]: I am not.
State:
Great Goblin: Angry
Thorin: Fear

The above trace generates the plan structure shown in Fig. 4.

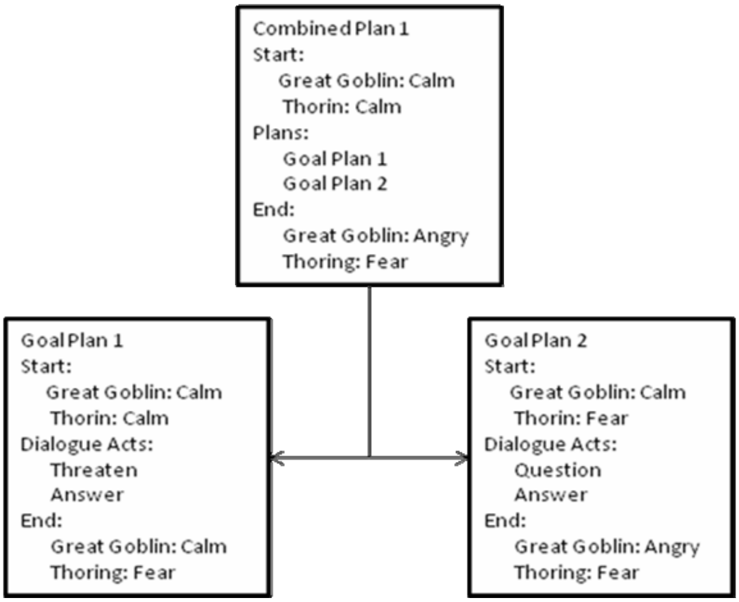


Fig. 4. A sample plan structure

Note that the size of a goal plan is not restricted to only two dialogue acts as shown in the example in Fig. 4. Similarly, a combination plan can have more than two goal plans encapsulated within it. We use the above learning procedure to improve our plan base with both goal plans and combination plans.

4.2.3 Case Based Planning

When an author needs suggestions from the system to fill certain parts of a dialogue, Darmok2 is called to retrieve reasonable plans. The plan retrieval phase ranks the plans in the plan base based on the current state in the dialogue and the target state. Two scores are calculated to retrieve the best plan: (i) goal match score – this heuristic calculates the similarity between the goal state of a plan in the case base and the targeted goal state; (ii) initial state match score – this heuristic calculates the similarity between the initial state of a plan in the case base and the retrieval query's initial state. In our current system, the retrieval phase returns up to 10 different plans and provides the Plan Adaptation phase with more options to choose the right plan.

The Plan Adaptation uses the plans obtained in the retrieval phase to perform dialogue generation. This scenario can be visualized as a gap-filling problem because a typical plan retrieval query would have a start state and end state and the system attempts to generate dialogue for the start and end states. Thus, the system attempts to come up with the most appropriate dialogue to narrate the gap between the start and the end state. The Plan Adaptation module picks the best plan from the list of retrieved plans to fit this gap. There are 4 possible events that could occur in this case: (i) the retrieved plan would completely fit the gap (ii) the retrieved plan would be a complete fit for the start state but not for the end state (iii) the retrieved plan would be a complete fit for the end state but not the start state (iv) the retrieved plan would not fit either the start or the end state. If the gap is completely filled, then the adaptation phase does not have to perform further checking. In events 2-4, the plan adaptation tries to invoke a planner to find the missing pieces. The planner in the plan adaptation phase recognizes gaps (like in events 2-4) and recursively calls the retrieval phase until all the gaps are filled satisfactorily. Thus, the planner in plan adaptation can be thought of as an example of recursive case-base planning. Consider the query with the following initial and final states to the plan-base:

Initial	**Final**
Great Goblin: Calm	**Great Goblin:** Angry
Thorin: Angry	**Thorin:** Fear

Now, in adaptation phase if the best plan so far is from

Initial	**Final**
Great Goblin: Angry	**Great Goblin:** Angry
Thorin: Angry	**Thorin:** Fear

then the planner recognizes the gap and recursively queries the plan-base with

Initial	**Final**
Great Goblin: Calm	**Great Goblin:** Angry
Thorin: Angry	**Thorin:** Angry

such that the gap in the beginning of the retrieved plan is filled satisfactorily.

Moreover the adaptation phase also looks for substitutable plans if the recursive plan adaptation fails. In our current domain, we define approximation rules for emotional changes. For instance, for Thorin's character, an emotional change from Calm to Angry can be approximated by emotional change from Angry to Fear, assuming unpleasant information or actions can make a character's emotion becomes worse. In our story domain, being angry is a worse emotional state than being calm because it shows the character feels unhappy and dislikes the current situation. Being "fear" is worse than being angry because the character feels the threats and may start to react instead of only expressing the emotion. By adding approximation rules, the plan adaptation module ensures that the retrieval query gap gets filled completely. For our application, it is more important for our system to respond to every user query than to always provide a response of good logical sense. Since our system is used to generate movie dialogues, using approximation rules may not always help the system to generate a sensible dialogue, but can possibly produce interesting and funny effects that are also favorable in movie dialogue generation.

4.3 Dialogue Content Generation

Once we have the sequence of actions that will make up the story, we must then assign each dialogue action an utterance, which is a single-sentence string that represents what the character will say. As mentioned earlier, the issue here is that we cannot merely have a one-to-one mapping from action to utterance – a set of utterances that matches one sequence of dialogue actions may not match the same dialogue actions in a different sequence. The reason for this is that different sequences of actions will result in different contexts, and since the sequence of dialogue actions is user generated (though aided by this system), we must be able to support any context that is created. Users will not want to use this system if it cannot generate dialogue that makes sense in the story they create.

To solve this problem, we use an information state update approach to track dialogue content. An utterance library is pre-authored with sets of utterances for each possible dialogue act. Each utterance is tagged with a pre-context and a post-context, which are both represented by a set of topics. The pre-context indicates which topics have already been mentioned in the conversation in order for that particular utterance to make sense. The post-context indicates which new topics the utterance introduces to the conversation and occasionally which topics, if any, the utterance can settle. These topics indicate which information has or has not been disclosed in the dialogue. By updating these information states, we keep track of which topics should be in the dialogue content. Settled topics do not need to be discussed further.

Take the following exchange as an example:

Great Goblin: How do I know you are not here to spy on us?
Thorin: We are not spies.

It does not make sense for Thorin to mention that he is not a spy until he is first accused of being a spy. Therefore, Thorin's utterance would have a pre-context of "spies", since he cannot say that utterance until the topic of "spies" is introduced. The

Great Goblin's utterance would have a post-context of "spies" to indicate that once that utterance is selected to be part of the conversation the topic of "spies" has been introduced.

Internally, the dialogue generation system keeps track of which topics are currently active in the context as part of the information states. When given an action, it determines which of the utterances in the library match both the given action and the current information state. If there is more than one possible utterance that matches, the system looks at the last utterance in the conversation to determine which possible utterance most closely matches the current context. This is done by checking which of the possible utterances' pre-contexts most closely matches the post-context of the previous utterance in the conversation. The system then selects that utterance, sets the utterance as the output of the action, updates the information state according to the post-context of the utterance, and returns the updated action.

Emotion is considered to be part of the context. People speak differently when they are angry than when they are calm or afraid, and our utterance library needs to reflect that. So, some utterances have an emotion as part of the pre-context or post-context. When given an action tagged with a desired emotional outcome, in addition to the procedure outlined above, the system will also check the post-context of each utterance to determine if it will produce the desired emotional outcome. Any utterance that will not produce that emotional outcome is ignored, even if it matches the current conversation state.

4.4 Natural Language Dialogue Generation

In the experiments reported in this paper, the dialogue generation module uses canned dialogue lines rather than generating utterances programmatically. We feared that introducing a natural language generation system could negatively bias the results if we ended up with a sequence of actions for which the natural language generation system did not perform well. We have done some work with the NLG system Personage [12], which adjusts how an utterance is generated based on the personality of the speaker, to see if we could possibly apply it in this domain and cause it to generate an utterance differently based on the emotional state of the speaker in addition to their personality. That work, however, is beyond the scope of this paper.

5 Experiments and Results

We perform a two-phase preliminary experiment to evaluate our system. In the first phase, we invite three users (2 males and 1 female) to write stories using our system. These stories are used to construct the plan base for Darmok2. We choose to use user-generated stories instead of expert-authored stories because we want to evaluate our system in a realistic scenario, in which the planning system uses user-generated stories to provide recommendations upon user requests. In addition, if knowledge is engineered by expert authors, it conflates the creative abilities of the system and those of the authors'. Therefore, we do not use expert-authored stories to bias our system evaluation.

The three subjects were given the background story shown in Section 3. They were asked to generate 3 dialogues between the two characters in the story. None of the users have difficulties writing the stories although only one of them is familiar with English fantasy fictions like our background story. The average length of user generated dialogues is 5.67 turns. The average number of emotion changes per dialogue is 2.5 times. In total, the plan base constructed by user-generated dialogues has 53 cases.

These subjects were then asked to each create 2 cases in which they would want to get story recommendations from our system, i.e., to ask the system to generate the dialogue for them. In each case, they need to specify the start and end emotion states of both characters in the dialogue. The subjects were given enough prior background information about the domain story so that they could make an informed choice for the start-end pairs.

Once we obtained all 6 start-end pairs, we generated dialogues for each pair using three different versions of the plan base. The first version of the plan bases contained 33% of the user generated plans (17 plans) for the plan base to start off with, the second version contained 66% (34 plans) of the user generated plans and the third version contained 100% of the user generated plans (53 plans). Each subset of the plan base was randomly chosen. We want to test the impact of the size of the plan base on story quality. In total, we generated 18 (6*3) dialogues. We recruited a different set of judges to evaluate these dialogues because we did not want the judges to be biased when seeing portions of their own dialogues being used in new ways. We also decrease a judge's bias by assigning one dialogue to two judges and use the average ratings by the two judges as the final ratings of that dialogue. Our hypothesis is that by extending the CBR with a hierarchical planning strategy, the quality of generated dialogues would not be impacted by the different sizes of the plan bases.

We recruited 4 judges (3 males and 1 female) to assess the quality of generated dialogues. The judges were assigned 3 stories each. Each judge was also provided with dialogues generated by all the three versions of the plan base so they would be capable of providing a good comparison. We asked the judges to rate the dialogues on four evaluation measures, i.e., logic coherence, interestingness, fitness with background story, and fluency. We also asked the judges to give an overall rating of how well they like a dialogue. Ratings are on a 4-point Likert scale. We did not use a 5-point likert scale because we wanted our judges to clearly indicate whether our system's performance is good (a rating of 3 or 4) or bad (a rating of 1 or 2). We did not want the judges to give neutral answers.

