2009-01-01

Proceedings of the ICCBR 2009 Workshops

Sarah Jane Delany
Technological University Dublin, sarahjane.delany@tudublin.ie

Proceedings of the ICCBR 2009 Workshops

Sarah Jane Delany (Editor)

Preface

I am pleased to present the Workshop Proceedings of the Eighth International Conference on Case-Based Reasoning (ICCBR-09) which is held in Seattle in July 2009.

This year the workshop programme includes four workshops, two of which are workshops in well recognised CBR research areas continuing the well established, successful workshops on *Case-Based Reasoning in the Health Sciences* and on *Uncertainty, Knowledge Discovery and Similarity*. Along with these I am pleased to present two new workshops which explore developing areas of CBR research, namely *Reasoning from Experiences on the Web (WebCBR)* and *Case-Based Reasoning for Computer Games*. In addition, our workshop programme this year includes the papers from the finalists of the second Computer Cookery Competition, the follow-on from the very successful competition held at ECCBR in Trier, Germany in 2008.

I would like to thank all who contributed to the success of this workshop programme, including the presenters and authors of the workshop papers for their contributions along with the programme committees for the quality and timeliness of their reviews. In particular, my thanks goes to the workshop organisers for their support and co-operation in the organisation of this programme, especially to Kerstin Bach and to Derek Bridge whose skill in LaTeX helped me considerably.

I very much appreciate the support of the conference chairs Lorraine McGinty and David Wilson and value the opportunity to organise the workshops. My special thanks goes to the local chair Isabelle Bichindaritz for her help with the production of the proceedings book.

I sincerely hope that the participants enjoy this year's workshop programme and that this collection of papers will inspire and encourage more CBR-related research in the future.

Sarah Jane Delany July 2009

Table of Contents

WebCBR09: Reasoning from Experiences on the Web

Case-Based Reasoning for Computer Games

Uncertainty, Knowledge Discovery and Similarity in Case-Based Reasoning

Case-Based Reasoning in the Health Sciences

WebCBR:

Reasoning from Experiences on the Web

Workshop at the
Eighth International Conference on
Case-Based Reasoning
(ICCBR 2009)

Seattle, Washington, USA
July, 2009

Derek Bridge, Enric Plaza
and Nirmalie Wiratunga (Eds.)

Co-Chairs

Derek Bridge
University College Cork, Cork, Ireland
d.bridge@cs.ucc.ie

Enric Plaza
IIIA-CSIC, Catalonia, Spain
enric@iiia.csic.es

Nirmalie Wiratunga
Robert Gordon University, Scotland, UK
n.wiratunga@rgu.ac.uk

Programme Committee

Klaus-Dieter Althoff, University of Hildesheim, Germany
Ralph Bergmann, University of Trier, Germany
Sutanu Chakraborti, IIT Chennai, India
Paulo Gomes, University of Coimbra, Portugal
David Leake, Indiana University, USA
Ashwin Ram, Georgia Institute of Technology, USA
Barry Smyth, University College Dublin, Ireland
Brigitte Trousse, INRIA, France

Additional Reviewers
Kerstin Bach, University of Hildesheim, Germany
Kane Bonnette, Georgia Institute of Technology, USA
Alexandre Hanft, University of Hildesheim, Germany
Mirjam Minor, University of Trier, Germany
Meike Reichle, University of Hildesheim, Germany
Saurav Sahay, Georgia Institute of Technology, USA

Preface

We are pleased to present the proceedings of the Workshop in Reasoning from Experiences on the Web (WebCBR), which was held as part of the Eighth International Conference on Case-Based Reasoning in Seattle, July 2009.

Advances in web technology have led to vast amounts of user generated web content in the form of blogs, emails, reviews and opinions. Increasingly people search and browse other people's experiences on travel, medicine, retail, entertainment, etc. While they can be treated as documents, we take the view that they are more appropriately seen as a rich source of untapped experience data, a valuable asset that can be used to generate web experience bases through Case-Based Reasoning (CBR) technology.

Proliferation of web content invariably also means significant increases in web usage. Typically many users will have similar searching and browsing needs and should ideally benefit from this commonality. Currently user behaviour remains mostly un-captured but the potential to facilitate reuse of web usage experiences also creates an exciting opportunity for CBR research.

This workshop promotes CBR as a means to advance web technology in two ways: firstly by enabling better capture and representation of explicit yet unstructured experiential web content, and secondly by harnessing web usage experiences to improve browsing and searching.

The workshop was a forum for intense discussion of this research direction, provoked by the presentation of eight papers, which are collected in these proceedings. The papers include Web2.0 applications in resource sharing, content generation and visualisation. Importantly papers also addressed the opportunities and challenges for CBR when harnessing experience on the web.

We wish to thank all who contributed to the success of this workshop, especially the authors, the Programme Committee, the additional reviewers, the panelists and Sarah-Jane Delany (the ICCBR Workshop Chair).

Derek Bridge July 2009
Enric Plaza
Nirmalie Wiratunga

There and Back Again: Building Web-Scale CBR

Larry Birnbaum and Kristian J. Hammond

Northwestern University
Intelligent Information Laboratory
Evanston, Illinois USA
{birnbaum, hammond}@infolab.northwestern.edu

A Story

We open this paper on a personal note. The two of us practically cut our teeth as researchers during the early days of CBR, and we remain very proud of the work we did at that time (see, for example, Hammond, 1989; Birnbaum and Collins, 1989). The core insight of CBR—that effective human and machine reasoning must proceed on the basis of specific prior experiences—seems to us as true today as ever. And the semantic analyses of language and planning we developed in building systems based on this idea still seem to us not just true, but actually quite beautiful.

For more than a decade now, however, our work has been focused elsewhere, on technologies for intelligent information systems. These systems are aimed at delivering information in the broadest possible sense, including media, services, and connections to other people, on the basis of a user's current context, including task, location, and the content of the documents or media with which he is engaged. They are user-centric, context-aware, "frictionless" and proactive. We have built a score or more of such systems based on the technologies we've developed. Most importantly, these technologies, and so the systems built upon them, are robust and scalable, because they are built on top of a robust and scalable substrate: information retrieval.

And this gets to the nub of why we moved our research in the direction that we did. Case-based reasoning is based on the appealing idea that situation understanding and problem-solving can leverage specific prior experiences in large, already-organized "chunks"—in other words as narratives or stories in one form or another. This idea is intuitively appealing because we all experience being reminded of relevant stories by situations or problems, and because we regularly communicate with other people using them. It is computationally appealing because it provides a possible alternative to models based on logical reasoning "from scratch," with their well-known issues in terms of both scalability and robustness. So scalability and robustness were part of the original motivation for CBR in the first place.

Unfortunately, in its "classic" form, CBR did not seem to meet this promise. It still required a large corpus of highly-structured, well-represented cases, and the construction of such case libraries, like the construction of large logically-represented knowl-

edge bases, remains a continued barrier to scalability.[1] And it has proven extremely difficult to construct robust and scalable methods for adaptation (or "tweaking"), which were a core part of the original CBR vision.

CBR has surely made progress on both these fronts since that time. Nevertheless, the chief lessons we drew from these experiences seem to us uncontrovertible: First, *scalability is critical, and must be built in from the beginning*. The notion of scaling up methods after the fact rarely works, if ever. Second, *scalability is only possible by building on an inherently robust and scalable substrate*. Using these lessons, we would like to propose an approach to constructing CBR systems built, like our web-scale intelligent information systems, on top of information retrieval (IR). In these systems, the determination of relevance, utility, and interestingness, is under the control of explicit, context-sensitive rules. However, these rules "bottom out" not in logic or structured representations, but in standard term-based queries and filters, in turn executed by standard IR technology. The result is a set of intelligent information technologies that are powerful in scope yet responsive to human need. Perhaps it's time to take these technologies, originally inspired by our work in CBR, back to the development of the next generation of robust and scalable CBR systems.

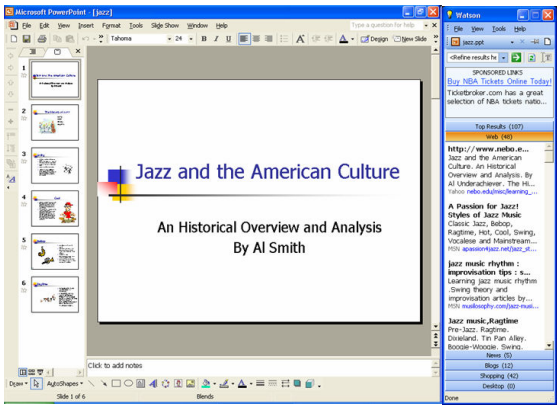
From Similarity to Utility

We start with a brief description of one of our first intelligent information systems, ***Watson***, to set the context. Watson (Budzik and Hammond, 2000; Budzik, Hammond, and Birnbaum, 2001) is a *proactive* information assistant that automatically finds information relevant to the document a user is currently reading or writing in a variety of application settings (including word processing and web browsing), and then presents this information to the user. Watson, in other words, takes the document a user is manipulating as an indicator of the user's task context, and tries to find information relevant to that context. The system works by analyzing the current document statistically and heuristically, using the resulting context model to automatically formulate queries, directing these queries to relevant online information sources (including web search engines), and then filtering, ranking, and presenting the results to the user in a "sidebar" as shown below.

Watson's results are almost always, in a technical sense, *relevant* to the user's context. In fact, our studies showed that users judged its results to be *twice as good*, in terms of precision, as the results returned from queries that they formulated themselves (over 60% vs. about 30% of the top 10 from each). But, while "on point," these results aren't necessarily very *interesting* or *useful*. Too often, in fact, the results are pretty much the same, from the user's point of view, as the information provided in the original document context.

---

[1] Although textual CBR systems have ameliorated this problem in certain domains by "extending" a well-represented core set of cases into the much larger space of unrepresented text. See for example the work described in Ashley and Lenz (1998).



The problem is that Watson uses similarity as a *proxy* for relevance: If a document is highly *similar* (in a standard IR sense) to the document you are looking at, then it is deemed *relevant* to that document. This is a basic assumption underlying information retrieval systems generally, but it is a simplification, and often an oversimplification (Budzik, *et al.*, 2000; Bergmann, 2002). To take the most extreme case, the document that is most similar to the document you have in your hand is another copy of that document itself. This obviously isn't very useful, because it doesn't give you any new information.

It's clear, then, that to be genuinely relevant, information must be similar in certain respects to the context, but *dissimilar* in certain other respects. The particular dimensions of (dis)similarity that matter in a given context will depend upon the user's information needs—which in turn depend upon the user, the user's task, and the domain at hand.[2]

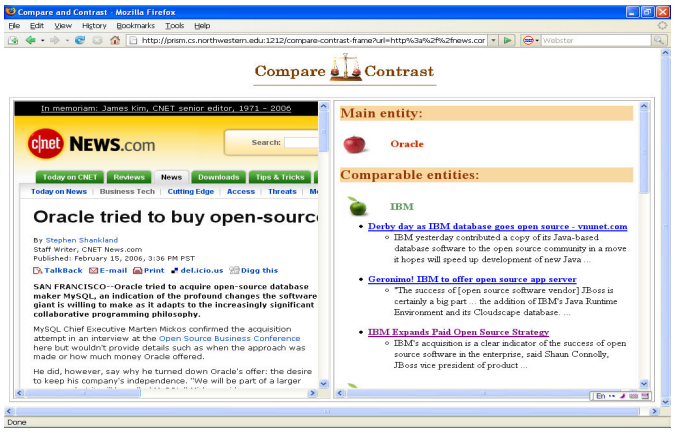
A key component of these methods must involve using some portions of the context as a focus or base, while systematically varying—tweaking!—other attributes of the information being sought in specific ways. In this sense, the methods we propose are ***transformational***. Starting from a model of the current context or situation, they apply transformations to generate queries and filters aimed at identifying information, in the broadest sense, that is relevant along identified and explicit dimensions, and aimed at satisfying specific information needs determined by the context and task. These transformations might include adding, deleting, or substituting terms in the queries or filters being constructed, or imposing additional constraints along other dimensions of variation such as authorship, sentiment, narrative structure, etc.

Our model, in broadest terms, is this: The domain-dependent aspects of a situational context are managed using term-based IR techniques. Domain knowledge is not explicitly represented, but is simply embodied in relatively unanalyzed documents

[2] Of course, the general notion that relevance depends on task will not come as a surprise to anyone working on CBR.

or media (perhaps with some simple metadata). This provides scalability and robustness. The current information needs, as determined by the task, are used to transform this basic representation, for example by adding additional terms that bias the results in directions that make it more likely they will satisfy those needs. The mechanisms for accomplishing this will generally be rule-based. This provides context-sensitivity and goal-directedness.

One of the systems we've built along these lines is called ***Compare & Contrast*** (Liu, Wagner, and Birnbaum, 2007). As the name suggests, this system, given a document, aims to identify and retrieve documents describing similar but distinct situations in order to support reasoning by comparing and contrasting the cases. In particular, given a news story, it aims to find news stories describing similar situations involving different agents. For example, given a story about Israel's use of cluster bombs during the 2006 Lebanon War, rather than retrieving hundreds or thousands of stories on the same topic, it finds and presents stories about US use of cluster bombs in the Iraq War, NATO use of cluster bombs during the Balkans campaign, and so on. In this sense, it is aimed at providing users with *information diversity* of a certain kind (McSherry, 2002; see also Leake *et al*., 1999; Budzik *et al*., 2000; Krema *et al*., 2002). In the example shown below, a story about Oracle's effort to acquire MySQL results in the retrieval and presentation of a number of stories about IBM's efforts to expand its open-source strategy through similar acquisitions.



The system does this by developing two "models" of a story it is given. The first consists of the named entities in the story, with particular effort made to discover the main actor. The second is a stop-listed histogram of the other terms in the text. The system finds comparable entities to the main actor, and then systematically combines these with terms from the second model—in effect, *substituting* these comparable entities for the original actor—to form queries which are then directed at news search engines. These comparable entities may be identified by using pre-existing hierarchies when available. However the current version of the system attempts to find these entities on the web itself (Liu and Birnbaum, 2007). It does this by first formu-

lating general queries based on the second model (terms other than named entities). The first *n* resulting documents are then compared with the original story to find entities that play a syntactic role similar to that of the main entity in the original story, i.e., that play a similar role in lexically similar sentences. All the documents ultimately retrieved by the system are on this initial list as well, of course; but searching without any named entities is an extremely noisy process, so they may be quite far down the list. Similarly, the list also includes a large number of stories that aren't at all relevant to the current context.

From Intelligent Information Systems to Case-Based Reasoning

From the above discussion, we can discern the outlines of a general architecture for CBR systems built using transformational IR technology, comprising the following processes:

1. *Develop a representation of the current context* (e.g., the documents users are currently manipulating, data from real-time systems, text streams, etc.), combining lexical elements and metadata from and about the context.
2. *Determine likely information needs* in that context, based either on some prior expectations about likely goals or interests in that context—i.e., the task—or on some indication by a user.
3. *Retrieve or generate appropriate transformations* specifying information dimensions or attributes relevant to the user's information needs, along with *filters* that specify additional such dimensions or attributes.
4. *Parameterize and apply these transformations* to generate queries and filters.
5. *Dispatch the queries* to appropriate information sources, such as case libraries, and manage the interaction with those sources.
6. *Filter and rank results* based on metadata supplied by these information sources and/or the entire result itself, using the filters identified in step 3 in conjunction with the initial context representation.

For example, in the earliest phases of situation understanding, a system built along the lines described above would take incoming reports and information, and, using standard IR techniques, search for prior cases involving similar reports and information. But if the reports and information concerned an adversary's actions, the system might also want to present the user with cases in which similar reports and information were the result of efforts by the adversary to *misdirect* possible observers. Such cases might be actively searched for simply by the addition of terms such as "feint" or "misdirection." Alternatively, finding such cases, if any, might require more complex transformations—e.g., taking the input reports, plus a user's assessment of what they might mean, and then searching for cases in which similar input reports are connected with quite different assessments of what they mean—along with historical data as to the ultimate correctness or utility of these assessments (i.e., outcomes). These kinds of transformations would be automatically applied to the current situation model,

based on the user's current task and its state, and used to retrieve the relevant prior cases, if any.

Similarly, in retrieving prior plans to address the situation, the system might utilize transformations aimed at uncovering different types of plans utilizing different resources—e.g., air operations, covert operations, etc. And finally, if the current task involves evaluating potential plans, it would be possible to bias retrieval of prior cases based on particular relevant dimensions, e.g., civilian casualties, US forces casualties, scale of forces utilized, etc.

Some of the query transformations or filters described above make use of document or metadata structure, as when, e.g., searching for a similar situation coupled with a different assessment. A critical challenge in utilizing text as opposed to more complex representations is, of course, its lack of explicit logical structure. (This is what IR gives up in exchange for robustness and scalability.) In some cases, this can be sidestepped, as in Compare&Contrast's simplistic but effective assumption that actors in news stories will be represented by named entities. In other cases, however, document or metadata structure can serve as a proxy for logical structure, with different transformations or filters applied differentially to different components of the documents or metadata. Many documents have a fairly rigid structure with sections explicitly labeled "Procedure," "Outcome," "Background," "Budget," etc. This makes it possible to systematically vary those portions of the queries or filters derived from certain sections while holding others fixed or applying different transformations to them.

We can also transform queries to bias retrieval towards particular kinds of narrative forms. The ***Buzz*** system (Owsley, Hammond, Shamma, and Sood, 2006) retrieves and presents *personal stories* on specified topics or relating to certain themes from the blogosphere. To find compelling stories, Buzz mines the blogophere, collecting blog posts in which the author describes an emotionally compelling situation—a dream, a nightmare, a fight, an apology, a confession, etc.—using syntactic story indicators in conjunction with topical or thematic terms to automatically generate appropriate queries. These story indicators are simply sets of words and phrases that we've empirically observed tend to be good cues that a document in fact contains a personal narrative. Such a process isn't going to cast a particularly wide net—i.e., the *recall* of this system in IR terms isn't very good. But at web scale, a high recall rate isn't necessarily a virtue: there are a *lot* of stories on the web—we're just looking for a few good ones.[3]

After retrieving blog posts containing both the required topical or thematic terms and one or more story indicators, Buzz applies affective classification to focus on entries with a heightened emotional state, and applies additional syntactic filters to find the most story-like entries and select the appropriate portions of those entries. After

[3] And indeed, many of the systems we have built tend to use fairly narrowly-formed queries and aggressive filtering, partly with an eye towards favoring precision over recall, but primarily in order to avoid being overwhelmed by enormous numbers of results.

passing through these filters, the resulting story selections are compelling and emotional. They can be presented in a variety of formats, including a performance model in which they are read aloud by animated characters. Here is an example of a story discovered by the system:

> My husband and I got into a fight on saturday night, he was drinking and neglectful, and I was feeling tired and pregnant and needy. It's easy to understand how that combination could escalate, and it ended with hugs and sorries, but now I'm feeling fragile. Like I need more love than I'm getting, like I want to be hugged tight for a few hours straight and right now, like I want a dozen roses for no reason, like a vulnerable little kid without a safety blankie.

From Finding Narratives to Constructing Narratives

Ultimately, CBR systems must be able not only to find apt stories for presentation to users, they must be able to actually generate stories. ***News at Seven*** (Nichols, *et al*., 2007) is a system that automatically generates a virtual news show. Totally autonomous, it collects, parses, edits and organizes news stories and then passes the formatted content to artificial "anchors" for presentation. Using information resources present on the web, the system goes beyond the straight text of the news stories to also retrieve relevant images, videos, and blogs with commentary on the topics to be presented. It then edits and combines these into a single, coherent presentation. Once it has assembled and edited its material, News At Seven presents the content to its audience using animated characters and text-to-speech (TTS) technology in a manner similar to the nightly news.

The initial implementation of News at Seven built reports following a single narrative model: a short news show with a few stories from topic areas of interest, and in-

cluding a single "man in the street" comment extracted from the blogosphere on one of these. The system has since been expanded to utilize *multiple* narrative models. One of the more interesting such models we've implemented so far builds movie review shows in the spirit of Siskel and Ebert, in which two animated "reviewers" present different points of view on the movie in question in an interactive and conversational manner, producing a virtual "argument." The narrative models themselves are expressed as templates specifying not the particular *topic* of the resulting story, but the *relationships* that must hold among the elements that comprise the story and which must be retrieved. These relationships are used to specify query templates which are filled out with terms extracted from the topical elements at hand to retrieve material which is on point and fills the appropriate role in the overall narrative model.

Finally, we have begun to investigate both the construction and detection of "higher order" narrative structures. We've developed a prototype system for constructing "rants" in which a virtual character begins to talk about a topic and then escalates his rhetoric saying more and more outrageous things about that topic. We've also built a prototype system for generating short, humorous comic strips. The text for these comics exploits ambiguity to generate puns using online lexical resources. Because the results to date are fairly hit-or-miss, a social media component is included to enable humans to collectively select the comics that actually work.

On the detection side, we've developed a prototype capable of detecting abstract themes (an exceptionally hard problem in natural language understanding) by propagating human-generated labels. For instance, the system can determine that the bank bailout of last fall might be construed as an instance of "setting the fox to guard the henhouse." The system works by combining terms from thematic descriptions (e.g., "fox," "henhouse," etc.) with topical terms (e.g., "bank bailout") to ascertain whether the given thematic label has been applied to that topic by a human being. It should be emphasized that the resulting system in no way understands the deep analogy between the theme and the specific topic at hand. Instead, it uses the "human computation" approach proposed by Luis von Ahn—with the notable difference that it mines labels previously attached by humans to these narratives, rather than eliciting them in an interactive mode.

Back to the Future

Our focus on web-scale intelligent information systems over the past ten or more years was based on a certain frustration with our earlier work in case-based reasoning. While we felt that there was much truth, and even beauty, in the CBR models we developed at that time, their lack of robustness and scalability eventually drove us to conclude that we were building on the wrong substrate, and needed to find an underlying set of mechanisms that would be inherently scalable and noise tolerant—which is to say, statistical mechanisms, in particular, information retrieval. Nevertheless the issues of story retrieval, transformation, and construction have remained in the back of our thoughts. We think the time is right to bring them to the fore again.

Acknowledgments: We thank all the member of the InfoLab, past and present, for their help and collaboration in developing our ideas as well as the systems described above. The rant project is being developed by Lisa Gandy; the web comics project by Patrick McNally; and the thematic labeling project by Earl Wagner. We also thank the National Science Foundation (under grants IIS-0325315 and IIS-0535231), and Motorola for their support.

References

Ashley, K., and Lenz, M. (eds.) (1998) *Textual Case-Based Reasoning, Papers from the AAAI-98 Workshop*, AAAI Technical Report WS-98-12 AAAI Press, Menlo Park, CA.

Bergmann, R. (2002) *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*. Springer.

Birnbaum, L., and Collins, G. (1989) Remindings and engineering design themes: A case study in indexing vocabulary. *Proceedings of the First Case-Based Reasoning Workshop*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 47-51.

Budzik, J., and Hammond, K. (2000) User interactions with everyday applications as context for just-in-time information access. *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, ACM Press.

Budzik, J., Hammond, K., and Birnbaum, L. (2001) Information access in context. *Knowledge-Based Systems*, vol. 14, pp. 37-53.

Budzik, J., Hammond, K., Birnbaum, L., and Krema, M. (2000) Beyond similarity. *Artificial Intelligence and Web Search: Papers from the AAAI Workshop*, AAAI Press, pp. 6-11.

Hammond, K. (1989) *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, San Diego, CA.

Krema, M., Birnbaum, L., Budzik, J., and Hammond, K. (2002) *Themometers* and *themostats*: Characterizing and controlling thematic attributes of information. Proceedings of the 2002 International Conference on Intelligent User Interfaces, ACM Press.

Leake, D., Birnbaum, L., Hammond, K., Marlow, C., and Yang, H. (1999) Integrating information resources: A case study of engineering design support. *Proceedings of the Third International Conference on Case-Based Reasoning (ICCBR-99),* Munich, Germany, pp. 482-496.

Liu, J., and Birnbaum, L. (2007). Measuring semantic similarity between named entities by searching the web directory. *Proceedings of the 2007 Conference on Web Intelligence*, Freemont, CA.

Liu, J., Wagner, E., and Birnbaum, L. (2007) Compare&Contrast: Using the web to discover comparable cases for news stories. *Proceedings of the 16th International World Wide Web Conference*, Banff, Canada, pp. 541-550.

McSherry, D. (2002) Diversity conscious retrieval. (2002) In Craw, S., and Preece, A., eds., *Advances in Case-Based Reasoning (Proceedings ECCBR-2002)*.

Nichols, N., Liu, J., Pardo, B., Hammond, K., and Birnbaum, L. (2007) Learning to gesture: Applying animations to spoken text. *ACM Conference on Multimedia*.

Owsley, S., Hammond, K., Shamma, D. and Sood, S. (2006). Buzz: Telling compelling stories. *ACM Conference on Multimedia Interactive Art*.

Using Experience on the Read/Write Web: The GhostWriter System

Derek Bridge and Aidan Waugh

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

Abstract. On the Read/Write Web, users are authors as much as they are searchers. We describe one concrete example of this, a waste exchange service where users submit descriptions of items that they have available but wish to get rid of. It is generally to the advantage of subsequent users if authors write comprehensive descriptions. We propose that *successful* descriptions, i.e. ones which did result in the user passing on an item for reuse, be used as cases. We describe the GhostWriter system that we have designed and built: it makes content authoring suggestions using feature-values extracted from the cases. We end with a preliminary, off-line ablation study, which shows promising results.[1]

1 Introduction

Web 2.0 is the era of the Read/Write Web. The current Web makes information-seekers but also information-authors of us all. On the one hand, we search and browse; on the other hand, numerous web pages offer us the opportunity to post our own content.

In his invited talk at the Ninth European Conference on Case-Based Reasoning, Enric Plaza argues that people increasingly use the Web as a repository for recording and sharing experiences [7]. These experiences include reviews of products, such as hotels, electronic goods, books and movies, in which people report their experience of consuming these products; and they include 'how-to' plans and recipes, in which people report their experience of carrying out tasks to achieve certain goals. Users search and browse web pages, blogs and forums, to retrieve, aggregate and reuse these experiences to help them make decisions in the real-world (e.g. which hotel to book) and to help them do things in the real-world (e.g. how to install a piece of software). Reasoning with these unstructured experiences is a new direction in CBR research.

[1] This research was funded by the Environmental Protection Agency of Ireland under Grant Number 2007-S-ET-5. We are grateful to Maeve Bowen and Catherine Costello of Macroom-E and wastematchers.com and to Lisa Cummins of University College Cork for their engagement in our research.

All of this emphasizes how important it is, when exploiting the write-capabilities of the Read/Write Web, that we author high quality content. In this paper, we show how to reuse existing Web content to support the authors of new content.

In Section 2, we present the concrete scenario in which we are conducting our research, namely a waste-exchange service. In Section 3, we present GhostWriter, our case-based approach to making content suggestions to authors. Section 4 contains a discussion, including a review of related work. Section 5 presents preliminary experimental results.

2 Waste Exchange Services: A Case Study

Waste exchange services connect organizations or households who have unwanted items with organizations or households who can use those items. The advantages of such a service include: it provides a way of diverting waste from landfill; it provides a way of saving on storage or disposal costs; and it provides a way of sourcing cheap or even free materials.

Many waste exchange services operate over the Web. Organisations or households use the web site to submit descriptions of the items that they have available. Other organisations or households either search and browse for items that they can reuse, or they submit descriptions of items and the waste exchange service automatically contacts them when such items become available. On-line waste exchange services are a great example of the Read/Write Web: their users engage in both search and authoring.

But, in a waste exchange service there are at least four reasons why an exchange may fail to take place:

- It may, of course, be that an unwanted item that is available through the service is not wanted by anyone else, or that an item that is requested through the service is not available from anyone else using the service.
- Surprisingly, the search facilities of these services are often quite rudimentary. They use simple search engines, which may fail to find matches between descriptions of items available and requested. Although not the focus of this paper, it is part of our work to use ideas from Information Retrieval and case-based retrieval to improve the search engine of the waste exchange service that we are working with.
- The descriptions of items available or wanted that users submit are often quite short, which reduces the likelihood that the search engine will find a match. In the waste exchange service with which we are working, for example, the average length of descriptions of items available is just 8 words or 6 words if we ignore stop-words; and the average length of descriptions of items wanted is just 6 words or 4 words if we ignore stop-words.
- The service may find a match between a pair of descriptions but, when their authors make personal contact, it may turn out that the match found by the service is spurious and does not satisfy at least one of the persons involved. Note how the short descriptions that we mentioned in the previous bullet point increase the likelihood of spurious matches. A transaction may

be abandoned when features that were not included in a description (maybe the colour, the price, the delivery terms, etc.) become known.

From this analysis, it seems useful to consider how a waste exchange service can support people in authoring better descriptions. There is an obvious source of experience that we can exploit: successful descriptions.

Many waste exchange services have transaction closure facilities. Consider a user who had submitted a description of an unwanted item, for example. When she deletes the description, the service shows a form that requires her to explain why the description is being deleted. She may have sold or given away the item through the waste exchange service; she may have sold or given away the item but not through the service; she may have disposed of the item (e.g. by sending it to landfill); or she may have failed to dispose of the item. We propose that descriptions of items that have been sold or given away through the waste exchange service should be retained in a case base. These are successful descriptions: ones that work.

We can use these successful descriptions to make suggestions. When a user is authoring a new description, we can prompt her to think about including certain kinds of content: content that we find in successful descriptions. We explain the details of the way we do this in the next section.

Figure 1 shows an overview of a waste exchange service that includes a suggestion facility of the kind we have described. The left-hand side of the figure represents a standard waste exchange service. Users who have items insert them into a database of items available, and search a database of items wanted; users who want items insert them into a database of items wanted, and search a database of items available. Transaction closure results in the update of statistics. But, as the right-hand side of the figure shows, we propose the service also inserts successful descriptions into a case base. Then the service can use successful descriptions of items wanted to make content authoring suggestions to users who are describing wanted items, and use successful descriptions of items available to make content authoring suggestions to users who are describing available items.

3 GhostWriter: Case-Based Content Authoring Suggestions

GhostWriter is the system that we have designed and built. It relies on feature extraction, which we apply in advance to successful descriptions (cases) and incrementally to the author's own description as she writes it. The mechanism we have designed and built for making the suggestions is novel but is inspired by Conversational CBR techniques.

Up to now, our implementation, built using jColibri,[2], is suitable only for running off-line experiments, preliminary results for which are described in Section 5. Ultimately, however, we plan to implement an Ajax client that will proactively

[2] `http://gaia.fdi.ucm.es/projects/jcolibri/`

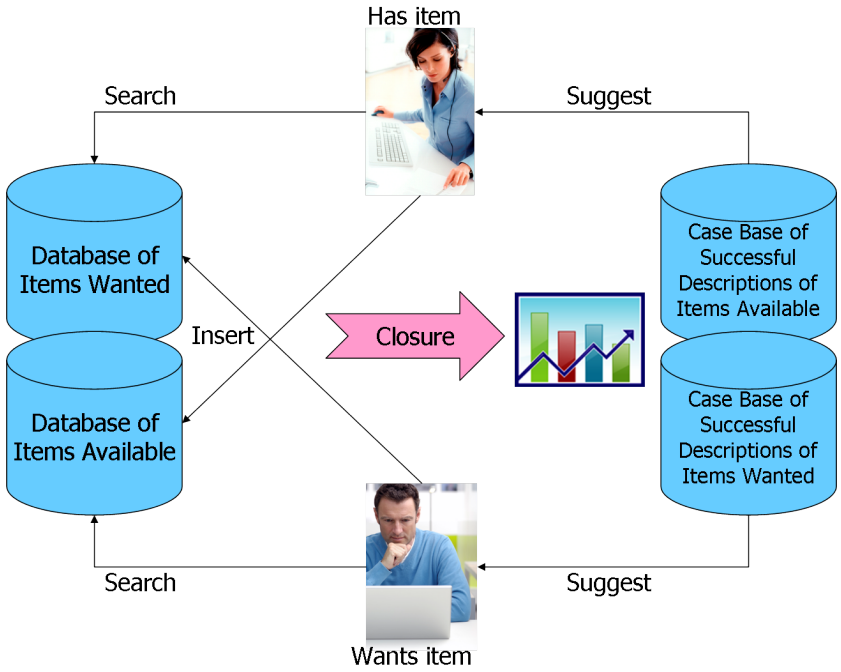


Fig. 1. A waste exchange service that includes a suggestion facility

send asynchronous requests to the server-side GhostWriter system: as the user's content grows, the client will send the new content in an asynchronous HTTP request to the server; the client will update a suggestions pane with GhostWriter's responses. In this way, the user is not interrupted from her normal work, either to invoke GhostWriter or to receive its results. The user can click on a suggestion in the suggestion pane if it is close enough to her current intentions and it will be incorporated into her content, where she can edit it. More likely, the suggestions will not be close enough to what is wanted but will prompt the user to include content she hadn't thought of including. For example, if one of the suggestions is "Will deliver within a 10 mile radius", this might prompt the user to include her own delivery terms, even if these are very different from the suggested ones. Hence, even if the use of suggestions means that descriptions more often have the same *features*, they may still be novel descriptions by virtue of not having the same *feature-values*.

3.1 Case and new item representation

As mentioned, a case is a successful description. It therefore primarily consists of free text. But, as it enters the case base, we apply Feature Extraction rules. For now in our work we produce these rules manually. Each is in essence a regular expression that aims to find and extract a particular feature-value pair

```
[ItemCondition]{0}(\w+\s+([Cc]ondition|[Qq]uality))
[ItemCondition]{0}(VGC|[Vv]gc|[Ww]orn|[Tt]or[en]|[Bb]roke|[Bb]roken)
```

Fig. 2. Example Feature Extraction rules in the format `[FeatureName]{FeaturePosition}RegularExpression`

from the text. Hence, the rules augment each case by a set, possibly empty, of feature-value pairs. Two example rules are shown in Figure 2. Both these rules extract the ItemCondition feature. The zero indicates that the rule extracts the entire expression. In the first rule, the regular expression matches phrases such as "excellent quality" and "very good condition"; the regular expression in the second rule matches 'stock' phrases and words for describing an item's condition such as "vgc" (very good condition), "worn", "torn", etc. (In GhostWriter, this rule has more disjuncts than are shown here.)

More formally then, a case c comprises some free text, $text(c)$, and a set of feature-value pairs, $fvs(c)$. We will denote a feature-value pair by $\langle f, v \rangle \in fvs(c)$. Note that cases do not comprise problem descriptions and solutions. There is no solution part to the cases. This is because making content authoring suggestions is in some sense a form of case completion [2]: we use cases to suggest content that the author might add to her description.

New items that the user is authoring have exactly the same representation as cases: free text and feature-value pairs. The only difference is that they grow in size, as the author adds to her content. We will denote a new item description as nid.[3]

3.2 Conversational case-based suggestions

The GhostWriter approach to making content authoring suggestions to the user is novel, but it is inspired by Conversational CBR (CCBR) [1]. In CCBR, a typical case has a problem description, comprising of a free text description and a set of question-answer pairs, and a problem solution, comprising a sequence of actions. This is very similar to our case representation, described above, except, as already mentioned, our cases have no solution component.

Aha et al's generic CCBR algorithm [1] starts with the user entering a free text query. Then the following repeats until the user selects a case or no further cases or questions can be suggested to the user: the system retrieves and displays a set of cases that are similar to the user's query; from these cases, the system ranks and displays a set of important but currently unanswered questions; then the user inputs more free text or answers one of the questions.

Figure 1 shows the GhostWriter approach to making content authoring suggestions. Recall that we invoke this algorithm repeatedly as the user's content grows. Each time we invoke it, it does the following:

[3] We avoid the word "query", which is more common in CBR, since we have found it leads to confusion.

Algorithm 1 GhostWriter's content authoring suggestion algorithm

Inputs: CB: case base
nid: new item description
k_1, k_2, k_3: number of cases, features and values, resp.

$R \leftarrow [\,]$
$C \leftarrow rank\_cases(nid, CB, k_1)$
$F \leftarrow rank\_features(C, nid, k_2)$
for each $f_i \in F$, taken in decreasing order **do**
  $V_i \leftarrow rank\_values(f_i, C, k_3)$
  insert $\langle f_i, v \rangle$ onto the end of R for each $v \in V_i$ taken in decreasing order
end for
return R

- It initializes the result R to the empty list.
- It retrieves k_1 cases C from the case base CB, ranking them on their similarity to the user's new item description nid. In fact, we compute similarity between the free text descriptions, $text(nid)$ and $text(c)$ for each $c \in CB$. It may be worthwhile to use a diversity-enhancing algorithm, e.g. [9], for this retrieval.
- From the cases retrieved in the previous step C, we obtain up to k_2 features F. Candidates for inclusion in F are all features in each $c \in C$, after removing duplicates and any feature that is already among the features of the user's item description $fvs(nid)$, irrespective of that feature's value in $fvs(nid)$. There are many ways of ranking these candidates. At the moment we use the simplest approach: frequency of occurrence across the cases in C. We place in F the k_2 features that have the highest frequency of occurrence.
- For each of the features obtained in the previous step $f_i \in F$, we obtain up to k_3 values for that feature V_i. Candidates for inclusion in V_i are all values for that feature in each of the cases $c \in C$, after removing duplicates. Again there are many ways to rank these candidates. At the moment, we use the original ranking of the cases C. In other words, if $\langle f_i, v \rangle \in fvs(c)$ and $\langle f_i, v' \rangle \in fvs(c')$ and $c \in C$ has higher rank than $c' \in C$, then v has higher rank than v'.
- We return the ranked list of up to k_2 features, each with their ranked list of up to k_3 values, for display to the user in the suggestion pane.

When the user makes sufficient change to nid, possibly by incorporating suggestions from the suggestion pane, we run GhostWriter again to make fresh suggestions. This continues until the user is satisfied with her description and submits it to the waste exchange service database.

4 Discussion

Our first goal in this section is to discuss related work. Several researchers have investigated ways of proposing completions for incomplete phrases and sentences,

representative of which are [4, 6]. This kind of work tends to focus on data structures for representing phrases and sentences in a way that supports fast matching with phrase and sentence prefixes. Lamontagne and Lapalme use CBR for the more challenging task of generating email replies [5]. But their work, and the work on phrase and sentence completion, is concerned with making suggestions in situations where there are 'stock responses'. We would argue that GhostWriter's task is different in nature. Its goal is to prompt the user to write a more comprehensive description. On occasion, 'stock responses' may be relevant, and the user may click on a suggestion to include it directly in her content. But just as likely, she will not accept any of the phrases (*feature-values*) that we suggest to her. Nevertheless, we hope that she will be prompted to include a phrase of her own, inspired by the *features* that we suggest.

In its goal, Recio-García et al's Challenger 1.0 system is much more similar to our own work [8]. Their system supports the author of air incident reports. However, their texts are longer and their techniques are quite different from ours. They have no feature-value pairs and do not draw ideas from CCBR. Instead, they use standard text retrieval coupled with clustering of the results.

Our second goal in this section is to discuss objections to what we have done. It might be objected that a simpler approach is the use of forms. These forms could include fields inviting the author to provide much more information than normal (e.g. the colour, the price, the delivery terms, etc.). But we think there are problems with a form-based approach: forms can become quite long; there is the difficulty of anticipating what fields to include on the form, although this could be solved by using Feature Extraction from successful descriptions; a form-based approach assumes greater regularity in the descriptions than our approach assumes; and a form-based approach may result in less distinctive descriptions, when in fact the author's real goal is to make her descriptions stand out. Our approach, by contrast, prompts the user dynamically, based on the current description and the content of related successful descriptions (cases).

Our final goal in this section is to discuss the generality of our approach. We have made our presentation more concrete by giving a context, namely waste exchange services. But we believe that the same approach can be used in any 'classified ads' service, where cars, jobs, housing, dates, and many other things are advertised. In fact, we intend doing experiments with data that we have scraped from craigslist.[4] We are also interested to apply our approach to support the authors of reviews of products such as hotels and electronic goods. The content of these reviews is even less predictable than that of classified ads, so our approach may then be even more promising than an approach based on form-filling. In the domain of product reviews, other users can often indicate whether they found a review to be useful or not. This is what we would use as a measure of whether a description is successful. It implies that the case base becomes a fuzzy set, where descriptions have different degrees of membership depending on how useful people have found them to be.

[4] `http://www.craigslist.org`

5 Experimental Evaluation

Here, we report the results of a preliminary, off-line ablation study. The results are promising, but they do show that we need to use a different dataset and we do need to make some changes to our experimental methodology.

From an operational waste exchange service, we took a set of 73 descriptions of items available. Most waste exchange services, including the one we work with, do not retain successful descriptions. Therefore, unfortunately, these 73 descriptions, which we use as a case base in this experiment, are not restricted to successful descriptions. As mentioned already, they are rather short: 8 words on average, and 6 if we exclude stop-words.

We use a leave-one-out methodology. We temporarily remove a case from the case base and delete a random proportion of its words. We treat this as the user's *nid*; the ablation simulates an incomplete description. We supply this *nid* to GhostWriter. GhostWriter is run with $k_1 = 10$ (the number of cases it retrieves), $k_2 = 2$ (the number of features it suggests), and $k_3 = 2$ (the number of values it suggests for each feature). Hence it returns up to four suggestions (two values for two features). We randomly select one of the suggestions and add the suggested feature-value to the *nid*. We keep doing this until GhostWriter is unable to make further suggestions. We repeat this for each case in the case base, and we repeat the whole procedure five times to average out differences that result from random ablation.

After we add a suggested feature-value to the *nid*, we measure the similarity between the current state of the *nid* and the original case from which we created the *nid*. We compute similarity using the standard cosine measure. The results are shown in Figure 3.