Table 2. Summary of average ratings across all dialogues and all judges

Logical Coherence	Interestingness	Fitness with background	Dialogue Fluency	Overall
3.03 (±0.56)	2.78 (±0.84)	3.14 (±0.68)	3.22 (±0.52)	2.69 (±0.73)

Table 2 summarizes the average ratings across all dialogues and all judges. The numbers in the parenthesis show the standard deviations. The average overall rating indicates that the judges' overall feedback on our system's performance is positive (2.69). It is not surprising to see that dialogue fluency gets a high rating because the natural language generation module uses pre-authored utterances. Logical coherence

(3.03) and fitness with background (3.14) scores show that the performance of our planning module is also positive. The interestingness of the dialogues (2.78) gets the lowest rating among all the measures. Using Pearson's correlation, we observe that there is a strong correlation between the length of the generated dialogues and its interestingness ($R = 0.71$, $p \leq 0.05$). In other words, judges tend to rate longer dialogues to be more interesting. When controlling for dialogue length, each of our four measures strongly correlate with judge's overall ratings, which shows that all our measures are important aspects that judges consider when rating a dialogue. When we use the four measures to build a regression model to predict the overall rating, fluency and interestingness have the highest coefficients, which show that these two measures have the highest weights in impacting the judge's overall ratings. Since our system generates relatively short dialogues (on average 5 turns) and therefore gets a lower interestingness score, the overall dialogue ratings are not very high. Here is an example of generated dialogues (4 turns) for the following conditions. In the beginning of this dialogue, Great Goblin is calm and Thorin is angry and at the end of the dialogue Great Goblin is angry while Thorin is scared.

Great Goblin: Who are you?
Thorin: I'm not answering your questions.
Great Goblin: Answer me truthfully, and I'll consider letting you live.
Thorin: No. I demand you let us go.

When comparing dialogues generated from the three different sizes of plan bases, we do not observe any statistical significance among any of the evaluation measures or the overall scores (p values larger than 0.05). This supports our hypothesis by showing that our system can work with a small plan base generated by naïve users. Although this result is based on our simple domain, we are encouraged by this result because being able to work with a small plan base makes it possible for users to switch our system to a new story domain quickly.

6 Conclusions

In this paper, we report our recent work on using a case based planning system for narrative story and dialogue generation. Our system acts as an intelligent tool to assist users to write their own movie scripts. When users use our system to write their movie lines, their scripts will be stored in the system. Later, when a user lacks the idea to write certain scenes, she can use our system to fill in the gap in the movie. Our system uses user-generated story to construct its plan base, and then uses the plan base to generate new stories to fill in gaps in new stories. Our task of applying a case based planning system on dialogue generation distinguishes from previous applications in the game domain because system actions in one dialogue need to be represented at a higher level in order to be reused in other dialogues. Therefore, in our system, we code each utterance in dialogues with a dialogue act to represent its conversational strategy and its effect on conversational partners' emotions. Also, we build a content planner using an information state update approach in supplement to the case based planner in order to generate the dialogue content.

Our results show that our system can generate logically coherent stories that reflect background contexts. The generated stories get positive user ratings in terms of their overall quality. Moreover, our system can work with a small case base contributed by naïve users. This feature enables our system to be used by users in new domains.

However, our stories have relatively lower ratings on their interestingness, which is found to be directly correlated with dialogue length. In the current system, we do not have a drama management module that shapes the generated dialogues in terms of its interestingness. In the future, we plan to add heuristic rules which can generate more interesting dialogues, but not the most direct dialogues.

Acknowledgements. The authors gratefully acknowledge the support of the U.S. Defense Advanced Research Projects Agency (DARPA) for this effort.

References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications 7(1), 39–59 (1994)
2. Aha, D.W., Breslow, L.A., Munoz-Avila, H.: Conversational Case-Based Reasoning. Applied Intelligence 14, 9–32 (2001)
3. Austin, J.: How To Do Things With Words. Harvard University Press, Cambridge (1962)
4. Core, M., Allen, J.: Coding Dialogues With The DAMSL Annotation Scheme. In: Working Notes of the AAAI Fall Symposium on Communicative actions in Humans and Machines, Cambridge, MA, pp. 28–35 (1997)
5. Fasciano, M.: Everyday-World Plan Use. Technical Report TR-96-07, University of Chicago (1996)
6. Gervas, P., Diaz-Agudo, B., Peinado, F., Hervas, R.: Story Plot Generation Based on CBR. Knowledge Based Systems 18(4-5), 235–242 (2005)
7. Hajarnis, S., Barve, C., Karnik, D., Riedl, M.O.: Supporting End-User Authoring of Alternate Reality Games with Cross-Location Compatibility. In: Proceedings of the 24th Conference of the Florida Artificial Intelligence Research Society, Palm Beach, FL (2011)
8. Hammond, K.F.: Case-Base Planning: Viewing Planning as a Memory Task. Academic Press, New York (1989)
9. Hammond, K.F.: Case-Based Planning: A Framework for Planning from Experience. Cognitive Science 14(3), 385–443 (1990)
10. Kambhampati, S., Hendler, J.A.: A Validation-Structure-Based Theory of Plan Modification and Reuse. Artificial Intelligence 55(2), 193–258 (1992)
11. Kolodner, J.: Case-based reasoning. Morgan Kaufmann, San Francisco (1993)
12. Mairesse, F., Walker, M.: PERSONAGE: Personality Generation for Dialogue. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), Prague (2007)
13. McCoy, J., Treanor, M., Samuel, B., Tearse, B., Mateas, M., Wardrip-Fruin, N.: Authoring Game-Based Interactive Narrative using Social Games and Comme il Faut. In: Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate (2010)
14. Mehta, M., Ram, A.: Run Time Behavior Adaptation for Real Time Interactive Games. IEEE Transaction of AI and Games 1(3) (2010)

15. Munoz-Avila, H., Cox, M.: Case-Based Plan Adaptation: An Analysis and Review. IEEE Intelligent Systems (2007)
16. Ontañón, S., Sugandh, N., Mishra, K., Ram, A.: On-line Case-Based Planning. Computational Intelligence 26(1), 84–119 (2010)
17. Ontañón, S., Ram, A.: Case-Based Reasoning and User-Generated AI for Real-Time Strategy Games. In: Gonzales-Calero, P., Gomez-Martin, M. (eds.) AI for Games: State of the Practice (2011)
18. O'Neil, B., Riedl, M.O., Nitsche, M.: Towards Intelligent Authoring Tools for Machinima Creation. In: Proceedings CHI, Extended Abstracts, Boston, MA (2009)
19. Peinado, F., Gervas, P., Diaz-Agudo, B.: A Description Logic Ontology for Fairy Tale Generation. In: 4th International Conference on Language Resources and Evaluation: Workshop on Language Resources and Linguistic Creativity (2004)
20. Penberthy, J.S., Weld, D.S.: UCPOP: A Sound, Complete, Partial-Order Planner for ADL. In: Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning, pp. 103–114 (1992)
21. Pérez y Pérez, R., Sharples, M.: MEXICA: A Computer Model of a Cognitive Account of Creative Writing. Experimental and Theoretical Artificial Intelligence 13, 119–139 (2001)
22. Ram, A., Santamaria, J.C.: Continuous Case-Based Reasoning. In: Proceedings of the AAAI 1993 Workshop on Case-Based Reasoning, Washington DC, pp. 86–93 (1993)
23. Riedl, M.O.: Story Planning: Creativity Through Exploration, Retrieval and Analogical Transformation. Minds and Machine 20(4), 589–614 (2010)
24. Rieser, V., Lemon, O.: Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz data: Bootstrapping and Evalutaion. ACL (2008)
25. Thomson, B., Young, S.: Bayesian Update of Dialogue State: A POMDP framework for spoken dialogue systems. Computer Speech and Language 24(4), 562–588 (2010)
26. Traum, D., Larsson, S.: The Information State Approach to Dialogue Management. In: van Kuppevelt, J., Smith, R. (eds.) Current and New Directions in Discourse and Dialogue, pp. 325–353. Kluwer, Dordrecht (2003)
27. Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Finkan, J., Blythe, E.: Integrating planning and learning: The prodigy architecture. Experimental and Theoretical Artificial Intelligence 7 (1995)
28. Weber, B.G., Mateas, M.: Case-Based Reasoning for Build Order in Real-Time Strategy Games. In: The Artificial Intelligence for Interactive Digitial Entertainment (2009)
29. Young, R.M.: Notes on the use of plan structures in the creation of interactive plot. In: Mateas, M., Sengers, P. (eds.) Narrative Intelligence: Papers from the AAAI Fall Symposium (Technical Report FS-99-01). AAAI Press, Menlo Park (1999)

Successful Performance via Decision Generalisation in No Limit Texas Hold'em

Jonathan Rubin and Ian Watson

Department of Computer Science,
University of Auckland, New Zealand
jrub001@aucklanduni.ac.nz, ian@cs.auckland.ac.nz
http://www.cs.auckland.ac.nz/research/gameai

Abstract. Given a set of data, recorded by observing the decisions of an *expert* player, we present a case-based framework that allows the successful generalisation of those decisions in the game of no limit Texas Hold'em. The transition from a **limit** betting structure to a **no limit** betting structure offers challenging problems that are not faced in the limit domain. In particular, we address the problems of determining a suitable **action abstraction** and the resulting **state translation** that is required to map real-value bet amounts into a discrete set of abstract actions. We also detail the similarity metrics used in order to identify similar scenarios, without which no generalisation of playing decisions would be possible. We show that we were able to successfully generalise no limit betting decisions from recorded data via our agent, SartreNL, which achieved a 2nd place finish at the 2010 Annual Computer Poker Competition.

1 Introduction

In 2008 the Second Man-Machine Poker Competition was won by a computer poker robot named Polaris [1]. Polaris challenged a group of professional human players to the game of limit Texas Hold'em, beating its competitors by a statistically significant margin. The success of Polaris can (at least in part) be attributed to the increasing popularity of Texas Hold'em poker as a research domain and advances in game theoretic equilibrium finding algorithms [2,3]. Polaris's victory occurred in the game of **limit** Texas Hold'em, where the amount able to be wagered is restricted by pre-determined bet sizes. Today, the most popular variation of the game is **no limit** Texas Hold'em, where players' bet sizes are no longer restricted. This allows a player to wager any amount they wish (up to the total amount of chips they possess). This simple rule change has a profound effect on the nature of the game, as well as on the development of computerised agents that wish to handle the no limit betting structure. While the Polaris system was able to beat world-class human opposition in the game of limit Hold'em, nothing like Polaris's victory has been achieved in the more complicated domain of no limit Hold'em. In fact, it is only relatively recently that many researchers have shifted their attention to the more complicated domain of no limit Texas Hold'em [4,5,6].

Our previous research has focused on the techniques of expert imitation to achieve strong performance in the domain of limit Texas Hold'em [7]. In this current work, we apply the same principles of expert imitation and decision generalisation to the more complicated domain of no limit Texas Hold'em. As mentioned above, while the transition from restricted betting to a no limit betting structure ostensibly is a simple rule change, the result is a profound impact on the nature of the game, as well as on the construction of computerised agents. For automated no limit poker agents, a non-trivial translation phase is now required to map quantitative bet amounts into discrete betting categories. Furthermore, our approach requires the construction of metrics to determine similarity between complicated no limit betting sequences (without which no generalisation would be able to take place). Our approach allows the successful imitation of any designated *expert player* (artificial or human), given a large enough set of training data. We describe the discrete betting categories used by our system, the translation process that maps real values into appropriate categories and the metrics required that allow successful generalisation of an expert player's decisions based on a set of hand histories.