On the y-axis is similarity; on the x-axis is the number of feature-values that we have added to the *nid*. There are different lines according to the starting amount of ablation. For example, one line records what happens when we form the *nid* by ablating 20% of the original case; another plot measures what happens when there is 40% ablation; and so on. The x-axis goes up to 6. But GhostWriter will not make 6 suggestions for every *nid*. For some *nid*s, GhostWriter may run out of suggestions much earlier: if the features of the retrieved cases C are all already present in the *nid*, then GhostWriter can make no fresh suggestions. The percentages alongside each data point record this information. For example, on the line for 0% ablation, we were able to add one feature-value pair to 85% of *nid*s; we were able to add two feature-value pairs to 63% of *nid*s; three to 38%; four to 25%; and so on.

In interpreting these results, the question is: when we add suggested feature-values to a *nid*, are we restoring some of the original content that we ablated earlier? If this is so, then the suggestions are useful ones. In judging this, we must compare with the line for 0% ablation. When there is 0% ablation, any feature-values we add inevitably reduce the similarity between the *nid* and the original case: we are adding content that was not originally there. Provided the gradient in the other lines is not as steep as the gradient in the 0% ablation line, then we know that the content that we are adding is at least partly restoring

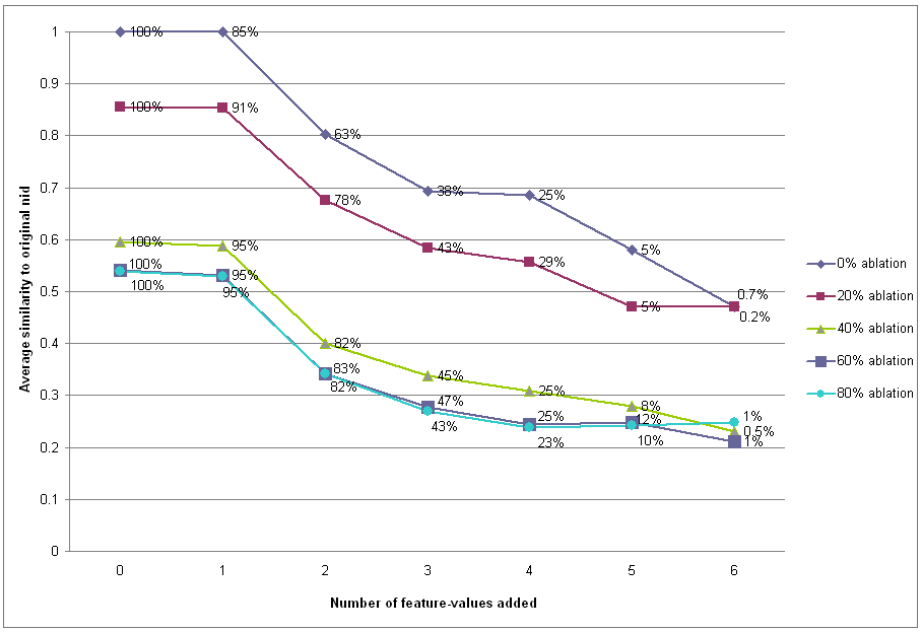


Fig. 3. Average similarity between the *nid* as we add suggested feature-values and the case from which it was created

ablated content. This does seem to be so: with 0% ablation similarity falls from from 1.0 to 0.47, i.e. by 0.53; for 20% ablation it falls from 0.86 to 0.47, i.e. by 0.39; for 40% ablation it falls by 0.37; for 60% it falls by 0.33; and for 80% it falls by 0.29. We believe this shows that the GhostWriter algorithm is promising.

The main lesson from this preliminary experiment, however, is that, going forward, we need to change the experimental set-up. We need a case base with more comprehensive descriptions, more akin to what we hope to find in a case base of successful descriptions. We also probably need to take item category into account so that when GhostWriter suggests content to someone describing a desk, it should only use feature-values that come from other descriptions of furniture, and not from descriptions of electrical appliances, for example. In the experiment at present, this restriction is not in place. At the moment also the ablation deletes a proportion of a case's words at random. It may be a fairer experiment to delete a proportion of a case's phrases (i.e. its existing feature-values) and to use a similarity measure that rewards GhostWriter the earlier it suggests the right kinds of features, even if the suggested feature-values do not match those in the original case.

6 Conclusions and Future Work

In this paper, we have argued that the Web contains experience in the form of successful descriptions, which we can treat as cases in making suggestions to the authors of new content. We have presented a concrete scenario, that of a waste exchange service, where this perspective can be useful. We have presented a novel algorithm, implemented in the GhostWriter system, for making these suggestions, inspired by work in Conversational CBR. And we have reported some promising preliminary results.

This is early-stage research, with many lines of future inquiry. In particular, we want to apply the idea in other domains, especially classified ads and product reviews. We want to try some of the many ways of learning the Feature Extraction rules, see, e.g., [3]. We want to investigate variants of the algorithm, where we use different ways of ranking the cases, features and feature-values. We mentioned, for example, the use of diversity-enhanced methods for retrieving the cases. We want to use a different dataset and make some changes to the methodology in our off-line ablation study. Finally, we want to carry out evaluations with real users.

References

1. David W. Aha, Leonard A. Breslow, and Héctor Muñoz-Avila. Conversational case-based reasoning. *Applied Intelligence*, 14:9–32, 2001.
2. Hans-Dieter Burkhard. Extending some concepts of CBR — Foundations of case retrieval nets. In M. Lens et al, editor, *Case-Based Reasoning Technology: From Foundations to Applications*, pages 17–50. Springer, 1998.
3. Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.
4. Korinna Grabski and Tobias Scheffer. Sentence completion. In M. Sanderson et al, editor, *Procs. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 433–439. ACM Press, 2004.
5. Luc Lamontagne and Guy Lapalme. Textual reuse for email response. In P. Funk and P. A. González Calero, editors, *Procs. of the 7th European Conference on Case-Based Reasoning*, LNCS 3155, pages 234–246. Springer-Verlag, 2004.
6. Anranb Nandi and H. V. Jagadish. Effective phrase prediction. In C. Koch et al, editor, *Procs. of the 33rd International Conference on Very Large Data Bases*, pages 219–230. ACM Press, 2007.
7. Enric Plaza. Semantics and experience in the future web. In K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, editors, *Procs. of the 9th European Conference on Case-Based Reasoning*, LNCS 5239, pages 44–58. Springer Verlag, 2008.
8. Juan A. Recio-García, Belén Díaz-Agudo, and Pedro A. González-Calero. Textual CBR in jCOLIBRI: From retrieval to reuse. In D. C. Wilson and D. Khemani, editors, *Procs. of the Workshop on Textual Case-Based Reasoning, 7th International Conference on Case-Based Reasoning*, pages 217–226, 2007.
9. B. Smyth and P. McClave. Similarity vs. diversity. In D. W. Aha and I. Watson, editors, *Procs. of the 4th International Conference on Case-Based Reasoning*, LNCS 2080, pages 347–361. Springer, 2001.

Many Cases Make Light Work for Visualization in Many Eyes

Jill Freyne and Barry Smyth *

CLARITY: Centre for Sensor Web Technologies
School of Computer Science and Informatics
University College Dublin,
Dublin, Ireland.
jill.freyne@csiro.au ***, barry.smyth@ucd.ie

Abstract. Visualization is among the most powerful of data analysis techniques and is readily available in standalone systems or components of everyday software packages. In recent years much work has been done to design and develop visualization systems with reduced entry and usage barriers in order to make visualization available to the masses. Here we describe a novel application of case-based reasoning techniques to help users visualize complex datasets. We exploit an online visualization service, Many Eyes, and explore how case-based representation of datasets including simple features such as size and content types can produce recommendations of visualization types to assist novice users in the selection of appropriate visualizations.

1 Introduction

Manipulating complex data is now a familiar part of our everyday lives. and to help us there are a wide range of data analysis tools, from general purpose spreadsheets to more complex statistical analysis packages. Visualization is among the most powerful of data analysis techniques and is readily available either as standalone systems or as key components of common software packages such as spreadsheets. Great strides have been made in bringing a wide range of visualization options to the masses. For example, Microsoft's Excel offers 11 different types of chart (bar, line, pie etc.) and a total of 73 basic variations on these charts. Apple's Numbers spreadsheet is similarly well equipped and even Google's free Spreadsheets programme offers access to about 25 different variations of 6 different chart types.

Surely all of this puts sophisticated visualization within reach of the average user? The problem, of course, is that the average user is not a visualization expert and producing the right sort of visualization for a given dataset is far from trivial. Previous work in the area of visualization recommendation includes research

*** This author is now with the CSIRO Tasmanian ICT Center, Hobart, Australia

* This work was supported by Science Foundation Ireland through grant No. 07/CE/I1147.

into articulated task-orientated systems [4], early data property based systems [9, 8], hybrid task and data based systems which examine both user intent and the data at hand [11, 2] and more recent work which aims to discover patterns in user behaviour in preparation of a dataset in order to predict visualization requirements [6]. This work returns to the early data property based research as we exploit case-based reasoning techniques to make visualization recommendations. We believe that case-based reasoning [1] is very well suited to providing useful assistance in this type of task and in this paper we describe a case-based recommender system that is designed to do just this.

The starting point for this work is a Web based "social" visualization platform called *Many Eyes*. In brief, Many Eyes is a web-based visualization platform, developed at IBM Research, that allows users to upload datasets, chose from a wide variety of visualizations, and make the results available to others. To date over 33,000 datasets have been uploaded by nearly 8,000 users, creating 24,000 different visualizations. These "visualization experiences" encode important visualization knowledge in terms of the decisions taken by a user about how to visually represent a given dataset. In this way each visualization can be viewed as a *case*, with features of the dataset providing the *case specification* and the resulting visualization configuration providing the *case solution*. In this paper we propose that these visualization cases can be reused in the context of a new dataset, to make suggestions about appropriate visualizations.

2 Many Eyes

Many Eyes (http://manyeyes.alphaworks.ibm.com/manyeyes/)is a browser based visualization tool designed specifically to make sophisticated visualization easily accessible to web users but also to make the process of visualization a social one, where people can come together to discover and share what they see in publicly visualized data [10]. Many Eyes differs from other visualization software in that all human contributed data (datasets, visualizations and comments) are publicly accessible. As Many Eyes is an experimental system the visualization options vary from the ordinary (histograms and pie charts) to experimental (word trees and matrix charts) and users have little assistance other than small graphics and a short textual description when choosing a visualization for their dataset.

Many Eyes has four core processes, data upload, visualization creation and social discovery and discussion. Due to the system's open access policy each process can be undertaken independently, for example any user can create a visualization on any dataset contributed or comment on any dataset or visualization created. This platform creates an ideal online environment for collaboration, cooperation and communication around a set of data and its visualizations.

2.1 Data Upload

Raw data uploaded to ManyEyes can be freeform text or tab-delimited data. In an effort to keep entry barriers to using Many Eyes low the system has the

capability to recognise and process tab-delimited data and makes assumptions as to the type of data, textual or numeric, contained in each column. Uploaders are required to provide textual information such as a title for the dataset and encouraged to provide other relevant information such as the source and description of the data. The system appends further metadata including the creation date and creator's details before making it public.

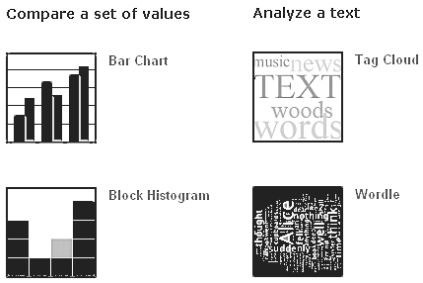


Fig. 1. Selection of category and visualization types in Many Eyes.

2.2 Visualization Creation

Many Eyes has 6 categories of visualizations, containing a total of 16 visualization types, some of which can be further sub-categorized. Sample category titles include "track rises and falls over time", "analyze text" and "seeing the world" amongst others. Each subcategory or chart type in the option list is accompanied by 1-2 explanatory sentences to guide the user in their decisions. Further information relating to each chart type describing its strengths and weaknesses and its appropriateness for varying data types is available but users must navigate away from their current process in order to locate this information [5]. Understandably not all of the visualization types in Many Eyes are suitable for displaying both unstructured text and tabular data. Six of the 33 visualization types have been used for text data visualization and 31 of the visualization types have been used to chart tabular data. On selection of a chart type Many Eyes automatically generates a visualization, assigning chart parameters such as mapping axis to columns etc. when only one suitable option is available and asking for user confirmation when multiple options exist.

2.3 Sharing & Discovery

Many Eyes was designed to enable a new kind of social data analysis. As a collaborative visualization tool it provides users with a platform for discovery, sharing and discussion around people, datasets and visualizations. Each member has a profile page containing personal details, watchlists, topic hubs and details of activity on the site.

In the context of the "knowledge worker", the availability of datasets and associated visualizations provides a rich environment from which non-expert visualizers can learn. Novice or inexperienced users may discover datasets similar to theirs in order to decide how to effectively uncover the messages contained in their raw data. Many Eyes provides various methods for browsing and searching its repository of data and visualization pairs. We believe that case-based reasoning techniques could automate the process of discovering suitable visualizations for contributed datasets. By creating cases which represent simple dataset features such as the presence of numeric and textual content as well as the size of the dataset we aim to capture the expertise demonstrated by expert visualizers to assist users in selecting the best chart for their data.

2.4 The Dataset

The dataset used for this work represents approximately 21 months of usage of Many Eyes from January 2007 and covers 33,656 separate dataset uploads and 24,166 unique visualizations from 15,888 registered users. It is worth noting that only about 43% of uploaded datasets are actually successfully visualized. In turn, just over 60% of users who uploaded datasets went on to store a visualization. This is surely a telling comment on the challenges faced by users when it comes to choosing and configuring suitable visualizations of their data. It seems that in many cases users just did not have the visualization experience (or the time) to select from the many different charting options and configurations that are offered. In general there are two basic types of dataset in Many Eyes. *Text* datasets are a *bag-of-word* type datasets whereas *tabular* datasets are column-based datasets, using a mixture of data types.

3 A Case-Based Recommender for Many Eyes

The Many Eyes repository of datasets and visualizations is more than a simple collection of raw datasets and charts. It is reasonable to assume that each combination of dataset and chart is the result of a deliberate visualization exercise. As such it encodes some latent decision making process by which the dataset 'owner' came to settle on a particular visualization option which addressed his/her particular objectives. Of course such objectives may extend beyond the simple need to visually summarise a particular dataset. In many cases it is reasonable to assume, for example, that the user will have considered the aesthetics of particular visualization choices, adding an extra dimension to their decision making.

In short then, the combination of dataset and visualization encodes an *experience*. It is a *case* in the classical view of case-based reasoning. And in this paper we propose to take advantage of this perspective in order to develop a case-based recommender system that is capable of suggesting good visualizations to users based on the characteristics of their particular dataset. This will be of particular interest and benefit to less experienced Many Eyes users, who, in the past, have failed to produce visualizations for their datasets. Of course the recommendations may also be of interest to more experienced users by highlighting alternative visualization options that they may be less familiar with.

3.1 Case Representation

We will begin by assuming each case represents a single visualization of a single dataset. Thus, each case, c_i is made up of a dataset component, d_i and a visualization component, v_i as shown in Eq. 1. In fact there is additional information that is sometimes available such as the rating associated with a particular visualization, r_i. In case-based reasoning parlance the dataset component corresponds to the *specification* part of a case, the visualization component corresponds to the *solution* part of a case, and the rating component can be viewed as the *outcome* of the solution. In this paper we will focus on the specification and solution side of visualizations cases, largely because the Many Eyes dataset is very sparse when it comes to the availability of ratings data.

$$c_i = \{d_i, v_i\} \tag{1}$$

The representation of the visualization component is straightforward, at least for this paper, since each case solution is just the type of visualization used, $chart(v_i)$, because we are focusing at the moment on recommending a particular visualization type when faced with a new dataset. Going forward, one can envisage more complex solution features if we wish to reason about particular features of the visualization, such as the axis placement, label usage etc.

Each dataset is characterised by a set of simple features that relate to the type of data contained in the dataset. We distinguish between text and tabular datasets by extracting different features for each. For example, for text datasets we extract features that include the total number of terms ($terms$), the number of unique terms $unique$ as part of the specification; see Eq. 2. For tabular datasets we can extract features such as the number of textual columns, col_{txt}, the number of numeric columns, col_{num} and the number of data points, $rows$. In this way each case is represented as a feature-based dataset and solution as in Eq. 3.

$$c_i = \{terms, unique\}, chart(v_i) \tag{2}$$

$$c_i = \{col_{txt}, col_{num} rows\}, chart(v_i) \tag{3}$$

3.2 Similarity and Retrieval

Given a new target case c_T (made up of a particular dataset) the task of the recommender system is to locate a set of similar cases that can be used as a source of visualizations. For the purpose of this paper we concentrate on some tried and tested similarity techniques using the above case representations. For example, to compute the similarity between tabular dataset cases we use the similarity metric shown in Eq. 4 which calculates the relative difference between the number of textual and numeric columns and rows between the target dataset and the case dataset; in this instance uniform weighting is used and so $w_f = 0.33$.

$$sim(c_T, c_i) = 1 - \sum_{f \epsilon \{col_{txt}, col_{num}, rows\}} w_f \bullet \frac{|c_T(f) - c_i(f)|}{max(c_T(f), c_i(f))} \tag{4}$$

A similar approach to similarity assessment is used for the text based dataset, by comparing the datasets by the total number of terms and total unique terms. While these similarity techniques are extremely simple, they provide a useful starting point for this work. In the evaluation section we will demonstrate that even these simple techniques work well when it comes to driving high quality recommendations, while at the same time leaving a number of options open for more sophisticated similarity techniques as part of future work. Thus, given a target case, c_T, we can use the above similarity techniques to produce a ranked list of n similar cases as the basis for recommendation.

3.3 Generating Recommendations

Each of the n cases retrieved will be associated with a single visualization. The same visualization type may occur in more than one case and so we can identify a set of k different visualization types from these n cases. We need a way to rank these visualizations so that those that are associated with more similar cases are preferred over those that are associated with fewer, less similar cases. To achieve this Eq. 5 scores each of the n visualizations, v_i, as the sum of the similarity scores associated with the retrieved parent cases; $chart(v_i, c_j) = 1$ if v_i is the chart used in c_j and is 0 otherwise. The result is a ranked list of visualization recommendations, $v_1, ..., v_k$ in descending order of their aggregate similarity scores as per Eq. 5.

$$score(v_i, c_T, c_1, ..., c_n) = \sum_{\forall j=1...n} sim(c_T, c_j) \bullet chart(v_i, c_j) \tag{5}$$

4 Evaluation

As mentioned we believe that the low visualization rate of datasets is at least in part due to the confusion of choice that faced novice first-time uploaders. Our hypothesis is that even a simple form of case-based recommendation will help to improve the visualization rate by making proactive suggestions to the user. In this section we will describe the results of a recent large-scale, off-line evaluation using the Many Eyes dataset.

4.1 Set-up

The core Many Eyes dataset was transformed into a set of 22,935 visualization cases covering 14,582 different unique datasets and 33 visualization types. These cases included 6800 text cases and 16135 tabular cases. For the purpose of this evaluation we are interested in understanding the extent to which our simple CBR strategy can produce useful visualizations, compared with a number of benchmark strategies, which differ in terms of how cases are selected or recommendations are produced. The different techniques are summarised as:

1. *CBR* - the basic CBR approach described above is used to produce a ranked list of the top k visualizations from a set of n similar cases
2. *Popular* - this strategy simply recommends the k most popular visualizations (globally) in Many Eyes.
3. *Exact* - this is a hybrid recommender strategy which identifies a pool of similar cases like the CBR approach but only identifies cases which exactly match the target features. It then calculates the popularity of each visualization type for this pool. Specifically in the case of tabular data the set of similar cases identified match the number of text and numeric columns in the target case and in the text based datasets the word count of similar cases is within a close defined range of the word count of the target dataset.
4. *PopularContext* - similar to *Popular* but it treats textual and tabular visualization types separately.
5. *Random* - recommend a set of k random visualizations.

Obviously the CBR and $Exact$ strategies provide a form of local recommendation that is based on the particular features of the target dataset, whereas the remainder provide simpler global recommendation strategies.

4.2 Methodology

Our evaluation takes the form of a standard leave-one-out test. For each target case, c_T, we use its specification features to represent a new dataset and generate a set of k visualizations using each of 5 recommendation strategies; note that the k is based on the number of unique visualizations retrieved by the CBR strategy. There are two factors to consider when evaluating the quality of the resulting recommendations. One is to look at how often the target visualization is present in the set of k recommendations; so an *accuracy* of 60% means that the target visualization is present in 60% of the recommendation sets of size k. Another option is to look at the average position of the target visualization in the recommendation lists. And there are two ways to do this. One is to focus on those recommendation lists that do have the correct target visualization and then compute the average position of the target visualization in the final recommendation list. This so-called *average position* approach ignores recommendation lists that do not contain the correct visualization though, and therefore benefits the less accurate strategies. As an alternative we can compute a position value across all recommendation lists by assigning a $k + 1$ penalty to those lists that do not contain the target visualization *adjusted position*. This is a conservative penalty because it assumes that the correct visualization is actually in position $k + 1$, which may not be, but it serves to at least remove some of the bias associated with *average position*.

4.3 Results

Recommendation Accuracy Fig. 2(a)-(b) show the accuracy results separately for the textual and tabular cases. These results clearly support the use

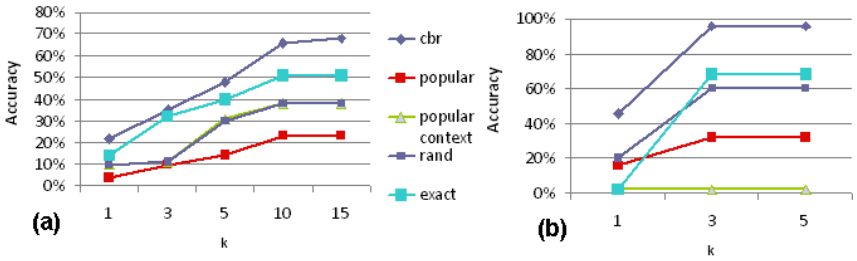


Fig. 2. Accuracy of predicted visualization types (a) tabular and (b) textual.

of the CBR recommendation strategy. Overall CBR is seen to outperform all other techniques with particularly impressive results for the CBR technique in the easier textual case recommendation scenario. There is also a very consistent benefit associated with the similarity-based technique used by CBR compared with the simpler matching used by $Exact$, with the former delivering relative improvements of 25%-50% across a wide range of k values.

Recommendation Position The results of the positional analysis of the recommendation techniques are presented in Fig. 3 (a)-(b). In terms of the average position statistic the local recommendation techniques such as CBR and $Exact$ are delivering improved performance compared to the global benchmarks, although there are a number of anomalies. For example in Fig 3(a), at $k = 3$, we see that CBR delivers its correct recommendations with an average position of 1.5. However, the popularity-based techniques achieve a better average position of just over 1. But remember, at this setting CBR is recommending a correct visualization among its top 3 recommendations more than 20% of the time versus 5% of the time with popularity-based approaches. By introducing a positional penalty we find that the local techniques do consistently better than all other benchmarks; see Fig. 3(c)-(d).

4.4 Summary

Even a relatively simple approach to case reuse has delivered useful results which may make a difference to Many Eyes users in practice. In each case we have found the case-based approach to outperform all of the other benchmarks that were tried, consistently producing more accurate recommendations nearer to the top of the recommendation list. Of course these findings need to be validated. They may be based on real-user data but they have not been tested on live users in the field. Nevertheless with these findings we can be optimistic about the prospect of success in such a future trial.

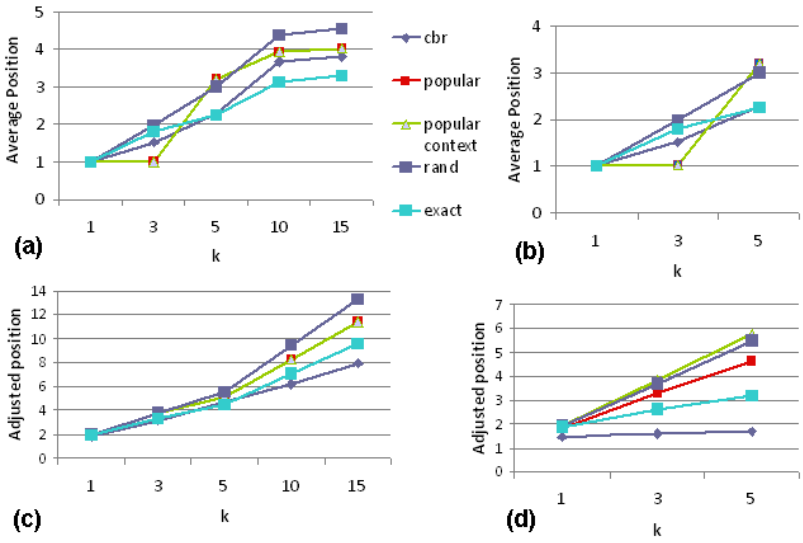


Fig. 3. Average position of the target visualizations (a) tabular, (b) textual and adjusted position of the target visualizations:(c) tabular, (d) textual.

5 Conclusions

The objective of this work is to help users of a Web based visualization system to produce better visualizations by recommending visualizations that have been previously used for datasets that are similar to their own. To that end we have started with a very simple case recommendation technique, but this has performed very well in practice, significantly outperforming a number of benchmarks. However, there remains plenty of room for improvement and as future work a number of obvious next steps present themselves.

More case features. For sure, there is ample opportunity to improve the case representation and the similarity techniques used. In this work we have used a very simple case representation for the purpose of exploring the potential of CBR in this domain. More representative cases would encompass more sophisticated details pertaining to datasets including in the case of numeric columns the features that reflect the maximum, minimum, average, and standard deviations of the columns ($min_i, max_i, avg_i, dev_i$) and for string columns we will extract the terms themselves ($t_1, ..., t_{terms}$). For all datasets we can extract a bag-of-words textual description derived from any metadata associated with the dataset, *desc* (e.g., column headings, title, source etc). Also example, recognising that a particular data field is a date or a currency can help to significantly improve the matching. Incorporating some notion of semantics into the representation and similarity computation should be possible. *Comparison to other Classification*

Techniques. In this work we have compared our simple CBR technique to very simple alternative measures. With work into creating sophisticated CBR resprresentations planned we can also compare the performance of the CBR method with other classification approaches such as Naive Bayes, decision trees or neural networks.

Introducing Adaptation. There is considerable scope for adaptation in this domain since recommending a visualization type is really just one part of a larger decision support problem. Users will benefit greatly from configuration support when it comes to actually using a particular visualization. This includes deciding which fields are associated with which axes, scale settings, etc. and these all provide opportunities for post-retrieval adaptation.

Ratings & Provenance. Many Eyes maintins rating information and information about the creator of the particular visualization. In recent years there has been new work in the area of *provenance* [7] and *reputation* [3] that could be used to greatly improve the recommendation algorithms by harnessing information about the source of a case and the reputation of the creator.

References

1. Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39 – 59, 1994.
2. Elisabeth André and Thomas Rist. The design of illustrated documents as a planning task. pages 94–116, 1993.
3. Peter Briggs and Barry Smyth. Provenance, trust, and sharing in peer-to-peer case- based web search. In *ECCBR*, pages 89–103, 2008.
4. Stephen M. Casner. Task-analytic approach to the automated design of graphic presentations. *ACM Trans. Graph.*, 10(2):111–151, 1991.
5. Catalina M Danis, , Fernanda B. Viegas, Martin Wattenberg, and Jesse Kriss. Your place or mine?: visualization as a community component. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 275–284, New York, NY, USA, 2008. ACM.
6. David Gotz and Zhen Wen. Behavior-driven visualization recommendation. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 315–324, New York, NY, USA, 2009. ACM.
7. David B. Leake and Matthew Whitehead. Case provenance: The value of remembering case sources. In *ICCBR*, pages 194–208, 2007.
8. Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2):110–141, 1986.
9. Steven F. Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. Interactive graphic design using automatic presentation knowledge. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 112–117, New York, NY, USA, 1994. ACM.
10. Fernanda B. Vigas, Martin Wattenberg, Frank van Ham, Jesse Kriss, and Matt McKeon. Manyeyes: A site for visualization at internet scale. In *IEEE Transactions on Visualization and Computer Graphics*, volume 13(6), pages 1121–1128, 2008.
11. Michelle X. Zhou and Min Chen. Automated generation of graphic sketches by example. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 65–74. Morgan Kaufmann, 2003.

Extraction of Adaptation Knowledge from Internet Communities

Norman Ihle, Alexandre Hanft and Klaus-Dieter Althoff

Intelligent Information Systems Lab
University of Hildesheim, Germany
{ihle, hanft, althoff}@iis.uni-hildesheim.de

Abstract. Acquiring knowledge for adaptation in CBR is an demanding task. This paper describes an approach to make user experiences from an Internet community available for the adaptation. We worked in the cooking domain, where a huge number of Internet users share recipes, opinions on them and experiences with them. Because this is often expressed in informal language, in our approach we did not semantically analyze those posts, but used our already existing knowledge model to find relevant information. We classified the comments to make the extracted and classified items usable as adaptation knowledge. The first results seem promising.

1 Introduction

Adaptation is a central part of the case-based reasoning process model [1]. A good adaptation is of very high importance if the case base is restricted to a small number of cases or if the variation of the problems to be solved is very high. Adaptation has not been the most current topic in recent years of CBR research [2], but in the last year the research effort increased again [3–5]. Often adaptation means justifying values in a bounded range [3] and is done via rules created and maintained by a domain knowledge or system developer [6].

Knowledge acquisition for adaptation (Adaptation Knowledge acquisition: AKA) is a cost intensive task since it is highly domain dependent and the hard-to-get experts are needed for acquiring and maintaining the necessary knowledge. To solve this problem, research on automated adaptation knowledge acquisition has been done, but mainly focused on automated AKA from cases in the case base [7–9].

Besides the case base, the Internet and the Web 2.0 with its user-generated content are a large source of any kind of knowledge and experience. Following the Web 2.0 paradigm of user-interaction people provide their experience, opinions and advice on any kind of topic. Although the people are not necessarily experts in the domain, the hope is that the mass of users will correct mistakes as practiced for example in the Wikipedia project.

In this paper we present an approach to make knowledge from Internet communities accessible as adaptation knowledge using the domain model that usually exists in structured CBR applications [10]. In the first part of the paper the

adaptation background inside CBR is introduced before we describe the domain and the existing application we worked with in our approach. After a short introduction of the tool we used, we will explain the approach in detail before we close with our ideas for the evaluation, related work and an outlook.

2 The Cooking Domain

For most people cooking is "everyday" knowledge. Almost everybody has encountered the problem of preparing a meal with a restricted amount of ingredients. This explains why a huge number of people are willing to share their recipes as well as their experience with the preparation of these recipes. Cooking communities on the Internet offer a platform for this. They provide the possibility to share recipes and also the chance to review and comment them. Usually they also offer additional information like cooking hints, background information on groceries or preparation methods. Besides the fact of the existence of active cooking communities, we chose the cooking domain for the investigation on adaptation knowledge because it has some advantages compared to other areas of interest discussed in communities like computer problems for example. First of all, it is relatively easy to describe the (near) complete context of preparing a meal. Hence, it is possible to reconstruct the experience of others by preparing the meal and trying the adaptation suggestions oneself. The context can be described according to the following characteristics:

1. all ingredients can be listed with exact amount and quality
2. ingredients can be obtained in standardized quantities and in comparable quality
3. kitchen machines and tools are available in a standardized manner
4. (in case of a failure) the preparation of a meal can start all over again every time from the same initial situation (except that we have more experience in cooking after each failure).

The latter one is not given in many other domains, for example setting up a computer costs much time and a certain installation situation cannot always be restored. Additionally, cooking and the adaptation of recipes is (some basic understanding presumed) relatively uncritical. In contrast to medical applications it does not endanger human health, except for some rare meals like fugu (pufferfish). The costs of a failure are low. It is mostly covered by the price of the ingredients plus the preparation time. Cooking is also an appropriate application domain for adaptation, because cooking mastery depends on the variation and creativity, not only on following strictly preparation advices for a meal [11].

3 CookIIS and Adaptation in CookIIS

CookIIS [12] is a CBR-based recipe search engine that competes in the Computer Cooking Contest (CCC). When the user provides possible ingredients, it searches

for suitable recipes in a case base. Doing that it considers ingredients the user does not want or cannot use because of a certain diet. If recipes with unwanted ingredients are retrieved, CookIIS offers adaptation suggestions to replace these ingredients. According to the CCC the set of recipes is restricted. Besides the retrieval and adaptation of recipes CookIIS also offers recipes for a complete three course menu from the given ingredients (and maybe some additional ones).

CookIIS is using a very detailed domain model which is described in [12]. It was created using the empolis:Information Access Suite (e:IAS) [13], which offers a knowledge modeling tool called Creator and with the Knowledge Server a component to build client-server based applications. It also provides a rule engine for the completion of cases and queries and for the adaptation of cases after the retrieval. Some more technical details are described in [14].

3.1 Adaptation with the empolis:Information Access Suite

As stated above, the e:IAS offers the possibility to use completion rules which are executed before building the case index or before the retrieval to extend cases or queries with meta-information and adaptation rules, which are executed after the retrieval, to modify retrieved cases. The Creator offers a rule editor to model completion and adaptation rules with an own syntax. The rules follow the classic IF ... THEN ... schema. They have read and write access to all modeled objects and their values, but only adaptation rules have access to the retrieved cases since they are executed after the retrieval. A large amount of predefined functions help to manipulate single values. Both rule types use the same e:IAS specific syntax, which after compilation is stored in the format of the Orenge Rule Markup Language (ORML), an XML-language.

3.2 Case Representation and Similarity for Adaptation in CookIIS

The case representation is based on structured CBR. 11 classes of ingredients (e.g. Vegetables, Fruit, etc.) plus some classes for additional information (e.g. Type of Meal, Tools, etc.) are modeled, which represent about 2000 concepts of the cooking domain. A case consists of 11 attributes, one for each possible ingredient class. Each attribute of ingredients can have multiple values per recipe (sets). Most concepts of the different classes are ordered in specific taxonomies. These and some custom similarity measures are used to compute the similarity between the query and the cases. Thereby the different attributes have different weights corresponding to their importance for a meal. Additional meta-information like the type of cuisine of a recipe is established during the indexing process of the recipes and also stored in the case.

The approach for adaptation that was first realized in CookIIS is to replace forbidden ingredients (according to a certain diet oder explicitly unwanted) with some similar ingredients of the same class. While executing a query unwanted (forbidden) ingredients are collected in extra attributes. Besides the explicit exclusion, four different methods can be distinguished to handle dietary practices, where more conditions have to be considered [14]. One of those methods is the

same approach as above: ingredients that have to be avoided due to a diet are replaced by similar ones.

Adaptation rules take these forbidden ingredients and check if at least one of them is an ingredient used in the retrieved case (recipe). Then, making use of the taxonomies and a similarity-aware set-function offered by the rule engine, the most similar ingredients to the unwanted one are retrieved and offered as replacement. The functions of the rules are described in detail in [14]. If no similar ingredient is found that can be used for following the diet, then the suggestion is to just omit that ingredient.

Shortcomings of the Existing Adaptation Approach Since the used adaptation approach makes use of the modeled taxonomies the results are often inappropriate. The method returns sibling concepts to the unwanted one as well as parent and child concepts. Only the siblings are the ones who are interesting for adaptation, but the others cannot be avoided with the provided rule functions. Also the number of siblings is often too high. For one unwanted ingredient one or two ingredients as an adaptation suggestion would be preferable. A detailed analysis of the problems with the adaptation and the ways to handle it with the e:IAS Rule mechanism is described in [15].

4 CommunityCook: A System to Extract Adaptation Knowledge from Cooking Communities

In this chapter we will present our approach to extracting adaptation knowledge from a German cooking community. For this purpose we use our existing knowledge model from the CookIIS application and the TextMiner provided by e:IAS to extract ingredients from recipes and comments on those recipes and classify them. One of the classes can then be used as adaptation knowledge.

4.1 Idea behind the Approach

Our idea is to make knowledge from a cooking community accessible for our CookIIS application to have better adaptation suggestions in case a recipe contains an unwanted or forbidden ingredient. We were especially interested in comments that people posted in reply to provided recipes. In these comments users express their opinion on the recipe, before as well as after cooking it. They write about their experience with the preparation process and also tell what they changed while preparing the recipe. Thereby they express their personal adaptation of the recipe and frequently give reasons for this. Since this is written down in natural language text, often using informal language, we had the idea not to semantically analyze what people said, but to just find the occurrences of ingredients in the comment texts and then compare them to the ingredients mentioned in the actual recipe. We propose to classify them into three classes, depending on whether the ingredients mentioned in a comment appear in the recipe or not. The classification idea is described in the following sections.

4.2 Analysis of Example Cooking Communities

In Germany, *chefkoch.de*[1] is a well known cooking community with a large number of active users. So far, over 131'000 recipes have been provided by the users with an even larger amount of comments on them. The users also have the possibility to vote on the recipes, send them to a friend per email or even add pictures of their preparation. Besides the recipes, chefkoch.de features an open discussion board for all kinds of topics on cooking with more than 7.8 million contributions. Their English partner site *cooksunited.co.uk*[2] is unfortunately much smaller with only about 2200 recipes and 3500 posts.

But with *allrecipes.com*[3] a big platform with a huge amount of recipes and over 2.4 millions reviews is available in English. It has representable big localizations for the United States, Canada, the United Kingdom, Germany, France and others. Allrecipes.com explicitly provides variants of an existing recipe. Hence it also seems to be also a good source candidate. Another large cooking German community is *kochbar.de*[4] with over 160'000 recipes. Besides these large communities a number of smaller communities exist in the Web with more or less similar content. For our approach we decided to use a large German community since the recipes and the corresponding comments are presented on one page with a standardized HTML-code template, which makes it easier to crawl the site and extract relevant information items.

4.3 Extraction of Information Items from a Cooking Community

From a large German community we collected about 70'000 recipes with more than 280'000 comments by crawling the site. This way we got one HTML source-code page for each recipe with the corresponding comments. From this source code we extracted the relevant information entities using customized HTML-filters which we built using the HTML Parser tool[5]. For the recipes these entities were primarily the recipe title, needed ingredients and the preparation instructions, but also some additional information on the preparation of the recipe (e.g. estimated time for the preparation, difficulty of the preparation, etc.) and some usage statistics (e.g. a user rating, number of times the recipe has been viewed, stored or printed, etc.). If users commented on the recipe, we extracted the text of the comment, checked if the comment was an answer to another comment and if the comment has been marked as helpful or not. We also remembered the recipe ID of the related recipe. All this information we stored in a database to have an efficient access to it.

In the next step we used the e:IAS and indexed all recipes and all comments into two different case bases using a slightly extended CookIIS knowledge model. One case base consists of the recipes and one of the comments. For each recipe

[1] http://www.chefkoch.de, last visited 2009-04-22
[2] http://www.cooksunited.co.uk, last visited 2009-04-23
[3] http://allrecipes.com, last visited 2009-04-23
[4] http://www.kochbar.de, last visited 2009-05-22
[5] http://htmlparser.sourceforge.net, last visited 2009-04-18

and each comment we extracted the mentioned ingredients and stored them in the case using our knowledge model and the e:IAS TextMiner during the indexing process. Since our knowledge model is bilingual (English and German) we were also able to translate the originally German ingredient names from the comment text into English terms during this process and this way had the same terms in the case bases that we use in our CookIIS application.

4.4 Classification of Ingredients

Having built up the two case bases we first retrieved a recipe and then all of the comments belonging to the recipe and compared the ingredients of the recipe with the ingredients mentioned in the comments. We then classified the ingredients mentioned in the comments into the following three categories:

- *New*: ingredients that are mentioned in the comment, but not mentioned in the recipe
- *Old*: ingredients that are mentioned in the comment as well as in the recipe
- *OldAndNew*: two or more ingredients of one class of our knowledge model, of which at least one was mentioned in the recipe and in the comment and at least one other one was only mentioned in the comment, but not in the recipe

We interpret the classification as follows:

- *New*: New ingredients are a variation of the recipe. A new ingredient (for example a spice or an herb) somehow changes the recipe in taste or is a tryout of something different or new.
- *Old*: If an ingredient of a recipe is mentioned in the comment it means that this ingredient is especially liked or disliked (for example the taste of it), that a bigger or smaller amount of this ingredient has been used (or even left out), or it is a question about this ingredient.
- *OldAndNew*: This is either an adaptation (e.g. instead of milk I took cream) or an explanation/specialization (e.g. Gouda is a semi-firm cheese).

For the adaptation the last class is the interesting one. For each ingredient classified as *OldAndNew* we also stored whether it is the new or the old one. We tried to distinguish between adaptation and specialization by looking for hints in the original comment text and by using the taxonomies of our knowledge model. Therefore we tried to find terms in the comment during the text-mining process that confirm if it is an adaptation (e.g. terms like: instead of, alternative, replaced with, ...) and stored those terms in the corresponding case. Additionally we looked in the taxonomy of the ingredient class whether the one ingredient is a child of the other (or the other way around). If an ingredient is a child of the other we interpreted this as specialization or explanation, because one ingredient is a more general concept than the other. This way we could avoid adaptations like: "instead of semi-firm cheese take Gouda".

For the classes *Old* and *New*, which we consider as variations of the recipe, we also tried to find terms in the comment that closer describe the function of

the mentioned ingredient. For example, if an ingredient was classified as *Old*, we looked for terms like 'more', 'less' or 'left out'. If the ingredient of the comment is of the supplement class of our CookIIS knowledge model, and the recipe did not contain any supplement, then we took this as a suggestion for a supplement (e.g. bread for a soup recipe).