Despite the extra complications introduced by the no limit Hold'em domain, we show that our approach is still able to achieve successful performance. In particular, we present results from the 2010 Annual Computer Poker Competition (ACPC), where our entry to the competition, SartreNL, achieved a 2nd place finish in the no limit equilibrium run-off event [8].

In Section 2 we describe the game of no limit Texas Hold'em. Section 3 provides the necessary background for the current work. Section 4 provides an overview of our approach. Sections 5 and 6 focus on how no limit betting is discretised and on the non-trivial translation phase that maps quantitative betting values into their discretised categories. Section 7 introduces the metrics required in order to generalise an expert player's observed decisions and Section 8 lists results obtained from the 2010 ACPC followed by a discussion of these results. Finally, a conclusion is provided in Section 9.

2 No Limit Texas Hold'em

Here we briefly describe the game of Texas Hold'em, highlighting some of the common terms which are used throughout this work. We focus on 2-player no limit Hold'em, as our system has been specialised for this domain. When a game consists only of two players, it is described as a **heads-up** match.

The game of heads-up, no limit Texas Hold'em is played in 4 stages – **preflop**, **flop**, **turn** and **river**. During the preflop both players are dealt two **hole cards**, which only they can see. Before any betting takes place, two forced bets are contributed to the pot, i.e. the **small blind** and the **big blind**. The big blind is typically double that of the small blind. The possible betting actions common to all variations of poker are described as follows:

- **Fold:** When a player contributes no further chips to the pot and abandons their hand and any right to contest the chips that have been added to the pot.
- **Check/Call:** When a player commits the minimum amount of chips possible in order to stay in the hand and continue to contest the pot. A check requires a commitment of zero further chips, whereas a call requires an amount greater than zero.
- **Bet/Raise:** When a player commits greater than the minimum amount of chips necessary to stay in the hand. When the player could have checked, but decides to invest further chips in the pot, this is known as a bet. When the player could have called a bet, but decides to invest further chips in the pot, this is known as a raise.

In a **limit** game all bets are in increments of a certain amount. However, in a **no limit** game a player may bet any amount up to the total value of chips that they possess. In a standard game of heads-up, no-limit poker both players' chip stacks would fluctuate between hands, e.g. a win from a previous hand would ensure that one player had a larger chip stack to play with on the next hand. In order to reduce the variance that this structure imposes, a variation known as **Doyle's Game** is played where the starting stacks of both players are reset to a specified amount at the beginning of every hand.

Once the betting is complete, as long as no player has folded, play continues on to the next stage. Each further stage involves the drawing of **community cards** from the shuffled deck of cards as follows: **flop** – 3 community cards, **turn** – 1 community card, **river** – 1 community card.

During each stage, players combine their hole cards with the public community cards to form their best 5 card poker hand. Each stage involves a further round of betting. A **showdown** occurs after the river where the remaining players reveal their hole cards and the player with the best hand wins all the chips in the pot. If both players' hands are of equal value, the pot is split between them.

3 Background

Our previous work has focused on expert imitation via a case-based approach within the domain of limit Texas Hold'em [7]. Expert decisions recorded from a set of training data are encoded into cases. Playing decisions are then made at runtime by searching the case-base.

While we employ a similar framework for our current work, the transition to a no limit domain results in unique challenges that are not encountered in a limit poker environment. First, there is the issue of establishing a set of abstract betting actions that all real actions will be mapped into during game play. This is referred to as **action abstraction** and it allows the vast, quantitative domain of no limit Hold'em to be approximated by a much smaller *abstract* state space. Second, given an established set of abstract actions, a **translation** process is required that determines how *best* to map real actions into their appropriate abstract counterparts, as well as a **reverse translation** that maps abstract actions back into appropriate real-world betting decisions.

Both **action abstraction** and **state translation** are issues that are also required to be addressed in the construction of no limit ϵ-Nash equilibrium strategies via algorithmic game theory. A pair of strategies are said to be an ϵ-Nash equilibrium if either player cannot gain more than ϵ by deviating their strategy. An early attempt to construct a no limit Hold'em agent via game theoretic methods is described by Andersson [9]. Andersson extended the procedures used to construct ϵ-Nash equilibrium-based limit poker agents [10] to the no limit domain. Betting amounts were discretised based on the amount of chips currently in the pot. Four abstract actions were created: half the pot, the full amount in the pot, $2 \times$ the pot and all-in (all the player's remaining chips). All bet amounts were required to be mapped into one of the above four abstract actions. Andersson notes that simply assigning a bet amount into the abstract action with the closest absolute distance can result in strategies that are able to be exploited. Instead Andersson advocates a probabilistic mapping based on the inverse absolute distance between abstract actions. Due to processor and memory limitations, [9] was only able to produce a game theoretic strategy for very small starting chip stacks.

Another no limit poker agent produced via game theoretic algorithms is Tartanian [6]. Tartanian was able to build models for much larger starting chip stacks than the work presented by [9]. Gilpin et. al. [6] advocate a translation process that uses a relative distance to perform the mapping, rather than an absolute distance. Using relative distance, a bet amount that falls between two abstract actions is mapped into the appropriate action by considering their corresponding ratios.

State translation in extensive form games has been formalised by Schnizlein et. al. [4]. Schnizlein et. al. define the concepts of **hard translation** and **soft translation**. Hard translation refers to the deterministic mapping of bet amounts into their appropriate categories. Soft translation refers to the probabilistic mapping of these values. Schnizlein et. al. investigate both hard and soft translation in the domains of Texas Hold'em poker and Leduc Hold'em — a much smaller, specialised poker domain useful for experimental analysis. In both domains [4] show that the use of hard translation produces strategies that are more easily exploitable than soft translation.

While our current work is required to deal with many of the same issues and challenges faced by ϵ-Nash Equilibrium strategies, the focus of our approach is more to do with expert imitation and investigating the generalisation of playing decisions from game traces.

We are now in a position to present the main components that make up our system. The resulting no limit poker agent is referred to as SartreNL. We begin with an overview of the representation used to record game scenarios by processing a collection of training data. The exact *action abstraction* used by SartreNL and the details of how and where *state translation* occurs are described, along with the metrics that allow decision generalisation to take place.

4 Overview

Given a set of (artificial or real-world) training data, SartreNL is able to generalise the decisions recorded within the data by constructing and storing a collection of cases. Each case attempts to capture important game state information that is likely to have an impact on the final betting decision. Table 1 depicts a collection of attribute-value pairs that, when taken together, captures a particular game scenario. SartreNL uses four attribute-value pairs to describe the current state of a match. Three of the four attributes (*hand strength*, *betting sequence*, *board texture*) are the same as those used by the limit variation of Sartre [7]. The *stack commitment* attribute was introduced especially for the no limit variation of the game. All attributes were selected by the authors, given their importance in determining a final betting decision.

Table 1. The case representation used by SartreNL. The four features capture important game state information. A solution is made up of an action and outcome tuple.

Feature	**Type**	**Example**
1. Hand Strength Bucket	Integer	1 – 50
2. Betting Sequence	String	*pdc-cqc-c*, *cc-*, *dc-qc-ci*, ...
3. Stack Commitment	Integer	1,2,3,4
4. Board Texture	Class	*No-Salient*, *Flush-Possible*, *Straight-Possible*, *Flush-Highly-Possible*, ...
Action	n-tuple	(0.0, 1.0, 0.0, 0.0, ...), ...
Outcome	n-tuple	($-\infty$, 36.0, $-\infty$, $-\infty$, ...), ...

Each case also records a solution. A solution is made up of two n-tuples, one which specifies action probabilities and another which specifies the average outcome of taking the observed action in the past. The entries within each tuple correspond to a particular betting decision. The entries within the action tuple must sum to one.

During game play values are assigned to each attribute and the previously stored collection of cases are searched for attributes with similar values. The case with the highest global similarity is assumed to be most similar to the current situation. Once a similar case has been retrieved a betting decision is made by re-using the tuples within that case's solution.

Each attribute-value pair is described in more detail below:

4.1 Hand Strength Bucket

To evaluate the strength of a player's hand the *expected hand strength squared* metric is used $E[HS^2]$. The $E[HS^2]$ metric computes the probability of winning at showdown against a random hand. This is given by *rolling out* all possible combinations of community cards and determining the proportion of the time the player's hand wins against the set of all possible opponent holdings. The hand strength value for each community card *roll-out* is then squared and the final $E[HS^2]$ is given by averaging these values.

Given the large variety of values that can be produced by the $E[HS^2]$ metric, bucketing takes place where similar values are mapped into a discrete set of buckets that contain hands of similar strength. SartreNL uses a total of 50 buckets for each post-flop betting round.

4.2 Betting Sequence

The betting sequence attribute is given by a string of characters where each character is either an observed game action or round delimiter. Round delimiters are represented by hyphens and indicate the transition to the next round of betting. As any quantitative betting value can be observed in the real game, a discrete set of abstract actions are chosen to represent real betting actions. This is known as *action abstraction* and is described in more detail in Section 5. The action abstraction used by SartreNL is given in Table 2.

Betting sequences consist of every abstract action that was observed up until the current decision point in the game. This includes a player's own previous actions and actions for all previous rounds of play. By including actions that belong to previous rounds, similarity assessment is made more difficult, but this allows a more informed context about the game environment at the time a playing decision was made.

4.3 Stack Commitment

In the no limit variation of Texas Hold'em players can wager any amount up to their total stack size. The proportion of chips committed by a player, compared to the player's stack size, is therefore of much greater importance, compared to limit Hold'em. The betting sequence maps bet amounts into discrete categories based on their proportion of the pot size. This results in information that is lost about the total amount of chips a player has contributed to the pot, relative to the size of their starting stack. Once a player has contributed a large proportion of their stack to a pot, it becomes more important for that player to remain in the hand, rather than fold, i.e. they have become **pot committed**.

The **stack commitment** feature maps this value into one of N categories, where N is a specified granularity:

$$[0 - \frac{1}{N}], [\frac{1}{N} - \frac{2}{N}], \ldots, [\frac{N-2}{N} - \frac{N-1}{N}][\frac{N-1}{N} - 1]$$

Hence, for a granularity of $N = 4$, a stack commitment of 1 means the player has committed less than 25% of their initial stack, a stack commitment of 2 means that player has contributed somewhere between 25% up to 50% of their total stack, and so forth.

4.4 Board Texture

The **board texture** attribute highlights important information about the public community cards that all players share to make their best five card hand. For instance, on the last round of betting if four of the five community cards were all the same suit, the chances that a player's opponent has a flush is high as they only require one card of that suit within their personal hole cards. The board texture attribute maps a collection of community cards to one of nine categories. The categories were selected by the authors and are believed to distinguish between the salient aspects of the public community cards. The board texture categories are listed in Table 4.