For each classified ingredient we assigned a specific score, which depends on the following factors:

- the number of ingredients found in the comment text
- whether the comment was marked as helpful or not
- whether a term was found that indicates the classification assigned or not
- whether a term was found that indicates a different classification or not

After assigning the score we aggregated our classification results. We did this in two steps: First we aggregated all classified ingredients of all comments belonging to one recipe. Thereby we counted the number of the same classifications in different comments and added up the score of the same classifications. Then we aggregated all classifications without regarding the recipe they belong to. This way we could select the most common classifications out of all classifications. Since we are using a CBR tool and have cases, we also checked if similar recipes have the same ingredients with the same classification mentioned in the comments. We did this for each recipe first with a similarity of at least 0.9, then with a similarity of 0.8. If many of the same classified ingredients exist in similar recipes, this supports our results.

	id integer	ingr_class text	oldingr1 text	oldingr2 text	newingr1 text	newingr2 text	score double precis	specification text	card_recipes integer
1	52	Comment_Ingredient_Milk	cream		milk		79.9583333333	adaptation	128
2	63	Comment_Ingredient_Milk	milk		cream		48.2	adaptation	78
3	254	Comment_Ingredient_Supplement	potatoes		sauce		36.1333333333	adaptation	61
4	238	Comment_Ingredient_Supplement	potatoes		broth		31.025	adaptation	54
5	386	Comment_Ingredient_OilAndfat	margarine		butter		29.4166666666	adaptation	46
6	128	Comment_Ingredient_Meat	bacon		ham		28.175	adaptation	45
7	20	Comment_Ingredient_Supplement	noodle		sauce		27.8166666666	adaptation	48
8	127	Comment_Ingredient_OilAndfat	butter		olive oil		27.0333333333	adaptation	43
9	393	Comment_Ingredient_OilAndfat	butter		margarine		25.55	adaptation	41
10	39	Comment_Ingredient_Vegetable	onion		green onion		21.55	adaptation	34
11	230	Comment_Ingredient_Milk	cream		yogurt		20.35	adaptation	31
12	332	Comment_Ingredient_Milk	creme fraiche		smetana		20.15	adaptation	33
13	1072	Comment_Ingredient_SpiceAndHerb	salt		pepper		18.55	adaptation	32
14	160	Comment_Ingredient_Supplement	rice		broth		17.025	adaptation	29
15	21	Comment_Ingredient_Milk	smetana		sour cream		16	adaptation	25
16	785	Comment_Ingredient_Vegetable	onion		paprika pepper		14.725	adaptation	26
17	57	Comment_Ingredient_Milk	cheese		cream		14.525	adaptation	25
18	60	Comment_Ingredient_Vegetable	onion		leek		14.25	adaptation	23
19	64	Comment_Ingredient_Milk	cream		coconut milk		14.1916666666	adaptation	22
20	73	Comment_Ingredient_Drinks	rum		amaretto		13.9	adaptation	22
21	660	Comment_Ingredient_Milk	cream		cheese		13.85	adaptation	24
22	98	Comment_Ingredient_Meat	ham		bacon		13.85	adaptation	22
23	424	Comment_Ingredient_Milk	yogurt		cream		13.775	adaptation	23
24	249	Comment_Ingredient_Meat	ham		salami		13.65	adaptation	21
25	385	Comment_Ingredient_Minor	honey		maple syrup		13.125	adaptation	21

Fig. 1. Some suggestions for adaptation

4.5 Usage as Adaptation Knowledge

OldAndNew-classified ingredients can be used to generate adaptation suggestions. This can be done in two different ways: independent from the recipe or with regard to the recipe. Considering the fist way, we look in the database table for the ingredient to adapt and use the result where the ingredient that needs to be adapted is categorized as old and appears in the most recipes or has the highest score. It is possible to retrieve two or more adaptation suggestions to be more manifold. Using this approach we got more than 6200 different adaptation suggestions of which we only used the most common (regarding the number of appearances in the comments and the score) per ingredient. Figure 1 shows some of these suggestions, e.g. in the first line a suggestion to replace cream with milk which appears in comments to 128 different recipes.

We integrated this approach into our CookIIS application: at first we look for two adaptation suggestions from CommunityCook. If no suggestions are provided, the set of more general adaptation rules (see section 3.2) determine adaptation suggestions.

5 Evaluation of the Results

A first look at the results of the most common adaptation suggestions is promising. Only the ingredient class "supplement" reveals problems which are due to the fact that too many different ingredients are integrated into this class. This can be changed by further improving the modeling.

A complete evaluation still has to be done. For that we want to follow two different approaches. At first we want to check if our classification and the interpretation correspond to the intentions written in the original comments. This can be done manually by comparing the classification results and their interpretation to the original comments. The second evaluation will be done on the results of the overall aggregated adaptation suggestions. We want to find out if adaptation suggestions with a high score are good adaptation suggestions for any kind of recipe. This is more difficult since we need domain expertise for this. Our idea is to take a representative number of recipes and present them with adaptation suggestions to real chefs. These chefs then rate the adaptation suggestions. For this we will design a questionnaire and present it to different chefs, because each chef may have a different opinion.

6 Related Work

JULIA [16] and CHEF [17] are early CBR systems giving preparation advice for meals. CHEF is a planning application which builds new recipes in the domain of Szechwan cooking. To satisfy the goals of a request for a new recipe it anticipates and tries to avoid problems. Therefore it stores and retrieves occurred problems and ways of dealing with them. JULIA integrates CBR and constraints for menu design tasks. It uses a large taxonomy of concepts and problem decomposition

with fixed decomposition plans. Unlike our approach their knowledge was built by experts and was not captured from communities.

The idea presented here closely relates to the research of Plaza [18], especially the EDIR cycle, however they concentrate more on gathering cases from web experience. In [4] they use the presented IakA approach for the acquisition of adaptation knowledge (and cases) by asking an oracle, which is described as an "ideal expert", but the presented prototype IakA-NF works (only) for numerical function domains. Furthermore Acquisition of Adaptation Knowledge from cases was done by by [8] or with the CABAMAKA System by [9].

The procedure of looking at first for concrete adaptation suggestions and apply afterwards, if the first step yields no results, more general rules, was done also by [6] with DIAL, which at first attempt to retrieve adaptation cases.

Our approach presented here goes with the vision of Collaborative Multi-Experts Systems (CoMES) [19] and is modelled following the SEASALT architecture [20], an instance of CoMES. Mapping this to the CommunityCook System the collection of recipes and comments corresponds to the task of the *Collector Agent*. The further analysis and interpretation match to their role of a *Knowledge Engineer*.

7 Conclusion and Outlook

Adaptation knowledge acquisition is an demanding and expensive task since it needs experts. In this paper we presented an approach to use experience from Internet communities for adaptation knowledge. Our approach is based on the idea of comparing the ingredients mentioned in a recipe to the ones mentioned in the comments that relate to the recipe. From comments which contain ingredients also existing in the recipe and others which are not contained in the recipe the adaptation suggestions are created and aggregated over all comments to 6200 suggestions. First evaluation results are promising, however ideas for a more complete evaluation which has still to be done are sketched.

The approach described here has a lot of advantages. For finding ingredients we can use our existing CookIIS knowledge model which has the benefit of taking care of synonyms, independence from slang and grammatically deficient language. By using a large number of recipes and comments we hope to balance out wrong classifications. We integrated the extracted adaptation suggestions in our CookIIS application.

In the future we want to be able to use the adaptation suggestions with regard to the recipe they belong to. Therefore we will find similar recipe out of our pool of 70'000 recipes to the one that has to be adapted and consider only comments of these recipes following the principle that similar recipes need similar adaptations.

Following the SEASALT architecture we also want to realize a multi-agent system that continuously monitors the community for new experiences with the recipes and adapts our adaptation knowledge if necessary.

References

1. Kolodner, J.L.: Case-Based Reasoning. Morgan Kaufmann, San Mateo (1993)
2. Greene, D., Freyne, J., Smyth, B., Cunningham, P.: An analysis of research themes in the cbr conference literature. [21] 18–43
3. Cojan, J., Lieber, J.: Conservative adaptation in metric spaces. [21] 135–149
4. Cordier, A., Fuchs, B., de Carvalho, L.L., Lieber, J., Mille, A.: Opportunistic acquisition of adaptation knowledge and cases - the iaka approach. [21] 150–164
5. Leake, D.B., Dial, S.A.: Using case provenance to propagate feedback to cases and adaptations. [21] 255–268
6. Leake, D.B., Kinley, A., Wilson, D.C.: Learning to improve case adaptation by introspective reasoning and cbr. In Veloso, M.M., Aamodt, A., eds.: ICCBR. Volume 1010 of LNCS., Springer (1995) 229–240
7. Wilke, W., Vollrath, I., Althoff, K.D., Bergmann, R.: A framework for learning adaptation knowledge based on knowledge light approaches. In: 5th German Workshop on CBR. (1996) 235–242
8. Hanney, K., Keane, M.T.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In Leake, D.B., Plaza, E., eds.: ICCBR. Volume 1266 of LNCS., Springer (1997) 359–370
9. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case base mining for adaptation knowledge acquisition. In Veloso, M.M., ed.: IJCAI, Morgan Kaufmann (2007) 750–755
10. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. Volume 2432 of LNAI. Springer-Verlag (2002)
11. This-Benckhard, H.: Rätsel und Geheimnisse der Kochkunst. Piper (2001)
12. Hanft, A., Ihle, N., Bach, K., Newo, R., Mänz, J.: Realising a cbr-based approach for computer cooking contest with e:ias. In Schaaf, M., ed.: ECCBR Workshops, Hildesheim, Berlin, Tharax Verlag (2008) 249–258
13. empolis GmbH: Technical white paper e:information access suite. Technical report, empolis GmbH (January 2008)
14. Hanft, A., Ihle, N., Bach, K., Newo, R.: Cookiis – competing in the first computer cooking contest. Künstliche Intelligenz **23**(1) (2009) 30–33
15. Hanft, A., Ihle, N., Newo, R.: Refinements for retrieval and adaptation of the cookiis application. In Hinkelmann, K., Wache, H., eds.: Wissensmanagement. Volume 145 of LNI., GI (2009) 139–148
16. Hinrichs, T.R.: Problem solving in open worlds. Lawrence Erlbaum (1992)
17. Hammond, K.J.: Chef: A model of case-based planning. In: American Association for Artificial Intelligence, AAAI-86, Philadelphia. (1986) 267–271
18. Plaza, E.: Semantics and experience in the future web. [21] 44–58 invited talk.
19. Althoff, K.D., Bach, K., Deutsch, J.O., Hanft, A., Mänz, J., Müller, T., Newo, R., Reichle, M., Schaaf, M., Weis, K.H.: Collaborative multi-expert-systems - realizing knowledge-lines with case factories and distributed learning systems. In Baumeister, J., Seipel, D., eds.: KESE. Volume 282 of CEUR Workshop Proceedings. (2007)
20. Bach, K., Reichle, M., Althoff, K.D.: A domain independent system architecture for sharing experience. In Hinneburg, A., ed.: Proceedings of LWA 2007, Workshop Wissens- und Erfahrungsmanagement. (September 2007) 296–303
21. Althoff, K.D., Bergmann, R., Minor, M., Hanft, A., eds.: Advances in Case-Based Reasoning, 9th European Conference, ECCBR 2008, Trier, Germany, September 1-4, 2008. Proceedings. Volume 5239 of LNCS., Heidelberg, Springer (2008)

Reuse of Search Experience for Resource Transformation

Peter Milne[1], Nirmalie Wiratunga[1], Robert Lothian[1], and Dawei Song[1]

School of Computing
The Robert Gordon University
Aberdeen AB25 1HG, Scotland, UK
{p.m.milne,n.wiratunga,r.m.lothian,d.song}@rgu.ac.uk
http://www.comp.rgu.ac.uk

Abstract. The advent of Web 2.0 has created a proliferation of resource sharing sites where individual users tag resources. Retrieval performance is good when users share the same vocabulary, but deteriorates when users have diverging vocabularies. In this paper we propose a novel method of reusing search experience to transform the underlying representation of tagged resources. The aim is to favour those tags that best correspond to community consensus. A CBR approach is presented to learn from user search histories, modifying resource tags in response to implicit user feedback. We evaluate this method on a prototype image retrieval system iFETCH. Our evaluation shows that resource transformation progressively increases the ranking of those images that are generally deemed relevant by similar search sessions. Our results also confirm that the casebase weight update mechanism is more robust to erroneous user feedback compared to a naive constant weight update strategy.

1 Introduction

Social searching and browsing have in recent years become increasingly popular with the advent of Web2.0 applications. These applications allow users to share resources such as documents, images, videos, music and Blogs on the Web. The absence of textual content presents a significant challenge for index creation for multimedia retrieval [8]. Resource annotation in the form of tagging is freely used and refers to the free association of keywords to resources by members of a given community using their own vocabulary [12]. The term folksonomy is now commonly used to refer to the bottom up structures that emerge as a result of such social tagging [1]. In recent years, social tagging has become popular, presenting itself as a useful indexing knowledge source, with Wikipedia, Flikr and del.icio.us websites a testament to this trend. An interesting emergent problem is the absence of a controlled tag vocabulary. Although this is attractive for users it is obviously not ideal for vocabulary management and multimedia content comparison. In this paper we investigate how to utilise user search and browsing experiences to evolve indexing vocabularies so as to capture consensus. We achieve this by transforming resource representations by altering individual tag weights thereby moving the resource closer to the queries that are commonly used to search that resource.

Resource transformation involves the modification of the underlying representation of a given resource [9]. Unlike feature weighting here feature values are incremented or alternatively decremented. Identifying which feature values to update and by how much are key concerns for transformation. Here we rely on user interaction and implicit relevance judgments. We introduce a casebased tag weight update mechanism by capturing previous user search experiences. Local neighbourhood similarity values are used to control the amount by which a weight is updated. We demonstrate our approach on an image retrieval task. Initial evaluation results are promising, showing that the technique increases the ranking of those images that the majority of users have deemed as relevant. Our results also show that the casebase weight update mechanism is more robust in the presence of atypical users.

In Section 2 we discuss the role of CBR within Web2.0 applications. Resource transformation using a casebased approach appears in Section 3 followed by a description of the image retrieval prototype, iFETCH, in Section 4. Our experimental design and initial results are discussed in Section 5. We conclude in Section 7 after presenting related work in Section 6.

2 Learning from Searching and Browsing Experiences

With information retrieval (IR) systems, users initiate retrieval through keyword-based search queries. Standard meta-search engines typically return a set of resources ranked according to their relevance to the query. A resource for example can be a document, image, video or Blog; and relevance typically estimates the level of term overlap in textual content. Like CBR, resource representation, indexing and similarity measures are key IR system design considerations. The vector space model is typically used for resource representation combined with an inverted list for indexing and the cosine similarity metric for retrieval [2]. Term importance or term weighting within each resource is typically captured as a function of term frequency and is known to improve search retrieval and ranking.

Tagging provides an additional source of knowledge for the standard feature vector representation. In fact studies suggest a greater overlap between the tag-query vocabularies when compared to the content-query vocabularies [11]. As a result it is increasingly common to represent resources based on just the tagging vocabulary. However the notion of tag importance within each resource is hard to capture, this is because a tag is simply assigned to a resource and is either present or absent. The situation is further exacerbated by the absence of a controlled tagging vocabulary, particularly with broad folksonomies where users can tag a resource with any set of tags that they see fit.

The question then is what knowledge can we use to establish within resource tag importance? User consensus on tags (tag popularity) and user selection patterns are both useful for this purpose. Unlike tag popularity which is easily captured by means of tag counters, user selection patterns require more sophisticated capture and integration mechanisms. For instance, if a user selects a resource as relevant to a query, then the query terms that are also used to tag this resource must be deemed important by this user. This allows us to implicitly identify which tag weights should be increased or alternatively decreased. However, some users might mistakenly select a resource as

relevant or be atypical and diverge from the majority. Therefore, instead of considering single user behaviour in isolation, it is far more useful to rely on collective user selection patterns. We need to also consider how much weights should be altered after each user session? The answer depends very much on the level of similarity in user selection patterns. To this end we exploit search history information to learn how much to update a tag weight by. We propose to maintain a casebase of previous queries, together with user selected resources. The more similar the cases to the recent user session the more reinforcement for tag weight update.

3 Case Representation and Transformation

A case consist of a query and a set of resources judged as relevant to the query by the user. The reliability of a user's feedback is a function of the similarity between the current search and the most similar cases in the casebase. These cases represent previous searches where the query and the relevant resources are the most similar to the current search. Each tag associated to a resource is given a weight proportional to how representative this tag is for this resource. This weight is consistently updated as the casebase evolves and more information is deduced from previous searches.

3.1 Case Representation

A case is a pair (Q, RR), where Q is a query and RR is a set of resources judged relevant to this query. A query Q is represented by a set $\{q_1, \ldots, q_l\}$ of keywords q_i present in the query. A set of relevant resources RR is represented by a set $\{R_1, \ldots, R_m\}$ of resources R_j judged by a user as relevant to the query Q. Each resource R_j is itself represented by a set $\{t_1, \ldots, t_n\}$ of tags t_k, where each tag has an associated weight w_k. A resource is normalised so that the weights of all associated tags t_k sum up to 1. In order to ascertain and utilise collective user judgments, the similarity between 2 cases is computed by aggregating the similarity between the query and the similarity between the set of relevant resources. The more similar the cases, the more similar the user search experiences.

3.2 Resource Transformation

The purpose of maintaining a casebase of previous searches and computing the similarities to the current search keywords is to optimise the tag weights associated to each resource. We propose to penalise tags associated to resources selected as relevant if they are not in the query. The weight reduction of such tags occurs as a side effect of the weight renormalisation stage. The weight update is a function of the similarity between the current search, or test case, and previous searches, or training cases. The more similar the test case is to the nearest cases in the casebase, the more the current users's feedback is aligned with previous users' feedback and therefore deemed as more reliable. The tags weights of a resource $R \in RR$ are updated as follows:

$$w_i{}^{temp} = w_i * update(R, Q)$$
$$update(R, Q) = \begin{cases} 1 & \text{if } t_i \in Q \\ f(CB, Q) & \text{if } t_i \notin Q \end{cases}$$

$$f(CB, Q) = (1 - \overline{Sim}_k(CB, Q))^{\alpha}$$

Here $\overline{Sim_k}$ is the average similarity between Q and its top k neighbours in CB. We use the cosine similarity metric and α is a parameter controlling the severity of the penalty. Other similarity metrics could have been applied, but cosine is most commonly used when comparing unstructured text. The fact that our tag vectors are normalised also makes cosine a natural description of similarity. Each dimension in a vector representation is a different term, where a non-zero value indicates the presence of that term. After each run, a resource's representation is updated so that the weights of each resource tag not present in the query is reduced relatively to the similarity between the current query and the most similar cases in our casebase. In turn, weight renormalisation will have the effect of increasing the weights of each resource tag present in the query. The renormalisation is achieved as follows:

$$w_i{}^{new} = \frac{w_i{}^{temp}}{\sum_{j=1}^{n} w_j{}^{temp}}$$

The overall impact of updating weights is to increase the importance of commonly used tags in the similarity metric used during retrieval, therefore achieving a ranking of resources more aligned with a global consensus of opinion.

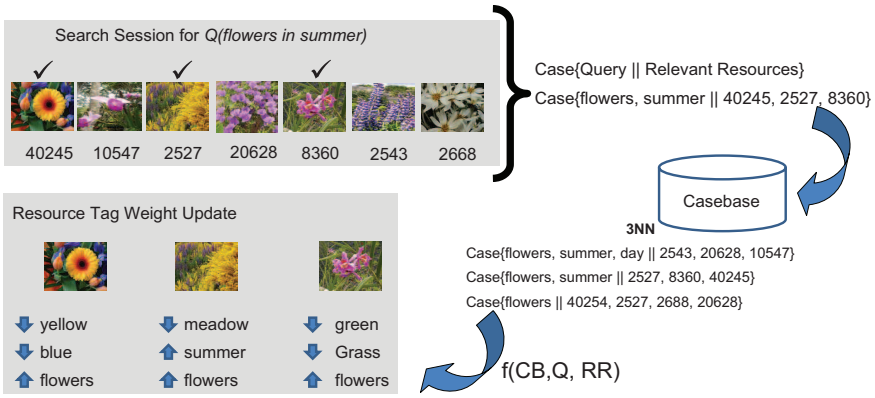


Fig. 1. Tag weights updated using a casebase

Figure 1 illustrates an example of a tag weight update session for images. The user executes a search with keywords *flowers in summer*. From the search results the users

selects 3 images 40245, 2527, 8360 that they feel are relevant to their information need. A new case made up of both the keywords and the identifiers for the images selected is then created and presented to the casebase. The k nearest neighbours are identified by calculating the cosine similarity between the new case and those in the casebase. A function of this similarity is then used to update the weights of each of the tags, where tags that were not in the query have their weights decreased, and tags common to the query have their weights increased. Finally tag weights of each of the selected resources are updated before the new case is added to the casebase for future searches.

4 The iFETCH Prototype

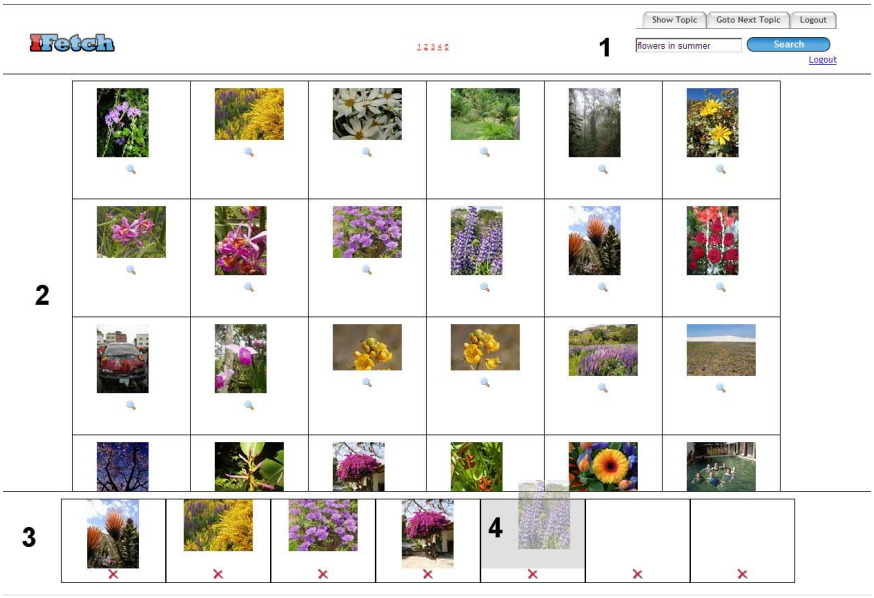


Fig. 2. The iFETCH user interface

To test our resource transformation method a demonstrator Web2.0 application, iFETCH was developed for image retrieval. Figure 2 shows the main components of the user interface. The interface header panel contains the search box to the top right (area 1) allowing users to input their search terms. Also in the centre of the header panel is the page selectors which are shown when there are multiple pages of results returned. Search result appear in the center (in area 2). Each image in this panel can be enlarged by clicking a magnifying glass icon below the image. Relevant images can be dragged into the task panel (shown by area 3) at the bottom of the interface. Each of the images in the task panel can be swapped back and forth or removed using a cross icon below

each image. This is illustrated in the figure with an image being dragged by a user into the task panel (in area 4). For demonstration purpose the task panel currently allows a selection of upto seven images. This is in keeping with Miller's [15] findings that the usual limit of the human short-term memory store is around seven units. IFETCH also allows users to execute multiple searches for the same information need while retaining any previously selected relevant images in the task panel. This is to allow the user to combine selections from multiple searches for the same information need. This is maintained as part of the user's profile, allowing them to amend previous queries and track their history of information needs.

5 Experimental Evaluation

We evaluate IFETCH incorporating the casebased weight update mechanism with a constant update function without a casebase. We also investigate the robustness of retrieval in the presence of noise. By noise we mean user's that may incorrectly identify resources as relevant or deviate from majority consensus. Precision at 10 is a good measure for evaluating the performance of web based retrieval systems [14]. This is due to the fact that the majority of people will only examine the first page of 10 results. Mean Average Precision (MAP) is a more recent metric that calculates the precision after each relevant document's retrieved rank [5] . These scores are then averaged over the number of relevant documents retrieved. For instance, if a run retrieves 3 relevant documents, one ranked 3rd, one ranked 5th and one ranked 10th. The precision at rank 3, 5 and 10 is calculated and averaged. In our evaluation, we used both metrics to compare the 2 systems: Precision at 10 (P10) and Mean Average Precision (MAP). It is generally accepted that evaluation is a challenge when user interaction is central to the techniques being evaluated. This is mainly due to the cost involved with large scale user trials, requiring live user participation. In order to get round this problem we developed a test harness to simulate user query generation and resource selection. Instead of collecting actual data from user trials we simulate search sessions using the IMAGECLEF'06 collection containing a set of images and their relevance to 60 topics.

5.1 Query Generation and User Trial Simulation

The IMAGECLEF 2006 dataset contains 20,000 images from many locations around the world. The majority of the images provided in the dataset are images from an independent travel company organising adventure and language trips to South America. The images within the dataset are varied and are realistic in terms of what we would expect to search in a web based image retrieval system. Figure 3 shows an example of a typical topic (animal swimming) and an associated image annotation. Topic annotations capture typical search needs and have been generated from search logs with the aim to provide a balanced and representative set of information needs. Importantly IMAGECLEF also has images labelled as relevant for each topic. We assume that these ground-truth images would be the ones that typical users might select for a given query (for a specific topic). As such the retrieval rank of ground-truth images provides the basis for comparing retrieval performance between user trials. The more ground truths

<TOPIC>
<NUM> Number: 5 </NUM>
<TITLE> animal swimming </TITLE>
<NARR>
Relevant images will show one or more animals (fish, birds, reptiles, etc.) swimming in a body of water.
Images of people swimming in water are not relevant.
Images of animals that are not swimming are not relevant.
</NARR>
</TOPIC>

<IMAGE>
<NO>annotations/34/34160.eng</NO>
<TITLE> a swimming Pelican near Paracas</TITLE>
<DESCRIPTION> a Pelican is swimming on the water;
</DESCRIPTION>
<NOTES></NOTES>
<LOCATION>Mondsee, Austria</LOCATION>
<DATE>December 2003</DATE>
<IMAGE>images/34/34160.jpg</IMAGE>
<THUMBNAIL>thum/34160.jpg</THUMBNAIL>
</IMAGE>

Fig. 3. The IMAGECLEF example topic and image annotation

ranked at the top the better the retrieval. Each image is allocated tags by extracting stemmed keywords from its description (see Figure 3). Tag weights are initialised and length normalised. We simulate a search task as selecting up to 7 images from those retrieved for a system generated query. Queries are generated by extracting keywords from a topic's title and as such play the role of a user query. For each query we then simulate 105 user trials, whereby 7 images are randomly selected as relevant from the retrieved set in a given user trial. A perfect user is simulated by randomly selecting ground-truth images from the set of retrieved images whilst a random selection forming a mix of ground truths and other irrelevant images simulates an imperfect user. In this way the level of error is managed by using a mixing parameter. A case is created for each simulated user trial and tag weights accordingly updated at each trial.

5.2 Results

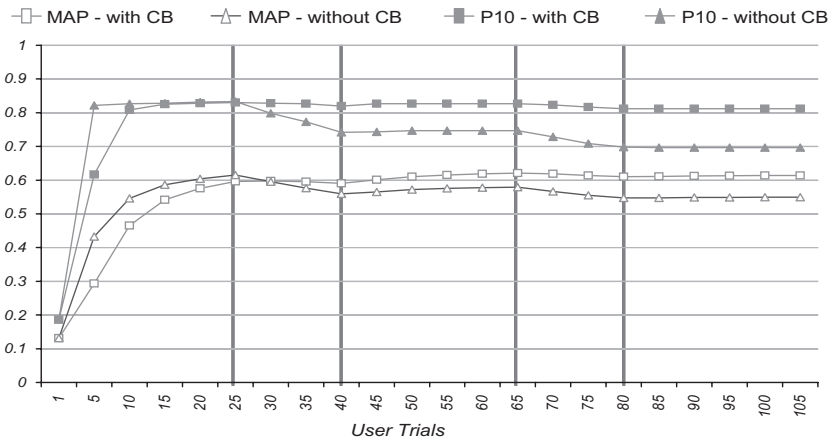


Fig. 4. Effect of erroneous feedback on MAP and P10

Figure 4 illustrates the evolution of MAP and P10 results averaged over the 60 topics for each ordered trial. During the first 25 runs, we assume a perfect user. The following 15 runs simulate unreliable user feedback. This is simulated by randomly selecting only 20% of ground-truth images and 80% of irrelevant images. This pattern is repeated for the following 65 runs. The first trial indicates the MAP and P10 result when no resource transformation is used. As expected, resource transformation significantly improves upon this first trial in consequent trials. Both MAP and P10 results peek faster when no casebase is used initially. This is because, during the 25 first runs, ground-truth images are selected and the relevant tag weights of each of these images are promoted after every run. However, with a casebase, the same image has to be selected multiple times before the similarity between the new case and cases in the casebase is sufficient to significantly promote tag weights. This initial phase can be likened to a standard CBR system's performance during the case generation or authoring phase. The advantage of the matured casebase becomes more apparent with the introduction of user error. This can be seen with the introduction of erroneous user feedback between trials 25 to 40, where MAP drops considerably when not using a casebase, whilst remaining relatively stable when using the casebase. This is because, with a casebase, an irrelevant image has to be selected multiple times before it can be promoted to a higher rank i.e. the majority of users must agree that these images fulfill the information need better than those previously selected. In the absence of a casebase, an irrelevant image gets promoted quickly to a higher rank as soon as it gets selected. This phenomenon is illustrated even more clearly in Figure 4, with the P10 graphs. Without the use of a casebase, P10 suffers a drop at each introduction of erroneous feedback and does not recover, even when consistent relevant feedback follows. It is also important to note that the average P10 achieved over the 60 topics reaches a level of over 0.8. This is much higher than the average MAP. While MAP illustrates the spread of relevant images over the retrieval, P10 illustrates the amount of relevant images ranked in the top 10. This is a more relevant result as, in a real web based system, users would tend to look at the top images within the first page and are less likely to browse other pages.

6 Related Work

Many studies have been conducted into user tagging behaviour with a view to identify how best to utilise tags as meta-knowledge sources for collection organisation and searching [12, 13]. Although the absence of any controlled vocabulary is viewed as a drawback, studies have shown that frequently used tags are still a good indicator of user query terms [11]. In our work we leverage on user queries as a means to refine the explicit association of tags to images. A casebase is used to capture tag popularity and social consensus about tags. Case similarity values are then used to implicitly refine the tag-to-image associations. The approach is sufficiently generic and can be transfered to other resources commonly encountered within folksonomies.

Refining associations between tags and their resources can be viewed as a form of tag recommendation but also as a form of relevance feedback. In traditional IR relevance feedback is used to refine a query entered by the user. The popular Rocchio algorithm uses feedback from a user's explicit preferences to form two centroids; rep-

resentative of relevant and irrelevant resources [6]. Using a linear combination query terms are then promoted if they appear in the relevant centroid and demoted otherwise. Here promotion can also be viewed as query expansion. Reliance on users for explicit feedback is a drawback which has been addressed by research into implicit feedback capture. Typically click-throughs and mouse moves are used for this purpose [7]. An interesting aspect here is o facilitate unobtrusive and effective feedback capture though clever interface design [8]. IFETCH also adopts a similar emphasise whereby standard areas for query entry and ranking are augmented with additional drag-and-drop work areas to allow for task management.

Document transformation work in machine learning, aims to refine a document's representation by learning from queries that led to its access. Unlike explicit and implicit relevance feedback mechanisms from IR, here querying experiences are utilised as a means to assist future user's with similar search needs. The idea is to reuse query and relevance feedback knowledge to improve descriptions of selected resources. In [9] indexing descriptors of documents on the web are modified, by incrementally updating it to better match each query that is used to retrieve it. However to directly apply such an approach within a folksonomy setting would be naive. This is because one needs to influence the update according to consensus and not simply on the basis of individual user queries. We achieve this by using an update function that is directly influenced by similarity to previous user search sessions i.e. their queries and chosen relevant resource sets. The more consensus amongst users about a query and its relevant resource set the higher the average similarity leading to higher promotion and demotion of tags.

Use of consensus within a community of like-minded people has been applied to improve document ranking in [10]. A separate representation of documents is maintained so as to to capture document relevance preferences for similar queries entered by users within a community. A new representation for selected documents is generated from snippet text that is returned by standard meta-search engines such as Google or Yahoo. This approach has the useful property of altering surrogate representations of documents instead of the original document representations themselves. As such one can imagine the maintenance of multiple community views through multiple surrogate representations. One drawback here is its reliance on textual snippets and as such does not easily lend itself to other forms of resources within folksonomies: such as images and videos. However the general idea of maintaining multiple community views is still very interesting and an area we intend to explore in the future.

7 Conclusion and Future Work

This paper presents and initial approach to transforming resource representations by altering the weights of tags associated to resources. We present a novel casebased approach to help control the amount by which weights are updated. The casebase maintains previous search experiences consisting of query and relevance judgments. A small-scale evaluation of the IFETCH demo system shows that resource transformation to result in far superior retrieval performance as it learns from each user session. Furthermore the casebase weight update approach remains resistant to user error, and outperforms a constant weight update approach.

The simulation of query generation and user trials in our evaluation is particularly novel. In future work it would be interesting to include new features to this evaluation methodology, for example, to allow variation in queries within topics, thus simulating user's differing vocabularies. Another interesting future direction would be the inclusion of multiple representations for groups of users, transforming these representations based on the group's vocabulary and the behavior of individual's within that group. We also intend to evaluate our weight update strategies using real users with the IFETCH prototype to ascertain if the technique works in a real life scenario.

References

1. Jaschke, R, Marinho, L., Hotho, A., Schmidt-Thieme, L., and Stumme, G.: Tag recommendations in folksonomies. *Proceedings Knowledge Discovery in DBs* 506–514, Springer (2007)
2. Salton. G., Wong, A., Yang, C.: A vector space model for automatic indexing *Communications of the ACM* 18(11) 613–620, (1975)
3. Hotho, A., Jschke, J., Schmitz, C., Stumme, G.: FolkRank: A Ranking Algorithm for Folksonomies. In *Proceedings of Fachgruppe Information Retrieval (FGIR)* (2006)
4. Jansen, B. J., Spink, A., Saracevic, T.: Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management* 36 207–227 (2000)
5. Buckley, C., Voorhees, E.M.: Evaluating evaluation measure stability. In *Proceedings of the 23rd Annual International Special Interest Group in Information Retrieval (SIGIR)* 33-40 ACM Press (2000)
6. Salton, G., Buckley, C.: Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science* 41(4) 288–297 (1990)
7. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback *Proceedings of the 28th Annual International Special Interest Group in Information Retrieval (SIGIR)* 154–161 ACM Press (2005)
8. White , R., Ruthven, I., Jose, J.: An implicit feedback approach for interactive information retrieval *Information Processing and Management* 42(1) 166–190 (2006)
9. Kemp, C., Ramamohanarao, K.: Long-Term Learning for Web Search Engines, *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)* 263–274 Springer (2002)
10. Boydell, O., Smyth, B.: Enhancing Case-Based, Collaborative Web Search. *Proceedings of the 7th International Conference on Case-Based Reasoning* 329–343 Springer (2007)
11. Suchanek, F., Vojnovic, M., Gunawardena, D.: Social tags: meaning and suggestions. *Proceedings of the 17th Conference on Information and Knowledge Management (CIKM)* 223–232 ACM Press (2008)
12. Van Damme,C., Hepp,M., Coenen, T.: Quality Metrics for Tags of Broad Folksonomies. *Proceedings of International Conference on Semantic Systems (I-SEMANTICS)* 118–125 (2008)
13. Farooq, U., Kannampallil, T., Song, Y. Ganoe, C., Carroll, J. Giles, L.: Evaluating tagging behavior in social bookmarking systems: metrics and design heuristics *Proceedings of International Conference on Supporting Group Work (GROUP)* 351–360 ACM Press (2007)
14. Anh, V. N., Moffat, A.: Improved retrieval effectiveness through impact transformation. *Australian Computer Science Communications* 24(2) 41–47 (2002)
15. G., A., Miller.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review* 63 81–97 (1956)

Principle and Praxis in the Experience Web: A Case Study in Social Music

Enric Plaza and Claudio Baccigalupo

IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain.
Email: {enric,claudio}@iiia.csic.es
WWW: http://www.iiia.csic.es

Abstract. The praxis of the users in some particular domain characterizes the Experience Web. In this paper we focus on analyzing *usage of musical objects*, and how the knowledge discovered can be used to design and implement a CBR system. We perform a case study of the Poolcasting CBR system in order to analyze the role of content provided by the praxis of users in the musical domain and the techniques used to acquire the knowledge required by a CBR system in the context of the Experience Web.

1 Introduction

Among the wide diversity of user-contributed content on the web, there is a particular kind of content that has the potential of being put to good use by intelligent systems: *human experiences*. We are now familiar with different forms of content that are provided by the users that reflect not merely an opinion or a belief, but rather express an individual experience: this we may call the Experience Web.

For instance, when a user has experienced a travel with an air carrier company or a stand at a hotel, the comments of that user concerning the air carrier A or the hotel are not merely issues of opinion or belief, they are expressing and recording a concrete and factual experience. That is to say, that the plane was delayed or that the hotel H didn't attend a client's request, they are not merely subjective estimations: (i) they are statements that certain facts occurred and (ii) they are evidence with respect to the likelihood oh these facts being a recurring pattern (A's planes tend to be delayed, H's staff tends to be unfriendly).

Human experiences recorded on the web offer *practical knowledge* concerning a wide variety of real world objects and situations. This practical knowledge is different from theoretical knowledge as that which can be provided by the Semantic Web. For instance, the Semantic Web approach can offer theoretical knowledge about hotels as in the statement "Hotel H is three stars (according to European Standards)", which means that some authority has classified hotel H so because it satisfies certain properties adjudicated to that class. Although

this knowledge also provides evidence on the quality and features of hotel H, it is in fact knowledge about the "three star hotel class" rather than about H itself.

Theoretical knowledge is, by definition, about the concept or the class — while practical knowledge is mostly about the object or the instance. What is there about the instance that is not in the class? Well, basically an instance has a concrete *cluster of relationships* with other instances, with its environment (that includes how people use that object or instance). Most of these relations are outside the purview of a theoretical/semantic definition of a concept or a class. Experiences, on the other hand, being concrete, are precisely those *clusters of relationships among instances*.

We have now introduced a core notion: *usage*. Thus human experiences, when expressed, essentially provide a description of *how people have used an object* — and therefore a description of relevant relations of that object with its (physical and conceptual) environment. In previous papers I've emphasized the fact that a large number of experiences in the web are described using text [6, 7]. There are situations, however, where such experiential knowledge are recorded on the web as different forms of data instead of free text; although these situations may seem minority or less general, they may be more amenable to analysis and reuse of those experiences by an automatic process.

In the rest of the paper we will present a case study in the domain of music, where human experiential knowledge is recorded and available as different forms and sources of data, and we will show how this experiential knowledge may be analyzed, interpreted, and reused to automatically to perform a particular task. The task we want to automate is that of play a DJ in a radio channel, as implemented in the Poolcasting system [4]. The task is to convey a selection of songs that are satisfying for a dynamic audience (i.e. a group of individuals) and that play smoothly one after the other (i.e. each song is musically associated with the next and not merely chosen at random). The purpose of the paper is to elucidate how web data can be analyzed as *experiential knowledge* and used in a case-based reasoning process. That is to say, how data can be interpreted as records of actions performed by human beings, and thus represents their *musical praxis*, and how the practical knowledge that can be discovered or inferred from that praxis can be exploited in a system that reasons from people's experiences.

The structure of the paper is as follows. Since Poolcasting has to satisfy two criteria, namely song sequence smoothness and group audience satisfaction, we will show in the next two sections how to acquire experiential knowledge for the criteria of musical smoothness (§2) and audience satisfaction (§3). The paper will close with a discussion section.

2 Social Music Praxis

In order to establish a succession of songs whose order in musically meaningful or appropriate, we need to acquire knowledge about which songs "play well together." Moreover, since this is a matter of degree, we will call *(musical) asso-*

ciation between two songs the likelihood that these two songs "play well one after the other." There are two ways to approach the acquisition of such a *musical association model*, one based on principles, and one based on praxis.

Acquiring a musical association model from *principles* means that we have some theory, some general knowledge, such that when applied to a particular pair of songs yields a degree of association. in the music domain these are called content-based approaches, since they analyze the song's musical or acoustic content. A simple principle could be that songs classified as belonging to the same genre should, in principle, play well together — similarly to the three-star hotel above, we are using information about the class (the genre). We may use any kind of class partition for this purpose, e.g. the songs performed by the same artist, or written by the same composer. The most common way is to represent each song by a collection of acoustic features extracted from the audio signal. For instance, some authors posit the principle that two songs are highly associated when their global timbre quality is similar [1]; this approach then focuses on ways to analyze the spectral shape of songs and ways to assess their similarity. Other approaches use beat and tempo analysis to assess which songs "sound similar" [8].

The other approach is analyzing the praxis of people in situations where they deal with songs that "play well together." One example of this approach would be analyzing the behavior of real DJ's in charge of playing music that flows smoothly over time. Nielsen Broadcasting Data has compiled a large amount of data on music broadcasts in more than 1,600 radio station, but their data base is not available without a fee.

The social web has revolutionized the scope and availability of all forms of content, and specially in the domain of popular music. Different social web platforms that focus on music concerns have compiled user-provided playlists in the order hundred of thousands. A *playlist* is in essence a collection of songs that someone has considered "play well together." Analyzing hundred of thousands of playlists we may discover which songs are more associated with respect to a community of users of a social web platform.

The association model we developed for Poolcasting analyzes this social data to find those songs that appear together in playlists, following two intuitions: (a) that the closer they occur in a playlist, the more associated they are, and (2) the more playlist two songs co-occur, the more associated they are. However, this initial analysis was insufficient, and we needed to take into account what we called a song's *popularity*: the more playlist a song occurs in, the more popular the song is. The reason is that, without taking into account song popularity co-occurrence of pairs of songs in playlists was biased in favor of popular songs. That is to say, we detected mainly situations where one of the co-occurring song was highly popular, but we missed co-occurring pairs of songs where neither of them was popular.

The final measure of *song association*, re-normalizing with respect to relative popularity [2], was applied to 599,565 playlists provided by the social web platform MyStrands.com. Moreover, once song association is estimated we can infer

Destiny's Child	
Poolcasting/ beta=0.5	*Kelly Rowland, City High, Ciara, Fantasia, Christina Milian, Beyonce, Ashanti, Girls Aloud, 3LW, Dru Hill*
MyStrands	*Ciara, Pussycat Dolls, Usher, Beyonce, Nelly, 50 Cent, Mariah Carey, Chris Brown, Gwen Stefani, Eminem*
AllMusic	*Mariah Carey, Jennifer Lopez, Aaliyah, Xscape, Ginuwine, Deborah Cox, Kelly Price, Faith Evans, Brandy, Usher*
Yahoo	*Cruel Story Of Youth, Jessica Simpson, Ryan Cabrera, Ashlee Simpson, Faith Evans, Nick Lachey, Vitaly Romanov, Janet Jackson*
Last.fm	*Beyoncé, Mariah Carey, Jennifer Lopez, Usher, Aaliyah, Rihanna, TLC, Ciara, Ashanti, Christina Aguilera*

Fig. 1. Comparisons of artists associated with Destiny's Child.

association degrees among artists. The results were compared with other models of "musical similarity" among songs and artists, like *All Music Guide* where they are provided by expert editors, *Yahoo! Music* where information come from user feedback, and Last.fm where similarity comes from overall listening habits of users. Although our order-based *association* is asymmetric and is not a measure of *similarity* as the measures provided by these sources, the general results are roughly equivalent in practice. However, our measure did find out more obscure associations (because of the popularity renormalization) than others did not detect. For instance, Figure 1 shows that *Poolcasting* associates Destiny's Child with Kelly Rowland; this association is a good one, because Kelly Rowland is the lead singer of Destiny's Child.

Regardless of the details, the focus of this paper is on the fact that we are analyzing how people *use* their music. Playlists embody some particular instance of the notion of "songs sounding well together," and the social web platform merely provides a conduit where this experiential knowledge, from many users, and about tens of thousands of songs, is expressed and stored. The fact that MyStrands.com is a "social web platform" is relevant in as much as it facilitates that a large number of users contribute their musical experiences. There is no difference in analyzing user's playlists in a personal computer or shared via a website: social web platforms are useful in motivating and facilitating the sharing of experiential knowledge, not necessarily creating that experiential knowledge. Nevertheless, the openness of user-contributed experiential content is very important in practice: (1) the number of songs and artists from different countries and sources (e.g. bootleg concerts, independent bands) is larger than any particular endeavour (like All Music Guide, based on experts) could ever achieve; (2) the responsiveness to include newly created songs is also much higher.

Reasoning from experiences on the web is not only a matter of acquiring and analyzing experiential knowledge. In our musical domain, for instance, playlists are contentless — i.e. they contain references to the songs (and the artists) but not the songs themselves. Thus, to put the Poolcasting system into practice we needed to identify the songs references and the artists names: recover from misspellings, unify denominations and establish a unique (and possibly shared)

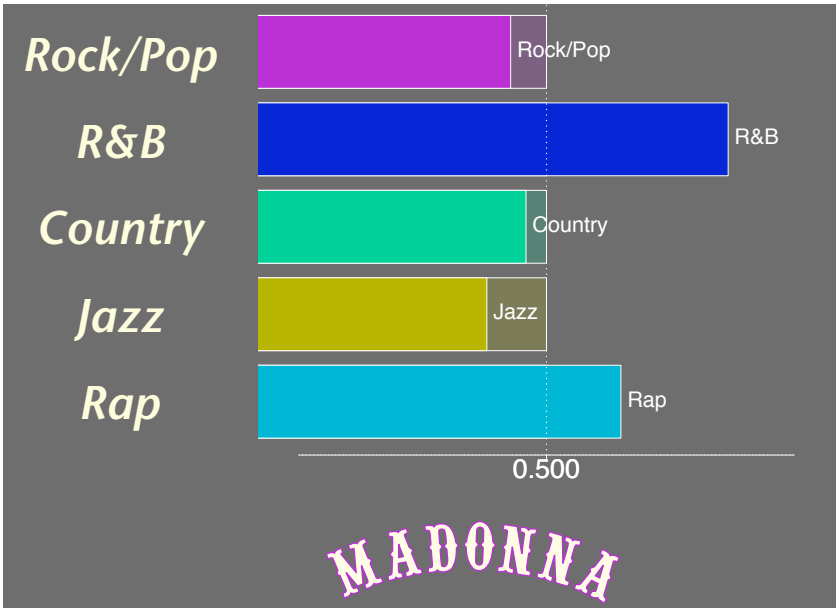


Fig. 2. Genre affinity profile of Madonna.

ID for each object in the domain. This Data Web concern is orthogonal to our approach on the Experience Web, and is currently being addressed by research on the Data Web; in this approach objects like cities or persons are identified by RDF triplets. The "linked data" approach proposed by Tim Bernes-Lee [5] seems more congruous with the Experience Web than the semantic web approach: the Data Web focuses on representing the *clusters of relationships among instances* that we talked about before as the way concrete experiences may be represented.

Analyzing and discovering higher-level relationships from experiences is not technically different from analyzing data, but the fact that the discovered relationship come from data recording practice is what makes a difference. Analyzing how people use and combine songs in playlists we find how songs and artists are associated. Therefore, we can analyze how songs or artists cluster together to form groups. Moreover, songs and artists are already categorized into *genres*, but his application of principles assigns only one genre label to each artist, sacrificing a more nuanced characterization. However, analyzing users' musical praxis we can discover new relations between artists, clusters of artists, and genres.

For instance, we can revise the principle-based categorization of artists and propose that artists have a graded *affinity* to multiple genres [3]. This characterization of artists is closer to reality, since artists do not belong to one genre and are excluded from belonging to any other genre; rather, they have high affinity to some genres (e.g. Madonna has high affinity with Pop and R&B) and low affinity to others (e.g. Madonna has very low affinity with Jazz). Moreover, the

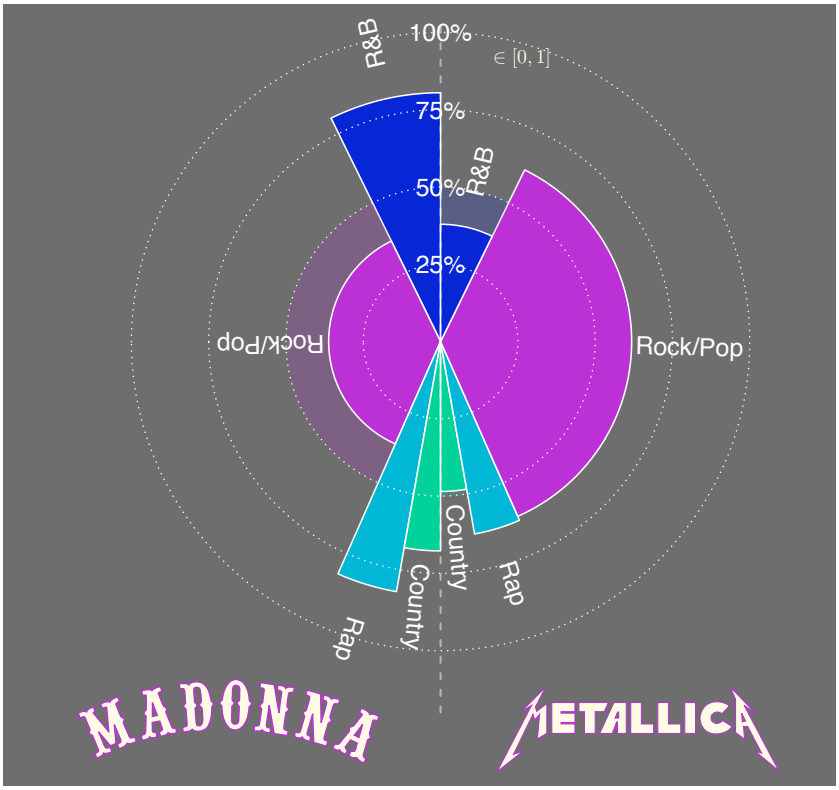


Fig. 3. Genre-centrality comparison of two artists originally labelled as Rock/Pop.

affinity vector of each artist with respect to genres provides a new way to describe musical performers, as shown in Figure 2, where the affinity degree spans from 0 to 1.

Moreover, we can detect which artists are "central" to specific genres — i.e. they are good representatives of that genre. The *genre-centrality* of an artist x to a genre g is the percentage of artists whose genre affinity to g is lower or equal than the genre affinity of x to genre g. For instance, on the Soundtrack genre the most central artists are James Horner, Alan Silvestri and Michael Giacchino, who are famous composers of original movie scores (e.g., James Horners Titanic Original Soundtrack), and not Pop artists who have only sporadically performed famous songs which appeared in movies (e.g., Celine Dions My Heart Will Go On). Moreover, artists can be compared on how central they are to different genres, as shown in Figure 3 where we may compare Madonna and Metallica (both originally classified in the Rock/Pop genre).

The *usage of musical objects* by users is the basis of these analysis, it's the availability of this data that is crucial for the Experience Web. The fact that

this data is available in "social web platforms", or the kinds of data mining techniques used to analyze them, these are secondary issues: the praxis of the users in some particular domain characterizes the Experience Web.

3 Individual Music Listening Praxis

The second criterion for the *Poolcasting* system is to customize the music selection to a dynamic audience —namely the group of users registered at a musical channel. Therefore, the system needs knowledge to estimate how satisfying is selecting a song over another (1) for the individuals in the audience and (2) for the overall satisfaction of the group as such. The overall satisfaction is essentially some sort of average of the individual satisfaction, so we needed to acquire knowledge about which songs and/or artists an individual prefers. For this purpose, we analyzed how individuals used their music libraries on their computers, specifically on iTunes players. The data available on iTunes library database includes which songs are rated higher, which songs had been played frequently, etc.

The strategy is similar to the one in the previous section, but now we are focusing on examining each individual music player as a repository of data about their musical listening praxis. We considered that each library database may be interpreted as an "individual case base" and can thus be used in a CBR system like *Poolcasting* to predict the degree of satisfaction of each user in the audience with respect to a specific song. However, as indicated in the original paper [4], they were not strictly "individual case bases" since they not contained *cases*. The core idea of cases in CBR is very close to that of examples in Machine Learning, maybe for historical reasons; a case may have a representation that is simpler or more complex, but is composed of two *separate* objects: a problem (represented on the *problem space*) and a solution (represented on the *solution space*). Similarity of two problems is defined over the problem space with the purpose of estimating how similar their solutions might be in the solution space.

However, the *Poolcasting* system did not had access to this kind of experiential knowledge represented as cases. Thus, the approach we took was to take a step back, and recall that CBR is also classically defined as reasoning and learning from past experience. Within this interpretation, we did have knowledge of the users' usage of songs for listening purposes by analyzing the digital music player library data. The knowledge that can be derived is (qualitatively) straightforward: the more often a song has been played, the more star-based ranking a song has, the more songs of an artist the user has, then the more likely is that the user likes that song or that artist.

Moreover, we already have seen in section 2 how to acquire knowledge about song association. The schema in Figure 4 shows how the the *Poolcasting* system combines knowledge about musical association among songs (obtained from playlists contributed by a community of users) and knowledge about audience song preferences (obtained from digital music player library data of the members of the audience). The Retrieve process uses the musical association knowledge to

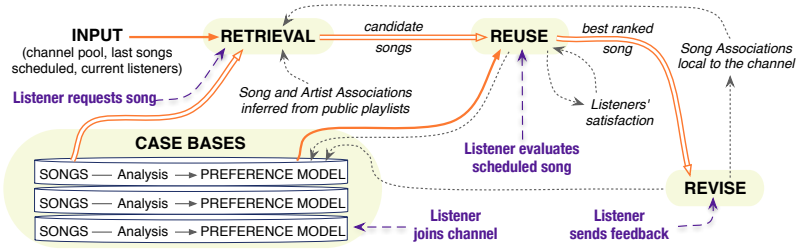


Fig. 4. The CBR schema of the Poolcasting system.

filter, from all possible songs, a small number of songs that are musically associated with the last song being played, while the Reuse process used the audience song preferences knowledge to select the song that will keep the audience (and individual members) satisfied.

4 Discussion

In summary, *Poolcasting* as a CBR system focuses on acquiring and harness knowledge coming from the praxis of users in a domain, analyzing their *usage* of the objects in a domain for specific purposes. In this case study we analyzed how people put together songs in a playlist; they do it because for them these songs (for some unknown reasons or purpose) "sound well together." We also analyzed how individuals use music stored in their digital music players; we interpreted them as repositories of data recording the music listening praxis of each individual.

The Experience Web is therefore characterized by a certain viewpoint on a specific type of content. The content is data representing specific actions, the praxis of individuals in a given domain; the viewpoint is that we interpret those actions, that praxis, as experiences from which new knowledge and insight can be gained and harnessed by developing intelligent systems for achieving specific goals.

The Semantic Web and the Data Web are orthogonal endeavors withe respect to the Experience Web approach. They are in fact required to be able to harness the Experience Web. We have focused on this paper on domains where experiences are directly recorded as data, not free text. Although text-based experiences are qualitatively and quantitatively very important, we would argue that a careful examination of existing non-textual content will uncover areas where the available data can be analyzed and interpreted as experiential content, and be amenable to partake of the Experience Web approach.

References

[1] J. J. Aucouturier and F. Pachet. Music similarity measures: what's the use? In *Proceedings of the 3rd international symposium on music information retrieval (ISMIR02), Paris*, October 2002.

[2] C. Baccigalupo and E. Plaza. Mining music social networks for automating social music services. In *Workshop Notes of the ECML/PKDD 2007 Workshop on Web Mining 2.0*, pages 123–134, 2007.

[3] Claudio Baccigalupo, Justin Donaldson, and Enric Plaza. Uncovering affinity of artists to multiple genres from social behaviour data. In *Proc. 9th Int. Conference on Music Information retrieval ISMIR-08*, pages 275–280, 2008.

[4] Claudio Baccigalupo and Enric Plaza. A case-based song scheduler for group customised radio. In Rosina Weber and Michael M. Richter, editors, *Case-Based Reasoning Research and Development, 7th International Conference on Case-Based Reasoning, ICCBR 2007*, volume 4626 of *Lecture Notes in Artificial Intelligence*, pages 433–448. Springer, 2007.

[5] Tim Berners-Lee. The next web of open, linked data. In *TED2009 Conference*, 2009.

[6] Enric Plaza. Semantics and experience in the future web. In K D Althoff, R Bergmann, M Minor, and A Hanft, editors, *Advances in Case-Based Reasoning: 9th European Conference, ECCBR 2008*, volume 5239 of *Lecture Notes in Artificial Intelligence*, pages 44–58. Springer Verlag, 2008.

[7] Enric Plaza. On reusing other people's experiences. *Künstliche Intelligenz*, 9(1):18–23, 2009.

[8] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE Transactions on*, 10(5):293–302, 2002.

Collaborative Information Access: A Conversational Search Approach

Saurav Sahay, Anushree Venkatesh, and Ashwin Ram

College of Computing
Georgia Institute of Technology
Atlanta, GA

Abstract. Knowledge and user generated content is proliferating on the web in scientific publications, information portals and online social media. This knowledge explosion has continued to outpace technological innovation in efficient information access technologies. In this paper, we describe the methods and technologies for 'Conversational Search' as an innovative solution to facilitate easier information access and reduce the information overload for users. Conversational Search is an interactive and collaborative information finding interaction. The participants in this interaction engage in social conversations aided with an intelligent information agent (Cobot) that provides contextually relevant search recommendations. The collaborative and conversational search activity helps users make faster and more informed search and discovery. It also helps the agent learn about conversations with interactions and social feedback to make better recommendations. Conversational search leveragesthe social discovery process by integrating web information retrieval along with the social interactions.

1 Introduction

Socially enabled online information search (social search) is a new phenomenon facilitated by recent Web technologies. This collaborative social search involves finding specific people in your network who have the knowledge you're looking for or finding relevant information based on one's social network. Social psychologists have experimentally validated that the act of social discussions have facilitated cognitive performance[1]. People in social groups can provide solutions (answers to questions)[2], pointers to databases or other people (meta-knowledge)[2][3] , validation and legitimation of ideas[2][4], can serve as memory aids[5] and help with problem reformulation[2]. Guided participation[6] is a process in which people co-construct knowledge in concert with peers in their community[7]. Information seeking is mostly a solitary activity on the web today. Some recent work on collaborative search reports several interesting findings and the potential of this technology for better information access. [8] [9] [10] [11]

We are building a system called Cobot [1] to address these challenges. Cobot introduces a conversational environment that provides social search through conversations integrated with intelligent semantic meta-search from the web. Users

[1] We use the term Cobot for Cobot system as well as Cobot agent interchangeably

want to simplify their experience when performing an information finding task. Conversational Search is about letting users collaboratively search and find in natural language, leaving the task of user intent comprehension on the system. The participating agent interacts with users proving recommendations that the users can accept, reject, like, dislike or suggest.

Figure 1 captures the online search space divided into web search and social search on one axis and aggregated, personalized and semantic search on the other axis. Cobot falls in the space of Social and Semantic Search space. It is social because it uses the user's social graph to find socially relevant results. It is semantic because it understands the queries (to some extent), concepts, relationships and indexes terms by their enclosing semantic types.

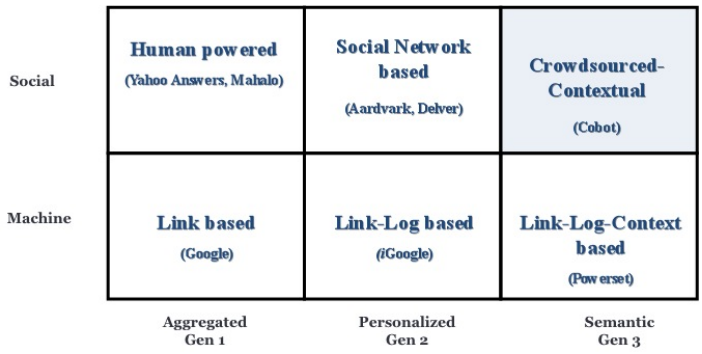


Fig. 1. Web Search Space

2 The Problem

The need to make the world wide web information universally accessible has accelerated research and development in Information Retrieval (IR) systems. Most web search systems are based on general Information Retrieval (IR) principles. Many of these IR systems are general purpose search systems that index millions, if not billions of pages and use state of the art advanced statistical modeling techniques to make them findable using keyword based matching. Google, for example, models webpages using link cardinality on the hypertext web graph to calculate the relative importance of webpages. They use several other parameters like proximity, anchor text and cardinality to build their full text search index. One of the design goals of the initial Google system was to handle the common case of queries well, topical information and under-specified queries. Even today, these popular search engines are not able to find precise, specific and non topical information efficiently.

Conversational Search differs from traditional search paradigms in some ways. The focus is user centric collaborative information access from the web; it is not acceptable to return hundreds of results matching a few keywords even if two or three of the top ten are relevant. Unlike traditional information retrieval, the

problem requires synthesis of information and interaction, the system along with the users of the system must analyze the results collectively to create an effective solution.

3 Proposed Solution

3.1 Conversational Search

Conversational Search(CS) is an interactive and collaborative information finding interaction. The participants engage in a conversation and perform a social search activity that is aided by an intelligent agent. The collaborative search activity helps the agent learn about conversations with interactions and feedback from participants.

It uses the power of semantics with natural language understanding to provide the users with faster and relevant search results without being overwhelmed by information. It moves search from being a solitary activity to being more participatory activity for the user. The search agent performs multiple tasks of finding relevant information and connecting the users together; participants provide feedback to the agent during the conversations that allows the agent to perform better.

CS is different from Information Retrieval (IR) [12] or Question Answering (QA) [13]. The focus of IR systems is on retrieving relevant documents from a large document collection in response to a query. If the user's information need is complex, browsing through retrieved results to find solutions to problem queries is time consuming and inefficient. Moreover, IR generally does not deal with the process of understanding the meaning of queries when posed in natural language e.g. in the form of a question or paraphrases.

In Question Answering (QA), researchers are developing different algorithms and techniques to obtain effective responses for specic information requests. The solution is generally present in a paragraph, sentence, or phrase. These snippets of information contain possible answers to the posed questions. While QA deals with understanding the meaning of natural language queries, it does not involve a back and forth interaction to continuously adapt the results and find out more and explore in continuum about some information or questions.

CS involves a continuous exchange of information between the sender and recipients; allowing for mutual learning and benefit. Fusion of IR and QA can be imagined to be a part of the CS approach. It is an intelligent problem solving AI technique applied to address the problem of search differently. There are several hard problems and challenges involved in CS besides the inherent problems in IR and QA. Some of the additional problems in CS are as follows:

- *How to model CS as a collaborative information finding activity?* The process of modeling an artifact involves giving it structure and organization for representing its intension and extension. The challenge lies in modeling it such that it can lend itself naturally for carrying out tasks for which it needs to be modeled. CS brings relevant information and people to the participants.

These recommendations are generated in progression with users' social interactions. Taking these functionalities into account, modeling CS involves creating dynamic data structures that are socially aware of the participants of the conversation and its content.

- *How do we use the model as a basis for providing recommendations?* The socio-semantic models of conversations along with the user models, case index of conversations and the semantically searchable document index act as different levels of memory structures for the system. Users' interactions and social feedback are registered in the system to bring in suitable recommendations and also improve contextual relevance of the data being generated.
- *How do we dynamically connect cohorts based on the conversations?* The system is aware of the users' social network along with the users who are online in the system. The user models consist of an aggregate of user's knowledge as a result of his past interactions. We dynamically connect cohorts by overlaying the user models with the social network taking into account the users that are online in the system.
- *How does the agent find relevant information to insert in the conversation besides providing relevant recommendations?* Besides providing the recommendations for the conversations, the agent's goal is to insert possible answers to questions directly into the conversation. The "Text Analysis and Processing Engine(TAPE)" analyzes the conversation and the recommendations to identify possible answers for the natural language questions.

The approach we have taken to address CS problems is by developing dynamic data structures that model it. We call this structure the "Socio-Semantic Conversation Net" - these conversation nets maintain in memory models of the conversation, participants, participants' immediate social connections, concepts, relationships and information flow.

3.2 Socio-Semantic Conversation Model

"The core problem that context-sensitive asynchronous memory addresses is how to get the information an agent needs when it doesnt know how to ask the right question and doesnt have the time to exhaustively search all information available to it. The key to this solution is to interleave remembering with thinking and doing, thus making the context of thought and action available to guide remembering."[14]

The Socio-Semantic Conversation Model that we are developing is a dynamic memory data structure based on principles of experience based agent architecture. [15] It supports interleaved retrieval of information by applying different memory retrieval algorithms such as PageRank or Spreading Activation. The model maintains the user's social graph, the conversation graph with the extracted semantic net for the conversation.

Some essential properties of the model are as follows:

- The model should be socially aware of the participant and his social network's availability (to aid with Cohort Matching)

- The model should provide bi-directional recommendation and feedback.
- The model should understand domain terminology and be able to find semantic relationships amongst concepts extracted from conversations.
- The model should be aware of user's basic profile (such as interests) for the agent to be able to use that information if needed.

The Socio-Semantic Model aims to provide storage and memory based retrieval for dynamic representation, update and reuse of users' knowledge and experiences. Figure 2 depicts the user-centric domain information modeling approach to jointly model the information context from users' perspective.

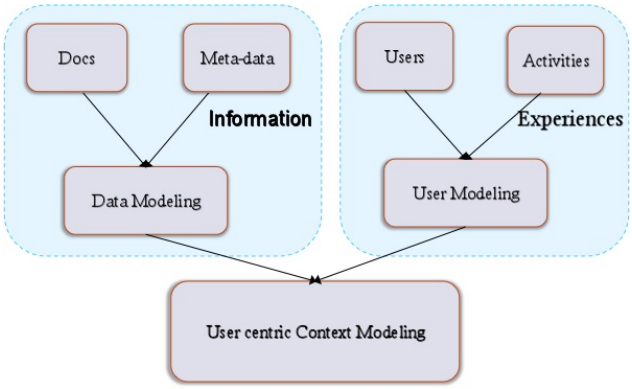


Fig. 2. User-centric Domain Information Modeling

4 Cobot System Prototype Design

Cobot is an intelligent agent platform that connects users through real-time and offline conversations. Cobot lives in a community, has a limited understanding of domains through ontologies and brings relevant information to the users by participating in the conversations. Cobot's 'conversation engine' monitors user conversations with other users in the community and provides/receives recommendations (links and snippets) based on the conversation to the participants. Cobot's community engine' models conversations to capture user-user and user-information interactions.

The following design goals are being adhered while developing the Cobot system.

1. Near real time conversational agent
2. Personalized recommendations
3. Agent learns with interaction
4. Uses a structured internal organization of content
5. Connects conversations to people.

Figure 3 depicts the high level architecture of the Cobot system. The Conversational Agent uses different modules for conversation analysis (TAPE), search and recommendation and maintains a short-term conversation memory for each conversation. The socio-semantic index is analogous to the agent's long term memory model where it stores all processed information about users, conversations, activities and content descriptors.

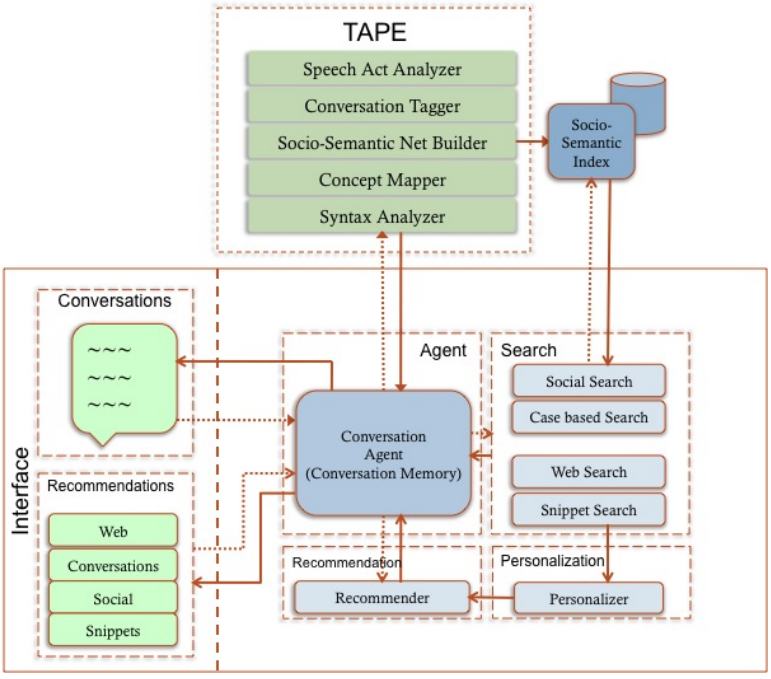


Fig. 3. System Architecture

Figure 4 shows one screenshot of the initial system prototype. This prototype is fine tuned for health related searches by incorporating medical ontologies in order to better understand the conversations. Users actively engage in conversations by multi-user chat, rating or adding recommendations. The agent monitors the environment to build user interaction models and to improve search relevance.

5 Key Research Challenges for Conversational Search

Search results are not tailored to the users goals or information need, or to his/her specific situation. Cobot system is agent based and agent assisted. We browse, find and soon forget what we have found. The agent based system builds and maintains user models and finds relevant information from the web or his past interactions, network and experiences.

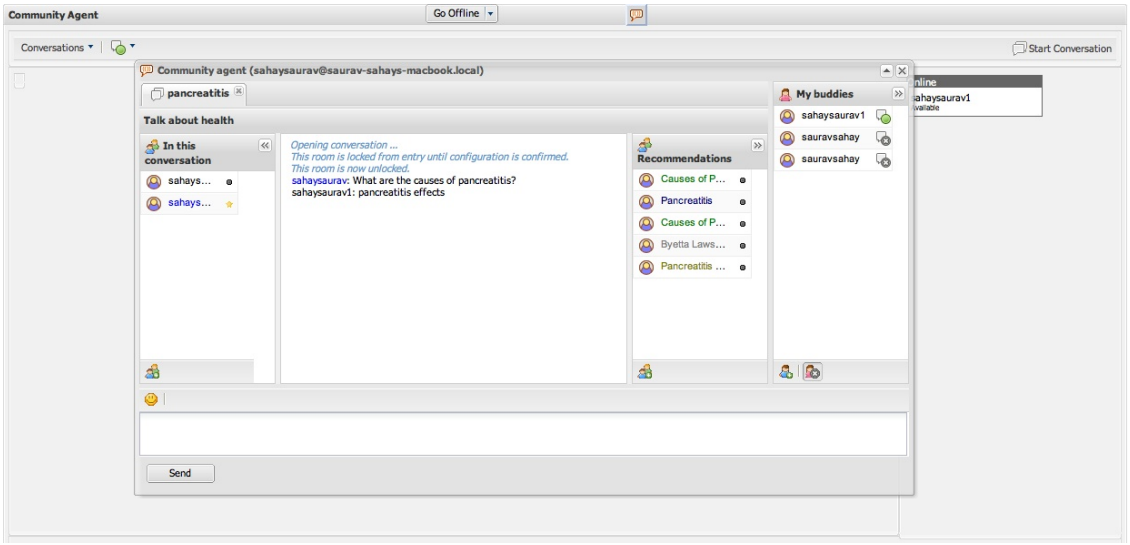


Fig. 4. Prototype Interface

5.1 Precise Search

Identifying relevant documents for a particular user's need without extensive search, in conversational manner is the key objective for precise search. The right search queries need to be figured out with situation assessment from the conversational snippets. It is not desirable to return dozens or hundreds of remotely relevant results, even if some of them will be highly relevant. The aim is to retrieve successive recommendations that try to address the search problem precisely.

5.2 Knowledge Representation and Synthesis

Any form of knowledge that needs to be captured has to be expressed in some representation medium. This representation of knowledge is one of the fundamental intelligence design problems that has been extensively studied in Articial Intelligence research. Textual search based systems work on natural language. Natural language, unlike math or logic, does not intrinsically lend itself to computational reasoning. In order to intelligently reason from text, it needs to be abstracted in a form that becomes amenable to communicating that meaning to any user. Combining the document models with the user models in an integrated representation will lead to development of systems that intrinsically lend their model to user centric personalization efforts. We are developing a graph based representation of our information model that includes data entities as well as user based entities.

5.3 Socio-Semantic Conversation Modeling

Language and interaction creates usable memories, useful for making decisions about what to do, what to retain, or how to plan future moves. Data Modeling is the process of providing some organization and structure to data. Knowledge bases and databases are built by adhering to some data model and populating the model with occurrences of data. We build socio-semantic nets as discussed earlier to model the social behavior of people on the system across the semantic data nets.

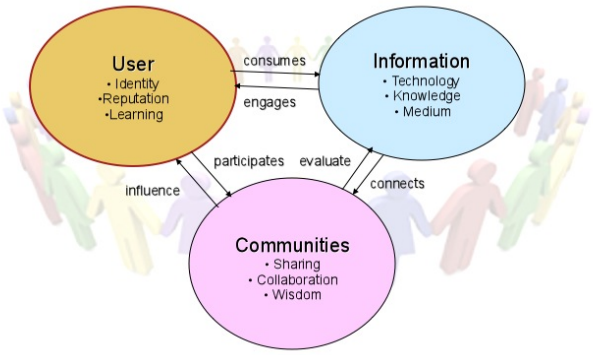


Fig. 5. The Socio-Information Cycle

Figure 5 tries to depict the relationships between the user, information and the communities. Communities are made up of users who are grouped by different information needs into dynamic cohorts. These online communities, through effective sharing and collaboration, increase the utility of systems and help solve individual problems more effectively.

5.4 Case based Reasoning for Longitudinal Search

Case-based reasoning is an artificial intelligence approach, in which past cases are used to solve new problems [16] [17]. The key lies not in running a smarter search engine against a set of documents, but in understanding which documents contain appropriate answers to users' different kinds of queries using his past experiences. While driven by information retrieval techniques, there is a learning component that goes beyond simply matching queries against documents to matching queries against past episodes. Cases are stored in a case library and represent the systems experience or historical record of previous queries and responses.

Conversational Search uses the familiar CBR cycle (retrieve, select, apply, learn) but with the following differences:

– there is a separate acquisition and representation phase which builds the knowledgebase by acquiring information from the web

- these is a social retrieval component that finds people based on their related past conversations; it also retrieves information from the user's social network based on conversations
- retrieve and select require text analytics (in our case, NLP, search), since the knowledgebases and cases are unstructured text instead of traditional AI representations
- the apply phase merges the social and web based recommendations to create a new case for the problem
- learn requires human-in-the-loop relevance feedback and requires storing new cases in the case library

Instead of matching queries against keywords in documents, the system develops a case library of past problem-solving sessions containing previous queries the system has seen and corresponding solutions the system has proposed. More specifically, this approach builds on the following research issues, as can be associated with a CBR system:

- Knowledge Representation: What information does a case contain apart from the given knowledgebase representation? How is this information represented?
  Information is modeled by capturing user-user and user-data interactions for every user so that the system can reason with their experiences. A particular case is associated with the problem as posed by the user and the solution which in turn keeps a record of the suggested and finally chosen solution(s).
- Indexing and Retrieval: How are cases organized to enable relevant cases to be found later? How are cases retrieved in response to a users query? How is the relevance of a case determined?
  Relevance of a case is dependent on the user model in the system along with the experience (feedback) associated with it.
- Learning: How are new cases learned? How are indexes and cases updated through experience?
  For every case that is finally created, there is social feedback associated with it. The system uses such implicit and explicit feedback to learn and update associated user models and also to gauge relevance of cases.

6 Discussion

This paper proposes a collaborative system for conversational search. We are hypothesizing that such a Conversational Search system is more usable for information access as compared to a solitary web search experience. We briefly describe the challenges involved in construction of a Socio-Semantic Conversation Model for Conversational Search. Socio-Semantic Conversation Modeling using Experience-based Agency is a unified approach for solving Conversational Search problem. This approach leverages the individual Social and Semantic Approaches efficiently. The dynamic and self configuring memory structures and the semantic net details enable memory retrieval from the storage. Automatic

Cohort Matching based on Conversations describes a methodology to dynamically pull users for conversations. Unlike users themselves having to find relevant conversations, the conversations find the users using this approach.

References

1. Ybarra, O., Burnstein, E., Winkielman, P., Keller, M.C., Manis, M., Chan, E., Rodriguez, J.: Mental Exercising Through Simple Socializing: Social Interaction Promotes General Cognitive Functioning. Pers Soc Psychol Bull **34**(2) (2008) 248–259
2. Cross, R., Rice, R.E., Parker, A.: Information seeking in social context: structural influences and receipt of information benefits. IEEE Transactions on Systems, Man, and Cybernetics, Part C **31**(4) (2001) 438–448
3. Fox, E.A., Hix, D., Nowell, L.T., Brueni, D.J., Rao, D., Wake, W.C., Heath, L.S.: Users, user interfaces, and objects: Envision, a digital library. J. Am. Soc. Inf. Sci. **44**(8) (1993) 480–491
4. Evans, B.M., Chi, E.H.: Towards a model of understanding social search. In: CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work, New York, NY, USA, ACM (2008) 485–494
5. Karasavvidis, I.: Distributed Cognition and Educational Practice. Journal of Interactive Learning Research (2002) 11–29
6. Rogoff, B.: Apprenticeship in thinking: Cognitive development in social context. Oxford University Press New York (1990)
7. Wilson, B., Meij, H.: Constructivist learning environments: Case studies in instructional design. IEEE Transactions on Professional Communication (1997) 0361–1434
8. Feng, D., Shaw, E., Kim, J., Hovy, E.: An intelligent discussion-bot for answering student queries in threaded discussions. In: Proceedings of the 11th international conference on Intelligent user interfaces, ACM New York, NY, USA (2006) 171–177
9. Boydell, O., Smyth, B.: Enhancing case-based, collaborative web search. Lecture Notes in Computer Science **4626** (2007) 329
10. Amershi, S., Morris, M.: Cosearch: a system for co-located collaborative web search. (2008)
11. Paul, S., Morris, M.: Cosense: enhancing sensemaking for collaborative web search. In: Proceedings of the 27th international conference on Human factors in computing systems, ACM New York, NY, USA (2009) 1771–1780
12. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern information retrieval. Addison-Wesley Harlow, England (1999)
13. Lehnert, W.: The process of question answering. (1977)
14. Francis, Anthony G., J.: Context-sensitive asynchronous memory : a general experience-based method for managing information access in cognitive agents. PhD thesis, Georgia Institute of Technology (2000)
15. Ram, A., Francis, A.: Multi-plan retrieval and adaptation in an experience-based agent. Case-Based Reasoning: experiences, lessons, and future directions (1996) 167–184
16. Kolodner, J.: Case-based reasoning. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (1993)
17. Leake, D.: Case-based reasoning: Experiences, lessons and future directions. MIT Press Cambridge, MA, USA (1996)

The Case-Based Experience Web*

Barry Smyth[1], Pierre-Antoine Champin[1,2], Peter Briggs[2], and Maurice Coyle[2]

[1] CLARITY: Centre for Sensor Web Technologies
School of Computer Science and Informatics
University College Dublin, Ireland.
barry.smyth@ucd.ie
http://www.www.clarity-centre.org
[2] LIRIS, Université de Lyon, CNRS, UMR5205,
Université Claude Bernard Lyon 1, F-69622, Villeurbanne, France
pchampin@liris.cnrs.fr

Abstract. With the rise of user-generated content (blogs, wikis, ratings, reviews, opinions etc.) the web is evolving from a repository of content into a repository of experiences, and as it evolves there are many opportunities to harness these experiences. In this paper we consider some of the challenges associated with harnessing online experiences by adopting a case-based reasoning perspective, and highlighting how existing case-based approaches might be adapted to take advantage of this new world of the experience web. To make this discussion more concrete we will draw on examples from one recent case-based attempt to harness the experiences of communities of users in the area of web search.

1 Introduction

The *Social Web* [9] reflects an important change in the nature of the web and its content. Since 1999, the rapid growth of blogs, as a simple way for users to express their views and opinions, ushered in this new era of *user-generated content* (UGC) as many sites quickly began to offer a whole host of UGC alternatives including the ability to leave comments, write reviews, as well as the ability to rate or vote on the comments/opinions of others. The result has been an increased emphasis on people rather than content and, in combination with social networking services, this has precipitated the growth of the *social web* as a platform for communication and collaboration. What has all of this got to do with *experiences*? The essential point is that the combination of traditional web content (the *digital artifacts* that are the fundamental units of conventional web content) with the opinions, views, and ratings of users, is the very stuff of experiences [10]; see Fig. 1. More generally, the combination of a digital artifact plus its *usage information* is an experience repository. Usage information encompasses not only the *explicit* forms of user-generated content mentioned above but also the *implicit* usage information, such as the navigation trails and/or search queries that led to a particular digital artifact, usage information that is recorded within server logs. Echoing the views of [10], the web is the ultimate decision-support tool and there is a significant opportunity to harness these

* This work is supported by Science Foundation Ireland under grant 07/CE/I1147, the French National Center for Scientific Research (CNRS), and HeyStaks Technologies Ltd.

many and varied types of experiential knowledge in order to help people make the right decisions using the right information at the right time.

Fig. 1. The Evolution of the Experience Web.

The ability to harness and reuse these online experiences has tremendous potential and the purpose of this paper is to explore how we might go about meeting this challenge, and to what end. Our starting point is the research of the *case-based reasoning* (CBR) community, where more than 20 years of research effort has been devoted to exploring different aspects of *reasoning from experiences* [1, 5, 17]. Our aim is to identify challenges and pose questions rather than to propose answers through fully worked solutions. Specifically we wish to consider the type of tools and techniques that need to be developed in order to support (personal) experience reuse as a basic web service, in much the same way as web search is a basic service today. In particular we will focus on 3 core challenges as follows:

1. *Capturing Personal Experiences.* How might we capture, organise, and share the online experiences of web users? How can current tools and applications be augmented to accommodate experience capture and reuse, leading to the creation of shared personal case bases.
2. *Coping with Noise.* Personal experience creation is a departure from traditional approach to *expert-led* CBR (in which cases are created by domain experts or as a direct result of expert problem solving). Capturing the ad-hoc experience of individuals introduces a significant quality risk. Not only are our opinions and views subject to change, but the way they are collected on the web may not always reflect our own perception, introducing a considerable amount of noise. How might an experience reuse system cope with repositories of experiences that are extremely noisy?
3. *Reuse in Context.* How can we leverage the right experience at the right time and in the right context, bearing in mind that relevant experiences may be distributed across multiple cases or even case bases? In particular, understanding the provenance of a case and the reputation of the case creator — while dealing with

the attendant privacy issues — will play a significant role in the development of *experience-based interfaces* that will integrate experience reuse into our everyday tasks.