5 Action Abstraction

Recall that *abstraction* is a concept used by game theoretic poker agents that derive ϵ-Nash equilibrium strategies for the game of Texas Hold'em. As the actual Hold'em game tree is much too large to represent and solve explicitly, it becomes necessary to impose certain abstractions that help restrict the size of the original game. For Texas Hold'em, there are two main types of abstraction:

1. **Chance abstraction** – which reduces the number of chance events that are required to be dealt with. This is typically achieved by grouping strategically similar hands into a restricted set of buckets.
2. **Action abstraction** – which restricts the number of actions a player is allowed to perform.

Action abstractions can typically be avoided by poker agents that specialise in limit poker, where there are only 3 actions to choose from: fold (f), check/call (c) or bet/raise (r). However in no limit, where a raise can take on any value, some sort of action abstraction is required. This is achieved by restricting the available bet/raise options to a discrete set of categories based on fractions of the current pot size. For example, a typical abstraction such as: *fcpa*, restricts the allowed actions to: f – fold, c – call, p – bet the size of the pot a – all-in (i.e. the player bets all their remaining chips). Given this abstraction, all actions are interpreted by assigning the actual actions into one of their abstract counterparts.

While SartreNL does not attempt to derive an ϵ-Nash equilibrium solution for no limit Hold'em, it is still required to define an action abstraction in order to restrict the number of actions allowed in the game and hence reduce the state space. SartreNL uses the following action abstraction: *fcqhipdvta*. Table 2 provides an explanation of the symbols used.

Table 2. The action abstraction used by SartreNL

f	fold
c	call
q	quarter pot
h	half pot
i	three quarter pot
p	pot
d	double pot
v	five times pot
t	ten times pot
a	all in

6 Translation

Given that all bets need to be mapped into one of the actions listed in Table 2, a translation process is required to define the appropriate mapping. For SartreNL, this translation phase needs to occur in 3 places:

1. During case base construction – where hand history logs from previously played hands are encoded into cases.
2. During actual game play - where betting actions observed during a hand are required to be mapped into appropriate abstract actions. Equivalent to the translation process required of ϵ-Nash equilibrium agents that solve an abstract extensive form game [9,6,4]
3. A final *reverse translation* phase is required to map a chosen abstract action into a real value to be used during game play.

Recall, Schnizlein et. al. [4] formalise two types of translation: hard translation and soft translation.

– **Hard Translation:** is a many to one mapping that maps an unabstracted betting value into an abstract action based on a chosen distance metric. Given a unique unabstracted betting value, hard translation will always map this value into the same abstract action. A disadvantage of hard translation is that an opponent can exploit this mapping simply by selecting particular betting values.
– **Soft Translation:** is a probabilistic state translation that uses normalised weights as similarity measures to map an unabstracted betting value into an abstract action. The use of a probabilistic mapping ensures that soft translation cannot be exploited like hard translation can.

SartreNL uses both hard and soft translation. The type of translation that takes place differs depending on where translation occurs within the system. The exact details of the translation used within the different areas of the system are now presented.

Define $A = \{q, h, i, p, d, v, t, a\}$ to be the set of abstract betting actions. Note that the actions f and c are omitted as these require no mapping.

1. During case-base construction SartreNL uses the hard translation function $T_h : \Re \rightarrow A$, as follows:

$$T_h(b) = \begin{cases} a & \text{if } \frac{a}{b} > \frac{b}{c} \\ c & \text{otherwise} \end{cases} \tag{1}$$

   where $b \in \Re$ is the proportion of the total pot that has been bet in the actual game and $a, c \in A$ are abstract actions that map to actual pot proportions in the real game and $a <= b < c$. The fact that hard translation has the capability to be exploited is not a concern during case-base construction. Hard translation is used during this stage to ensure that re-training the system with the same hand history data will result in the same case-base.
2. During actual game play SartreNL is required to map opponent betting actions (as well as its own actions) to abstract categories. Observant opponents have the capability to exploit deterministic mappings during game play, hence SartreNL uses a soft translation function for this stage, $T_s : \Re \rightarrow A$, given by the following probabilistic equations:

$$P(a) = \frac{\frac{a}{b} - \frac{a}{c}}{1 - \frac{a}{c}} \tag{2}$$

$$P(c) = \frac{\frac{b}{c} - \frac{a}{c}}{1 - \frac{a}{c}} \tag{3}$$

   where once again, $b \in \Re$ is the proportion of the total pot that has been bet in the actual game and $a, c \in A$ are abstract actions that map to actual pot proportions in the real game and $a <= b < c$. Note that when $b = a, P(a) = 1$ and $P(c) = 0$ and when $b = c, P(a) = 0$ and $P(c) = 1$. Hence, a betting action that maps directly to an abstract action in A does not need to be probabilistically selected. On the other hand, when $b \neq a$ and $b \neq c$, abstract actions are chosen probabilistically.
3. The final place that translation is required is when SartreNL has determined an appropriate abstract action to play. A reverse mapping is then required to map the abstract action into an appropriate real betting value, given the current game conditions. SartreNL uses the following function to perform reverse translation, $T_r : A \rightarrow \Re$:

$$T_r(a) = a' \pm \Delta a' \tag{4}$$

   where $a \in A$ and $a' \in \Re$ is the real value corresponding to abstract action a and $\Delta a'$ is some random proportion of the bet amount that is used to ensure SartreNL does not always map abstract actions to their exact real world counterparts. For example, when $a' = 100$ and $\Delta = 0.3$, SartreNL could bet any amount between 70 - 130 chips.

7 Similarity

In order to generalise no limit betting decisions, it is first required to locate similar scenarios for which solutions have been recorded in the past. Given a target case, t, that describes the immediate game environment, a source case, $s \in S$, where S is the entire collection of previously recorded cases and a set of features, F, global similarity is computed by summing each feature's local similarity contribution, sim_f, and dividing by the total number of features:

$$G(t, s) = \sum_{f \in F} \frac{sim_f(t_f, s_f)}{|F|} \tag{5}$$

We now present the local similarity metrics (sim_f) required in order to generalise betting decisions from a collection of data.

7.1 Hand Strength Bucket

The following metric is used to determine similarity between two hand strength buckets (f_1, f_2).

$$sim(f_1, f_2) = max\{1 - k \cdot \frac{|f_1 - f_2|}{T}, 0\} \tag{6}$$

Here, T refers to the total number of buckets that have been defined, where $f_1, f_2 \in [1, T]$ and k is a scalar parameter used to adjust the rate at which similarity should decrease. SartreNL uses values of $T = 50$ and $k = 2$.

7.2 Stack Commitment

The stack commitment metric uses an exponentially decreasing function.

$$sim(f_1, f_2) = e^{(-|f_1 - f_2|)} \tag{7}$$

where, $f_1, f_2 \in [1, N]$ and N refers to the granularity used for the stack commitment attribute. This function was chosen as small differences between two stack commitment attributes (f_1, f_2) should result in large drops in similarity. SartreNL uses a granularity of $N = 4$.

7.3 Betting Sequence

SartreNL uses the following bet discretisation: *fcqhipdvta*. Within this representation there are some non-identical bet sizes that are reasonably similar to each other. For example, a bet of half the pot (h) is quite close to a bet of three quarters of the pot (i). The betting sequence similarity metric we derived compares bet sizes against each other that occur at the same location within two betting sequences.

Let S_1 and S_2 be two betting sequences made up of actions $a \in A \cup \{f, c\}$, where the notation $S_{1,i}, S_{2,i}$ refers to the i^{th} character in the betting sequences S_1 and S_2, respectively.

For two betting sequences to be considered similar they first need to satisfy the following conditions:

1. $|S_1| = |S_2|$
2. Both $S_{1,i} = c$ and $S_{1,j} = a$ whenever $S_{2,i} = c$ and $S_{2,j} = a$

i.e. each sequence contains the same number of elements and any calls (c) or all-in bets (a) that occur within sequence S_1 must also occur at the same location in sequence S_2[1].

Any two betting sequences that do not satisfy the initial two conditions above are assigned a similarity value of 0. On the other hand, if the two betting sequences do satisfy the above conditions their bet sizes can then be compared against each other and a similarity value assigned.

Exactly how dissimilar two individual bets are to each other can be quantified by how far away from each other they occur within the bet discretisation string, displayed in Table 3.

Table 3. Bet Discretisation String

q	h	i	p	d	v	t

As h and i are neighbours in the discretisation string they can be considered to occur at a distance of 1 away from each other, $\delta(h, i) = 1$, as opposed to say $\delta(q, t) = 6$, which are at opposite ends on the discretisation string.

For two betting sequences S_1, S_2 overall similarity is determined by (8):

$$sim(S_1, S_2) = \begin{cases} 1 - \sum_{i=0}^{|S_1|} \delta(S_{1,i}, S_{2,i})\alpha & \text{if } |S_1| = |S_2|, \\ & S_{1,i} = c \Rightarrow S_{2,i} = c, \\ & S_{1,j} = a \Rightarrow S_{2,j} = a \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where α is some constant rate of decay. SartreNL uses a rate of decay where $\alpha = 0.05$.

Betting Sequence Similarity Example. Here we offer a concrete example of how similarity is computed for two non-identical betting sequences.

Consider two betting sequences, $S_1 = ihpc$ and $S_2 = dqpc$.

Here, $|S_1| = 4$ and $|S_2| = 4$ and wherever there exists a check/call (c) in S_1, there exists a corresponding c in S_2.

[1] A betting sequence consists of one or more betting rounds, the above conditions must be satisfied for all betting rounds within the betting sequence.

As both conditions are satisfied we can evaluate the top half of Equation (8):

$$\begin{aligned} sim(S_1, S_2) &= 1 - [\delta(i,d)\alpha + \delta(h,q)\alpha + \delta(p,p)\alpha + \delta(c,c)\alpha] \\ &= 1 - [2 \cdot \alpha + 1 \cdot \alpha + 0 \cdot \alpha + 0 \cdot \alpha] \\ &= 1 - 3\alpha \end{aligned}$$

Using a rate of decay of $\alpha = 0.05$, gives a final similarity of: $1 - 0.15 = 0.85$.

7.4 Board Texture

To determine similarity between board texture categories a similarity matrix was derived. Rows and columns in Figure 1 represent the different categories defined in Table 4. Diagonal entries refer to two sets of community cards that map to the same category, in which case similarity is always 1. Non-diagonal entries refer to similarity values between two dissimilar categories. These values were hand picked by the authors. The matrix given in Figure 1 is symmetric.

$$\begin{array}{c|ccccccccc} & A & B & C & D & E & F & G & H & I \\ \hline A & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B & 0 & 1 & 0.8 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0.8 & 1 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ D & 0 & 0.7 & 0.7 & 1 & 0 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 1 & 0.8 & 0.7 & 0 & 0.6 \\ F & 0 & 0 & 0 & 0 & 0.8 & 1 & 0.7 & 0 & 0.5 \\ G & 0 & 0 & 0 & 0 & 0.7 & 0.7 & 1 & 0.8 & 0.8 \\ H & 0 & 0 & 0 & 0 & 0 & 0 & 0.8 & 1 & 0.8 \\ I & 0 & 0 & 0 & 0 & 0.6 & 0.5 & 0.8 & 0.8 & 1 \end{array}$$

Fig. 1. Board texture similarity matrix

Table 4. Board Texture Key

A	No salient
B	Flush possible
C	Straight possible
D	Flush possible, straight possible
E	Straight highly possible
F	Flush possible, straight highly possible
G	Flush highly possible
H	Flush highly possible, straight possible
I	Flush highly possible, straight highly possible

8 Results

A version of the system described above was submitted to the 2010 Annual Computer Poker Competition [8]. Our entry to the competition was trained on data from the best no limit agent of the 2009 competition. The ACPC is the premier computer poker event and has been held each year at either AAAI or IJCAI conferences since 2006. The ACPC involves separate competitions for different varieties of Texas Holdem, such as limit and no-limit competitions, as well as heads-up and multiple-opponent competitions. Entrance into the competition is open to anyone and the agents submitted typically represent the current state of the art in computer poker.