In what follows we will make our discussion concrete by drawing on a particular implementation of one attempt to harness the experience web in the area of web search. HeyStaks is a *search utility* that is designed to work with mainstream search engines by allowing users to organise, share and reuse their particular search experiences; see www.heystaks.com for a live beta. It comes in the form of a browser toolbar and back-end server to provide users with an experience-based web search support system that is fully integrated with Google. While it will not be possible to describe the technical details behind HeyStaks the interested reader is referred to [13] for further information.

2 Challenge 1 - Capturing Personal Experiences

Experience-like information is now commonplace on the web, as many sites and services attempt to supplement their core content with the opinions, ratings, and comments of users. The challenge for end-users is that these experiences are often ad-hoc and usually fragmented. As a user my opinions and reviews are thinly spread across many different sites and the trusted opinions of my social network are all but invisible to me. How might we support individual users when it comes to the creation and sharing of their personal experiences? As a user, I want to be able to keep track of my experiences and the relevant experiences of my trusted friends and colleagues. I want to be reminded of similar experiences as I interact with services online; if I am booking conference accommodation (through the conference web site) I would like to be reminded if I have stayed at a particular hotel before, or if one of my colleagues has stayed there, especially if the experience was good or bad.

To meet this challenge there is the need for common representational formats as a way to represent digital artifacts; this has been a long-time goal of semantic web initiatives [2]. In addition, experience creation, organisation and sharing needs to be built into the very fabric of the web, and the tools that we use to interact with web services. In short, there is a need for experience creation and management tools that are as much part of the web experience as the browser and search engine are today. This is in contrast to the work of the case-based reasoning community which, to date, has focused on the the provision of dedicated CBR tools. These tools are mainly designed to be used by domain experts, allowing for the creation of standalone CBR systems and case bases. If we are to incorporate experience reuse into our online-lives then a different sort of approach is needed, one that sees experience management fully integrated into the many and varied tools and services that we naturally use, from search engines and portals to e-commerce services to online word processors etc. [8].

HeyStaks addresses these challenges in the domain of web search by allowing users to create repositories for search experiences related to a particular topic or task. Each repository is called a *search stak* and is effectively a case base of search cases. Each case corresponds to a single result page that has been 'selected' for this stak during a user's searches. Each case is anonymously associated with a number of implicit and explicit

interest indicators, including: the total number of times the result has been selected during a search, the query terms that led to its selection, the snippet terms associated with the result when it was selected, the total number of times a result has been tagged and the terms used to tag it, the total votes it has received, and the number of people with whom it has been shared. In addition each term (query, tag, snippet) is linked to a hit-count that reflects the number of times that this term has been associated with the page in question.

For example, Fig. 2-left shows how a user can use the HeyStaks toolbar to create a search stak to capture their searches related to the experience web. Then, as they search they can select a suitable stak prior to, or during a search, as a way to ensure that the current search experience is stored within the appropriate stak; see Fig. 2-right.

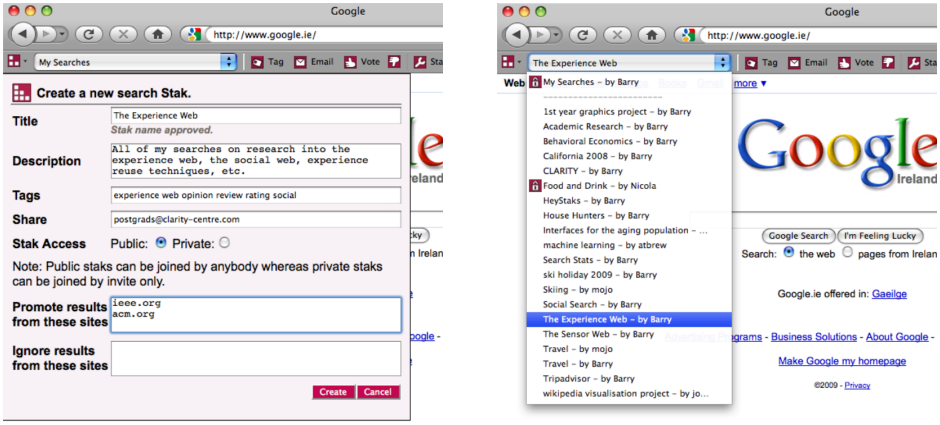


Fig. 2. Creating a new search stak (left) and selecting a search stak prior to search (right).

In this way users can create and share different repositories of search experiences. As they browse and search, these repositories are enriched with additional searches. For example, while browsing users can use the HeyStaks toolbar to vote on any particular page, or they can tag a page or share it with a friend. As they vote, tag, and share this information (tag terms, votes) is associated with the page in question in the current stak. In turn, as users search, their result click-thrus are taken as implicit indicators of interest so that click-thru frequency information is also associated with a given result for a given query in the current stak. Moreover, the HeyStaks toolbar augments conventional search result-lists to provide access to tagging, voting, and sharing actions at the level of individual results: as the user mouses-over individual results popup HeyStaks icons provide access to voting, tagging, and sharing features as shown. HeyStaks also *promotes* a number of results, which we will discuss in more detail in section 4.

The power of staks as search experience repositories comes to be fully felt when they are shared with others. This facilitates the aggregation of search experiences across groups of friends and colleagues. In the case of our Experience Web search stak, by sharing it with a group of interested searchers this stak will be added to their own

toolbar, and will therefore quickly grow to accumulate a significant store of related search experiences as the basis for targeted promotions during future searches by stak members. In this way, stak members will benefit from results found by other members for similar queries in the past. In a recent beta deployment across 95 users, over a 3 month period, we found stak sharing to be commonplace. The average user created 3.2 search staks and joined a further 1.4 staks and 70% of users shared staks with at least some other users; see [13] for further information.

3 Challenge 2 - Coping with Noise

In the previous section we argued for experience management and creation tools as a necessary feature of future web infrastructure, and we described HeyStaks as a point example of how this has been achieved in the context of web search. Providing for the capture of online experiences will ensure that our experience repositories grow quickly to reach some critical mass, especially if these experiences are created based on implicit as well as explicit actions and activities. For example, in the case of HeyStaks, every search by a user that results in at least one result click-thru is translated into an experience (search case). The problem now becomes one of quantity versus quality and, specifically, the extent to which these experiences will serve as a reliable basis for future actions and decision making.

For example it will not always be possible to infer the right context for a given experience so the resulting experience may be misrepresented or misclassified. This is a problem in HeyStaks, exacerbated by the need for users to select the current search stak at search time. If the user does not select the correct stak then their new search will be stored in whatever stak happens to be active at search time. This is in part addressed by using recommendation techniques to automatically select an appropriate stak (from the user's stak list) based on their current search query. Reliable recommendations cannot always be made, however, especially if there are few experiences in the user's staks as the basis for query matching, and so searches continue to be misclassified by HeyStaks.

Thus, a key challenge when it comes to personal experience capture and management concerns the ability to deal with potentially significant levels of noise. For a number of years the case-based reasoning community have looked a variety of techniques for editing experiences, under the heading of *case-base maintenance*; see for example the work of [4, 6, 7, 16]. However, existing techniques are usually designed to manage case bases with relatively low amounts of noise and work best with cases where there is an objective measure of when a case can be used to correctly *solve* some target problem. These assumptions are less likely to hold in the experience web. For example, in HeyStaks we currently find a significant number of search experiences to be stored in an incorrect stak, largely due to failings in the stak recommendation feature mentioned above, and because many users "forget" to select an appropriate stak at search time.

One technique for coping with high-levels of experience noise is to identify what we call the *experience kernel*, a notion related to the notion of a *competence footprint* in the work of [14, 15]. In the case of HeyStaks this is a subset of pages (cases) that are assumed to be relevant to that stak. Experience kernels can be computed using a variety of techniques. Because of the extent of noise within search staks we have explored a

number of different approaches based on the clustering of cases into meaningful groups of related experiences. For the purpose of this paper we will briefly review the simplest *term-based clustering* approach here. Two query terms used in a stak are considered related if at least n pages contain both of these terms[3]. Briefly, we use a complete linkage clustering algorithm [12] to build clusters of related terms according to this measure. Then, we remove all clusters containing less than s terms. The kernel of the stak is composed of all the pages in that stak that were retrieved with at least one term from the remaining (i.e. large enough) clusters.

An interesting feature of this method is that, although it retains a small number of terms (at most 15% for some staks, and 5% globally), it keeps a reasonably high number of pages in the kernel. For example, the histogram in Fig. 3 shows the relative number of pages contained in a subset of staks, including the size of each stak's kernel. The relative size of the kernels gives a clear indication of the amount of (potential) noise that is contained within staks with some staks being dominated by potential noise while others enjoy much larger kernels.

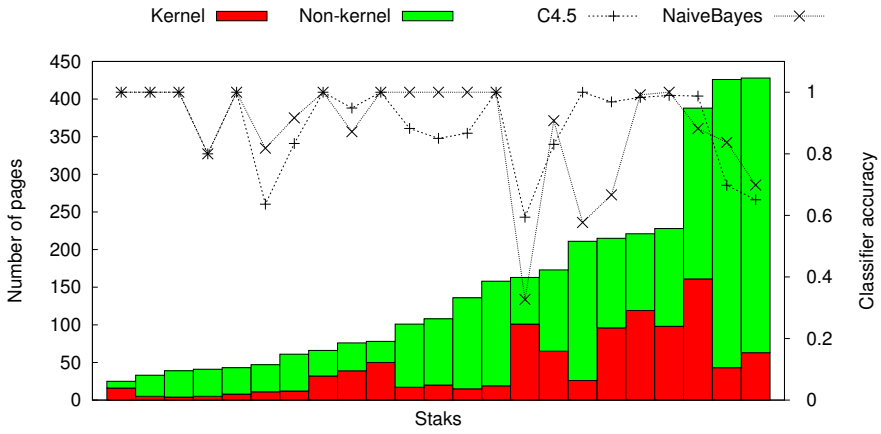


Fig. 3. Number of kernel and non-kernel pages (histogram) and accuracy of the classifiers, for each test stak (lines).

The ability to identify reliable experience kernels leads to a number of ways to improve the manner in which experiences are captured and organised. For example, we can construct a classifier from the experience kernels and use this classifier to predict the right stak for a given search experience. In this scenario each instance corresponds to a page from the relevant stak kernel with the stak id used as the class. For instance, from the staks described in Fig. 3 we constructed a C4.5 [11] decision tree and a naive bayesian classifier, trained with the kernel pages, to use as test classifiers. A standard 10-fold cross validation evaluation delivers stak-by-stak classification results shown as line-plots in Fig. 3; the average accuracy is 89% for the decision tree and 83% for the naive bayesian classifier.

[3] More precisely, n is weighted by how many times each term was used to retrieve each page.

The predictive power of both classifiers is generally good. Both mostly agree on the "difficulty" of certain staks. This suggests that the kernel building technique is capable of identifying collections of core pages that are at least reasonably predictable within a stak, and lends confidence to the prospect of using the classification approach as a way to associate non-kernel pages with their "correct" staks as part of a maintenance process.

These preliminary results merely scratch the surface of some of the maintenance challenges associated with the experience web. They serve to highlight the potential for high degrees of noise in personal experience repositories where the inadvertent actions of the user can lead to experiences being misclassified. They also point in the direction of a potential solution since if experience kernels can be reliably identified then they can also be used to guide experience maintenance. In the future it may be interesting to quantify how mature or consensual an experience repository (stak) has become, or on the contrary to detect when it is subjected to an abrupt change. This kind of information could provide additional context when using these experiences for recommendation.

4 Challenge 3 - Reuse in Context

In this section we consider some of the reuse challenges that must be addressed when it comes to actually putting stored experiences to good use. Once again there is an integration challenge, related to how relevant experiences might be incorporated into a particular application interface. In addition, there is the obvious challenge of experience selection (and ranking) as the right experiences need to be chosen at the right time and in the right context. Moreover, we note that when experiences are created and shared within communities of users, experience reuse creates a new form of collaboration network between community members.

Search experiences can be reused as a way to recommend actual result-pages during web search, and this is the approach adopted by HeyStaks: individual pages that have been frequently selected, tagged or voted on, for similar queries are highlighted, promoted, or inserted directly into the Google result-list by the HeyStaks toolbar. Briefly, to generate these promotion candidates, HeyStaks uses the current query as a probe into each stak, to identify a set of relevant cases. Each candidate case is scored using a similar technique to that described by [3] by using a TFIDF (*term frequency • inverse document frequency*) function as the basis for an initial recommendation ranking; this approach prefers cases that match terms in the query which have occurred frequently in the case, but infrequently across the case base as a whole; see also [13] for a more detailed analysis of HeyStaks' promotion mechanism.

In addition, however, HeyStaks is designed to explore another form of reuse, at the level of the case base, rather than the individual case. In this context, for a given target query, and in addition to the results that may be promoted from the currently active stak, HeyStaks will also consider experiences that are stored in other staks that the user is a member of, with a view to identifying relevant experiences in these alternative contexts. Consequently, HeyStaks can also recommend to the user a list of alternative staks as a source of further recommendations. Indeed, HeyStaks can also recommend public staks

from the wide HeyStaks community — staks that the user has not yet joined — if they also contain similar experiences to the current search context.

Many of the recommendations that are made to the current user may come from their own personal search histories but many also come from the search histories of other users who also participate in their shared staks. This type of collaboration is commonplace within HeyStaks as the results of a recent beta deployment demonstrate. For example, a *net producer* is defined as a user who has *helped* more other users than they themselves have been helped by; in other words they have contributed more search experiences, which have been reused by others, than they themselves have reused. Conversely, a *net consumer* is defined as a user who has been helped by more users than they themselves have helped; in other words they tend to benefit a lot from the experiences of others but don't contribute many new experiences of their own for others to benefit from. For example, [13] highlights how 47% of HeyStaks users are net producers (that is, almost half the users are helping others, by their search experiences, more often than they themselves are helped in return) and, when we look at the promotions that users actually select during their searches we find that, on average, 33% of these are so-called *peer promotions*, promotions that are directly derived from the experiences of others, where as 66% are so-called *self promotions*, promotions that come from the searcher's own personal experiences. In this way, experience reuse in HeyStaks pervades conventional web search as results and staks (case bases) are suggested, on the fly, to searchers.

5 Conclusions

The web provides a rich source of explicit and implicit experiences but, by and large, these experiences are either diluted across a great many different sites and services, or never captured in the first place. From a case-based reasoning perspective we know how to represent and reuse experiences and so there is considerable opportunity for the CBR community to turn its attention to the web as a new source of experiential knowledge that is just waiting to be harnessed. In this paper we have taken the first tentative steps in this regard, in an attempt to explore this type of experience reuse, and the challenges that it presents.

Our vision is one that reflects a bottom-up approach to experience reuse. We have argued the need for experience capture and reuse facilities to be integrated into our online tools and services, so that individual users can benefit from their own past experiences to begin with. We have also argued the need for experiences to be shared among groups of related users and interested parties, so that people can benefit from aggregate community experiences. We have also highlighted issues of reliability and noise when it comes to personal experience capture and argued the need for new techniques to cope with high degrees of experience noise that would be considered to be unusual in a conventional expert-created case base.

Throughout the paper we have attempted to provide concrete examples with reference to one particular online experience reuse system that has been deployed in the domain of Web search. As such, the HeyStaks system illustrates many of the points that have been made. It integrates experience capture and reuse as part of the traditional web search interface and allows for the creation and sharing of personal search experiences.

These experiences are inherently noisy and we have described one approach to coping with this noise by identifying experience kernels within a case base. Finally we have demonstrated how current HeyStaks users are benefiting from their own search experiences and those of others, leading to an effective form of search collaboration as a side-effect of experience reuse and sharing.

References

1. A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, 1994.
2. T. Berners-Lee. Semantic web roadmap. Draft, W3C, Sept. 1998.
3. O. Boydell and B. Smyth. Enhancing case-based, collaborative web search. In *ICCBR*, pages 329–343, 2007.
4. S. Craw, S. Massie, and N. Wiratunga. Informed case base maintenance: A complexity profiling approach. In *AAAI*, pages 1618–1621, 2007.
5. B. Leake, David. *Case-Based Reasoning: Experiences, Lessons and Future Directions*. MIT Press, Cambridge, MA, USA, 1996.
6. D. B. Leake, B. Smyth, D. C. Wilson, and Q. Yang. Special issue on case-based maintenance. Computational Intelligence, 17(2) 2001.
7. D. B. Leake and D. C. Wilson. Categorizing case-base maintenance: Dimensions and directions. In *EWCBR*, pages 196–207, 1998.
8. A. Mille. From case-based reasoning to traces-based reasoning. *Annual Reviews in Control*, 30(2):223232, Oct. 2006. Journal of IFAC.
9. C. Perey and D. Hazael-Massieux, editors. *W3C Workshop on the Future of Social Networking*, Barcelona, Jan. 2009. `http://www.w3.org/2008/09/msnws/`.
10. E. Plaza. Semantics and experience in the future web. In *ECCBR*, pages 44–58, 2008.
11. J. R. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.
12. C. Romesburg. *Cluster analysis for researchers*. Lulu. com, 2004.
13. B. Smyth, P. Briggs, M. Coyle, and M. P. O'Mahony. Google? shared! a case-study in social search. In *User Modeling, Adaptation and Personalization*. Springer-Verlag, June 2009.
14. B. Smyth and M. T. Keane. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *IJCAI*, pages 377–383, 1995.
15. B. Smyth and E. McKenna. Competence guided incremental footprint-based retrieval. *Knowl.-Based Syst.*, 14(3-4):155–161, 2001.
16. B. Smyth and E. McKenna. Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001.
17. I. Watson. *Applying case-based reasoning: techniques for enterprise systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

Case-based Reasoning for Computer Games

Workshop at the
Eighth International Conference on
Case-Based Reasoning
(ICCBR 2009)

Seattle, Washington, USA
July, 2009

Luc Lamontagne
and Pedro González Calero (Eds.)

Co-Chairs

Luc Lamontagne
Université Laval, Canada
luc.lamontagne@ift.ulaval.ca

Pedro González Calero
Complutense University of Madrid, Spain
pedro@sip.ucm.es

Programme Committee

David Aha, Naval Research Laboratory, USA
Hans-Dieter Burkhard, Humboldt University Berlin, Germany
Pádraig Cunningham, University College Dublin, Ireland
Belén Díaz-Agudo, Complutense University of Madrid, Spain
Babak Esfandiari, Carleton University, Canada
Hector Muñoz-Avila, Lehigh University, USA
Santi Ontañón, Georgia Institute of Technology, USA
Ashwin Ram, Georgia Institute of Technology, USA

Preface

The motivation for this workshop was to encourage the exchange of information and ideas about CBR as it is embedded within and provides support for computer gaming environments. Computer games are receiving increasing attention as a means for testing CBR concepts and to extend current CBR paradigms (e.g. real-time issues, uncertainty, online learning). The goals of this workshop have been to:

1. Provide a medium of exchange for information on games-related CBR research.
2. Provide an opportunity for participants to demonstrate some game-related CBR prototypes and hence to illustrate some of the challenges and issues faced by CBR researchers.

Seven papers were accepted for this workshop. Various gaming environments were represented in the contributions including real-time strategy games, RoboCup game, football simulation, and Texas Holdem poker. And technical contributions addressed themes such as real-time control strategies, character decision making, opponent modeling and behavior authoring.

Floyd and Esfandiari present a learning-by-demonstration problem in the RoboCup simulation environment and present an evaluation that shows that an example-based classifier outperforms a decision tree, and SVM and a Naive Bayes classifier on the problem.

Laviers et al. describe a CBR and SVM-based approach to recognizing adversary plans in the context of the Rush 2008 game, a football simulator. They exploit the spatio-temporal structure of the team behaviors and make use of support vector machines as a technique to classify and recognize the opponent's defensive play.

Mehta et al. propose a case-based planning approach to guide a gaming system to learn behavior sets from its interactions with a human user. By learning from demonstration, the authors aim to ease the process of defining game characters and to reduce the amount of coding necessary to program their behavior.

Rubin and Watson submitted two contributions to this workshop. In their first paper, they give an overview of a case-based poker-playing agent with a similarity metric that is comprised of the betting sequence, the quality of the hand as well as the quality of the community cards. In their second paper they tackle the problem of general game playing and propose to address this through analogy and lazy learning techniques. The paper explores various types of memories and their potential use for this problem as well as lazy learning techniques.

The paper by Sanchez-Ruiz et al. proposes the use of abstract cases automatically generated through planning to assist game designers when building Behaviour trees, an expressive mechanism that let designers author complex behaviours along the lines of the story they want to tell without requiring programming knowledge.

Szczepanski and Aamodt propose a CBR approach to control, at a micromanagement level, the actions of individual units of a real-time strategy game (Warcraft 3). They propose various strategies for case matching and action adaptation. They also report on experiments conducted on different testing configurations and compare their approach to scripted computer opponents.

We wish to thank all who contributed to the success of this workshop, especially the authors, the Program Committee, and Sarah Jane Delany, the ICCBR09 Workshop Coordinator.

Luc Lamontagne July 2009
Pedro González Calero

Comparison of Classifiers for use in a Learning by Demonstration System for a Situated Agent

Michael W. Floyd and Babak Esfandiari

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, Ontario

Abstract. In *learning by demonstration* systems, learning is performed by an agent observing how an expert behaves in response to a given input. The learning task is more difficult when the agent is situated in a dynamically-changing environment, especially when only a partial view is available at any time. In this paper we propose a comparison of different learning by demonstration techniques, namely case-based reasoning, decision trees, support vector machines and naive bayes classifiers, in the dynamic environment constituted by the RoboCup robotic soccer simulation. An initial look at the results indicate that our case-based reasoning algorithm behaves well across all experiments and outperforms the other classifiers. Case-based reasoning even outperforms a decision tree classifier when learning from an expert whose internal reasoning is represented as a decision tree.

1 Introduction

Case-based reasoning has been successfully applied to a variety of gaming applications [1–5] and, more recently, has been used to learn to play a game by watching an expert play. In these *learning by demonstration* systems, cases are generated by observing how an expert behaves in response to the current state of the game and then behaving similarly when presented with similar game states. Using case-based reasoning for learning by demonstration has been used in a variety of games including robotic soccer [6, 7], real-time strategy [8–10] and Tetris [11].

In some games, like chess or Tetris, the game playing agent will have a complete world view at all times and will be fully aware of the state of the environment. However, in many real-world domains the agent will only have a partial world view. This is common when the agent is able to move around a large, dynamic environment. If the agent is only able to view a portion of the environment at a given time, it will never be fully aware of the state of the environment at all times.

This partial world view can pose a challenge since the environmental stimuli may not be consistent over time. For example, as the agent moves around the environment objects may move in and out of its field of vision. Also, there is

no guarantee that any specific object will be visible to the agent at a particular moment in time. In fact, the agent may never know how many objects of any kind are present in total in the environment. One approach that has been used to overcome this is to have other agents provide information about unseen areas of the environment [3] although this requires multiple cooperating agents that are adequately distributed over the environment.

This difficulty in dealing with data from a partial world view is not limited to case-based reasoning, but exists in most learning by demonstration work. Typical approaches to deal with a partial world view include only using omnipresent objects as inputs [12], ignoring external stimuli [13], or only reasoning using commonly visible objects [14, 15]. Finding a reasoning technique that does not require such limitations or restrictions would be a preferable solution as it would allow for learning by demonstration even if the agent only had a partial world view.

The goal of this paper will to be compare case-based reasoning to several other classifiers to determine which is most applicable to a learning by demonstration system that receives the sensory information of an agent with a partial world view as inputs. In Section 2 we describe the raw data available when observing an expert perform a task and Section 3 presents a method to transform the raw data so it can be used by common classifiers. Section 4 provides the experimental results and, finally, Section 5 discusses the conclusions we can draw from those experiments.

2 Sensory Stimuli

When a software agent or a robot is situated in a large, dynamic environment it will likely have a partial, and always changing, world view. For example, the field of vision of a soccer player is shown at three points in time in Figure 1. At different points in time, as shown in Table 1, the player observes different objects. Additionally, the player may not have the ability to uniquely identify objects. For the soccer player, this would mean it would be unable to differentiate between the various *players* or know if it is observing the same *ball* at the different points in time.

Each sensory stimulus, S, of an agent can then be modelled as a collection of multi-valued attributes, V_i. If the agent is able to differentiate between n types of objects, there will be n multi-valued attributes. Going back to the soccer example, there would be two types of objects: balls and players.

$$S = \{V_1, \ldots, V_n\} \tag{1}$$

The multi-valued attribute for each type of object contains the objects, of that type, that are currently visible to the agent. Each of these multi-valued attributes is an unordered set, with the size of that set being variable. This variability in set size can cause different types of objects to have different sized sets but can also cause the same object type to have differently sized sets at different points in time if objects move in or out of the agent's field of vision.

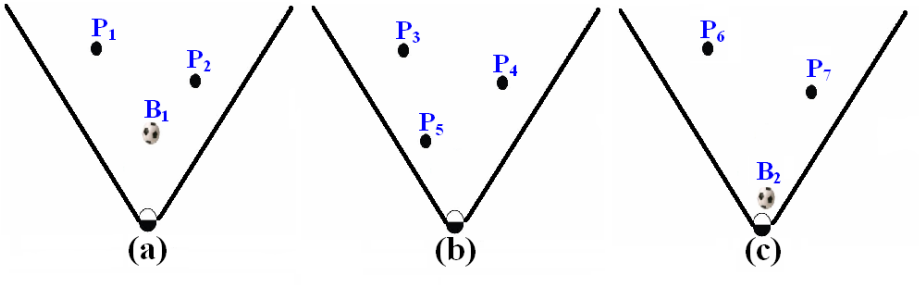


Fig. 1. A graphical representation of a soccer player's field of vision at three different points in time.

Field of Vision	Balls	Players
(a)	B_1	P_1, P_2
(b)		P_3, P_4, P_5
(c)	B_2	P_6, P_7

Table 1. The attributes of each field of vision.

3 Data Transformation

Most classification algorithms expect each feature to be single-valued. We therefore need to break down our multi-valued features into sets of single-valued ones, while ensuring a systematic way for each value of a stimulus to be mapped to an appropriate feature. This can be defined as a problem of matching the visible objects in a stimulus, $S = \{o_1, \ldots, o_m\}$, to a set of single-valued features, $F = \{f_1, \ldots, f_n\}$ (where $m \leq n$).

If, in the soccer example, there were always exactly one *ball* and one *player* visible in every stimulus then a feature vector could be constructed that contained two objects. The first object in the feature vector, representing the *ball*, could always be compared to the first object in other feature vectors. However, if there were multiple instances of each type of object then a correspondence problem arises. If a pair of stimuli each contain two *player* objects, it then becomes necessary to determine how those objects will be matched when comparing the stimuli. For example, we would need to determine if the first player in the first stimulus would be compared to the first player or the second player in the second stimulus. This correspondence problem is further complicated by the fact that

Ball	Distance	Direction
B_1	3	0°
B_2	1	10°

Table 2. The attributes of each ball.

different stimuli could have different numbers of objects of a given type. When comparing two stimuli this can result in objects that do not have a matching object to be compared to. An example where this correspondence issue would be a problem is in a decision tree. If a node of the tree makes a decision based on a single object of a specific type, there is no way to know which object of that type should be used.

There has been some work examining how data with multi-valued attributes can be used in a top-down induction of decision trees (TDIDT) algorithm [16–18]. In these approaches, when traversing the tree all possible valid paths are considered. When a test input contains a multi-valued attribute, any decisions based on that attribute are made using all of the values of the attribute so that several paths through the tree are taken simultaneously. This results in examining a series of paths through the tree that only involve a single value of each attribute and then combining the results. Since these approaches only examine a single object of each type during each traversal of the tree, they are not applicable in applications where information from multiple values of a multi-value attribute are needed. In the soccer example, this implies that only a single *player* object would be considered during each traversal of the decision tree, so no decisions involving multiple players would be possible.

Another possible approach would be to convert Table 1 into first normal form. This would involve replacing the *Balls* and *Players* columns with separate tables that contain a single row for each object. Using normalized relational tables, inductive logic programming (ILP) [19] could be used for multi-relational classification [20, 21]. However, similar to the TDIDT techniques described above these, ILP approaches treat each row containing a multi-valued attribute as multiple rows containing single-valued attributes, thereby not allowing all of the multiple values of an attribute to be used simultaneously.

In order to make this data usable by most classifiers, we will transform the data into a form that mirrors the biases imposed by our existing case-based reasoning system [6, 7]. In our previous work we have used case-based reasoning to control the behaviour of a soccer agent. The agent receives sensory input through its field of vision, which contains the positions of the objects that are visible to the agent. The current field of vision is then compared to a case base where each case is composed on a field of vision and an associated action. Using the most similar cases, the agent is then able to select an action to perform. Since the case base is created by observing a teacher, the agent should ideally perform the same actions as the teacher given similar sensory inputs. The case-based reasoning system adds two primary biases:

1. **Set Ordering:** In order to provide an ordering on the multi-valued attributes, objects are matched with similar objects when comparing two sensory stimuli. If there are an unequal number of objects then some objects may not have a match. This does not provide a fixed ordering of the objects, since the matching can be different depending on the other sensory stimulus that is being compared to.

2. **Extra Objects:** When calculating the distance between two sensory stimuli, an object that does not have an appropriate matching object in the other sensory stimulus results in a penalty value being added. This penalizes stimuli containing a different number of objects.

We will attempt to keep similar biases in the transformed data so as to avoid giving an advantage to any of the classifiers during the experimental comparison. The data transformation involves the following steps:

1. **Fixed Sized Feature Vector:** Each sensory stimulus will be represented as a feature vector of a fixed length. Initially, a set of training stimuli will be examined to determine the maximum size, M_i, of each of the multi-valued attributes. The feature vectors will then be created so that each attribute is able to contain the maximum number of values. If there are N attributes, the vector will be able to hold $\sum_{i=1}^{N} M_i$ objects. Since each object is complex and has both a distance and a direction, relative to the agent, the feature vector will actually be twice that size ($Vectorlength = 2\sum_{i=1}^{N} M_i$). For example, the first $2 \times M_1$ values of the vector will contain data related to the 1st multi-valued attribute.
2. **Object Ordering:** Objects are ordered by sorting them based on their distance from the agent. Objects that are closer to the agent will be placed in the feature vector before more distant objects of the same type (when two feature vectors are compared the objects closest to the agent will be compared to each other, second closest objects compared, etc.). This does not allow for more precise object matching, since the ordering is performed in advance, but is similar to the matching performed by our case-based reasoning system since it is computationally inexpensive and can be used in real-time.
3. **Normalization:** Similar to how a training set was mined to find the maximum size of each multi-valued attribute, it will also be mined to find the minimum and maximum values of the object distances and directions. These maximum and minimums will be used to normalize the distances and directions, between *0 and 1*, so that all elements of the feature vector are of a similar scale.
4. **Padding:** If a sensory stimulus contained fewer than the maximum number of values for any of the multi-valued attributes, the remaining entries in the vector will be padded with values that represent *unseen objects*. These values are used to mirror the penalty values used by the case-based reasoning system, since a visible object will have values that are dissimilar to the unseen object values.

4 Evaluation

Our experiments will look to compare the performance of our existing case-based reasoning system [6, 7] with three popular classification methods: J48 decision trees, SMO support vector machines and naive bayes. For these three algorithms the Weka [22] implementations were used.

4.1 Experimental Setup

The data will be generated by observing simulated robotic soccer players in the RoboCup Simulation league [23]. Two teams of players will be observed: Krislet agents [24] and CMUnited agents [25]. Krislet agents behave in a simple manner. They turn until they can see the soccer ball and then run toward the ball. When they get to the ball they attempt to kick it toward their opponent's goal. Krislet was selected because it is a simple, reactive team that should be easily represented as a decision tree. CMUnited is far more complex and was the former champions of the RoboCup Simulation League. They use a layered learning architecture and a number of strategies including formation strategies and agent communication. CMUnited players can have multiple states of behaviour and maintain internal models of the world, so their behaviour is likely more similar to that of a human expert.

The agents were observed while playing games of simulated soccer. During the games, each team was comprised of *11* players per team and the opposing team was always made of Krislet agents. Both Krislet and CMUnited agents were observed playing *25* complete games of soccer resulting in approximately *100000* observations being collected per team. Each classifier was trained using *5000* randomly selected observations[1] and tested using 10-fold cross validation. This testing was performed *25* times, for both the Krislet and CMUnited data, using each classifier.

The complete set of collected observations was mined in order to determine the maximum number of objects of each type visible during an observation and the maximum and minimum distance and direction values to use for normalization. The maximum and mean occurrences of each type of object[2], in the Krislet data, are shown in Table 3. Therefore, each stimulus for the Krislet data will be represented by a vector of length *114* (57 object each with a distance and direction value). The CMUnited data is similar to the Krislet data except the maximum number of flags is slightly higher. This is because the CMUnited agents do not use the standard sized field of vision, but instead use a wider field of vision (but this wider field of vision also leads to noisier estimates of the position of objects).

The feature vectors were padded with distance and direction values of *-1* when fewer than the maximum number of objects, of a specific type, were visible. This value was chosen since it will never occur in any visible objects since they are normalized between *0 and 1*.

Lastly, each of the classifiers had its performance optimized using feature selection. Using a methodology similar to that used in previous work [7], the most important types of objects were found for each classifier. Data from any objects that did not positively affect the classification performance was removed. It should be noted that, with only a few exceptions, each classifier selected

[1] Approximately the number of observations collected during a single game.

[2] The objects in simulated RoboCup soccer are: soccer ball, flags, boundary lines, goal nets, teammates, opponents and unknown players (their team is unknown due to noise).

the same features to use and those features were consistent with our previous studies[3].

	Ball	Flag	Line	Goal	Team.	Opp.	Unk.	Total
Max	1	16	1	2	10	11	16	57
Mean	0.6	5.8	1.0	0.5	4.4	3.2	1.8	17.3

Table 3. The maximum and mean occurrences of each type of object in the Krislet data.

4.2 Results

Our results measure the ability of each classifier to predict the action the expert would have performed given a similar stimulus. The possible actions are *kicking*, *dashing* and *turning*. Each action also has associated parameters, like the dashing power, but we ignore those and only attempt to get the action correct. For each classifier we measure the performance using the *f-measure*. The f-measure, which is a function of the precision and recall of each action, was selected because it is an acceptable metric to use when data is extremely imbalanced (only around *0.2%* of the data is for the kick action).

Table 4 shows the average f-measure values over the *25* tests. Examining the results from Krislet, we can see that both our case-based reasoning (CBR) algorithm and the J48 algorithm perform best. It was expected that the J48 algorithm would work well, since the reasoning logic of Krislet can be represented as a decision tree, but it is interesting to note that our CBR approach actually performs slightly better (although not a statistically significant difference).

	CBR	J48	SMO	NaiveBayes
Krislet	0.83 +/- 0.002	0.82 +/- 0.005	0.70 +/- 0.003	0.66 +/- 0.017
CMUnited	0.61 +/- 0.002	0.42 +/- 0.007	0.47 +/- 0.011	0.16 +/- 0.024

Table 4. The f-measure results using each classifier.

An initial look at the results in Table 4 indicate that our CBR approach clearly outperforms the other classifiers on the CMUnited data. While we have not thoroughly looked at optimizing the parameters of any of the algorithms,

[3] Most classifiers found the ball to be important for Krislet and the ball, goal and flags to be important for CMUnited. However, naive bayes also found teammates to be important for both Krislet and CMUnited and did not find the goal important for CMUnited.

it is promising to see that our CBR system outperforms the others using default parameters. This is likely due to the complexity of the reasoning used by CMUnited. Whereas the Krislet agent only reasons using the soccer ball and opponent's goal net, the CMUnited agents use a variety of objects. For the balls and goals, there are only ever one or two instances of those objects. Whereas with flag objects there are many of them, so more objects influence the classification process. Additionally, the data does not contain all of the information the CMUnited agents use during reasoning. Things like internal states and inter-agent communication are not included in the data so the stimuli may become more difficult to separate using rules or generalizations. The benefit of our CBR approach is that it keeps the data stimuli unchanged, so important information is not discarded as noise.

Further examination of the results show that the decision trees generated by the J48 algorithm on the Krislet data only contain a few decision nodes. These decisions are quite close to the decisions that the Krislet agent actually uses during reasoning. However, it often underestimates the decision bounds resulting in some misclassification during testing. On the other hand, with the CMUnited data the decision tree uses hundreds of decision nodes. The decisions are often not generalizations of the data, but instead make rules that completely describe individual training instances. Looking at the SMO support vector machines algorithm and naive bayes, they also do fairly well on the Krislet data but have significant trouble on the CMUnited data. Similar to the J48 algorithm, these approaches tended to over train the class boundaries based on the training instances.

While the results we have presented are not from an exhaustive comparison of all possible classifiers, or all combinations of algorithm parameters, they do show our case-based reasoning approach performs well when learning from agents of various complexities. More importantly, our CBR approach is competitive with the J48 algorithm when learning from an agent who reasons using a decision tree.

5 Conclusions

In this paper we examined the data available when an agent only has a partial view of the world and how the objects in its field of vision can change as the agent moves around the environment. This only gives the agent limited information about what it can see and causes a correspondence problem when attempting to use this information in learning by demonstration systems. The primary reason for this correspondence problem is an inability to uniquely identify objects, so the agent only knowns what *type* of object it sees, rather than *which* object of that type it sees.

Our results show that using our case-based reasoning system on such agent sensory data outperforms a variety of other classification algorithms when learning by observing a simulated soccer agent. Our experiments did not attempt to perform an exhaustive comparison of all possible classifiers or use all possible

algorithm parameters, but instead looked to show that our case-based reasoning system performs well on both simple and complex data and requires no transformation of the data. Even if the agent being learnt from can easily be represented by a set of rules or a decision tree, our CBR approach still performed well compared to a classifier more suited to that form of learning.

An additional benefit of an instance based learning approach, like case-based reasoning, is that the input data can be compared with actual stimulus rather than generalizations of the stimulus. This is important when ordering the multi-valued attributes since the objects can be matched using more sophisticated matching algorithms rather than being ordered using a fixed ordering rule (like being ordered by distance to the agent). These findings lead us to believe case-based reasoning is an appropriate technique to use when learning by demonstration as it can be used regardless of the complexity of the teacher and allows for the use of all knowledge contained in the training data since no generalization occurs.

Future work will involve examining a broader range of data sets. By performing a larger study we hope to be able to further identify the type of data our case-based reasoning approach performs well with and also if there is any data our approach performs poorly on. Also, we will look to compare our case-based reasoning approach to a variety of state-of-the-art classification algorithms.

References

1. Molineaux, M., Aha, D.W., Moore, P.: Learning continuous action models in a real-time strategy environment. In: 21st International Florida Artificial Intelligence Research Society Conference. (2008) 257–262
2. Auslander, B., Lee-Urban, S., Hogg, C., Muñoz-Avila, H.: Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In: 9th European Conference on Case-Based Reasoning. (2008) 59–73
3. Ros, R., de Mántaras, R.L., Arcos, J.L., Veloso, M.M.: Team playing behavior in robot soccer: A case-based reasoning approach. In: 7th International Conference on Case-Based Reasoning. (2007) 46–60
4. Watson, I., Rubin, J.: Casper: A case-based poker-bot. In: 21st Australasian Joint Conference on Artificial Intelligence. (2008) 594–600
5. Powell, J.H., Hauff, B.M., Hastings, J.D.: Evaluating the effectiveness of exploration and accumulated experience in automatic case elicitation. In: 6th International Conference on Case-Based Reasoning. (2005) 397–407
6. Floyd, M.W., Esfandiari, B., Lam, K.: A case-based reasoning approach to imitating RoboCup players. In: 21st International Florida Artificial Intelligence Research Society Conference. (2008) 251–256
7. Floyd, M.W., Davoust, A., Esfandiari, B.: Considerations for real-time spatially-aware case-based reasoning: A case study in robotic soccer imitation. In: 9th European Conference on Case-Based Reasoning. (2008) 195–209
8. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: 7th International Conference on Case-Based Reasoning. (2007) 164–178

9. Mehta, M., Ontañón, S., Amundsen, T., Ram, A.: Authoring behaviors for games using learning from demonstration. In: Workshop on CBR for Computer Games at the 8th International Conference on Case-Based Reasoning. (2009)
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: Workshop on Learning Structural Knowledge From Observations at the International Joint Conference on Artificial Intelligence. (2009)
11. Romdhane, H., Lamontagne, L.: Reinforcement of local pattern cases for playing Tetris. In: 21st International Florida Artificial Intelligence Research Society Conference. (2008) 263–268
12. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Fourteenth International Conference on Machine Learning. (1997) 12–20
13. Coates, A., Abbeel, P., Ng, A.Y.: Learning for control from multiple demonstrations. In: 25th International Conference on Machine Learning. (2008) 144–151
14. Thurau, C., Bauckhage, C.: Combining self organizing maps and multilayer perceptrons to learn bot-behavior for a commercial game. In: Proceedings of the GAME-ON Conference. (2003)
15. Grollman, D.H., Jenkins, O.C.: Learning robot soccer skills from demonstration. In: IEEE International Conference on Development and Learning. (2007)
16. Chen, Y.L., Hsu, C.L., Chou, S.: Constructing a multi-valued and multi-labeled decision tree. Expert Systems with Applications **25**(2) (2003) 199–209
17. Chou, S., Hsu, C.L.: MMDT: a multi-valued and multi-labeled decision tree classifier for data mining. Expert Systems with Applications **28**(4) (2005) 799–812
18. Li, H., Zhao, R., Chen, J., Xiang, Y.: Research on multi-valued and multi-labeled decision trees. In: Second International Conference on Advanced Data Mining and Applications. (2006) 247–254
19. Muggleton, S.: Inductive logic programming. Morgan Kaufmann (1992)
20. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artificial Intelligence **101**(1-2) (1998) 285–297
21. Yin, X., Han, J., Yang, J., Yu, P.S.: Efficient classification across multiple database relations: A crossmine approach. IEEE Transactions on Knowledge and Data Engineering **18**(6) (2006) 770–783
22. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2nd edition edn. Morgan Kaufmann (2005)
23. RoboCup: Robocup official site. http://www.robocup.org (2009)
24. Langner, K.: The Krislet Java Client. http://www.ida.liu.se/ frehe/RoboCup/Libs (1999)
25. Stone, P., Riley, P., Veloso, M.M.: The CMUnited-99 champion simulator team. In: RoboCup. (1999) 35–48

Opponent Modeling and Spatial Similarity to Retrieve and Reuse Superior Plays

Kennard Laviers[1], Gita Sukthankar[1], Matthew Klenk[2], David W. Aha[2], Matthew Molineaux[3]

[1] School of EECS, University of Central Florida, Orlando, FL,
{klaviers, gitars}@eecs.ucf.edu
[2] NCARAI, Naval Research Laboratory, Washington, DC,
{matthew.klenk.ctr, david.aha}@nrl.navy.mil
[3] Knexus Research, Springfield, VA,
matthew.molineaux@knexusresearch.com

Abstract. Plays are sequences of actions to be undertaken by a collection of agents, or teammates. The success of a play depends on a number of factors including, perhaps most importantly, the opponent's play. In this paper, we present an approach for online opponent modeling and illustrate how it can be used to improve offensive performance in the Rush 2008 football simulator. In football, team behaviors have an observable spatio-temporal structure, defined by the relative physical positions of team members over time. We demonstrate that this structure can be exploited to recognize football plays at a very early stage. Using the recognized defensive play, knowledge about expected outcomes, and spatial similarity between offensive plays, we retrieve an offensive play from the case base. This play is then (partially) reused to improve an in-progress offensive play. We call this process a *play switch*. Empirical results indicate that spatial similarity is central to play retrieval, and that substituting only a subset of the current play yields greater improvement over a full play substitution.