Table 5 presents a cross-table of results between competitors at the 2010 heads-up no limit competition. A green cell indicates a win for the row player and a red cell indicates a loss for that player. Cells with a lighter background represent matches that were not statistically significant. The figures presented in Table 5 are in milli big blinds per hand (mb/h), a milli big blind is 0.001 times the big blind value. Therefore, mb/h are calculated by dividing the total number of big blinds won by the number of hands played, followed by multiplying the result by 1000. A result of +1000 mb/h means that a player won, on average, one big blind per hand.

Table 5. Crosstable of all matches. Results are from the perspective of the row player. Values are in milli big blinds per hand. Confidence intervals are omitted due to space considerations. This table is replicated from the 2010 ACPC [8].

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) c4tw.iro	–	–	−4181	–	−7557	−2289	−5534	–
(2) c4tw.tbr	–	–	–	−3562	−6977	−2241	–	−8669
(3) Hyperborean.iro	4181	–	–	−83	775	200	248	122
(4) Hyperborean.tbr	–	3562	83	–	795	272	364	220
(5) PokerBotSLO	7557	6977	−775	−795	–	−193	−108	−159
(6) SartreNL	2289	2241	−200	−272	193	–	42	−13
(7) Tartanian4.iro	5534	–	−248	−364	108	−42	–	−80
(8) Tartanian4.tbr	–	8669	−122	−220	159	13	80	–

Table 6. Bankroll instant run-off results. This table is replicated from the 2010 ACPC [8].

	Round 0	Round 1	Round 2	Round 3
(1st) Hyperborean.iro	1351 ± 44	408 ± 27	224 ± 31	200 ± 39
(2nd) SartreNL	581 ± 34	12 ± 23	-79 ± 27	-200 ± 39
(3rd) Tartanian4.iro	1338 ± 33	-60 ± 27	-145 ± 34	–
(4th) PokerBotSLO	1620 ± 187	-359 ± 32	–	–
(5th) c4tw.iro	-4891 ± 213	–	–	–

In the no limit competition, the **Doyle's Game** rule variation was used where both player's begin each hand with 200 big blinds. All matches played were duplicate matches. In a heads-up duplicate match N hands are played between two agents after which the agents memories are wiped and the N hands played again, but in the reverse direction, i.e. the cards that were initially given to player A are instead given to player B and vice-versa. This way both players get to play both sets of N cards and this reduces the variance that is involved in simply playing a set of N hands in one direction only. Many duplicate matches are played in order to achieve a significant result. In the 2010 heads-up, no limit competition each competitor played 200 duplicate matches (each consisting of $N = 3000$ hands) against every other competitor.

Table 6 presents the final results of the instant run-off competition. The instant run-off competition uses a recursive winner determination algorithm that repeatedly removes the agents that performed the worst against a current pool of players. In the 2010 competition SartreNL was ranked in second place.

Table 5 indicates that SartreNL suffers a statistically significant loss against only one competitor i.e. Hyperborean. Where a competitor's name ends with .iro or .tbr this indicates the competitor was specifically submitted to a particular division (i.e. instant run-off or total bankroll). In general, SartreNL does not achieve as large a bankroll as some of the other competitors and this is mostly due to its performance against the competitor c4tw. It is clear that c4tw is the weakest competitor, having lost all of its matches, however SartreNL does not achieve as great a profit as some of the other competitors do against this opponent. Moreover, the amounts won against this opponent (the first two columns in Table 5) are by far larger than any other values in the table. This means that matches involving c4tw have a much greater impact on the final total bankrolls. This results in SartreNL achieving only a slight overall profit i.e. 581 ± 34. Table 6 shows that when c4tw is removed from the pool of players via the instant run-off winner determination procedure, SartreNL actually performs a lot better overall.

9 Conclusion

Given a set of data, recorded by observing an *expert* player, the framework presented in this paper allows the successful generalisation of those decisions. Our results support the idea that generalising decisions via expert imitation has the ability to produce strong, sophisticated strategies in complex, imperfect information environments. Moreover, our results show that these strategies can lead to successful performance compared with alternative approaches. In previous research we have shown this to be the case in the domain of limit Hold'em [7]. This work extrapolates our approach to the domain of no limit Hold'em. The transition to a more complicated no limit betting structure required issues to be addressed that were not a concern within the limit domain. In particular, a suitable action abstraction was required in order to reduce the large no limit state space. Given a chosen abstraction, state translation needs to take place at various locations within the system. The formulas required for both hard and soft translation were explained, as were the local similarity metrics that allow the identification of similar scenarios so that decision generalisation can take place. The SartreNL system produced by the framework achieved a second place finish at the 2010 ACPC no limit, instant run-off competition.

References

1. Bowling, M., Risk, N.A., Bard, N., Billings, D., Burch, N., Davidson, J., Hawkin, J., Holte, R., Johanson, M., Kan, M., Paradis, B., Schaeffer, J., Schnizlein, D., Szafron, D., Waugh, K., Zinkevich, M.: A demonstration of the Polaris poker system. In: AAMAS 2009: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, Richland, SC, pp. 1391–1392. International Foundation for Autonomous Agents and Multiagent Systems (2009)
2. Sandholm, T.: The State of Solving Large Incomplete-Information Games, and Application to Poker. AI Magazine (Winter 2010)
3. Rubin, J., Watson, I.: Computer poker: A review. Artificial Intelligence 145(5-6), 958–987 (2011)
4. Schnizlein, D., Bowling, M.H., Szafron, D.: Probabilistic State Translation in Extensive Games with Large Action Sets. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009, pp. 278–284 (2009)
5. Van den Broeck, G., Driessens, K., Ramon, J.: Monte-Carlo Tree Search in Poker Using Expected Reward Distributions. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS, vol. 5828, pp. 367–381. Springer, Heidelberg (2009)
6. Gilpin, A., Sandholm, T., Sørensen, T.B.: A heads-up no-limit Texas Hold'em poker player: discretized betting models and automatically generated equilibrium-finding programs. In: 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), pp. 911–918 (2008)
7. Rubin, J., Watson, I.: Similarity-Based Retrieval and Solution Re-use Policies in the Game of Texas Hold'em. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 465–479. Springer, Heidelberg (2010)
8. ACPC. The Annual Computer Poker Competition (2011), http://www.computerpokercompetition.org/
9. Andersson, R.: Pseudo-Optimal Strategies in No-Limit Poker. Master's thesis, Umea University (2006)
10. Billings, D., Burch, N., Davidson, A., Holte, R.C., Schaeffer, J., Schauenberg, T., Szafron, D.: Approximating Game-Theoretic Optimal Strategies for Full-scale Poker. In: IJCAI 2003, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 661–668 (2003)

Rule-Based Impact Propagation for Trace Replay

Raafat Zarka[1,2], Amélie Cordier[1,3],
Elöd Egyed-Zsigmond[1,2], and Alain Mille[1,3]

[1] Université de Lyon, CNRS
[2] INSA-Lyon, LIRIS, UMR5205,
F-69621, France
[3] Université Lyon 1, LIRIS, UMR5205,
F-69622, France
`firstname.lastname@liris.cnrs.fr`

Abstract. To help end-users master complex applications, it is often efficient to enable them to "replay" what they have done before. In some situations, it is even more useful to enable them to modify some values of the actions they are replaying so that they can see the consequences of the modification. Unfortunately, it is not always possible to replay series of actions after a modification of a prerequisite. Hence, the replay process has to deal with impact propagation of changes. In this paper, we describe our models to enable replay of user's interactions and to manage impact propagation of changes during the replay process using impact rules to perform the adaptation. These models are built upon traces, i.e. digital objects that enable us to record user interactions and to reuse them in different ways. We have implemented the replay process in a Web application called SAP-BO Explorer, an application assisting business users in managing large amounts of information. Our tool helps users to better understand the application.

Keywords: impact propagation, macro recording, bookmarks, trace-based reasoning, replay traces, human computer interaction.

1 Introduction

With the multiplication and the rapid development of software systems and applications, we now have access to more and more tools, which are usually more and more rich, and difficult to master. While using these tools, we are often lost, usually because we lack time to understand applications, to get used to them and to exploit them efficiently. In response to this problem, some application designers came up with solutions for helping users either to discover the application or to be more efficient while using it. Providing a relevant assistance to users becomes a real challenge for application designers. Among the proposals for assistance strategies, we usually find tutorials, how-to, videos, assistants, training courses, etc. However, all these assistance strategies rest upon a static description of the application, hard-coded *a priori*. The same assistance is provided to all the users. Assistance is not always well suited to particular needs of specific users.

To overcome this issue, we have proposed in a previous work to use Trace-Based Reasoning in order to provide a contextualized help to end users [1]. Trace-Based Reasoning draws its inspiration from Case-Based Reasoning and reuses past experiences to solve new problems. The main difference is that, in Trace-Based Reasoning, the main knowledge container is a base of interaction traces.

Interaction traces are relatively new digital objects. An interaction trace is a rich record of the actions performed by a user on a system. In other words, a trace is a story of the user's actions, step by step. Hence, traces enable us to capture users' experiences. Traces are recorded according to a pre-established model, so that they can be reused in different ways: replay, exploration, modification, modification plus replay, etc. Working with traces raises numerous research issues. How to collect, represent, store, and visualize traces? How to implement a replay mechanism in a pre-existing system? How to take into account privacy issues for traces?

Recent research enables us to work within an existing framework for manipulating traces (see [2], [3] and [4]). In this paper, we focus on a specific research question: how to replay a trace in a system and which issues are raised by the replay when the initial situation has been modified? To better understand this problem, let us consider the following example. A user makes a sequence of manipulations to improve a colored picture: transformation in gray-scale, selection of a scale of gray, luminosity attenuation for the selection, blur effect on the selection. Not satisfied with the result, he decides to go back to the initial state (the original picture) and to replay the whole set of actions, except from the transformation in gray-scale. The question is: "are the remaining actions still possible?"

The issue we address in this paper is then: how to enable a trace replay while monitoring the impact of a modification in the trace on the remaining of the process? In order to address this issue, we have firstly elaborated a mechanism enabling to do a simple replay of a trace (i.e. with no modification) from any point in the trace. Then, we have defined a model to manage impact propagation after a modification of the trace. The trace-replay mechanism has been implemented in the SAP-BO Explorer application [5], a web application enabling users to load, explore, visualize and export large quantities of data. SAP-BO needs a tool to help their users better understand their application so we designed this solution for them. We have instrumented the initial application in order to collect interaction traces and we have developed a graphical interface that displays the traces according to an *ad-hoc* knowledge representation. Using this interface, any user can replay his recorded traces. The application is operational and a demo video is available[1].