1 Introduction

To succeed at American Football, a team must be able to successfully execute closely-coordinated physical behavior. Teams rely on pre-existing sets of offensive and defensive plays, or *playbooks*, to achieve this coordinated behavior. By analyzing play history, it is possible to glean critical insights about future plays. In American Football, quarterbacks frequently call *audibles*, changes of play based on an assessment of the opponent's play. This task involves identifying the opponent's play and then selecting a new play for the offensive team.

In physical domains (military or athletic), team behaviors often have an observable spatio-temporal structure, defined by the relative physical positions of team members. This structure can be exploited to perform behavior recognition on traces of agent activity over time. This paper describes a method for recognizing defensive plays from spatio-temporal traces of player movement in the Rush

2008 Football Simulator. Rush 2008 simulates a modified version of American Football and was developed from the open source Rush 2005 game [1].

Using knowledge of play histories, we present a method for executing a *play switch* based on the potential of other plays to improve the yardage gained and their similarity to the current play. From a case-based reasoning perspective [2], this involves retrieving a superior play and adapting it to the current situation. In retrieving a superior play, we show that considering the relative similarity of the current play compared with the candidate play improves performance. Furthermore, we show that limiting the play switch to a subgroup of players is preferable to switching them all.

We begin by describing the Rush Football simulator. Next we describe our play switching approach with a detailed discussion of opposing play recognition, play similarity, and play adaptation. We outline the system that implements these ideas and present an empirical evaluation. We close with related and future work.

2 Rush Football

Football is a contest of two teams played on a rectangular field that is bordered on lengthwise sides by an end zone. Unlike American Football, Rush teams have only 8 players on the field at a time out of a roster of 18 players. The field is 100 yards by 63 yards. The game's objective is to out-score the opponent, where the offense (i.e., the team with possession of the ball), attempts to advance the ball from the line of scrimmage (i.e., the starting position of the ball) into their opponent's end zone. Therefore, an offensive play's success can be measured by the number of yards gained. Offensive plays contain the following positions:

Quarterback (QB): is given the ball at the start of each play, and will initiate either a run or pass to a receiver.
Running back (RB): begins behind the quarterback. The running back is eligible to receive a handoff or pass from the quarterback.
Fullback (RB): serves the same purpose as the RB.
Wide receiver (WR): executes passing routes and is the primary receiver for pass plays.
Offensive lineman (OL): is responsible for preventing the defense from reaching the ball carrier.
Tight end (TE): serves either as a lineman or as a receiver.

A Rush play is composed of (1) a starting formation and (2) instructions for each player in that formation. A formation is a set of (x,y) offsets from the center of the line of scrimmage. By default, instructions for each player consist of (a) an offset/destination point on the field to run to, and (b) a behavior to execute when they get there. Play instructions are similar to a conditional plan and include choice points where the players can make individual decisions as well as pre-defined behaviors that the player executes to the best of their physical capability. Rush includes three offensive formations (power, pro, and split) and

four defensive formations (23, 31, 2222, 2231). Each formation has eight different plays (numbered 1-8) that can be executed from that formation. Offensive plays typically include a handoff to the running back/fullback or a pass executed by the quarterback to one of the receivers, along with instructions for a running pattern to be followed by all the receivers. Defensive plays direct players to certain areas or toward individual offensive players with the goal of tackling the offensive player with the ball.

3 Offensive Play Switches

In American Football, the quarterback often dynamically changes the play based on the defensive formation and their reactions to offensive actions before the beginning of the play. Although Rush does not allow for actions before the play, the Rush simulator allows us to alter the play shortly after it has begun.

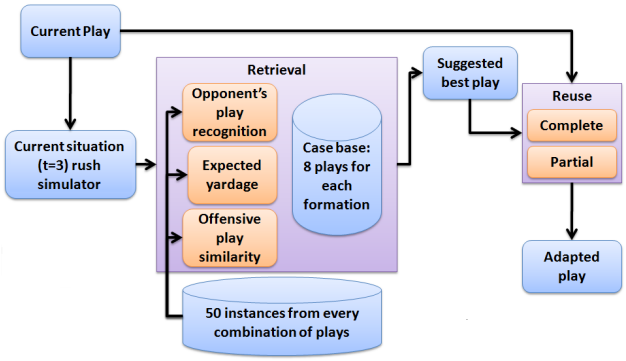


Fig. 1. Play-switching approach.

Our approach focuses on two aspects of case-based reasoning: retrieval and reuse [2]. At this early stage, we are not concerned with the revision or retention of play-switching episodes for future use. Our play switch approach is summarized in Figure 1. Our retrieval method selects an expected best offensive play by quickly recognizing the opponent's play, predicting the results of different offensive plays against it, and computing similarities between each offensive plays and the current situation. The retrieved play is reused by giving new actions to players in the current situation. Retrieval is performed using a case base of 24 plays (i.e., 8 plays for each of the three offensive formations).

The system's background knowledge includes 50 instances of every offensive and defensive play combination. These instances are used to train the recognition system, generate an expected yardage table for every combination of plays, and compute similarity between the offensive plays. The next sections describe the play recognition and similarity metric used in retrieval, followed by a discussion of how the retrieved play is adapted for the current situation.

3.1 Play Recognition using SVMs

Given a series of observations, our goal is to recognize the defensive play as quickly as possible in order to maximize our team's ability to intelligently respond with the best offense. Thus, the observation sequence grows with time unlike in standard offline activity recognition where the entire set of observations is available. We approach the problem by training a series of multi-class discriminative classifiers, each of which is designed to handle observation sequences of a particular length. In general, we expect that the early classifiers will be less accurate since they are operating with a shorter observation vector and because the positions of the players have deviated little from the initial formation.

We perform this classification using support vector machines [3]. Support vector machines (SVM) are a supervised algorithm that can be used to learn a binary classifier; they have performed well on a variety of pattern classification tasks, particularly when the dimensionality of the data is high (as in our case). Intuitively an SVM projects data points into a higher dimensional space, specified by a kernel function, and computes a maximum-margin hyperplane decision surface that separates the two classes. Support vectors are those data points that lie closest to this decision surface; if these data points were removed from the training data, the decision surface would change. More formally, given a labeled training set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$, where $\mathbf{x}_i \in \Re^N$ is a feature vector and $y_i \in \{-1, +1\}$ is its binary class label, an SVM requires solving the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

constrained by:

$$y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + \mathbf{b}) \geq 1 - \xi_i,$$
$$\xi_i \geq 0.$$

The function $\phi(.)$ that maps data points into the higher dimensional space is not explicitly represented; rather, a *kernel* function, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$, is used to implicitly specify this mapping. In our application, we use the popular radial basis function (RBF) kernel:

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0.$$

Several extensions have been proposed to enable SVMs to operate on multi-class problems (with k rather than 2 classes), such as one-vs-all, one-vs-one, and error-correcting output codes. We employ a standard one-vs-one voting scheme where all pairwise binary classifiers, $k(k-1)/2 = 28$ for every multi-class problem in our case, are trained and the most popular class is selected. Many efficient implementations of SVMs are publicly available; we use LIBSVM [4].

We train our classifiers using a collection of simulated games in Rush collected under controlled conditions: 40 instances of every possible combination of offense (8) and defense plays (8), from each of the 12 starting formation configurations. Since the starting configuration is known, each series of SVMs is only trained with data that could be observed starting from its given configuration. For each configuration, we create a series of training sequences that accumulates spatio-temporal traces from $t = 0$ up to $t \in \{2, \ldots, 10\}$ time steps. A multiclass SVM (i.e., a collection of 28 binary SVMs) is trained for each of these training sequence lengths. Although the aggregate number of binary classifiers is large, each classifier employs only a small fraction of the dataset and is therefore efficient (and highly paralellizable). Cross-validation on a training set was used to tune the SVM parameters (C and γ) for all of the SVMs. Testing demonstrated near perfect recognition results, 96.88%, at $t = 3$, therefore this classifier was used to help select the most appropriate offensive play, as discussed below.

3.2 Play Similarity Metric

While knowledge about the opposing play is central to retrieving an effective offensive play, the similarity of the candidate plays to the current play estimates the feasibility of the play switch.

To calculate play similarities, we create a feature matrix for every formation/play combination based on background knowledge. The 13 features for each athlete A include max, min, mean, and median over x and y in addition to the following five special features:

FirstToLastAngle: Angle from starting point (x_0, y_0), to ending point (x_n, y_n), defined as $atan\left(\frac{\triangle y}{\triangle x}\right)$

StartAngle: Angle from the starting point (x_0, y_0) to (x_1, y_1), defined as $atan\left(\frac{y_1 - y_0}{x_1 - x_0}\right)$

EndAngle: Angle from (x_{n-1}, y_{n-1}) to the ending point (x_n, y_n), defined as $atan\left(\frac{\triangle y}{\triangle x}\right)$

TotalAngle: $\sum_{i=0}^{N-1} atan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$

TotalPathDist: $\sum_{i=1}^{N} \sqrt[2]{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$

These features are similar to the ones used in [5] and more recently by [6] to match pen trajectories in sketch-based recognition tasks, another spatio-temporal task. Here, they are generalized for use with multi-player trajectories. Feature set F for a given play c ($c = 1...8$, represents possible play matches per formation) contains all features for each offensive player in the play and is described as:

$$\overrightarrow{F_c} = \{A_{c1} \cup A_{c2} \cup \ldots \cup A_{c8}\}$$

Using the 50 play instances from background knowledge, we compute a similarity vector V for every combination of offensive formation, offensive play,

defensive formation, and defensive play combination. This vector includes 8 entries (the computed similarities between the offensive play and the other plays from that formation). We define the similarity between plays as the sum of the absolute value of the differences (L_1 norm) between features F_{c_i} and F_{c_j}. In the evaluation section, we compare the performance of a similarity-based play switch mechanism vs. a play switching algorithm that focuses solely on the predicted defensive play.

3.3 Play Reuse

To reuse the new play in the current situation, we must adapt the current play. The most straightforward approach involves changing the entire play (i.e., each offensive player follows the new play from this time forward). An alternative strategy, *subgroup switching*, involves modifying the actions of only a small group of key players while leaving others alone. By segmenting the team in this fashion, we are able to combine two plays that had previously been identified as alike with regard to spatio-temporal data, but different in regards to yards gained. Based on our domain knowledge of football, we selected three subgroups as candidates to switch: {QB, RB, FB}, {OL, OL, OL}, and {WR, WR, TE}.

4 Improving the Offense with Play Switches

To improve offensive performance, our agent evaluates the competitive advantage of executing a play switch based on 1) the potential of other plays to improve the yardage gained and 2) the similarity of the candidate plays to the current play. Our algorithm for improving Rush offensive play has two main phases: a preprocess stage, which yields a play switch lookup table, and an execution stage, where the defensive play is recognized and the offense responds with an appropriate play switch for that defensive play. We train a set of SVM classifiers using 40 instances of every possible combination of offensive (8) and defensive plays (8), from each of the 12 starting formation configurations. This stage yields a set of models used for play recognition during the game. Next, we calculate and cache play switches using the following procedure:

1. Collect data by running the Rush 2008 football simulator 50 times for every play combination.
2. Create yardage lookup tables for each play combination. This information alone is insufficient to determine how good a potential play is for a play switch. The transition play must resemble our current offensive play or the offensive team will spend too much time retracing steps and perform very poorly.
3. Compute the similarity matrix between offensive plays for all formation/play combinations.
4. Create the final play switch lookup table based on both the yardage information and the play similarity.

To create the play switch lookup table, the agent first extracts a list of offensive plays L given the requirement $yards\,(L_i) > \epsilon$ where ϵ is the least amount of yardage gained before the agent changes the current offensive play to another. We used $\epsilon = 1.95$ based on a quadratic polynomial fit of total yardage gained in 6 tests with $\epsilon = \{MIN, 1.1, 1.6, 2.1, 2.6, MAX\}$ where MIN is small enough so that no plays are selected to change and MAX is set so that all plays are selected for change to the highest yardage play with no similarity comparison. Second, from the list L find the play most similar to our current play, and add it to the lookup table.

During execution, the offense uses the following procedure:

1. At each observation less than 4, collect movement traces for each player.
2. At observation 3, use LIBSVM with the collected movement traces and previously trained SVM models to identify the defensive play, j.
3. Access the lookup table to find $best(i, j)$ for our current play i.
4. If $best(i, j) \neq i$, Send a change order command to the offensive team to change to play $best(i, j)$.

As described in Section 3.3, our system allows for different methods of using the retrieved play. The agent can switch the play for either every offensive player or a subset.

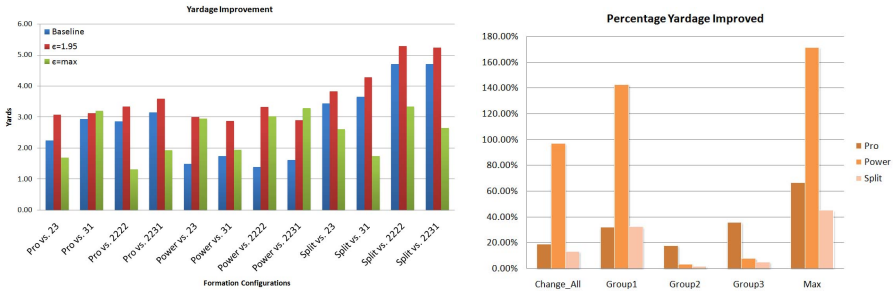
5 Empirical Evaluation

Our goal is to the answer the following questions:

1. Does our play switching algorithm improve yardage gained?
2. Does retrieval incorporating similarity with the current play outperform a greedy strategy that selects solely based upon expected yardage gained?
3. What are the effects of subgroup switching on play performance?

To answer the first two questions, we ran the RUSH 2008 simulator for ten plays on each possible play configuration under three conditions: a baseline without any play switching, our play switch model (using the yardage threshold $\epsilon = 1.95$ as determined by the quadratic fit), and a greedy play switch strategy based solely on the yardage table ($\epsilon = MAX$). The results are shown in Figure 2(a).

Overall, the average performance of the offense went from 2.82 yards per play (in the baseline condition) to 3.65 yards per play ($\epsilon = 1.95$) with an overall increase of 29%, $\pm$1.5% based on sampling of three sets of ten trials. An analysis of each of the formation combinations (Figure 2(a)) shows the yardage gain varies from as much as 100% to as little as 0.1%. Power vs. 23 is dramatically boosted from about 1.5 yards to about 3 yards per play, doubling yards gained. Other combinations, such as Split vs. 23 and Pro vs. 32 already gained high yardage and improved less dramatically (i.e., about .2 to .4 yards more than the gains in the baseline sample). Overall, our model's performance is consistently better for every configuration tested.



(a) Results by play similarity (b) Results by subgroup swtiching

Fig. 2. Similarity-based switching (shown in red) outperforms both the baseline Rush offense (blue) and a greedy play switch metric (green). Changing the play for just Group 1 improves performance over changing the entire play.

Results with $\epsilon = MAX$ clearly shows simply changing to the play with greatest expected yardage generally results in poor performance. When the similarity metric is not used, the results are drastically reduced. The reason appears to be mis-coordinations between teammates accidentally introduced by the play switch; by maximizing the play similarity simultaneously, the possibility of mis-coordinations is reduced.

To evaluate the subgroup switching, we ran the simulation in three additional trails. In each trial, our play switching method was allowed to switch only one of the offensive player subgroups. Using the improvement in yardage, we compared these trials to the full offense switch and the best offensive play against the defense.

The results (shown in Figure 2(b)) clearly indicated the best subgroup switch (consistently Group 1) produced greater gains than the total team switch, which still performed better than the baseline. The Max category presents the results of an agent given the opposing play at $t = 0$, providing a ceiling. Early play recognition combined with subgroup switching yields the best results.

6 Related Work

Previous work on team behavior recognition has been primarily evaluated within athletic domains, including American Football [7], basketball [8], and Robocup soccer simulations [9–12]. In Robocup, most of the research on team intent recognition focused on coaching. Techniques have been developed to extract specific information, such as home areas [13], opponent positions during set-plays [10], and adversarial models [9], from logs of Robocup simulation league games. However, the coaching agents use offline processing to improve their team's performance in future games. In contrast, our agent immediately takes action on the recognized play to evaluate possible play switches. Ros et al. present a similar approach involving similarity between offensive and defensive alignments for

selecting plays in robocup soccer [12]. Our retrieval approach differs by using traces of player movement and a prediction concerning the opposing play. Furthermore, we demonstrate the utility of switching the play for only a subset of the offensive players. On the other hand, their representations include aspects of the overall strategy, including the score and the amount of time remaining in the game. Adding knowledge of this type is necessary for our agent to effectively play an entire football game.

Comparatively few case-based reasoning researchers have investigated spatial reasoning. Most focus on retrieving precedents based on quantitative and qualitative features [14] without any adaptation. Using insights from research on pen stroke recognition [6], our spatial similarity metric incorporates spatio-temporal knowledge into retrieval, which is then used to adapt the current situation. Galatea [15] uses stored visual problem-solving episodes consisting of visual transformations, which are employed analogically to arrive at a solution for new problems. While transfer in Galatea is iterative, our play switch is a one-shot process. Furthermore, Galatea places little emphasis on retrieval. Our model uses spatial knowledge throughout retrieval, first in categorizing the opposing team's play, then in determining the most similar play from the case base.

Rush 2008 was developed as a platform for evaluating game-playing agents and has been used to study the problem of learning strategies by observation [16]. Intention recognition has been used within Rush 2008 as part of a reinforcement learning method for controlling a single quarterback agent [17]. In this paper, our approach addresses policies across *multiple* agents.

7 Conclusion

Accurate opponent modeling is an important stepping-stone toward the creation of interesting autonomous adversaries. In this paper, we present an approach for online strategy recognition in the Rush 2008 football simulator. After identifying the defense's play, our agent evaluates the advantage of executing a play switch based on the potential of other plays to improve the yardage gained and their similarity to the current play.

We have shown that spatio-temporal features enable online strategy recognition in the early stages of a play. Furthermore, by incorporating spatial similarity into the selection of the appropriate play switch, our method avoids miscoordinations between offensive players, increasing the yardage gained. Additionally, we demonstrate that limiting the play switch to a subgroup of key players further improves performance.

In future work, we plan on extending our game playing agent to play the entire game. While our focus on gaining more yards is central to successful offense, in the complete game, offensive strategy becomes more complex, including scoring and clock management. As discussed previously, we plan to explore methods for automatically identifying key player subgroups for adapting the play by examining motion correlations between players. Finally, we plan to explore these ideas of online strategy recognition in other domains.

References

1. : Rush (2005) http://sourceforge.net/projects/rush2005/.
2. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications
3. Vapnik, V.: Statistical Learning Theory. Wiley & Sons, Inc (1998)
4. 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.
5. Rubine, D.: Specifying gestures by example. Computer Graphics, Volume 25, Number 4 (1991) 329–337
6. Wobbrock, J., Wilson, D., Yang, L.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In: Symposium on User Interface Software and , Proceedings of the 20th annual ACM symposium on User interface software and technology. (2007)
7. Intille, S., Bobick, A.: A framework for recognizing multi-agent action from visual evidence. In: Proceedings of National Conference on Artificial Intelligence. (1999)
8. Jug, M., Pers, J., Dezman, B., Kovacic, S.: Trajectory based assessment of coordinated human activity. In: Proceedings of the International Conference on Computer Vision Systems (ICVS). (2003)
9. Riley, P., Veloso, M.: On behavior classification in adversarial environments. In Parker, L., Bekey, G., Barhen, J., eds.: Distributed Autonomous Robotic Systems 4. Springer-Verlag (2000)
10. Riley, P., Veloso, M.: Recognizing probabilistic opponent movement models. In Birk, A., Coradeschi, S., Tadorokoro, S., eds.: RoboCup-2001: Robot Soccer World Cup V. Springer Verlag (2002)
11. Kuhlmann, G., Knox, W., Stone, P.: Know thine enemy: A champion RoboCup coach agent. In: Proceedings of National Conference on Artificial Intelligence. (2006)
12. Ros, R., Veloso, M., de Mantaras, R.L., Sierra, C., Arcos, J.: Rertrieving and reusing game plays for robot soccer. In: 8th European Conference on Case-Based Reasoning (ECCBR-06). (2006)
13. Riley, P., Veloso, M., Kaminka, G.: An empirical study of coaching. In Asama, H., Arai, T., Fukuda, T., Hasegawa, T., eds.: Distributed Autonomous Robotic Systems 5. Springer-Verlag (2002)
14. Holt, A., Benwell, G.: Case-based reasoning with spatial data. In: Proceedings of the 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems (ANNES '95). (1995)
15. Davies, J., Goel, A., Nersessian, N.: Transfer in visual case-based problem-solving. In: Proceedings of the 6th International Conference on Case-Based Reasoning. (2005)
16. Li, N., Stracuzzi, D., Cleveland, G., Langley, P., Konik, T., Shapiro, D., Ali, K., Molineaux, M., Aha, D.: Constructing game agents from video of human behavior. In: Proceedings of AIIDE-09. (2009)
17. Molineaux, M., Aha, D., Sukthankar, G.: Beating the defense: Using plan recognition to inform learning agents. In: Proceedings of Florida Artifical Intelligence Research Society, AAAI Press (2009) 337–343

Authoring Behaviors for Games using Learning from Demonstration

Manish Mehta, Santiago Ontañón, Tom Amundsen, and Ashwin Ram

CCL, Cognitive Computing Lab
Georgia Institute of Technology
Atlanta, GA 30332/0280
{mehtama1,santi,amundsen,ashwin}@cc.gatech.edu

Abstract. Behavior authoring for computer games involves writing behaviors in a programming language. This method is cumbersome and requires a lot of programming effort to author the behavior sets. Further this approach restricts the behavior set authoring to people who are experts in programming. This paper will describe our approach to design a system that will allow a user to demonstrate behaviors to the system, which the system will use to learn behavior sets for a game domain. With learning from demonstration, we aim at removing the requirement that the user has to be an expert in programming, and only require him to be an expert in the game. The approach has been integrated in a easy to use visual interface and instantiated for two domains, one a real time strategy game and another an interactive drama.

1 Introduction

State-of-the-art computer games are usually populated with many characters that require intelligent and believable behaviors. However, even though there have been enormous advances in computer graphics, animation and audio for games, most of the games contain very basic artificial intelligence (AI) techniques. In the majority of computer games traditional AI techniques fail to play at a human level because such games have vast search spaces in which the AI has to make decisions in real-time. Such enormous search spaces cause the game developers to spend a large effort in hand coding specific strategies that play at a reasonable level for each new game. Game designers are typically non-AI experts, and thus defining behaviors using a programming language is not an easy task for them. They might have a clear idea in mind of the behavior they want particular characters in the game to exhibit, but the barrier is encoding those ideas into actual code. Ideally, we need an approach that can allow game designers to easily author behavior sets for particular games.

Human learning is often accelerated by observing a task being performed or attempted by someone else. In fact, infants spent a lot of their time repeating the observed behaviors [11]. These capabilities of the human brain are also evident in computer games where players go through a process of training and imitating

experienced players. These results have inspired researchers in artificial intelligence to study learning from imitation techniques. However except for a few attempts, there have been very few attempts at their integration in computer games. By observing an expert's actions, new behaviors can quickly be learnt that are likely to be useful; because they are already being used by the expert successfully. In this paper, we present an approach that utilizes this ability to extract behavioral knowledge for computer games from expert demonstrations. Using the architecture presented in this paper the game authors demonstrate the behavior to be learnt (maybe by controlling some game characters manually) instead of having to code the behavior using a programming language and the system learns from that demonstration. In order to achieve that goal, we use case-based reasoning (CBR) techniques, and in particular case-based planning [12]. The idea is to represent each behavior as a plan, and use case-based planning to reuse the behaviors learnt from demonstrations in order to play the game. Our architecture has been instantiated in two domains, one a real time strategy game and the other an interactive drama.

The rest of the paper is organized as follows. We present our architecture in Section 2. We discuss the concrete instantiation of the architecture in real time strategy game WARGUS (an open source clone of the popular game WARCRAFT II) in Section 3 and interactive drama domain in Section 4. The paper closes with related work and conclusions.

2 Learning from Demonstration Architecture

Our main goal is to create a system that allows a game designer to easily author AI behaviors using learning from demonstration, in constrast to having him encoding behaviors in some programming language. In order to achieve that goal, we have designed a learning from demonstration architecture (shown in Figure 1) that consists of four steps:

- *Demonstration*: The human plays the game, demonstrating the particular behavior he wants the system to learn. This process results in a trace, i.e. a log file that contains each action that the expert executed, together with their respective game state and time stamps.
- *Annotation*: The human annotates the trace specifying which goals (selected from a predefined set of goals) he was attempting with each action. In our experiments, annotation is performed using an easy to use GUI. Section 2.2 explains why annotation is desirable.
- *Behavior Learning*: The annotated trace is handed to a behavior learning module, which can automatically extract procedural behaviors from the annotated trace, and store them in a behavior base.
- *Behavior Execution*: Once the behavior base has been populated, the learnt behaviors can be executed in the game using a behavior execution engine. We propose to use a case-based planning [12] behavior execution engine, where each one of the behaviors is represented as a case.

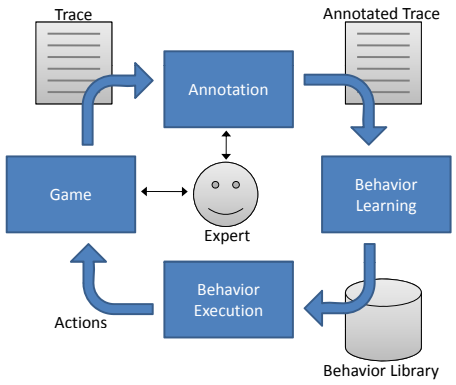


Fig. 1. Our general Learning from Demonstration Architecture, involving 4 steps: demonstration, annotation, behavior learning and finally behavior execution.

2.1 Demonstration

The game domain needs to provide a way to demonstrate behaviors. Depending on the game at hand, this can be done using the normal interface that a player would use to play the game, or through a special interface if required. The main idea is to let the expert use the basic set of primitives that are available within the game world. For example, in our RTS game domain, WARGUS the standard game playing interface can be used. However, in our interactive drama domain, Murder Mystery, a specific interface to control virtual characters inside the game world was developed. This was the case because the default game interface in that game did not generate traces nor allowed us to control the characters at the level of detail we wanted.

The author uses the demonstration interface to play the game. Apart from this interface, a basic mechanism to record the trace is required. In our architecture, a trace is composed of a list of *entries*, where each entry is a triple: time stamp, game state, primitive actions. Representing at a particular time in a particular state, the expert executed some primitive actions.

2.2 Trace Annotation

The next step is to annotate the trace. In this process, the expert specifies which goals was he pursuing for each particular action. This process requires a collection of goals being defined for each game for which the architecture is instantiated. Once a set of goals is defined, the expert can simply associate each of the actions in the game with one or more of the set of available goals.

The intuition behind annotation is that if a set of actions are labeled as achieving the same goal, then the system will put those actions in a single behavior that achieves the specified goal. Thus, annotations can be used in order to group together the actions that were demonstrated into individual behaviors.

We can now see that the set of goals that have to be defined for each game is a set of goals that allows the human to decompose the task of playing the game in subtasks for which behaviors can be learnt. Annotation could be partially automated (as we propose in [9]). However, an automatic process of annotation leaves the expert with less control over the learnt behaviors. An automatic annotation process is desirable if the goal is to build a system that can learn how to play the game autonomously. However, if the goal is to facilitate the task of a human author, annotation provides a simple way in which the author (or expert) can control which behaviors will be learnt. For example, in a given game, the expert might, by accident, achieve some particular goal during the game in a way that he did not want to demonstrate (just as a side effect of some actions). In an automated annotation process, that will result in the system learning an undesired behavior, which for the purposes of the system learning to play the game is desirable, but for the purposes of helping the author defining the behaviors he wants to define is undesirable. For that reason, we believe that annotation is desirable when the goal is to assist a human in behavior authoring.

2.3 Behavior Learning

In order to learn behaviors, the annotated trace is analyzed to determine the temporal relations among the individual goals appearing in the trace. In our framework, we are only interested in knowing if two goals are pursued in sequence, in parallel, or if one is a subgoal of the other. We assume that if the temporal relation between a particular goal g and another goal g' is that g happens *during* g', then g is a subgoal of g'.

From this temporal analysis of goals, procedural descriptions of the behavior of the expert can be extracted. Notice that an expert might assign more than one goal to each action. Thus, the system can learn hierarchical behaviors. Also, once the system has learned behaviors for each one of the goals used by the expert, a global behavior that uses these behaviors as "subroutines" can also be inferred (See [10] for more details).

Each one of the learnt behaviors are stored in a behavior library for future use. Notice that no generalization of the behaviors is attempted at learning time. Since we are proposing to use a case-based reasoning approach (where each behavior is considered to be a case), all generalization is left for problem solving time, i.e. for when the system is playing a game.

2.4 Behavior Execution

Once behaviors have been learned, they are ready to be executed in the game. Thus, a behavior execution engine is required. We propose to use a hierarchical case-based planner to perform this task. Each behavior will be seen as a partial plan to achieve a particular goal, and the hierarchical planner will combine them together to form full plans to achieve the goals of the character or characters the system is controlling.

Cycle	*Player*	*Action*	*Annotation*
8	1	Build(2,"pig-farm",26,20)	-
137	0	Build(5,"farm",4,22)	SetupResourceInfrastructure(0,5,2) WinWargus(0)
638	1	Train(4,"peon")	-
638	1	Build(2,"troll-lumber-mill",22,20)	-
798	0	Train(3,"peasant")	SetupResourceInfrastructure(0,5,2) WinWargus(0)
878	1	Train(4,"peon")	-
878	1	Resource(10,5)	-
897	0	Resource(5,0)	SetupResourceInfrastructure(0,5,2) WinWargus(0)
...	...	...	...

Table 1. Snippet of a real trace generated after playing WARGUS. The game states for each entry in the trace are omitted.

3 First Game Domain: WARGUS

Real-time strategy (RTS) games have several characteristics that make behavior authoring difficult: huge decision and state spaces [1, 2], non determinism, incomplete information, complex durative actions, and real time. WARGUS is a real-time strategy game where each player's goal is to remain alive after destroying the rest of the players. Each player has a series of troops and buildings and gathers resources (gold, wood and oil) in order to produce more troops and buildings. Buildings are required to produce more advanced troops, and troops are required to attack the enemy. In addition, players can also build defensive buildings such as walls and towers. Therefore, WARGUS involves complex reasoning to determine where, when and which buildings and troops to build.

In order to demonstrate a behavior set for WARGUS an expert simply plays a game. As a result of that game, we obtain a game trace. Table 1 shows a fragment of a real trace from playing a game of WARGUS. In the WARGUS domain, each trace entry is limited to a single action. For instance, the first action in the game was executed at cycle 8, where player 1 made his unit number 2 build a "pig-farm" at the (26,20) coordinates.

The next step is to annotate the trace. For the annotation process, the expert uses a simple annotation tool that allows him to specify which goals was he pursuing for each particular action. The annotation tool simply presents the execution trace to the expert (with small screenshots of the state of the game at every trace entry, to help the human remember what he was doing) and he can associate goals to actions. All the goal types defined for the WARGUS domain are available to the expert, and he can fill in the parameters of each goal when annotating. Figure 2 shows a screenshot of such tool.

In our approach, a *goal* $g = name(p_1, ..., p_n)$ consists of a goal name and a set of parameters. For instance, in WARGUS, some of the goal types we defined are: $WinWargus(player)$, representing that the action had the intention of making

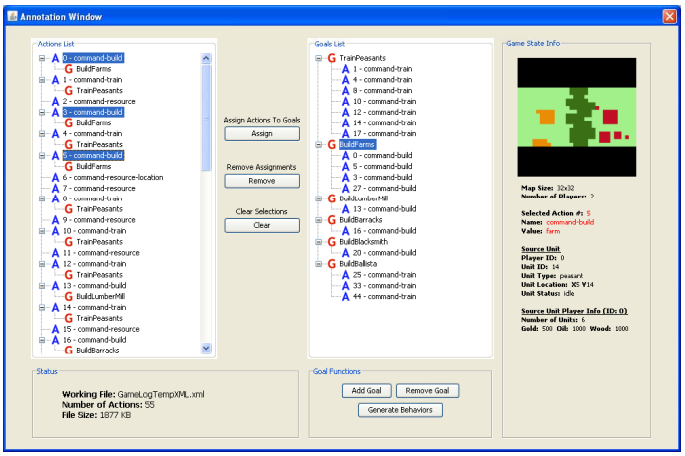


Fig. 2. A screenshot of our WARGUS trace annotation tool.

the player *player* win the game; $KillUnit(unit)$, representing that the action had the intention of killing the unit *unit*; or $SetupResourceInfrastructure(player, peasants, farms)$, indicating that the expert wanted to create a good resource infrastructure for player *player*, that at least included *peasants* number of peasants and *farms* number of farms.

The fourth column of Table 1 shows the annotations that the expert specified for his actions. Since the snippet shown corresponds to the beginning of the game, the expert specified that he was trying to create a resource infrastructure and, of course, he was trying to win the game. The annotated trace is next processed by the *behavior learning* module, which encodes the strategy of the expert in this particular trace in a series of behaviors.

Notice that in our system we don't attempt any kind of generalization of the expert actions. If a particular expert action in the trace is $Build(5, "farm", 4, 22)$, that is exactly the action stored in a snippet. Thus, using the learnt snippets to play a new scenario in WARGUS, it is very likely that the particular values of the parameters in the action are not the most appropriate for the new scenario (for instance, it might be the case that in the new map the coordinates 4,22 correspond to a water location, and thus a farm cannot be built there). In our WARGUS implementation, the behavior execution engine is responsible to adapt those parameters at run time.

Our execution engine in WARGUS is a case-based planner, that uses a set of adaptation rules in order to adapt the parameters of each of the actions in each behavior before executing it. Thus, in our implementation in the WARGUS domain, the game state in which the human demonstrated each action is stored together with the behavior. For details on how adaptation at run time is performed, see [10].

In order to evaluate our techniques in WARGUS, we developed an IDE from where users could launch WARGUS to start a demonstration, annotate

demonstrations, manipulate behaviors, and test them on the game [13]. Our results show that users were able to successfully demonstrate behaviors using our system, and that they felt demonstrating behaviors was an easier way to generate scripts, than coding them by hand.

4 Second Game Study: Murder Mystery

In recent years, there has been a growing interest in creating story based interactive systems where the player experiences a story from a first person perspective, interacts with autonomous, believable characters. Interactive drama presents one of the most challenging applications of autonomous characters, requiring characters to simultaneously engage in moment-by-moment personality-rich physical behavior, exhibit conversational competencies, and participate in a dynamically developing story arc. Hand authoring of behavior for believable characters allows designers to craft expressive behavior for characters, but nevertheless leads to excessive authorial burden [6]. Tools are needed to support story authors, who are typically not artificial intelligence experts, to allow them to author behaviors in an easy way.

The interactive drama we are developing is named Murder Mystery (MM).The story set up consists of six characters and is set up in a British mansion at the beginning of the 20th century. The player controls one of the character and is free to interact with the rest of the characters using natural language and also move freely around the house and manipulate some objects. In particular, the drama starts when two of the characters decide to celebrate an engagement party, and invite two friends to a dinner in their newly acquired mansion. The remaining two characters are the butler of the house and the father of the bride. Most of the characters have strong feelings (love or hate) for some of the other characters, and as the story unfolds the player will discover hidden relations between them. The player will take the role of one out of three possible characters and will be able to act freely in the mansion.

In order to demonstrate behaviors, the user observes a character from a third person perspective and is able to control it using a GUI. The GUI consists of a series of buttons and text fields that allow the user to perform the following actions: speak, move forward, move backward, move left, move right, rotate, and play an animation. Such an interface records a similar trace as for our WARGUS domain (an example is shown in Table 2. The context associated with each logged action describes the current game state and consists of information about the map and characters. Each object and player in the map is logged with as much information as possible (since it will help the CBR system to adapt actions at run-time).

In order to carry out the annotation, some of the goals that have been used are:

- *Greet*(*character*): representing that the action had the intention of greeting another.

Cycle	*Player*	*Action*	*Annotation*
8	Mary	Walk("230,400,1920", "230,400,1920", Mary)	-
137	Mary	Speak(Tracy, "Hi Tracy")	Greet(Tracy)
378	Mary	Wave ()	Introduce(Tracy)
500	Mary	Speak(Tracy, "I am Manuel Sharma")	-
678	Mary	Smile ()	-
800	Mary	Speak(Tracy, "I am working as a technician")	-
938	Mary	Speak(Tracy,"Could you pass me a drink?')	AskforObject(Tracy, drink)
...	...	...	...

Table 2. Snippet of a real trace generated after playing Murder Mystery .

- $AskforObject(character, object)$: representing that the action had the intention of asking for a particular object *object* from a character *character*.
- $Introduce(character)$: the action had the intention of introducing to a particular character
- $Insult(character)$: the action had the intention of insulting a particular character
- $Hurt(character)$: the action had the intention of hurting a particular character.

In the same way as for WARGUS this trace would then be given to the behavior learning module, that will learn behaviors from it. Figure 3 shows an example of a learnt behavior in Murder Mystery. Although an extensive evaluation of our system in the Murder Mystery domain is still part of our future work, initial evaluations suggest that it is easier to author behaviors using our demonstration interface than coding them by hand.

In an analogous way as for our WARGUS domain, in the Murder Mystery, the game state associated with each action is stored, so that the behavior execution engine (a case-based planner) can adapt those actions.

```
Introduce(tracy)
{
  Wave();
  Speak(Tracy, "I am Manuel Sharma");
  Smile();
  Speak(Tracy, "I am working as a technician");
}
```

Fig. 3. Snippet of a behavior learnt after behavior demonstration in Murder Mystery .

5 Related Work

Henry Lieberman describes a system called Tinker, that is able to learn from examples that a programmer demonstrates. Using this framework, a programmer can demonstrate sets of examples, starting with simple examples, and work up to more complicated ones. Using these examples, Tinker learns how to operate on its own. A more recent example is provided by Nakanishi et al. [7], who designed a system that learns biped locomotion by observing humans walking. Nakanishi et al. describe an approach of using dynamical movement primitives as a central pattern generator, which are then used to learn the trajectories for the legs in robot locomotion. Nicolescu [8] describes a modular architecture which allows a robot to learn by generalizing information received from multiple types of demonstrations, and allows the robot to practice under the demonstrator's supervision. This system, albeit in a robotic domain is quite similar to ours, and provides a general way to learn primitive behaviors through demonstration in order to accomplish a given task.

Floyd et. al. present an approach to train a RoboCup soccer-playing agent by observing the behaviour of existing players and determining the spatial configuration of the objects the existing players pay attention to [3]. Kaiser and Dillman [5] presented a general approach to learning from demonstration using sensor-based robots. They describe how skills can be acquired from humans, "learned" in such a way that they can be used to achieve tasks, and refined so that the agent's performance will constantly improve. The system uses action primitives that are very concrete and easy to predict, such as determining what angle to move a robotic arm. In our system, action primitives are parameterized like talking to another character in the game, which can potentially have results that are hard to predict. Finally, Floyd and Estefandiari [4] compare several techniques for learning form demonstration (CBR, decision trees, support vector machines and naive bayes), showing very strong results favoring case-based learners.

6 Conclusions and Future Work

Learning from demonstration is a powerful mechanism to quickly learn behaviors. In this paper, we discuss how the principle of imitation learning can facilitate the programming of computer game characters. Moreover, we demonstrated the approach by reporting two implemented systems based on the same learning from demonstration architecture.