This paper is organized as follows. In section 2, we survey related work. Then, in section 3, we show how we use traces in order to enable replay of user's interactions. In section 4, we discuss the consequences of a change during the replay, and we propose a rule-based impact propagation model. Section 5 gives implementation details. Open issues and discussion of our proposal are made in section 6. The paper ends with a conclusion and a description of future research issues.

[1] A demo video of trace replay and visualization is available at: `https://liris.cnrs.fr/~rzarka/ReplayTraceDemo/`

2 Related Work

Macro recorders are systems that record and play back mouse events and keystrokes. In most of existing macro recording systems, users have to be proactive: they need to start and stop macro recording. Bookmark systems are one of the most common macro recording systems. They enable users to “replay” web pages. With Koala [6], the user can record a sequence of actions and generate a script of keyword commands that can be replayed later. Recorded scripts are stored automatically on a wiki. CoScripter [7] is a Firefox plug-in created by an IBM Research group. It allows users to record and share interactions with websites. It records user actions and saves them in semi-natural language scripts. The recorded scripts are saved in a central wiki for sharing with other users. WebVCR [8] and WebMacros [9] record web browser actions as a low-level internal representation, which is not editable by the user or displayed in the interface.

All these systems require planning to enable recording while Smart Bookmarks [10] supports retroactive recording: it automatically captures users’ interactions while they navigate the web, and displays them through a graphical presentation. When users want to bookmark a webpage, the system automatically determines the sequence of commands needed to return to the page, and saves the sequence as a bookmark. While Smart Bookmarks lets users save or share actions from ongoing browsing sessions, ActionShot [11] enables users to share actions they have performed before by providing them with a visual interface for browsing their entire history. ActionShot is built on top of CoScripter. History data is reused through the re-execution of recorded steps. Sharing also is supported through Facebook, Twitter or *via* email. Both ActionShot and Smart Bookmarks are generic, but they are implemented as Firefox extensions which is a limitation because it cannot work with other browsers. Besides, they cannot work with dynamic pages (e.g. Ajax or Flash based).

In Smart Bookmarks, users can modify parameters values before the bookmark starts running. However, these new values may affect commands and cause inconsistent states in the application. Hence, it seems relevant to study impact propagation of these changes. Impact propagation analysis is widely studied in software engineering and database domains. In [12], the authors propose a UML model-based approach to impact analysis that can be applied before any implementation of the changes, thus allowing an early decision-making and change planning process. Most techniques to predict the effects of schema changes upon applications that use the database can be expensive and error-prone, making the change process expensive and difficult. In [13], the authors present a novel method for extracting potential database queries from a program, called query analysis. The impacts of a schema change can be predicted by analyzing the results of a query analysis, using a process they call impact calculation. Many systems also support impact analysis. One of them is Sybase Power Designer Modeling Tool that provides powerful methods for analyzing the dependencies between object models [14].

Some applications allow users to replay their actions like Photoshop [15], by using undo or go back commands. In Photoshop, graphics designers and photographers have a number of processes they frequently perform on their images. By creating macros called "actions" they can automate many routine tasks using simple text files that are recorded in a macro-style. Whether is the goal is to convert an image for the Web or to transform a color photo into a black and white photo, designers can reduce several steps to a click on a single button. Users can create their own macro scripts which are mini recordings of commands. This is also what we would like to provide, but in our case we need to apply macro recording for systems that do not support undo commands like most client-server applications. In addition, we do not want to ask the user to start or stop recording his actions. Table 1 shows a comparison between all these systems according to the way they allow visualization of past actions and if they support the replay with or without change of values.

Table 1. Comparison table of related work

System	Representation	Simple Replay	Replay with change	Adaptation
WebMacros WebVCR	No	Proactive	No	No
Koala	Wiki Scripts	Proactive	No	No
CoScripter	Text, Firefox Extension	Proactive	No	No
Smart bookmarks	screenshots, Firefox extension	Retroactive	Yes, without impact propagation	Classify buttons for side-effecting
ActionShot	Graphical text, Firefox extension	Retroactive	Yes, without impact propagation	No
Photoshop	Actions list	Macro and undo command	Yes	Yes
Power Designer	Does not trace	Undo command	No	Impact rules
Trace Replay (Our approach)	*M-Trace* with text explanations	Retroactive	Yes	Impact rules and adapted values

Trace replay is a step towards the goal of finding different personalized tasks to make applications more "task-aware". Some projects also tackle this problem. For example, TaskTracer [16] collects user's interactions to organize the user's information naturally according to tasks. They use machine learning techniques to learn and adapt solutions according to user specificities. CaBMA [17] (for: Case-Based Project Management Assistant) is another work providing assistance with project planning tasks. The system uses case-based reasoning by adding a knowledge layer on top of the traditional project management software, going beyond the editing and bookkeeping capabilities that this software is traditionally limited to. CaBMA automatically captures cases from previous project plans, and reuses them for planning. But in our future work we want to determine different personalized task and make use of case-based reasoning for reusing the experiences and replay tasks for any application, and not only in the project management applications.

3 Simple Trace Replay (Go Back to a Previous State)

To enable users to go back to a previous state, we propose to implement a “trace replay mechanism”. This mechanism enables users to replay their interaction until they reach the expected state of the application. In order to implement this mechanism, we have defined a trace model (see Fig. 1).

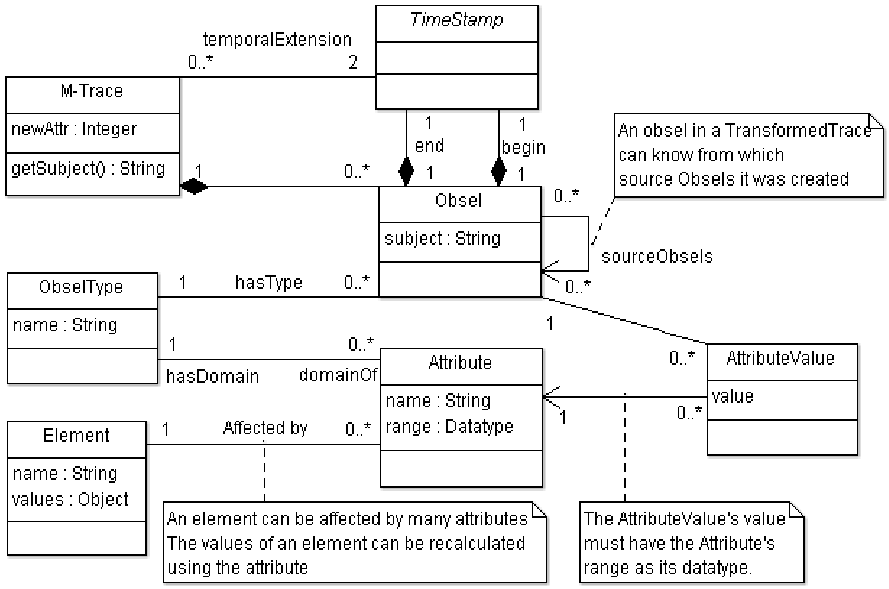


Fig. 1. Modified trace model to support trace replay

Each user’s session is represented by a *M-Trace* which consists of a set of observed elements (obsels). Each obsel has a type and two timestamps representing its beginning and ending instants. Each obsel type has a domain of attributes and indicates the values of its attributes respecting the range of the attribute type. An obsel can affect many elements at the same time. For example, pressing a “delete all” button can erase the values of many elements together. By using the obsel attribute values, we can calculate the new values for the related elements, where each obsel attribute concerns only one element. Using this model we can get all the obsels that can modify every element and all the elements that can be affected by an obsel. When capturing the traces we don’t need to store the values of elements at each time. We only store attributes and values of each obsel. For example if a user selects a chart, the value of the obsel will be the ID of the chart and not the whole information about the chart. We can find the complete information in the “selected chart” element.

3.1 Go Back Trace Process

Our solution to go back to a previous state of the system is to replay traces from a starting point (session start) and not by undoing last ones. When a user chooses to go back to a past state, he can choose the obsel that he wants to return to. The system will automatically go back to this state by replaying all the obsels that happened from the beginning of the session until the selected obsel; let's call it the triggered obsel (the obsel where we want the system to play back to). Fig. 2 [A] shows a simple trace replay, a list of obsels starting from *A* to *R*, where *R* is the replay obsel and *C* is the triggered obsel. In *R* the user asked to replay traces to back with the system to its state when clicking on *C*. We can see that all the obsels that happened between *C* and *R* will be ignored (*EDA*). This replay will be done by one command which means one call from the client to the server. After replaying traces the system will go back to the past state and the user will continue his usage to the system, and new obsels will be collected. An Obsel R means that at this point a replay action happened.

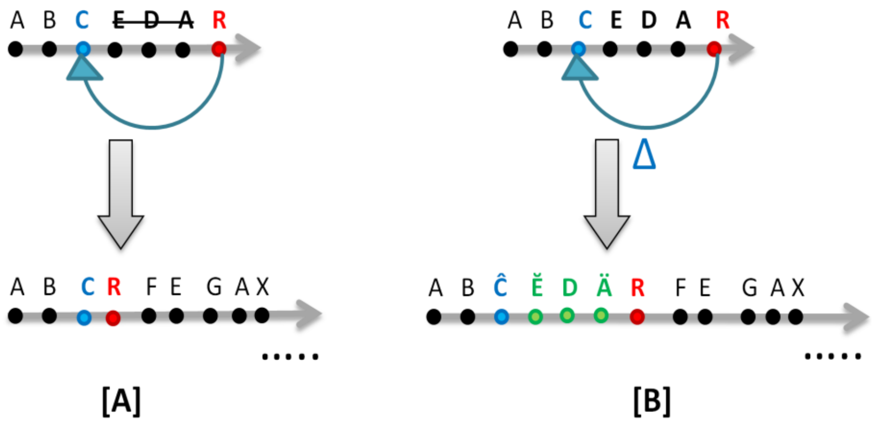


Fig. 2. [A] Simple Trace Replay, [B] Trace Replay with change

Algorithm 1. Simple replay gets *M-Trace* and the triggered obsel as input and goes back to a previous state. Firstly, it gets the subset of the trace that should be replayed starting from the first obsel to the triggered one by a chronological order. Then this trace will be optimized by using the optimization algorithm to delete extra obsels. Each element gets its default values and then a loop on all the obsels runs, where at each time the element values are updated according to the attributes of the current obsel. At the end, the new element values are updated making the system going back to this state. The replay event is also captured as a new obsel and taken in consideration during the analysis.

Algorithm 1. Simple replay

```
Program simpleReplay (M-Trace, TriggerredObsel)
  ReplayedTrace := getSubTrace(0, pos(TriggeredObsel));
  optimize(ReplayedTrace);
  Elements := getDefaultValues();
  For pos := 0 to getObselCount(ReplayedTrace)-1
    Obsel := ReplayedTrace[pos];
    Attributes := getAttributes(Obsel.Type);
    For each attribute in Attributes
      Value := getAttributeValue(Obsel, attribute);
      Elem := getAffectedElement(attribute);
      Elements[elem] := GenerateElementValue(value);
    End For each
  End For
  Return Elements;
End Program
```

3.2 Optimized Trace Replay Process

As not all the obsels play a role for changing the state of the system, replay process can be optimized by reducing the number of replayed obsels. In addition, in some cases many obsels can be ignored, either because they have been canceled by other obsels or because of reset values. We can get an optimized chronological list of obsels from the beginning of the session to the triggered obsel; this list will be used to generate the values for each element. Optimization algorithm tries to delete all unnecessary obsels that induce loops in the trace. For example, in the *simple replay* obsel, the subTrace from *replay obsel* to *triggered obsel* should be deleted. The same thing is also done for a *reset obsel* which means deleting all the obsels from the beginning to the *reset obsel*. So we consider that there is a list of unnecessary loop obsels in the trace that should be deleted as shown in Fig. 3.

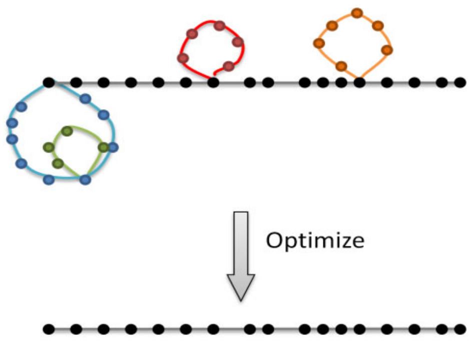


Fig. 3. Trace Replay Optimization

4 Replay Traces with Impact Propagation

In this section we describe how we can replay traces after modifying an obsel by handling the consequences of changes on elements. This is illustrated on Fig. 2 [B]. *R* is a *replay obsel* that triggers a replay of the trace after doing a change on the values of the *triggered obsel C*. Because of a change in one of the attribute values of *C*, the values of some other obsels could be inconsistently modified, like *E* and *A*, while other obsels may remain consistent, like *D*. We need to find the adapted values replay traces with these new values. After that the user can continue to use the system. We face many questions like: how can we determine the elements affected by a change? Can we be proactive and specify the appropriate new values, without asking the user to enter the new values? How can we replay the next obsels after applying this change? To answer these questions we propose to define impact rules of dependencies between the elements for manipulating the consequences of a change.

4.1 Impact Rules for Element Dependencies

Impact rules define the dependencies between the elements in the system in order to be able to identify the elements that are affected by a change in another element, and to specify the modifications that could be done on the affected elements to stay consistent and valid. Each rule includes a source element (modified element) and the condition on its values that specifies the dependence with a destination element (affected element) and the condition on its values. A rule says that if specific conditions for the values of the source element are fulfilled then some of the values of the destination element determined by the destination condition cannot exist, which requires replacing these values by an adapted value.

Definition. Impact rule

Let *Ȩ* be a set of elements. Each element has a name and some values. Let Ꝋ be a set of operations and Ƒ be a set of functions. We can define an impact rule Ʀ as an implication of the form:

$$\text{Ʀ} = (E_S, C_S) \rightarrow (E_D, C_D) : \text{Æ} \qquad (1)$$

Where E_S, E_D, Æ ∈ Ȩ, and E_S is the source element, C_S is the source condition, E_D is the destination element, C_D is the destination condition, and *Æ* is the adapted element. C_S and C_D are conditions based on operations and functions on the values of the elements. Conditions are composed of operations (Ꝋ) and functions (Ƒ) on elements values. Operations can be logical (and, or, not, etc), mathematical (+, -, *, /, etc) or others. Functions can be grouping functions like (max, sum, min, count, avg) or custom functions like (isNumber, isHoliday, etc).

For each application, system's experts define impact rules for the dependencies between the elements, to determine the consequences of modifying a past obsel. We can get all the impacted obsels for each rule from the entity of the relations between elements and obsels. If we find impact rules having the elements of the modified obsel as source elements and their values satisfying the source conditions, then, for each destination element, if its value satisfies the destination condition, we need to replace the destination element by the adapted one. Adapted values can be specified manually as default values or can be generated automatically using past traces. For example, in SAP-BO Explorer, we consider an impact rule like: if the number of selected measures is greater than one, the element "Chart" cannot be of type "Pie". If a user asks to replay a trace after modifying the number of selected measures that activated this rule, and if there was a successor obsel for changing the chart type to "Pie", then this obsel will not be valid anymore because of this rule, and the chart type will be automatically changed according to the adapted value to be "Vertical Bars". The rule will be as following:

E_S = Selected Measures $\quad$ C_S = (Count () > 1)

E_D = Chart $\quad$ C_D = (type = "Pie")

Æ = (Type = "Vertical Bar")

The user can replay a part of his session after modifying some of the obsels values. These modifications can be of many types like shifting obsel by changing their timestamps, thus causing a change in the order between obsels, updating a value for an attribute of an obsel, or even deleting an obsel. By using impact rules we can determine the consequences of a change and the adapted values. In case of not finding an adapted value of an element or the absence of an impact rule, the corresponding obsels will be invalid. Then the user will have to select the suitable value manually; otherwise the replay process will fail.

4.2 Retrieving Adapted Value from Past Traces

When a user adds a new impact rule, the system asks him to choose the adapted value from a list of possible values, or to keep the system calculating it automatically using past traces. For this purpose, we propose to use a retrieval algorithm similar to the algorithm we presented in [1]. In the original algorithm, obsel values are not taken in consideration when retrieving episodes similar to the current one, because we just wanted to know the next recommended obsels. An episode is a sequence of interactions that allow solving a problem. So, in order to make this algorithm useful for finding the adapted values, we need to make a comparison between the values of the obsels. In addition, we want to retrieve the adapted value for the destination element and not the next recommended obsels.

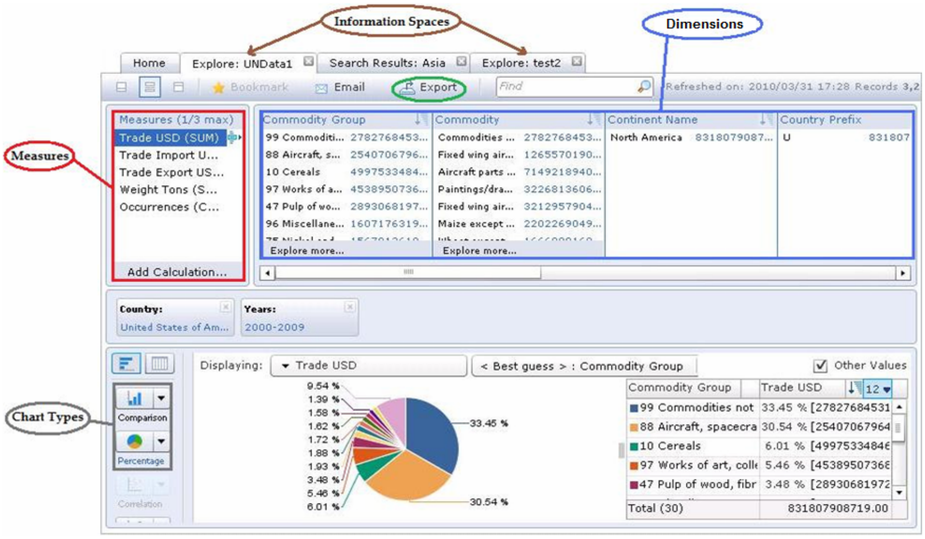
Get-adapted-value algorithm starts by selecting a subset of the trace from its beginning to the modified triggered obsel. Then it retrieves all the past similar episodes to the current one. Similarity includes values comparison. For each similar episode, it calculates the final value of the corresponding element (destination element in the impact rule) as we did in the simple replay, without updating the system. If there is more than one value, we take the one that occurs the most often and we consider it as the adapted one. If we are not able to retrieve any episode, we keep this element as an invalid element until another obsel modifies its value, otherwise the replay process will fail and the system will ask the user to choose the value manually.

5 Implementation

In the previous sections, we have described the models that we have defined to support replay of user interactions by exploiting traces. In this section, we show how we have implemented our trace replay model into the SAP-BO Explore application.

5.1 Trace Collecting and Visualization

Firstly we modified SAP-BO Explorer for being able to collect obsels. SAP-BO Explorer is divided into two parts. Server part is implemented in Java. The management of users' sessions is done in this part, thus enabling many users to work on the system at the same time. The client part is a Flex application; each user has a web application where he can do his exploration. The traces are collected in the client side. Fig. 4 shows a snapshot of the user interface. Each time a user tries to use the system, a new session is opened. Each session contains many obsels, and each action of the user is collected as an obsel presented in a XML format specifying the obsel type, timestamps, and the values of this obsel. We consider that the interface of SAP-BO Explorer is divided into task-oriented blocks, where each block contains obsels dedicated to similar kinds of tasks. The interface consists of blocks for measures, categories, visualization, export, search, etc. For example the measures block contains many types of obsels like select measure, add calculation, edit calculation, etc. For example, when a user tries to select a measure, we capture this action as an obsel of the type "Select measure" from the second block "Measures block". The obsel has for value "Trade USD" and is time stamped with the current timestamp.



Measures (1/3 max)
- Trade USD (SUM)
- Trade Import U...
- Trade Export US...
- Weight Tons (S...
- Occurrences (C...
- Add Calculation...

Commodity Group		Commodity		Continent Name		Country Prefix	
99 Commoditi...	2782768453...	Commodities ...	2782768453...	North America	8318079087...	U	831807
88 Aircraft, s...	2540706796...	Fixed wing air...	1265570190...				
10 Cereals	4997533484...	Aircraft parts ...	7149218940...				
97 Works of a...	4538950736...	Paintings/dra...	3226813606...				
47 Pulp of wo...	2893068197...	Fixed wing air...	3212957904...				
96 Miscellane...	1607176319...	Maize except ...	2202269049...				
Explore more...		Explore more...					

Country: United States of Am...