One of the key ideas introduced in this paper is that by the use of annotations in the demonstrations, the author can have control of the behaviors being learnt during the learning from demonstration process. Behavior authoring is ultimately a programming task, and as such is non-trivial when the set of behaviors that need to be authored are complex. However, we have seen that by using case-based planning techniques, concrete behaviors demonstrated in concrete game situations can be reused by the system in a range of other game situations, thus providing an easy way to author general behaviors.

Part of our future work involve trying to reduce the annotation task to a minimum, but that the author still has control over the behavior authoring process. One of the ideas is to implement a mixed initiative approach where the system will automatically annotate a trace, and the author will have the option (it desired) of changing the annotations. We are also working on implementing our approach in more domains to evaluate its strengths and weaknesses. In our initial evaluations we have seen that our approach is good for high level behavior demonstration, where as it is still not very good at low level reactive control.

References

1. David Aha, Matthew Molineaux, and Marc Ponsen. Learning to win: Case-based plan selection in a real-time strategy game. In *ICCBR'2005*, number 3620 in LNCS, pages 5–20. Springer-Verlag, 2005.
2. Michael Buro. Real-time strategy games: A new AI research challenge. In *IJCAI'2003*, pages 1534–1535. Morgan Kaufmann, 2003.
3. Michael W. Floyd, Babak Esfandiari, and Kevin Lam. A case-based reasoning approach to imitating robocup players. In *FLAIRS Conference*, pages 251–256, 2008.
4. Michael W. Floyd and Babak Estefandiari. Comparison of classifiers for use in a learning by demonstration system for a situated agent. In *Workshop on Case-Based Reasoning for Computer Games in ICCBR 2009*, page to appear, 2009.
5. M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. In *In International Symposium on Intelligent Robotics Systems*, pages 2700–2705, 1996.
6. B. Magerko, J. Laird, M. Assanie, A. Kerfoot, and D. Stokes. AI characters and directors for interactive computer games. In *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Confercence*, 2004.
7. Jun Nakanish, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion with dynamical movement primitives, 2003.
8. Monica Nicolette Nicolescu. *A framework for learning from demonstration, generalization and practice in human-robot domains*. PhD thesis, Los Angeles, CA, USA, 2003. Adviser-Maja J. Mataric.
9. Santiago Ontañón, Kane Bonnette, Prafulla Mahindrakar, Marco A. Gómez-Martín, Katie Long, Jainarayan Radhakrishnan, Rushabh Shah, and Ashwin Ram. Learning from human demonstrations for real-time case-based planning. In *The IJCAI-09 Workshop on Learning Structural Knowledge From Observations*, 2009.
10. Santiago Ontañón, Kinshuk Mishra, Neha Sugandh, and Ashwin Ram. Case-based planning and execution for real-time strategy games. In *Proceedings of ICCBR-2007*, pages 164–178, 2007.
11. Rajesh P. N. Rao, Aaron P. Shon, and Andrew N. Meltzoff. A bayesian model of imitation in infants and robots. In *In Imitation and Social Learning in Robots, Humans, and Animals*. Cambridge University Press, 2004.
12. L. Spalazzi. A survey on case-based planning. *Artificial Intelligence Review*, 16(1):3–36, 2001.
13. Suhas Virmani, Yatin Kanetkar, Manish Mehta, Santiago Ontañón, and Ashwin Ram. An intelligent ide for behavior authoring in real-time strategy games. In *AIIDE*, 2008.

SARTRE: System Overview
A Case-Based Agent for Two-Player 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

Abstract. SARTRE (Similarity Assessment Reasoning for Texas hold'em via Recall of Experience) is a `heads-up` (two-player) poker-bot that plays `limit` Texas Hold'em using the case-based reasoning methodology. This paper presents an overview of the SARTRE system. As far as we are aware SARTRE is the only poker-bot designed specifically to play `heads-up` Texas Hold'em using a CBR foundation. The design and implementation of the current system is discussed. Case features are illustrated and their reasons for selection are addressed. Finally, avenues for future areas of investigation are then listed.

1 Introduction

This paper will describe the design and implementation of a heads-up (two-player) poker-bot that plays limit Texas Hold'em using the case-based reasoning methodology. SARTRE (Similarity Assessment Reasoning for Texas hold'em via Recall of Experience) is the latest result of our ongoing research focused around the investigation into the role of memory in game AI. SARTRE is specifically tailored to play two-player poker, whereas our previous system, CASPER (CASe based Poker playER) was more suited to full-table game play (8 - 10 players) [1, 3]. Two-player poker offers its own unique challenges, where strategies for successful play differ markedly from those employed at a full table [2]. For a description of the rules of Texas Hold'em consult [1, 2].

2 Overview of SARTRE

A human poker player requires information to make their betting decisions. As SARTRE is a computer program, the information required needs to be easily recognised and able to be reasoned about algorithmically. Salient information needs to be identified and used to affect SARTRE'S final decision. While too much information is usually better than not enough, the utilisation of too much information could result in undue complexity which may deteriorate SARTRE'S performance.

The type of information that SARTRE has available at decision time includes items from the following list:

- The betting decisions of each player during the current hand.
- The betting decisions of each player for all hands that occurred previous to the current hand.
- The current stage of the hand.
- The *hole cards* that are only visible by Sartre.
- The *community cards* that are visible by all players.

The authors have hand picked three key factors from the above list to represent *indexed features* that SARTRE uses to determine a solution for a particular case:

1. The previous betting for the current hand.
2. The current strength of SARTRE'S hand given by combining personal *hole cards* with the publicly available board cards.
3. Information about the state of the current community cards, called the *texture of the board*.

Qualitative feature descriptions have been favoured over quantitative descriptions as they are more likely to be used by an expert, human player. Each case feature is described in more detail below, including the representation we have chosen to implement for the SARTRE system.

2.1 The previous betting for the current hand

The type of betting that can occur at each decision point in a hand consists of a fold *(f)*, check/call *(c)*, or bet/raise *(r)*. A combination of these symbols corresponds to all the decisions made during a particular hand.

As SARTRE is specifically designed to play only `heads-up` poker the number of betting patterns that can occur is drastically reduced compared to the combination of betting patterns that can occur at a table with ten players. An example betting pattern is presented and analysed below. The total bets allowed to be contributed by each player during each round is capped at four:

```
rrc-r
```

This particular example represents a feature that is used by SARTRE during the flop. The betting that occurred during the *pre-flop* is separated from the betting on the *flop* by a hyphen. This betting string can be described by the following situation:

- SARTRE is in the *small blind* (dealer) and makes a forced bet of 0.5. The opponent is in the *big blind* and makes a forced bet of 1.
- SARTRE is the first to act *pre-flop* and decides to *raise*. SARTRE has now committed a total of 2 bets.
- SARTRE'S opponent *re-raises* by committing another 2 bets to the pot.
- SARTRE calls 1 more bet completing the *pre-flop* betting and leaving 6 total bets in the pot.

– As SARTRE is the dealer he acts last on all *post-flop* betting rounds. SARTRE'S opponent has made 1 bet on the *flop* and it is now SARTRE'S turn to make a decision.

The above example indicates a lot of information is contained within the betting pattern. We have chosen to represent each betting pattern as a path within a betting tree. A betting tree succinctly enumerates all betting combinations up until a certain point in the hand. A path within this tree represents the actual decisions that were made by each player during this hand. This is represented graphically in Fig. 1.

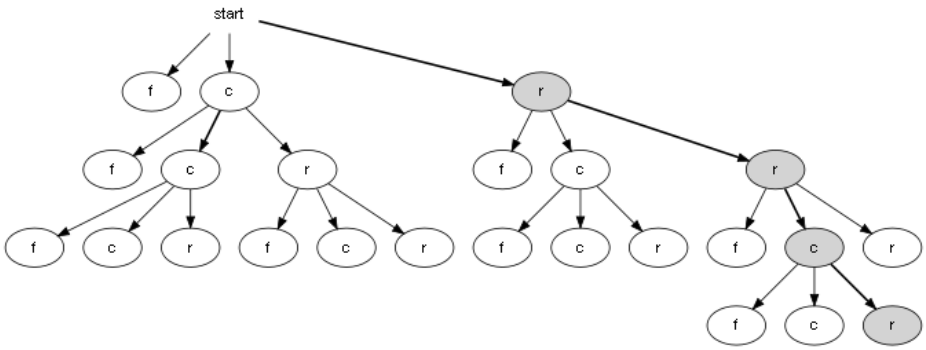


Fig. 1. A tree that describes betting decisions for two players during a hand of Texas Hold'em Poker. The highlighted nodes are the actual decisions that were made by each player.

Given this representation we can calculate the similarity between two separate trees (a target tree and a source tree) by comparing the betting path within each tree. If the betting path in the target tree is exactly the same as the betting path within the source tree a similarity value of 1.0 is assigned. Currently, Sartre will simply assign a value of 0.0 to any betting paths that are not exactly similar, however, we plan to investigate less stringent approaches for future implementations. For example, if one betting path mostly resembles that of another, with a small number of variations, a similarity value close to (but less than) 1.0 could be assigned.

2.2 The current hand category

The second case feature used to determine a betting action is a qualitative category describing SARTRE'S personal hand. During the *pre-flop* SARTRE'S hand simply consists of his personal *hole cards*, whereas for the *post-flop* stages of play SARTRE'S hand is constructed by combining his *hole cards* with the publicly available *community cards*, the best 5 card combination is used.

SARTRE'S best 5 cards are mapped to a category that describes the hand. The classic hand categories in poker include *no-pair, one-pair, two-pair, three-of-a-kind, straight, flush, full-house, four-of-a-kind* and finally a *straight-flush*. Each category has a greater strength than the previous one, where a *straight-flush*, consisting of the cards `Ten, Jack, Queen, King, Ace`, represents the highest rank possible (i.e. a *Royal Flush*).

During the *flop* and the *turn* all the community cards have yet to be dealt and therefore a player's hand has the ability to improve from one category to another, depending on which card is drawn next. It is therefore too simplistic to only consider the current hand category, so further classification is required for hands with the potential to improve. These types of hands are called *drawing hands* (in poker terminology). SARTRE considers two types of drawing hands: *flush draws* & *straight draws*. An example mapping is illustrated in Fig. 2.

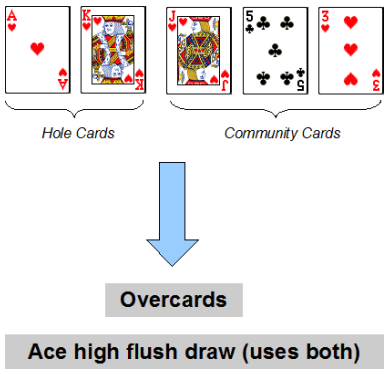


Fig. 2. Mapping a combination of five cards to a category that represents the current hand rank and the drawing strength of this hand.

The hand categories SARTRE uses to classify cards were decided upon by the authors. Fig. 2. shows a combination of two categories: *overcards* + *ace-high-flush-draw-uses-both* i.e. no pair has been made, but both *hole cards* have a higher rank than the community cards and this hand has the potential to become a *flush*.

Currently a simple rule-based system is used to decide which category a combination of cards belongs to. Similarity for this feature is currently either 1.0 when the category of the target case is exactly that of the source case, otherwise it is 0.0 when the categories are distinct.

2.3 The texture of the board

The final indexed feature attempts to summarise the state of the community cards without considering the *hole cards* of a player. The *texture of the board*

refers to salient information a human poker player would usually notice about the public cards, such as whether a *flush* is possible. Once again a set of qualitative categories were hand-picked by the authors to map various boards into. Some categories used by SARTRE'S current implementation that refer to flush and straight possibilities are *Is-Flush-Possible* (where three cards of the same suit are showing), *Is-Flush-Highly-Possible* (where four cards of the same suit are showing) & *Is-Straight-Possible* (where three consecutive card values are showing), *Is-Straight-Highly-Possible* (where four consecutive card values are showing).

If two boards are mapped into the same category they are given a similarity value of 1.0, whereas boards that map to separate categories have a similarity of 0.0.

2.4 SARTRE'S Case-Base

SARTRE'S case-base is generated by analysing the game logs of previous AAAI Computer Poker Competitions[1]. The current version of SARTRE uses approximately 250,000 cases for each stage of the game (*pre-flop, flop, turn, river*). Results against other computerised opponents will soon be available as we plan for SARTRE to compete in the upcoming Computer Poker Competition to be held at IJCAI '09.

3 Future Work

Work on the SARTRE system is still in an early phase and there is much room for investigation and improvement:

1. Experimental results are required. At present we have not had time to extensively measure SARTRE'S performance
2. Currently SARTRE'S similarity metrics are too stringent. Further work is required to improve the calculation of similarity values.
3. SARTRE uses no opponent modelling capabilities at present. We plan to augment SARTRE with a CBR opponent modelling system and assess the impact on performance.

References

1. Rubin, J., Watson, I.: Investigating the Effectiveness of Applying Case-Based Reasoning to the Game of Texas Hold'em. Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference. AAAI Press. 417–422 (2007)
2. Sklansky, D.: The Theory of Poker. Two Plus Two Publishing, Las Vegas, Nevada (2005)
3. Watson, I., Rubin, J.: CASPER: A Case-Based Poker-Bot. AI 2008: Advances in Artificial Intelligence, 21st Australasian Joint Conference on Artificial Intelligence. Springer (5360). Lecture Notes in Computer Science. 594–600 (2008)

[1] http://www.cs.ualberta.ca/ pokert/

Memory and Analogy in Game-Playing Agents

Jonathan Rubin and Ian Watson

Department of Computer Science
University of Auckland, New Zealand
http://www.cs.auckland.ac.nz/research/gameai
jrub001@aucklanduni.ac.nz, ian@cs.auckland.ac.nz

Abstract. We present our views and ideas about a possible approach to general game playing by utilising memory and analogy. We initially discuss the importance of memory in game playing agents. The forms that memory can take are examined and examples of successful agents who utilise memory are presented. Following this we focus on *experience-based, lazy learners* and justify why we believe they may be beneficial in a general game playing domain. Analogical reasoning is then introduced and its benefits considered. We conclude by formulating some example analogies and speculating how an experience-based, lazy learner could apply these to a general game playing environment.

1 Introduction

In this position paper we wish to describe the possibility of a lazy learning agent used to play multiple, arbitrary games using memory and analogy. Memory refers to the concept of storing scenarios in a database or case-base to represent an agent's knowledge. Analogy refers to the ability to recognise similarities between separate problem domains and to generalise solutions from one domain to another. Through examples and previous research we will attempt to outline how this system could be constructed and the possible benefits of this approach.

The following sections begin with a summary of the types of memory an agent has available and various systems that have achieved success through the use of memory. Followed by a review of experience-based, lazy learners that have been developed to play specific games such as chess, checkers and poker. Finally, analogy is discussed where an attempt is made to generalise memories an agent holds about a specific game to aid it in playing a game it has not previously encountered.

2 Memory in Games

Memory in a game playing agent can refer to any kind of persistent knowledge an agent has at it's disposal that it does not need to deduce algorithmically. Some examples include:

– Databases of powerful strategies in games such as chess or checkers.

- Tables that record opponent based information in games such as poker.
- Or, a collection of cases, in a case-based reasoning system, illustrating various plays and their level of success.

The addition of some kind of memory component to a game-playing agent can be beneficial for numerous reasons. Some of which are outlined below:

- Memory can take the form of knowledge gained through expert play such as in Grandmaster databases. Experts with years of game playing experience can encode their knowledge and strategies into databases which form the basis of the agent's memory. This provides the agent with a persistent knowledge of the game that may not be available through other strategies alone such as game-tree search. These proven lines of sophisticated play can then be used by the agent to aid its game playing decisions.
- A memory component can be used to hold perfect information about a game at a certain position and the outcome from that position i.e. win/loss/draw. This improves the performance of the agent by allowing it to identify when a win is available. If only draws or losses are present it also allows the agent to avoid moves that will lead to a loss.
- Memory allows an agent to learn from experience. By maintaining a memory the agent can record which decisions were beneficial and which were harmful.
- For games that rely heavily on how an opponent plays e.g. poker, memory is imperative. An adaptive agent will be required to remember how an opponent has played in the past and what type of playing style they may employ. Using this information a strong game-playing agent can then exploit weak opponents and avoid being exploited itself. To achieve this the agent will need to encode some sort of long-term memory about specific opponents as well as general playing styles.

While it is true that for extremely simple games such as *Tic-Tac-Toe* optimal agents can be constructed algorithmically without relying on any memory component [12], this is not necessarily true for games that involve more sophisticated strategies such as *Chess, Checkers* or *Go*. As the complexity of the game increases so does the resulting search space required for the game tree. Reasonably complex, deterministic games such as checkers typically rely on the use of the *Alpha-Beta* pruning algorithm to determine the next best move for the agent to make, however as the number of ply required to search further into the future increases so too does the computational complexity. World class game playing agents such as *Chinook* in Checkers [15] have resorted to the use of end-game databases to address this issue. End-game databases provide a persistent memory of the exact outcome of a game from a certain position. The use of this memory component has substantially improved the performance of the agent [16]. The world-class chess machine *Deep Blue* also included an end-game database although it's success depended less upon it and more on the inclusion of a database of Grandmaster games which were used to influence *Deep Blue's* decisions [4].

Chess and checkers are regarded as *deterministic* games with *perfect information*. There is no chance involved and both players can look upon the board and get all the information they need to make the best move possible. A separate category of games, classed as *stochastic* games with *imperfect information* are different, in that elements of chance play a role and information is hidden from the player. Poker is a game that involves chance and hidden information. Approaches to computer poker have mainly focussed around the use of game theory and adaptive imperfect information search [2, 21]. To be successful at games like poker a player has to be able to read his/her opponent, i.e. to compensate for missing information they have to make decisions based upon how their opponent has played in the past. We argue that at this point memory in a game playing agent not only becomes beneficial, but imperative. An early attempt to solve this problem was the poker playing program nicknamed *Poki*. *Poki* was developed by the University of Alberta Computer Poker Research Group[1] and used opponent modelling tables to keep track of how opponents played [3]. This memory for an opponent aided the agent in adapting its playing style accordingly.

The above examples have demonstrated the benefits that various forms of memory can have. This idea can be extended by considering programs that make decisions primarily based on memory, such as experience-based, lazy learners.

3 Lazy Learners

One approach to game AI focuses around the construction and traversal of game trees [4]. Another approach is to use machine learning algorithms to develop game playing agents [7, 18]. Many machine learning algorithms are classified as *eager learners*. An eager learner learns an approximation to a target function through training examples before any novel queries are encountered [11]. *Lazy learners* have also been developed and applied to game playing [13, 14]. Lazy learners usually bypass any computationally expensive training period and simply construct a local approximation to a target function when a new query is encountered [1]. When considering the problem of playing multiple, arbitrary games we believe that the use of lazy learners could prove beneficial due to the fact that lazy learners are highly adaptive to novel situations [1]. Experienced based or case-based systems can be considered lazy learners [10]. Detailed below are a number of experience-based agents that have been developed for specific games, with varying degrees of success.

Experience-based learning techniques were applied to the game of Othello in [6] with some success. The result was a system called GINA. De Jong and Schultz augmented a search-based Othello playing program with an experience base that was added to as GINA played more games. Each experience in the experience base was assigned a success rating which approximated the value that would have been found in a minimax search tree, coupled with a frequency counter that represented the confidence of the estimate. The results showed that the

[1] http://poker.cs.ualberta.ca/

use of experience-based learning was highly effective in improving both speed of decision making and skill in the game of Othello when challenging non-adaptive, minimax based opponents.

[13] produced CHEBR, a system to play the game of checkers via *automatic case elicitation*, whereby an agent with no prior domain knowledge acquired experience by simply playing games of checkers in real-time. CHEBR taught itself to play checkers better than an agent with initial knowledge of the game.

[17] reasoned that approaches to computer chess that used alpha-beta pruning algorithms employed a brute-force search strategy that considered many unnecessary lines of play. Sinclair focussed on forward pruning using example-based reasoning based on games played by human experts. An example base was built by analysing a collection of 16,728 expert games using *Principle Component Analysis* to reduce dimensionality and recording the move made given a board position. A separate test set of unseen positions was then used to assess the chosen move by the system. The results indicate that stronger moves were generated during earlier stages of the game when the example base held many instances and therefore similarity was high. However, this deteriorated as the move number increased as the example base became more sparse. Sinclair concludes that search based solutions do not transfer well to other problem domains they weren't designed for and proposes that example-based pruning may be well suited to handle imperfect information and problems where domain knowledge is incomplete [17].

Finally, case-based reasoning was applied to a stochastic, imperfect information environment in [14, 20]. A case-based poker player was developed (*Casper*) that made decisions in the game of Texas Hold'em by retrieving similar scenarios from it's memory and re-using these decisions. *Casper* was able to play evenly against the University of Alberta's *Pokibot*, upon which it was based, whilst avoiding the need for an intensive knowledge engineering effort.

As mentioned before the approaches discussed above have all focussed on specific domains. The next problem we wish to address is how an experience-based approach could be extrapolated to handle playing any arbitrary game it was presented with.

4 Analogy

Programs such as *Casper*, GINA and CHEBR described above have been created to focus on one particular domain - poker, othello and checkers respectively. Furthermore, one of the advantages of experience-based learning is an ability to generalise well [10]. Therefore, it is our opinion that a system with an initial memory of one or more game-related domains coupled with an ability to analogise beyond that domain could be used to address the problem of general game playing, where the same system uses knowledge it has obtained from previous games to inform it's decisions for a totally novel game.

Analogical reasoning has been successfully applied in the `Prodigy/Analogy` system [19]. Veloso combined derivational analogy with a base-level planning sys-

tem. Derivational analogy considers how a problem was solved rather than simply reusing old solutions for new problems [5]. This is achieved by taking the problem solving context into account. [19] describes the use of the `Prodigy/Analogy` system within a logistics transportation domain. When a problem is successfully solved an episodic solution trace is retained in a case base. This trace highlights justifications that support the decisions made. As new problems are encountered similar episodes are retrieved and their justifications are interpreted within the context of the new problem. One or more of the retrieved cases are then used to guide the problem solving process. This analogical reasoning process has allowed the successful transfer of skills within a complex domain, without a dependence on axiomatic domain knowledge. Resulting in a large increase in the amount and complexity of problems that can be solved compared to the base-level planning system [19].

Hinrichs and Forbus combine experimentation, analogy and qualitative modelling to the domain of a turn based strategy game [9]. The sub goal of optimizing food production within the Freeciv [8] strategy game is evaluated. Hinrichs and Forbus report that with the addition of learning from past failed cases, their experimental results indicate an improvement in the task of optimizing food production [9]. They propose that the use of analogy and qualitative reasoning is a viable approach to transfer learning, whereby a system is trained on one set of tasks and its learning subsequently measured on a set of related, but distinct tasks.

Analogical reasoning could perhaps produce similar results in a general game playing environment. Consider an example involving card games. The *Casper* system [14] plays Texas Hold'em poker entirely from memory. Through *Casper's* collection of experiences it has learned that cards such as *Jacks, Queens, Kings* and *Aces* are high valued cards. This knowledge could be generalised to other poker variants or even other card games as an initial assumption. If in fact this assumption proved incorrect (e.g. *Aces* are low) the system could compensate for its initial incorrect assumption by it's ability to quickly adapt to new situations. Of course for general game playing, games entirely outside the realm of card-games would need to also be considered. This requires further extrapolation, however we believe this is not unreasonable. Take, for example, a situation where we wish to generalise knowledge a system may have about card values to aid it in playing a game that involves the throw of a dice. Given that the system assigns high value to high card values an initial assumption the system could infer would be that the same applies for dice values. Hence, the system would value rolling double sixes opposed to double ones. By recording the outcome of a game, the system could successively evaluate whether this analogy is relevant or not.

As the system played more and more games and accumulated more knowledge about the games it has played it's experience-base would grow, allowing it to make further inferences and generalisations about different games it encountered. By consistently maintaining it's knowledge-base the system could drop analogies that proved incorrect and strengthen others that contributed to

successful outcomes. It is hoped this process would improve the general game playing abilities of the system.

We believe that an experience-based, lazy learner would provide the flexibility required to handle the type of generalisation described above.

5 Conclusion

In conclusion, the idea of a lazy learning agent has been proposed that relies on memory and analogy to generalise knowledge gained in one domain with the intention of applying it to another. We believe this approach could be beneficial to general game playing due to the fact that experience-based, lazy learners are able to adapt well to new situations and have been shown to be successful in a range of game environments e.g. deterministic vs. stochastic. Furthermore, analogical reasoning has demonstrated an ability to generalise skills within complex domains.

A discussion of the importance of memory in game-playing agents was presented. It was shown that memory can take many forms, but mostly relies on the encoding of specific game knowledge into databases or case-bases. Successful agents in board and card games have been used as examples to highlight the types of memory available and how it has been used effectively.

The Prodigy/Analogy system was discussed as an example that has achieved success via analogical reasoning. Finally, we speculated about the possibilities of analogy coupled with experience-based learners to generalise game knowledge which could be used as a basis for a general game playing agent.

References

1. Aha, D, W.: Editorial. Artificial Intelligence Review. 11(1–5):7–10, (1997)
2. Billings, D., Burch, N., Davidson, A., Holte, R, C., Schaeffer, J., Schauenberg T., Szafron D.: Approximating Game-Theoretic Optimal Strategies for Full-scale Poker. IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann. 661–668 (2003)
3. Billings, D., Davidson, A., Schaeffer, J., Szafron, D.: The challenge of poker. Artificial Intelligence. (134): 201–240 (2002)
4. Campbell, M., Hoane, J., Hsu, F.: Deep Blue. Artificial Intelligence. (134). 57–83 (2002)
5. Carbonell, J, G.: Derivational Analogy and Its Role in Problem Solving. Proceedings of the National Conference on Artificial Intelligence. AAAI Press. 64–69 (1983)
6. De Jong, K., Schultz, A, C.: Using Experience-Based Learning in Game Playing. ML. 284–290 (1988)
7. Fogel, D, B.: Evolving a checkers player without relying on human experience. Intelligence. (11). 20–27 (2000)
8. Freeciv. Freeciv official website. http://www.Freeciv.org/ (2006)
9. Hinrichs, T, R., Forbus, K, D.: Analogical Learning in a Turn-Based Strategy Game. IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence. 853–858 (2007)

10. Leake, D, B.: Case-based reasoning : experiences, lessons & future directions. Menlo Park, Calif. : AAAI Press ; Cambridge, Mass. : MIT Press, (1996)
11. Mitchell, T, M.: Machine learning. New York : McGraw-Hill, (1997)
12. Ostermiller, S.: Tic-Tac-Toe Strategy. http://ostermiller.org/tictactoeexpert.html January (2009)
13. Powell, J, H., Hauff, O, M., Hastings, J, D.: Utilizing case-based reasoning and automatic case elicitation to develop a self-taught knowledgeable agent. Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop (Technical Report WS-0404). AAAI Press. 77-81 (2004)
14. Rubin, J., Watson, I.: Investigating the Effectiveness of Applying Case-Based Reasoning to the Game of Texas Hold'em. Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference. AAAI Press. 417–422 (2007)
15. Schaeffer, J.: One jump ahead: challenging human supremacy in checkers. Springer-Verlag New York, Inc. New York, NY, USA. (1997)
16. Schaeffer, J., Björnsson, Y., Burch, N., Lake, R., Lu, P., Sutphen, S.: Building the Checkers 10-piece Endgame Databases. Advances in Computer Games, Many Games, Many Challenges, 10th International Conference. Kluwer, 193–210 (2003)
17. Sinclair, D.: Using Example-Based Reasoning for Selective Move Generation in Two Player Adversarial Games. Advances in Case-Based Reasoning, 4th European Workshop, EWCBR-98. Springer, Lecture Notes in Computer Science (1488). 126–135 (1998)
18. Tesauro, G.: Temporal Difference Learning and TD-Gammon. Commun. ACM. (38) 58–68 (1995)
19. Veloso, M, M.: Flexible Strategy Learning: Analogical Replay of Problem Solving Episodes. Proceedings of the 12th National Conference on Artificial Intelligence. AAAI Press. 595–600 (1994)
20. Watson, I., Rubin, J.: CASPER: A Case-Based Poker-Bot. AI 2008: Advances in Artificial Intelligence, 21st Australasian Joint Conference on Artificial Intelligence. Springer, Lecture Notes in Computer Science (5360). 594–600 (2008)
21. Zinkevich, M., Johanson, M., Bowling, M., Piccione, C.: Regret Minimization in Games with Incomplete Information. Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems. MIT Press. (2007)

Authoring Behaviours for Game Characters Reusing Automatically Generated Abstract Cases ⋆

Antonio A. Sánchez-Ruiz, David Llansó,
Marco Antonio Gómez-Martín and Pedro A. González-Calero

Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid, Spain
emails: {antsanch,llanso,marcoa}@fdi.ucm.es, pedro@sip.ucm.es

Abstract. Authoring the AI for non-player characters (NPCs) in modern video games is an increasingly complex task. Designers and programmers must collaborate to resolve a tension between believable agents with emergent behaviours and scripted story lines. Behaviour trees (BTs) have been proposed as an expressive mechanism that let designers author complex behaviours along the lines of the story they want to tell.
On the other hand, BTs appear too complex for non programmers. In this paper, we propose the use of *abstract cases* automatically generated through planning to assist designers when building BTs. In order to make this approach feasible within state-of-the-art video game technology, we generate the planning domain through an extension of the component-based approach, a widely used technique for representing entities in commercial video games.

1 Introduction

According to the number of papers dedicated to the subject in the editions 3 and 4 of the AI Game Programming Wisdom [8, 9], Behaviour Trees (BTs) are the technology of choice for designing the AI of NPCs in the game industry. BTs are proposed as an evolution for hierarchical finite state machines (HFSM) intended to solve their scalability problems by emphasizing behaviour reuse.

Behaviour trees have been proposed as an expressive mechanism that let designers author complex behaviours along the lines of the story they want to tell, but at the same time, BTs appear as a too complex mechanism for non programmers [2, 3]. Commercial game development teams usually build some support tools in the form of tree editors, where the designer can choose from a set of predefined composite nodes, conditions to be checked, and basic actions that can be included in the tree. Nevertheless, in practice, there is a tension between the freedom that the designers require to include their narrative in the game and the effort required from programmers to debug faulty AI authored by non-programmers.

In this paper we propose the use of ontologies and planning techniques to assist game designers when authoring the AI for non-player characters. The design of BTs

⋆ Supported by the Spanish Ministry of Science and Education (TIN2006-15140-C03-02 and TIN2006-15202-C03-03)

is usually an interactive process in which the designer incrementally adds new tree branches to make the NPC react to new situations. We propose an automatic way to compute abstract cases or solutions to those new situations, that designers may adapt before incorporating them to the current BT.

A drawback for using declarative knowledge-intensive AI techniques in games is the additional effort required to model the domain. In this case we require having a model of the actions that NPCs can do in the game world. In order to close this gap between academic and industrial game AI, we propose generating the planning domain through an extension of the component-based approach for representing entities, which is widely used in commercial video games.

The rest of the paper runs as follows. Next section introduces the two techniques from commercial video games incorporated into our approach: component-based game entities, and behaviour trees. Section 3 describes the main ideas of our proposal for authoring BTs from automatically generated abstract cases. Sections 4 and 5 provide the details and exemplify the approach, first describing the generation of the planning domain and then the use of abstract cases to author behaviour trees. Last section reviews related work and concludes the paper.

2 Background

2.1 Components

In the development of a virtual environment, the layer responsible of the management of the entities is usually created using an object-oriented programming language such as C++. Over the years this object-management system has been based on an inheritance hierarchy, where all different kinds of entities derive from the same base class often called `CEntity`.

Some of the consequences of this extensive use of class inheritance are, among others, an increase in the compilation time [5], a code base difficult to understand and big base classes. To mention just two examples, the base class of Half-Life 1 had 87 methods and 20 public attributes while Sims 1 ended up with more than 100 methods.

Due to all these problems developers tend to use a different approach, the so called component-based systems [13, 10]. Instead of having entities of a concrete class which define their exact behaviour, now each entity is just a component container where every functionality, skill or ability that the entity has, is implemented by a component. From the developer point of view, every component inherits from the `IComponent` class or interface, while an entity becomes just a list of `IComponents`.

As entities are now just a list of components, the concrete components (or abilities) that constitute them may be specified in an external file that is processed in execution time. This approach eases the creation of new kind of entities, because it does not require any development task but just the selection of the different skills we want our new entity to have from a set of components. In order to allow fine-grained adjustment of the behaviour (or skills) of different entities, their definition may also set the values of different attributes that components use as parameters of their behaviours.

Both, the list of components of an entity and the initial values of their parameters, are usually stored in a separated file that can be seen as the file that describes the main

```
<blueprint>
<entity type="goblin" ontType="Goblin" parentOnt="Monster">
  <components list="Take, MoveTo, TakeCover, MeleeAttack,
                    LongRangeAttack, Charge-At, ..."/>
  <attributes>
    <attrib name = "strength" value = "weak"/>
    <attrib name = "weapon-tech"
            value = "rudimentary, elaborate"/>
    <attrib name = "height" value = "short"/>
    ...
  </attributes>
</entity>
...
</blueprints>
```

Fig. 1. Partial list of blueprints file

characteristics of every entity. That file, usually known as the *blueprint*, is parsed by the game engine at the beginning of its execution. When the game loads a new level from the map file, it iterates over the list of the level entities and uses the blueprint file for creating and launching them.

Figure 1 lists a portion of one of such files that describes the goblin entity. The description of the entity has two main parts, the list of components and the list of attributes. Goblins have components that allow them, among other things, to go through the environment and to pick up objects. All these skills are parameterized in the `attributes` section. For example, the `strength` attribute influences the `Take` component, while the `height` predefines the kind of objects the `TakeCover` component should consider as protections.

2.2 Behaviour Trees

BTs define an AI driven by goals, in which complex behaviours can be created combining simpler ones using a hierarchical approach. Nodes in a BT represent behaviours, where an inner node is a composite behaviour and a leaf in the tree represents an action. To promote reusability, behaviours do not include the conditions that lead to transitions. Those conditions are represented as guards in nodes of the tree so that the same behaviour can be used in different contexts with different guards. To further promote reusability, behaviours may be parameterized, so that in a particular context parameters are bound to actual values in the map. This way, a node in such a behaviour tree is represented through: a behaviour (be it composite or a primitive action); bindings for the parameters of that behaviour; and a guard condition that must be true at run time for that behaviour to be activated.

From the different models of execution for BTs, we choose one where a BT has an active branch, going from the root to a leaf, of behaviours being executed. Every tick of the game, some guards may get evaluated and some behaviours may finish, be it

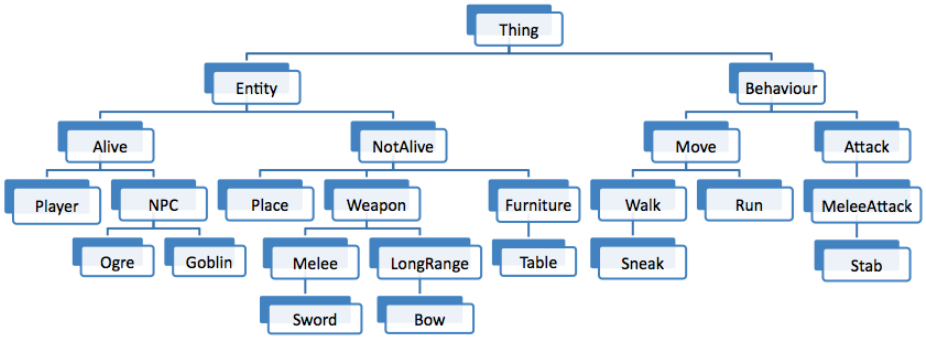


Fig. 2. Ontology that defines the domain vocabulary

successfully or with failure, eventually leading to the expansion of a new active branch in the tree.

Although more complex types of composites are described in the literature, for the goals of this paper we only require three types of composites: sequences, static priority list and dynamic priority list. A sequence composite behaviour executes its children in the order they are defined, succeeding when every children succeeds and terminating with failure whenever one of the children fails. Children behaviours of a sequence are not guarded by conditions. A static priority list is a composite node that evaluates its children guards in order and activates the first child whose guard is true. A dynamic priority list, in its turn, re-evaluates the guards of its children with higher priority (the first child being the one with highest priority) than the active one, and switches to a higher priority child whenever possible.

3 Generating Abstract Cases to Support BT Creation

The creation of BTs is a difficult task that designers usually perform by means of a try-and-fail process. The consequence is that the final quality of the BTs depends to a large extent upon the ability and experience of designers. Our proposal consists in helping them by means of abstract cases that are automatically computed using planning and ontologies. This way, we suggest sets of solutions that designers can adapt and add to the current BT.

In order to use planning we need to describe the domain and planning actions using a formal language.The description of a planning domain includes two main parts: (a) the description of the predicates that conform the domain vocabulary and how they are related among them, and (b) a set of planning actions. We propose the use of *ontologies* to represent the first part, that is, the vocabulary and the domain constraints. Figure 2 shows an example of ontology that intuitively describes different components of a game. The set of planning actions, on the other hand, is strongly related to the available basic behaviours in the game. Planning actions are described in terms of preconditions and effects using the vocabulary available in the domain ontology.

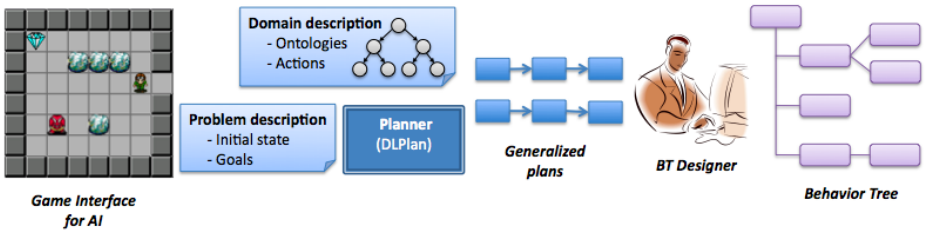


Fig. 3. Interactive process to create Behaviour Trees

Figure 3 summarizes our proposal to support the AI designer during the creation of BTs. By means of a graphical interface, that can be a simplified version of the game interface, the designer sets up a particular game scenario and some goals. Next, we automatically generate the equivalent symbolic description using the planning language, and by means of a planner, we compute all the possible plans that solve the problem. The planner that we use, DLPlan[1], is able to generalize the resulting plans using the domain ontology. This way, plans presented to the designer are not only specific plans for the current scenario but general strategies or abstract cases that can be reused in a broader set of situations (this will be described in detail later in Section 5). Then the designer can use those cases to complete the BT that is currently building. Let's remember that BTs are useful in a broad set of scenarios and thus, this is actually an interactive process in which the designer proposes different scenarios to the system and incrementally completes the BT using the retrieved solutions.

Finally, the process by which abstract cases are integrated to the current BT is, nowadays, manual, the designer is the only responsible of changing the BT to add the new branches. It is important to remark that the planner works with a limited model of the game, while the designer can take into account many more factors (like story plot or special situations) in order to select behaviours, modify preconditions under certain circumstances and set the priority of each alternative. The planner output, therefore, only shows different solutions that the designer must modify, validate or reject. Anyway, preconditions of the planning operators and the generalized plans computed by DLPlan are usually a good source of inspiration to define the guards of new nodes in the tree.

4 Generating the Planning Domain

To be able to use planning techniques, we need a symbolic representation of the world as well as the actions that each type of entity can perform. The basic approach is to create this description from scratch. However, this information is, at least partially, already in the C++ classes that programmers have to implement to develop the game and in the configuration files that define the different types of entities.

Our proposal is to use this information in order to automatically generate all the information required by the planner. In order to do that we have to minimally extend the information contained in the components implementation and *blueprint* file.

[1] Freely available at *http://sourceforge.net/projects/dlplan/*

The process starts with a base domain ontology that can be seen as the basic vocabulary of the game genre and it is independent of the concrete game being developed. This domain ontology is taken for granted and it includes the basic vocabulary for describing the new type of entities and actions that the game will incorporate. In other words, we start with an ontology similar to the one in Figure 2 but without the leaves that correspond to the concrete types of entities in the game.

In order to populate the ontology with the new entities, we use the *blueprints* file. As we showed in Section 2.1, this file has an entry per game entity, describing the set of components and attributes it has. The only new information we need to add to this file is two special fields in every entity, *ontType* and *parentOnt*, that set the corresponding symbolic name for this entity in the ontology and the branch or branches in which it must be added. This way, we can now add automatically new concepts in the ontology to represent these new types of entities.

Once the entity has been added to the ontology, we can also add information about their properties and the actions that they can carry out. This is done iterating over the list of components that the entity has, asking them which information has to be injected to the corresponding ontological entity. As an example, we would add the description of the `Goblin` entity that appears in Figure 1 with `canWalk`, `canTake`, `hasStrength.weak` and other properties.

As regards the operators the planner uses, we can extract them from the components. Most of the components are in fact the responsible of the execution of one or more actions over the environment. They first check if the action can be carried out and then execute it. Our method of automatically generating the planning operators consists in extending the components with an extra task: their self-description. In that sense, every component that represents a behaviour must be able to provide, through its programming interface, the planning action that describes it.

Figure 4 shows the operator description that components `MoveTo`, `TakeCover` and `Take` generate. The preconditions are specified in terms of the entities' properties.

5 Plan Generation and BT Authoring

This section describes how to build BTs using our approach, i.e., taking advantage of planning techniques to support designers during the process. Next, we introduce a concrete example in which a BT must be created to control a greedy goblin that has entered in a room to discover a diamond in the opposite corner. We will assume the existence of a graphical interface (As complex games usually provide [6]) to define different initial states and goals without having to deal with logical predicates but just setting items and units in the map and defining theirs attributes.