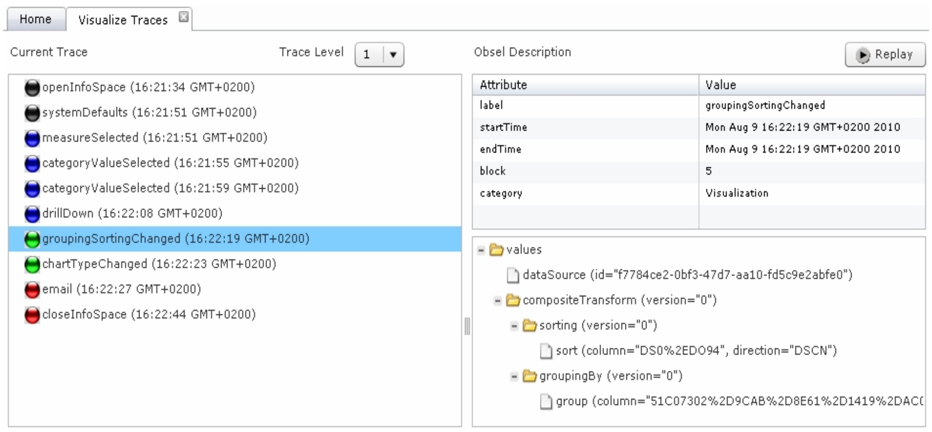
Years: 2000-2009

Commodity Group	Trade USD
99 Commodities not	33.45 % [27827684531
88 Aircraft, spacecra	30.54 % [25407067964
10 Cereals	6.01 % [49975334846
97 Works of art, coll	5.46 % [45389507368
47 Pulp of wood, fibr	3.48 % [28930681972
Total (30)	831807908719.00

Fig. 4. SAP-BO Explorer user interface

Each session is presented as an *M-Trace* stored in XML and has a unique session ID, contains the ID of the user who did this session, and the temporal list of obsels that happened in this session. When a user log himself in SAP-BO Explorer, a request to the server-side is sent in order to open a new session. This triggers the creation of a new XML output file for this session. Each time a new obsel is collected; it is formatted in XML format and sent to the server in order to be added to the session file. Each user can open and manipulate many Information Spaces at the same time. An Information Space is a collection of objects mapped to data for a specific business operations or activities. All the obsels of a session, whatever the Information Spaces they belong to, are stored in the same file.

We have developed a new interface to visualize users' traces displaying a graphical representation of what they have done so far (see Fig. 5). Each obsel is captured according to our model classified according the available types and represented as colored bullets. Obsels appear on the left side of the interface as a chronologically ordered list from the beginning of the session to the most recent obsel. By clicking on an obsel, we can see its description on the right side of the interface. Obsel's values are visualized in the form of a tree of attributes and their values.



openInfoSpace (16:21:34 GMT+0200)
systemDefaults (16:21:51 GMT+0200)
measureSelected (16:21:51 GMT+0200)
categoryValueSelected (16:21:55 GMT+0200)
categoryValueSelected (16:21:59 GMT+0200)
drillDown (16:22:08 GMT+0200)
groupingSortingChanged (16:22:19 GMT+0200)
chartTypeChanged (16:22:23 GMT+0200)
email (16:22:27 GMT+0200)
closeInfoSpace (16:22:44 GMT+0200)

Attribute	Value
label	groupingSortingChanged
startTime	Mon Aug 9 16:22:19 GMT+0200 2010
endTime	Mon Aug 9 16:22:19 GMT+0200 2010
block	5
category	Visualization

```
values
    dataSource (id="f7784ce2-0bf3-47d7-aa10-fd5c9e2abfe0")
    compositeTransform (version="0")
        sorting (version="0")
            sort (column="DS0%2ED094", direction="DSCN")
        groupingBy (version="0")
            group (column="51C07302%2D9CAB%2D8E61%2D1419%2DAC
```

Fig. 5. Trace Visualization Interface

5.2 Trace Replay Implementation

If a user wants to go back to a previous state, he can at any time select the triggered obsel from the list of captured obsels and click on replay button (see Fig. 5). The system will automatically replay traces to go back to this state. A new obsel will be added to the obsels list of type 'Replay'. Its values are set according to the values of triggered obsel. This new obsel indicates that a replay action has occurred here and has triggered a previous obsel. As we explained before, the optimization algorithm uses replay obsels to minimize the number of the replayed obsels by deleting the obsels that are skipped in the replay action.

Each element has different type and number of values from other elements. For not analyzing each element in a different way, we need to make it more general. By using introspection we can determine the type of an object at runtime. Introspection refers to the ability to examine something to determine what it is, what it knows, and what it is capable of doing. Introspection gives us a great deal of flexibility and control. To do that we used Object as type of the values attribute of an obsel, which means that this attribute can have any type of values. We do introspection on this attribute in order to determine the content of it and then to manipulate it in a general way.

6 Discussion and Open Issues

We have implemented our replay method within the SAP-BO Explorer application. However, this method can be applied in any system. To enable trace replay, the first step is to collect traces. For this purpose, we use a model, the *M-Trace,* enabling us to collect all the traces according to the same abstract model. We have experimented with our system by using many types of datasets and by considering all obsels types, opening many sessions together and trying to go back to previous states many times in the same session. We even succeeded to go back to all sessions at the same time by one single go-back command. The execution time of the replay process is very fast, it is like any other action in the application, which means the time of message exchanging between the client and the server. A demo video is available online[1].

Systems like ours face number of challenges like replaying traces for already closed sessions, optimizing replay after modifying past obsels and rechecking impact rules after modifying elements values. But they also face more general problems. For example, in [10], the following issues are raised: privacy of the user and his permission to be traced, security of the system while collecting and visualizing traces, protection of users from undesirable side-effects triggered by the replay, and the robustness of the replay after doing some changes. When implementing our system, we also faced specific problems. For example, in SAP-BO Explorer the same user can open many sessions at the same time. We had to deal with the problem of replaying the trace of a closed session. Our replay process can handle this case by reopening the session, with default values and by applying all the replayed obsels until the triggered one. As we have not implemented yet the replay with changes, we have not faced the problem of optimizing the replay after these modifications. Application of impact rules can be recursive; a modification on an obsel value can have an impact on other obsel values if obsels are related. To deal with these problems we need to develop a graph of impact propagation to solve loops problems and to know the dependencies between different obsels and elements. This is a future work.

When the trace includes obsels that have secure and sensitive information like passwords and credit card numbers, our system detects and obscures the password when visualizing it. But it still needs a lot of enhancements and rules to detect this information and secure it, by notifying the user about it or even asking him to re-enter it again. Our system continuously collects and records user's interactions which constitute a potential risk to privacy and security. This problem is share by all the systems that record rich history traces (web browsers, recommendation systems, etc.). Dealing with this issue is out of the scope of our study. However we do notify our users that all their interactions are recorded. Side-effects are another issue we have to deal with. Indeed, replaying a trace may have unexpected consequences and can damage the system or cause deletion of critical data. In our current implementation, we do not deal with this problem. However, we think that the proposed method described in [10] is relevant to solve such a problem. The idea is to classify obsels into two classes: side-effecting and non side-effecting. This makes easier the annotation of critical obsels. Last, we have to face robustness issues. Indeed, we have to make sure that the trace system is still usable after major changes either on data or on processes of the system. This question is also out of the scope of our study as it is mainly related to the trace collecting phase. We make the assumption that robustness issues are handled by the trace-based system, responsible for traces management.

7 Conclusion and Future Work

In this paper we have described an approach using of interaction traces to allow users to return to a particular state of an application. This approach is an alternative way of undoing actions in applications where undo commands are not available (such as client-server applications). For this purpose, we use play-back of interaction traces. Replay can be identical to the original trace or can introduce different action parameters. We analyze the impact propagation of changes performed on past actions. This work has been conducted in collaboration with SAP Business Objects and the application we used to implement our approach is SAP-BO Explorer. The aim of our contribution within this project was to support replay process in a client-server application, where classical undo commands cannot be implemented. The main contribution of this paper shows how we can replay interaction traces, in an optimized way, in order to go back to a particular state of the application. For that purpose, we have introduced the concept of predefined impact rules and we have built an algorithm that discovers adapted values of obsels affected by changes.

At the time being, the collect process and the simple replay process are implemented. In future work, we plan to address issues mentioned in the discussion concerning side-effects, robustness, and security. SAP-BO Explorer is our first test-bed application. However, our approach is generic enough to be applied to other application. Hence, another future work is to experiment with this approach in a different application domain (namely video annotation) and with a different category of end-user. Our goal is to study how interaction traces, and Trace-Based Reasoning can contribute to help user better understand and use applications.

Acknowledgments. We thank Francoise Corvaisier, member of SAP-BO enterprise for her support, thoughts and for giving us the opportunity to do this work in SAP-BO. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsors.

References

1. Zarka, R., Cordier, A., Corvaisier, F., Mille, A.: Providing assistance by reusing episodes stored in traces: a case study with SAP Business Objects Explorer. In: Le Ber, F., Renaud, J. (eds.) 18ème Atelier Raisonnement à Partir de Cas, hal-00497210, Strasbourg, pp. 91–103 (2010)

2. Champin, P.-A., Prié, Y., Mille, A.: MUSETTE: a framework for Knowledge from Experience. In: EGC 2004, RNTI-E-2 (article court), pp. 129–134. Cepadues Edition (2004)
3. Cordier, A., Mascret, B., Mille, A.: Extending Case-Based Reasoning with Traces. In: Grand Challenges for Reasoning from Experiences, Workshop at IJCAI 2009 (2009)
4. Settouti, L.S., Prié, Y., Champin, P.-A., Marty, J.-C., Mille, A.: A Trace-Based Systems Framework : Models, Languages and Semantics (2009), http://hal.archives-ouvertes.fr/inria-00363260/PDF/trace.pdf
5. SAP: SAP Business Objects Explorer: Explore your business at the speed of thought, http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/search-navigation/explorer/index.epx
6. Little, G., Lau, T., Cypher, A., Lin, J., Haber, E., Kandogan, Koala, E.: capture, share, automate, personalize business processes on the web (2007), http://portal.acm.org/citation.cfm?id=1240624.1240767
7. Leshed, G., Haber, E.M., Matthews, T., Lau, T.: CoScripter: automating & sharing how-to knowledge in the enterprise. In: CHI 2008 Proceeding of the Twenty Sixth Annual SIGCHI Conference on Human Factors in Computing Systems, pp. 1719–1728 (2008)
8. Anupam, V., Freire, J., Kumar, B., Lieuwen, D.: Automating Web navigation with the WebVCR. Computer Networks 33, 503–517 (2000)
9. Safonov, A., Konstan, J.A., Carlis, J.V.: Beyond Hard-to-Reach Pages: Interactive, Parametric Web Macros. In: Proc. Human Factors and the Web, pp. 1–14 (2001)
10. Hupp, D., Miller, R.C.: Smart bookmarks: automatic retroactive macro recording on the web. In: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, pp. 81–90 (2007)
11. Li, I., Nichols, J., Lau, T., Drews, C., Cypher, A.: Here's what i did: sharing and reusing web activity with ActionShot. In: Proceedings of the 28th International Conference on Human Factors in Computing Systems, pp. 723–732. ACM, New York (2010)
12. Briand, L.C., Labiche, Y., O'Sullivan, L.: Impact analysis and change management of UML models. In: Proceedings of International Conference on Software Maintenance, ICSM 2003, pp. 256–265. IEEE Comput. Soc., Los Alamitos (2003)
13. Maule, A.: Impact analysis of database schema changes (2010), http://eprints.ucl.ac.uk/19497/
14. Sybase: Power Designer: Impact and Lineage Analysis (2010), http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc38093.1520/doc/html/rad1232025100240.html
15. Harrington, R.: Understanding Adobe Photoshop CS4 The Essential Techniquesfor Imaging Professionals. Peachpit Press, Berkeley (2009)
16. Dietterich, T.G.: TaskTracer project, http://tasktracer.osuosl.org/
17. Xu, K., Muñoz-Avila, H.: CaBMA: a case-based reasoning system for capturing, refining, and reusing project plans. Knowledge and Information Systems 15, 215–232 (2007)

Author Index