Let's start with the simplest situation, where the goblin and the diamond are in the same room and there are no enemies near. This goblin is a warrior well armed with a short sword, a small knife, a short bow and a sling. The room, in turn, contains some furniture: a table, two chairs and a bookcase. Although the designer does not know it, behind the scene this information is been automatically translated to a symbolic representation for the planer using the vocabulary in the ontology. Now, when the designer

WALK−TO(?who: alive, ?target: entity)
vars: ?r
pre: canWalk (?who), inRoom(?who, ?r), inRoom(?target, ?r), aloneInRoom(?who)
post: nextTo(?who, ? target)

TAKE−COVER(?who: alive, ?c: cover)
vars: ?r, ?s1, ?s2
pre: canTakeCover(?who), uncovered(?who), inRoom(?who, ?r), inRoom(?c, ?r),
cover(?c), hasSize(?who, ?s1), hasSize(?c, ?s2), lessEqSize(?s1, ?s2)
post: covered(?who)

TAKE(?who: alive, ?what: resource)
vars: ?w, ?s
pre: canTake(?who), nextTo(?who,?what), hasWeight(?what, ?w),
hasStrength (?who, ?s), enoughStrength(?s, ?w)
post: inInventary (?who, ?what)

Fig. 4. Planning operators corresponding to some basic behaviours.

defines the goal (the goblin gets the diamond), the planner shows the only possible plan: walk until the diamond location and take it:

1. `WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`

Actually, using the abstraction capabilities of DLPlan we are able to point out that this plan is applicable in several more scenarios, because the plan only requires *goblin1* to be an entity that can walk and take things and that is alone in the room, and *diamon1* to be a small item. The generalization process followed by DLPlan to reach this conclusion is based on the ontological domain definition and it is described in [11].

Using this information, the designer builds the red branch, with dashed borders, of the BT shown in Figure 5, that represents the only plan available in this scenario. It is important to mention that plans generated using the planner are sequences of actions that correspond to the leaves of the BT. The definition of internal nodes in the tree to group basic actions and to represent different alternatives is responsibility of designers.

Next, the designer must complete this basic BT to make it useful in other scenarios as well, for example when there is an enemy in the same room that has already detected the goblin. This time the planner computes several more possible plans:

1. `ChargeAt(goblin1,sword1,enemy1), WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`
2. `ChargeAt(goblin1,knife1,enemy1), WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`
3. `TakeCover(goblin1,table1), LRAttack(goblin1,bow1,enemy1), WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`
4. `TakeCover(goblin1,table1), LRAttack(goblin1,sling1,enemy1), WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`
5. `TakeCover(goblin1,bookcase1), LRAttack(goblin1,bow1,enemy1), WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`

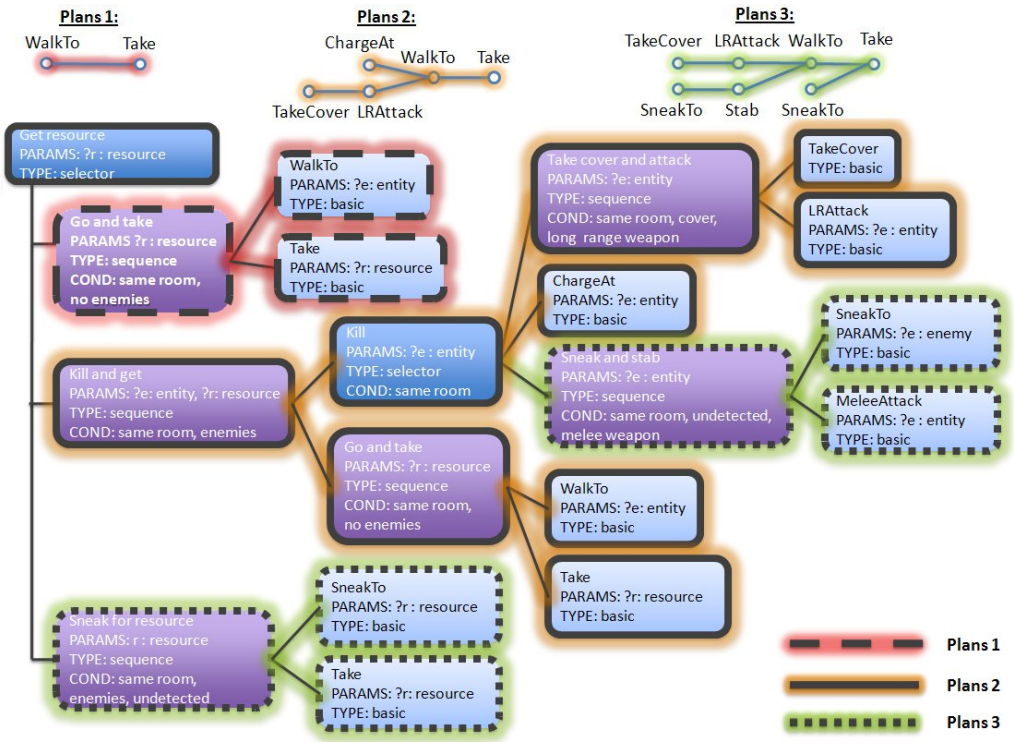


Fig. 5. Example of BT creation from different plans

6. `TakeCover(goblin1,bookcase1), LRAttack(goblin1,sling1,enemy1), WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`

It is important to mention that during the computation of these plans the planner has performed some interesting inferences using the domain knowledge. For example, the planner has used the table and the bookcase as possible covers and different weapons have been classified in melee or long range weapons. With those inferences, the six generated plans are in fact two different strategies parameterized with different values: charge against the enemy and then take the diamond; or look for a cover, attack the enemy from the distance and then take the diamond:

1. `ChargeAt(goblin1,sword1,enemy1), WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`
2. `TakeCover(goblin1,table1), LRAttack(goblin1,bow1,enemy1), WalkTo(goblin1,diamon1), Take(goblin1,diamon1)`

Again, using the abstraction capabilities of DLPlan, those plans are applicable in more general scenarios than the current one, in this example *sword1* represents any melee weapon and *bow1* any long range weapon.

The computation of plans and the later generalization is performed behind the scene, and so, the designer only sees the generalized plans. Then, he has to complete the previous behaviour tree to incorporate the new possibilities. The resulting BT is built adding the orange branches, with continuous borders, shown in Figure 5. Basically, the previous branch is only applicable if there are no enemies in the room, and in other case we have to kill the enemies first.

Finally, the designer wants to complete the BT with new branches that will be executed when there is an enemy in the room but he has not detected the goblin yet. This time, the planner computes several solutions that can be summarized in three strategies: (1) take a cover, attack from the distance, go until the diamond and take it; (2) sneak until the enemy, stab him, go until the diamond and take it; and (3) sneak until the diamond and take it (without killing the enemy).

Next, the designer adds these new alternatives to the BT, obtaining something similar to the Figure 5 . In this case, the strategy of looking for a cover and attacking using a long range weapon was already in the previous BT so we only need to add the other two plans (green branches with dotted borders).

6 Related Work and Conclusions

Authoring the AI for non-player characters in modern video games is an increasingly complex task. Behaviour Trees, as successor of Hierarchical Finite State Machines, are a promising and emergent technology to represent the complex behaviours that the market demands. However, designers that have to build these BT usually do not have a programming background and do not find them intuitive enough. Consequently, we have to provide the designers with good tools to support the BT creation process. In particular, we propose the combination of planning and ontologies to propose abstract cases to the designer, that will incorporate them into the BT after the required adaptations.

We have described an interactive process in which the designer proposes different scenarios and goals and the planner computes all the possible solutions and generates abstractions for the initial state. Then the designer uses that information to build a more robust and versatile BT that represents different plans as different branches. The use of off-line planning techniques, let us to explore the space of design possibilities, and does not have the computational cost of planning during the game execution. The use of ontologies, on the other hand, provides an intuitive way to describe scenarios and interact with the planner. In order to make this approach feasible within state-of-the-art video game technology, we generate the planning domain through an extension of the component-based approach for representing entities which is widely used in commercial video games.

Related approaches have been described in [7, 4]. Pizzi et al. [7] use a planner to compute every combination of actions to solve each level, and show them to the human designer like a comic, to let him check if there are any gaps in the storyline or in the design of the level. Another related work is the one described in [4] where the authors propose the use of planning to coordinate the behaviours for NPCs that are not main characters in the storyline of role games. Our approach does not generate the full behaviour for an NPC as [4], but proposes particular plans for particular situations, and

the human designer is responsible for incorporating those traces into the tree he is designing. On the other hand, we differ from [7] because we intend to assist the designer on building the behaviour for an NPC, instead of supporting a kind of validation of a game level by exploring the solutions that the player may try, so that the designer may choose to re-design the level in order to avoid certain solutions.

Regarding CBR in games, most of the works in literature focus on how to use cases during the execution of the game, either to improve the quality of the final AI or to improve the performance of the AI engine. In [1], cases consist of strategies applicable to specific situations in the real-time strategy game Stratagus. Using these strategies, they were able to beat opponents considered "very hard". Another interesting work presents the multi-layered architecture CARL [12], that combines CBR and reinforcement learning to achieve a transfer-learning goal in another real time strategy game.

As future work we will study how to improve the interface between the planner and the BT authoring tool, and how to semi-automate the translation process between both representations. We are also interested in developing debugging tools to help the designer to understand the inferences that the planner does why the planner proposes some solutions and rejects others in particular scenarios.

References

1. D. W. Aha, M. Molineaux, and M. J. V. Ponsen. Learning to win: Case-based plan selection in a real-time strategy game. In *ICCBR*, pages 5–20, 2005.
2. D. Isla. Handling complexity in the Halo 2 ai. In *Game Developers Conference*, 2005.
3. D. Isla. Halo 3 - building a better battle. In *Game Developers Conference*, 2008.
4. J. P. Kelly, A. Botea, and S. Koenig. Offline Planning with Hierarchical Task Networks in Video Games. In *AIIDE*, 2008.
5. J. Lakos. *Large Scale C++ Software Design*. Addison Wesley, 1996.
6. M. McNaughton, M. Cutumisu, D. Szafron, J. Schaeffer, J. Redford, and D. Parker. Scriptease: Generating scripting code for computer role-playing games. In *ASE*, pages 386–387, 2004.
7. D. Pizzi, M. Cavazza, A. Whittaker, and J.-L. Lugrin. Automatic Generation of Game Level Solutions as Storyboards. In *AIIDE*, 2008.
8. S. Rabin, editor. *AI Game Programming Wisdom 3*. Charles River Media, 2006.
9. S. Rabin, editor. *AI Game Programming Wisdom 4*. Charles River Media, 2008.
10. B. Rene. *Game Programming Gems 5*, chapter Component Based Object Management. Charles River Media, 2005.
11. A. A. Sánchez-Ruiz, P. A. González-Calero, and B. Díaz-Agudo. Abstraction in Knowledge-Rich Models for Case-Based Planning. In *Proc. of International Conference on Case-Based Reasoning*, 2009.
12. M. Sharma, M. Holmes, J. Santamaria, , A. Irani, C. Isbell, and A. Ram. Transfer learning in real-time strategy games using hybrid cbr/rl. In *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.
13. M. West. Evolve your hierarchy. *Game Developer*, 13(3):51–54, Mar. 2006.

Case-based reasoning for improved micromanagement in real-time strategy games.

Tomasz Szczepański and Agnar Aamodt

Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
NO-7491, Trondheim, Norway
szczepan@stud.ntnu.no, agnar@idi.ntnu.no

Abstract. Real-time strategy (RTS) gameplay can be divided into macromanagement and micromanagement. Other researches have employed Case-based reasoning (CBR) and case-based planning in real-time strategy games that have beaten static scripted computer opponents. Unlike much of the previous work where CBR and case-based planning is used to improve the macromanagement in RTS games, we present a CBR system that can be used to improve the micromanagement quality in RTS games. We explore various ways of case matching mechanisms suited for a micromanagement environment. By managing to beat a hard-coded computer opponent we conclude that our approach can be used to aid human players against computer opponents and increase the quality of the micromanagement of a computer player. Our experiments have been conducted within the Warcraft 3 gaming environment.

Keywords: case-based reasoning, real-time strategy games, micromanagement

1 Introduction

The computer player performance in the popular real-time strategy (RTS) game genre has always been poor. Although AI techniques have successfully been applied in other related game genres like classic board games, the computer players in RTS games (often commonly referred to as game AI) are still lagging behind and can easily be defeated by amateurs [4]. Unlike games like chess where each player waits until the opponent makes his move, game flow in RTS games is simultaneous and continuous. Players usually compete within areas of resource gathering, structure building and army management, watching hundreds or even thousands of interacting objects from a top-down perspective. Due to the genre's nature, the game AI has to make decisions in real-time in an inaccessible, non-deterministic, dynamic and continuous environment with vast search spaces. Here traditional search methods no longer apply [4, 6, 8].

RTS gameplay can further be divided into macromanagement and micromanagement. Micromanagement is the way a player manages his or her units during combat or resource gathering. In the RTS game context a unit is a single character that may have several associated attributes like attack type or movement

speed. In contrast to macromanagement, which focuses on overall game strategies involving a player's army like positioning or army composition, micromanagement describes all the small details that involve the individual units themselves (enemy unit targeting, spell casting etc.). The macromanagement has been addressed by many studies of CBR in relation to RTS games [5, 6, 7]. However, very little has been done in relation to micromanagement which is usually just handled by the game itself.

In the study presented here a case-based reasoning system has been implemented in the game Warcraft 3. The system focuses on the micromanagement of units during battles. Specifically, we are interested in how case-based reasoning can improve the quality of micromanagement in a real-time strategy game. We study how to beat the already implemented game AI in Warcraft 3 as well as how a CBR system can be used to aid human players. To avoid confusion, the already implemented game AI in Warcraft 3 will be referred to as "computer opponent" while our implemented CBR system will be referred to as "CBR player".

The rest of this paper is structured as follows: Chapter 2 gives a brief introduction to micromanagement in Warcraft 3. In Chapter 3, 4, 5 and 6 we present our implemented CBR system, with the results presented in Chapter 7. Chapter 8 continues with a discussion of the results. Finally Chapter 9 summarizes and provides conclusions about the project.

2 Background

Warcraft 3: Reign of Chaos is an RTS computer game released by Blizzard Entertainment in July 2002. During typical "melee" gameplay in Warcraft 3 each player starts with a main building and five workers. By gathering resources new buildings can be constructed and the player gains access to new units, technologies and structures (Figure 1). By specific build strategies and unit control in battle each player tries to get the upper hand to win the game by eventually destroying all opponents [2, 3]. The way Warcraft 3 is constructed makes micromanagement one of the most important aspects of the game. A micromanaging player can easily defeat a non-micromanaging player without loosing a single unit in most battle setups.

Before choosing Warcraft 3 as our game environment we also considered using the Wargus/Stratagus environment [10], an open source clone of the game Warcraft 2 (older version of Warcraft 3), and ORTS [11], an open source programming environment for RTS games. Wargus has been successfully used in other studies that show how a CBR approach can beat scripted computer opponents [6]. However, those environments focus mostly on other aspects of RTS games than micromanagement in battle [9]. Wargus is mostly designed for macromanagement while ORTS is focusing more on low level tasks like pathfinding, formations and resource gathering.

From a micromanagement perspective, environments like Wargus compared to Warcraft 3, are not very friendly for human players. Compared to Warcraft 3, units in Wargus die quickly, the interface does not give a good overview of unit status or support efficient unit management. Under such conditions, a human player has problems to react in time and can not micromanage efficiently. Since both Wargus and ORTS are open source, they can be converted to a suitable micromanagement

environment for our needs. The problem is the amount of work needed to do so. On the other hand, since Warcraft 3 is not open source, the whole CBR system must be created from scratch by tools offered by Warcraft 3. Luckily, Warcraft 3 has an internal scripting language [13] that makes this possible. It is also worth mentioning that despite its age, Warcraft 3 is still a popular game. This makes it easy to find suitable human testers for systems implemented in its game environment. Because of Warcraft 3's human-friendly interface, suitable human tester availability and due to our study's time limitation, we decided to use Warcraft 3 as out game environment.

Fig. 1. A screenshot from Warcraft 3.

3 System Architecture

When designing and implementing our system we had to keep in mind that it would run on an end-user's computer inside the Warcraft 3 game. Therefore we wanted our CBR system to be simple and efficient. We also limited our game environment from factors irrelevant to micromanagement. The system uses the three steps in the 4R CBR model [1] and can be summarized by the following algorithm:

1. Store the current game state as a new unsolved case in memory.
2. Compare the new case to the existing cases in memory and retrieve the most similar one.
3. Map the solution from the chosen case and execute the retrieved case.
4. If in training mode and a problem is observed, add a new case to the system.
5. Wait 1 second then go to 1.

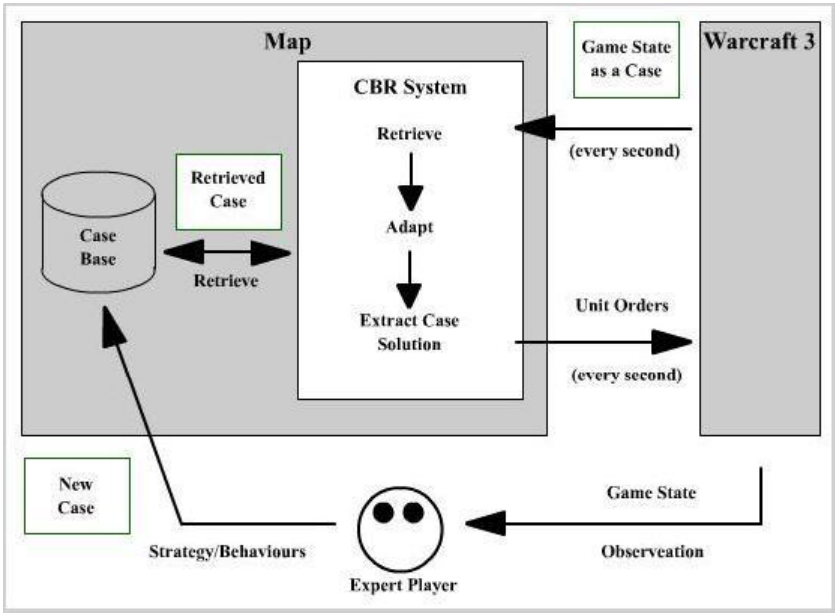


Fig. 2. Overview of the CBR system.

Figure 2 shows a more detailed diagram of the CBR system. Every second, the current game state is abstracted into a case and compared to previously stored and solved cases. The most similar case is retrieved and the solution provided by that case is executed. At any time during execution new cases can be added. This is typically done when the system is executing a wrong case or a new case is needed (decided by an expert). Adding a case is done by halting the game and entering a new strategy that is supposed to be followed next. For a screenshot of the CBR system see Figure 3.

4 Case Structure and Indexing

The game state relevant to our system consists of units battling each other in a battlefield area (the battlefield in our system is a square shaped region where the battle takes place). We are not interested in a player's resources, structures or units outside the battlefield. Our choice of relevant attributes that were added to the case descriptions describing the game state were "unit type", unit remaining "hit points" (determines the amount of damage a unit can withstand), unit remaining "mana points" (mana in Warcraft 3 is "magical energy" that some units can use to cast spells) and "unit position" relative to the battlefield (x and y coordinate).

Our case structure is a simplified version of the proposed structure by Cheng et al. [5] and consists of a condition part, description part and a solution part. The condition part of a case consists of a set of conditions that have to be fulfilled to be able to

execute the solution part of the case. This is also the primary indexing part of our cases, i.e. only cases where the condition parts can be fulfilled are compared to the current game state. If the solution part of a case has an action that orders a unit to cast a certain spell, the condition part needs to check whether such a unit exists on the battlefield and if it exists, whether it has enough mana points to cast the spell. The description part of a case simply describes the situation when the case was learned. It consists of two arrays, one for each player, which stores our chosen attributes (unit type, remaining hit points, remaining mana points, x-coordinate and y-coordinate for all units involved). The description part is used for case matching. Finally, the solution part consists of a strategy that contains actions and behaviors. Actions are simply individual unit orders like attacking an enemy unit, moving to a specific point, healing a friendly unit etc. A behavior is a set of actions that trigger when some condition or conditions are met. Behaviors are used to both decrease system reaction time (currently 1 second) and the number of cases. Something important might happen that requires a fast response between the 1-second intervals our CBR system currently uses. By using behaviors it is possible to react quickly by setting the important happening as a condition. An example of a behavior of a unit might be to retreat when getting low on health or cast a spell when having enough mana points. Using behaviors can also potentially decrease the number of cases needed. For example if a player has ten different units in his or her army and wants to retreat with a unit once it gets low on hit points, at least ten cases must be learned (one for each unit when its hit points are at a certain low value). By setting an escape behavior on the units involved, only one case would be needed.

5 Similarity Metric

Our case retrieval process uses a nearest neighbor algorithm. It is a similar algorithm that Ontañón et al. use in their project [7]. The difference is mainly that our approach does not use goals (therefore the part of the equation matching goals is not included). The similarity metric is as follows:

$$d(c1, c2) = dGS(c1.U, c2.U) \quad (\mathbf{1})$$

Here c1 and c2 are the compared cases while c1.U and c2.U are the sets of units contained in those cases. dGS( ) is a Euclidean distance between game states (between units). To measure distance between two units the following distance is used (2.1 if same unit type, 2.2 otherwise):

$$dGS(u1, u2) = \sqrt{\sum ((pi - qi) / Pi)^2} \quad . \quad (\mathbf{2.1})$$

$$dGS(u1, u2) = \sqrt{\sum 1} \quad . \quad (\mathbf{2.2})$$

Fig. 3. A screenshot from the CBR system during training. Left: similarity trace generated by comparing a new unsolved case to existing solved cases. Upper right: helping tool used by the expert during training. Lower right: overview over learned cases.

u1 and u2 are the units compared. Pi is the maximum value of the attribute i (pi and qi are the values of attribute i belonging to the compared units). When units are different, the distances between their corresponding attributes is set to 1.

This metric favors cases with equal amount of units. The similarity metric can have a value between 0 and 2√n. Here n is the maximum number of units contained in either the compared case or the current game state. n usually has values between 4 and 40, which is the amount of units involved in a typical Warcraft 3 battle. Similarity values of 2√n can be obtained by comparing two totally different cases, where n is the number of units in the largest case (the case containing most units). During case retrieval all executable cases are compared, by use of the described metric, with the current game state, and the most similar case (the case with the smallest d(c1, c2) value) is chosen. If no case is found, the units just follow their old orders from the last case executed.

6 Case Matching

The case matching process starts by retrieving the subset of cases from the case base which fulfill the condition part of the new case. This requires a 100% match and increases the performance of the system as well as prevents selection of cases that can not be executed. The similarity assessment is then executed among these cases based on the similarity of the description part of the cases (hit points, mana points, unit type

and position). We implemented four different case matching methods, as detailed by Szczepański [12]:

1. Unit sorting. Units are constantly sorted by remaining hit points. The existing cases are also sorted in the same way. During matching the first unit in the new case is matched with the first unit in an old case etc.
2. Unit similarity. Most similar units matched together, same metric as described in Chapter 5. The existing cases are not sorted.
3. Commitment. Units are numerated before battle, numeration does not change. During matching equally numerated units are compared.
4. Unit sorting with enemy in reverse order. Similar approach as 1. Differs by sorting of enemy units (sorted inversely).

Approach 1 and 4 are motivated by one of the most important principles of micromanagement: killing one unit at the time reduces the DPS (damage per second) of the enemy army faster that killing several units at the time (it is assumed that killing n units at the time takes n times longer than killing one unit). This is because in Warcraft 3 a unit deals the same amount of damage regardless of its remaining hit points unless it is dead. Approach 2 tried to find the most similar units in the current new case (current game state) and the retrieved case when applying strategies and behaviors. The 3rd approach was the simplest possible. Units were enumerated before battle and the enumeration did not change.

Each matching approach was tested in a very simple setting of two teams of 5 units battling each other. On that basis approach 4 was selected because it was the approach that needed the least amount of cases to beat a computer opponent. Enemy units are sorted such that the first unit has the least remaining hit points while the CBR system's units are sorted in the opposite way (the first unit has the most remaining hit points). This makes the cases reusable (attacking units with lowest remaining hit points is most often the correct/optimal play). The presorting of units in cases also serves the role of case adaptation. This is because units from the new case that are matched together with units from the retrieved case also receive the corresponding actions and behaviors from the solution part of the retrieved case (unit type must be the same).

7 Testing

The main purpose of the tests was to compare the performance of our CBR system against human players to the performance of the computer opponent in Warcraft 3 in the same setting. Having our limited case representation in mind (no information about terrain etc.) and that our CBR system focuses purely on micromanagement, we isolated the testing battlefield to a perfectly flat rectangular area. The battlefield, shown in Figure 3, contains two armies (controlled by the CBR player and the computer opponent) fighting each other in a mirror battle (both armies are equal). The testing was divided into four parts. We started by training the CBR system against a computer player. We then tested the trained system against the computer player

without training. Next, both the CBR system and the computer player were tested against human opponents. Finally, to test the applicability of the CBR system as an aiding tool, both human players and the CBR system were tested in a cooperative mode against the computer player. A test proceeded as long as there were units on the battlefield belonging to different players. The human players were divided into three categories: novice, casual and expert players. A player would be considered novice if he had some minor RTS experience while expert would be a person able to easily beat an "insane" computer opponent (the hardest difficulty of a computer opponent in Warcraft 3). We had 10 persons for the tests. Those were students (the novice players) and people from the Warcraft 3 gaming community (the casual and expert players).

The CBR player was trained by playing repeatedly against a computer opponent. Whenever the system did execute an inappropriate case an expert halted the game and added a new case. Whether or not an executed case was inappropriate with respect to the current game state, was solely determined by the expert player. After learning 25 cases, our system was able to beat the game AI in Warcraft 3 in our setting (details in Table 1). A screenshot from the training of the CBR system is shown in Figure 3.

Table 1. Testing resuslts during training against the computer opponent.

Training step	Cases added	Units lost by CBR system	Computer units killed
1	7	All	7
2	6	All	11
3	5	All	6
4	3	All	7
5	3	All	7
6	1	5	All

When finished with the training, we tested the trained CBR player against the insane computer opponent. The CBR player won all of the 10 games by an average loss of 2.5 units out of 14 per game. Next, we performed tests where human players played against a computer opponent. The feedback received from novice players was that the micromanagement provided by the insane computer opponent was simply too hard to beat. Similar feedback was received from the casual players. The expert players, on the other hand, complained that the insane computer opponent was too easy to beat (the expert players often managed to beat the computer opponent without loosing a single unit). The same response was received when human testers played against the CBR player. The major difference was that the expert players needed some games to figure out how to beat the CBR system. They did not manage to beat the CBR player without loosing only a few units. Finally players played with the CBR player on their side such that the CBR player was given the control of all unselected units. Interestingly, all groups of players managed to beat the insane computer opponent. The dependency observed was that the novice and casual players did better in this setting than the experienced players. The experienced players found this setting very disturbing because the CBR player was destroying their executed strategies. One example of this was when an expert ordered a near-death unit to retreat while a couple of seconds later the CBR player brought it back into battle. The novice and the casual

players liked this setting because they could relax and focus only on a few units instead of the whole army.

8 Discussion

Our approach managed to create a working system that was capable of beating both computer and novice/casual human players. The system also added more challenge to the micromanagement for experienced players. Having the CBR system on their side, novice and casual players were able to easily beat the insane computer opponent. By using a setup that gives a novice or casual player aid from a CBR player, while putting the CBR player against expert players, one should be able to increase the entertainment value of a played RTS game. However it is also important to note that since our number of human testers was low, the results are a mere indication of what to expect from our CBR system in the future.

Though working, the CBR system also has some weaknesses that were encountered during the training in Chapter 7. During the development phase the unit setup was rather simple and the five attributes used (unit type, hit points left, mana points left, x position and y position) sufficed. Those attributes do not give any information about the past history of what the various units have been doing for the last few seconds. This means that our CBR system does not distinguish between a very active, moving unit and a passive, stationary unit. The system will continue to attack a unit independent on whether or not the attacked unit has been moving around in circles, avoiding most of the attacks. We also observed that our similarity metric was inefficient in many situations. This was because we compared the position x and position y attributes directly instead of looking at the configuration and patterns of units on the battlefield. Using unit positions directly without abstracting it into more complex structures like unit formations, caused bad case reuse.

We also encountered a problem with unnecessary and unintended unit movement. Unit behaviors and actions are defined by the position in the unit list sorted by the attribute “remaining hit points”. When units swap places in this list, they also get new corresponding actions assigned. Repositioning is needed when such units are far from each other. If such units swap a lot, most of the game time will be used for moving units back and forth resulting in huge damage loss. To avoid this problem, an expert needs to foresee this and adjust the strategy such that swaps in the unit list sorted by the attribute “remaining hit points” occur as seldom as possible.

9 Conclusion

The work presented in this paper shows how a CBR system focusing on micromanagement for the RTS game Warcraft 3 can outperform the original game AI in addition to novice and casual human players. Even though we encountered some problems during testing against human players, our approach looks promising.

There are several ways our system can be improved. To prevent unit chasing and inefficient case choosing, our case representation can be extended to include

information about both opponent playing style and unit activity attributes. To avoid classifying units that run small distances back and forth as very active units, it is important to not only consider the total distance traveled but also the effective change in position during a time interval. An increased reusability of cases might be obtained by comparing unit position attributes as patterns and not directly. Better/new case adaptation approach is also needed to reduce unnecessary unit movement. One solution could be to have a dynamic case adaptation that sorts by remaining hit points when the unit count is low, but changes to some more suited case adaptation approach when the number of units increases. Complex actions/behaviors that need longer time to complete are not supported by our approach. Combining macromanagement with micromanagement might be one way to solve this. Another opportunity is to convert our approach into a case-based planning system.

Acknowledgments. The authors wish to thank Helge Langseth, Vidar Holen and Audun Marøy, who through their earlier work [9] inspired the startup of this research. Also thanks to our multinational testing team Aaron Holmes, Asgeir Aakre, Gaute Oftedal, Henrik Skov Jakobsen, Kenneth Wannebo, Marcin Szczepański, Marcus Persson, Rasmus Anker-Nilsen, Sigurd Dahl for their contributions to the evaluation of the system.

References

1. Aamodt, Agnar, and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39-59, 1994
2. Blizzard Entertainment, ed. (2002). Warcraft 3 Instruction Manual. pp. 19-30
3. Blizzard Entertainment. Warcraft 3 Strategy Guide. http://www.battle.net/WAR3/
4. Buro, M. Call for AI Research in RTS Games, AAAI-04 AI in Games Workshop, San Jose 2004
5. Cheng, D.C., & Thawonmas, R. (2004). Case-based plan recognition for real-time strategy games. *Proceedings of the Fifth Game-On International Conference* (pp. 36-40). Reading, UK: University of Wolverhampton Press, 2004
6. David Aha, Mattew Molineaux, and Marc Ponsen. Learning to win: Case-based plan selection in a real time strategy game. In *ICCBR'2005*, number 3620 in LNS, pages 5-20. Springer-Werlag, 2005.
7. Santiago Ontañón, Kinshuk Mishra, Neha Sugandh, Ashwin Ram: Case-Based Planning and Execution for Real-Time Strategy Games. ICCBR 2007: 164-178
8. Schwab, B. (2004). AI game engine programming. Charles River Media.
9. Vidar Holen, Audun Marøy. Reinforcement learning in Wargus. Project Report. NTNU-IDI, December 2007.
10. Marc J.V. Ponsen, Stephen Lee-Urban, Héctor Muñoz-Avila, David W. Aha, Matthew Molineaux: Stratagus: An Open-Source Game Engine for Research in Real-Time Strategy Games. IJCAI 2005 Workshop on Reasoning, Representation, and Learning in Computer Games. Edinburgh, July 2005
11. M. Buro. ORTS project. http://external.nj.nec.com/homepages/mic, March 2002
12. Tomasz Szczepański. Case-based reasoning for improved micromanagement in Real-time strategy games. Project Report NTNU-IDI, December 2008.
13. Pang, Jeff. JASS Manual. http://jass.sourceforge.net/doc/, 2003.

Uncertainty, Knowledge Discovery, and Similarity in Case Based Reasoning

Workshop at the
Eighth International Conference on
Case-Based Reasoning
(ICCBR 2009)

Seattle, Washington, USA
July, 2009

Kerstin Bach, Miltos Petridis, Michael Richter, Rosina Weber and Eyke Hüllermeier (Eds.)

Co-Chairs

Kerstin Bach
University of Hildesheim, Germany
bach@iis.uni-hildesheim.de

Eyke Hüllermeier
University of Marburg, Germany
eyke@informatik

Miltos Petridis
University of Greenwich, UK
m.petridis@gre.ac.uk

Michael Richter
University of Calgary, Canada
mrichter@ucalgary.ca

Rosina Weber
Drexel University, Philadelphia, USA
rosina.weber@drexel.edu

Preface

Knowledge discovery is a key element and challenge in the Case Based Reasoning problem solving process. By its nature, knowledge discovery is usually uncertain and in order to make effective use of discovered knowledge, the types of uncertainty need to be determined and dealt with using appropriate methods and techniques. Discoveries can be naturally imprecise, stochastic, fuzzy and subject to prescribed tolerances. Uncertainty can also affect the useful application of knowledge discoveries in the CBR cycle and can raise issues of confidence, possibly making the ensuing reasoning unconvincing to its end users. Further on, similarity also needs to deal with uncertainty, especially in knowledge areas with complex, approximate, imprecise cases and heterogeneous domains: The domain knowledge underlying the specification of similarity measures or the adaptation of retrieved solutions is usually uncertain and incomplete. Moreover, as problem solving in CBR is primarily of heuristic nature, various aspects of uncertainty also emerge within the case-based processing of knowledge. Indeed, these sources of uncertainty are inherent to CBR and actually concern all phases of the case-based problem-solving process and are relevant in all CBR knowledge containers.

Case-based reasoning must face the challenge to deal with uncertain, incomplete, and vague information, which leads to the need of suitable methods for modeling and reasoning under uncertainty, appropriately complemented by tools for learning and knowledge discovery. Over the past years there has been increased interest in formalizing parts of the CBR methodology within different frameworks of reasoning under uncertainty, and in building hybrid approaches by combining CBR with methods of uncertain and approximate reasoning and soft computing.

The objective of the workshop is to provide an opportunity for exchanging ideas related to the application of various techniques of uncertainty management, knowledge discovery, and similarity in CBR. The workshop aims at providing a forum for the discussion of recent advances in this research field and to offer an opportunity for researchers and practitioners to identify new promising research directions.

This workshop is a follow-up event of the workshops organized at the ECCBR'06 conference in Ölüdeniz/Fethiye, Turkey (namely the "Uncertainty and Knowledge Discovery in CBR" Workshop), the ICCBR'07 conference in Belfast, Ireland (the "Uncertainty and Fuzziness in Case-Based Reasoning" and the "Knowledge Discovery and Similarity" workshop) as well as the ECCBR'08 joint-workshop on "Uncertainty, Similarity, and Knowledge Discovery in CBR" in Trier, Germany.

Just like its forerunner, it aims at providing a forum for the discussion of recent advances in the application of uncertainty and knowledge discovery techniques in case-based reasoning, and to offer an opportunity for researchers and practitioners to identify new promising research directions. Among the submissions that we received, four contributions have eventually been accepted.

The first paper of Ulf Müller, Thomas Barth and Bernhard Seeger discusses the retrieval of 3D shapes using geometrical similarity search based on M-tree-based indexing. They present specific requirements for indexing-methods that can be used for 3D product data retrieval. The second workshop paper by Sutanu Chakraborti goes back to the roots of CBR and picks up on computational models of human memory introduced by Roger Schank. The paper carries Schank's ideas further and transfers them to a dynamical systems approach as a non-conventional model of memory based reasoning as well as it comments on the strength and weaknesses of this approach.

The remaining two papers focus on knowledge discovery and how to cope with uncertainty in CBR applications. Stelios Kapetanakis, Miltos Petridis, J Ma and Liz Bacon present an intelligent workflow monitoring system, which analysis and subsequently reduces the number not-finished workflows of an exam moderation system. They use a graph representation to model temporal relationships within workflows and show that the analysis of past workflow event logs can reduce the uncertainty in similarity based matching and improve the efficiency of the reasoning process. In the fourth workshop paper Rong Hu, Brian Mac Namee and Sarah Jane Delany present an active learning strategy that selects new examples using k-NN based confidence measures. In this paper they deal with the aspect that active learning selection strategies that measure certainty using factors rather than classification scores are more effective.

In closing this preface, we would like to recognize all the colleagues who helped to review the submissions and to guarantee the quality of the papers that have been included.

Kerstin Bach July 2009
Miltos Petridis
Michael Richter
Rosina Weber
Eyke Hüllermeier

Accelerating the Retrieval of 3D Shapes in Geometrical Similarity Search using M-tree-based Indexing

Ulf Müller[1], Thomas Barth[1], and Bernhard Seeger[2]

[1] Information Systems Institute, University of Siegen, Germany
{mueller, barth}@fb5.uni-siegen.de
[2] Dept. of Mathematics and Computer Science, University of Marburg, Germany
seeger@mathematik.uni-marburg.de

Abstract. Increasing the efficiency of knowledge-intensive processes is a major challenge especially in those domains where expert knowledge is the decisive factor for the successful and efficient fulfillment of complex tasks. In the engineering domain, one of the most knowledge-intensive and time-consuming core processes is the cost estimation, design, and construction of a product and its production tools (e.g. an automotive supplier part) answering a customer's specific request. Most of the knowledge necessary to fulfill these processes and which is in turn generated in the course of these processes is represented through the product's three-dimensional geometry. Hence, efficient retrieval of geometrical data is a key issue when supporting these processes with a case-based reasoning (CBR) approach. CBR techniques for supporting these processes enable a process participant to analyze whether there is the opportunity to speed up such a knowledge-intensive and time-consuming process by reusing sufficiently similar, previously generated product-related data and/or documents representing sophisticated expert knowledge.
In this paper we describe the specific requirements for adequate indexing-methods to be used in our system for 3D product data retrieval. An M-tree-based indexing structure is presented to support the k-nearest neighbor search answering a given query by a 3D geometry. As a proof of concept the M-tree indexing method was implemented in the context of our search engine and validated using the Purdue Engineering Shape Benchmark dataset.

Keywords: Geometrical similarity search, shape-based indexing, M-tree, product data retrieval

1 Introduction

The continuously increasing need for the optimization of processes – especially in the presence of knowledge-intensive tasks within these processes – already led to various approaches for managing the necessary knowledge. Among those one can roughly distinguish between organisational approaches and technically oriented approaches

focusing on retrieving data representing the necessary knowledge to solve a given task.

Due to the high degree of knowledge intensity of their processes and tasks, enterprises in the manufacturing industry – e.g. automotive manufacturers as well as suppliers – seek a way to accelerate and improve them. One approach to achieve a substantial improvement is to optimize and increase the knowledge management. In previous papers we have discussed a similarity search engine to support the knowledge management during the Product Lifecycle (PLC) by techniques from Case-Based Reasoning (CBR, [1]). The developed search engine and the related methods for similarity analysis as core parts of the PROXIMA framework [2] are capable of searching various data types relevant in Product Data and Product Lifecycle Management (PDM/PLM) such as numerical, alphanumerical and geometrical data related to products and their manufacturing processes. In the context of the design and construction processes in engineering, one main aspect of assessing similarity between two products is the similarity of their geometry represented by three-dimensional models (3D shapes). As a brief example for this CBR approach to efficient process support, one can think about the gain in efficiency when retrieving geometrical information about similar products by manually searching in large amounts of CAD files spread across several file servers or even searching in collections of printed drawings and communicating to colleagues about certain features of a product compared to the use of a search engine in a structured product data base. Since engineers spend about 30% of their time on searching for various kinds of information, improving this search process is of major importance to companies and their highly-skilled engineers.

A large number of methods have been developed to compute the “distance” (as a measure for similarity) between 3D shapes and to retrieve similar shapes from a given database [3, 4]. All these different methods have in common that the calculation necessary to compute the actual similarity takes a lot of effort since high-dimensional descriptors containing e.g. statistical measures of the color distribution must be derived from the 3D models. A subsequent challenge to the efficient retrieval of geometrical data is the efficient similarity search within large datasets containing complex descriptors [5].

Responding to these challenges, the following paper discusses requirements for indexing methods handling large datasets of complex, high-dimensional descriptors. The M-tree indexing method is identified as one way to answer these requirements.

Accordingly, the remainder of this paper is organized as follows. Section 2 surveys related work and compares our approach with similar approaches. In section 3, specific requirements for indexing large datasets containing 3D shape descriptors are derived. An M-tree based index structure for 3D shape descriptors that is answering those requirements is presented in Section 4. Section 5 contains the experimental validation of this approach using the implementation of the M-tree within the similarity search and the Engineering Shape Benchmark dataset. Section 6 concludes this paper with a brief summary and directions for future work.

2 Related Work

Azuaje et al. classify the retrieval of cases from a given case base into three main groups: Distance-based, indexing-based and hybrid approaches [6]. The simplest approach is the distance-based approach where the distance between two cases can be calculated by measuring the distance between the attributes describing the case. Most 3D retrieval systems follow this kind of strategy for shape retrieval [7-11]. Two different types of retrieval are used in this context: Retrieval by an especially created 3D shape descriptor [9] or retrieval by example. This example can be an existing 3D model [8, 9], a 3D model created from scratch or 2D sketches of a 3D model prepared by the user [7, 8].

A drawback of all of the aforementioned systems is the increasing time for answering a request with an increasing number of entities in the databases. There are only a few approaches which try to avoid this problem of increasing request times by shifting the retrieval strategy from distance-based to index-based [12-18]. Two examples for index-based retrieval on 2D data are the approaches by Mahmoundi and Daoudi [12] and Garcia-Pèrez et al. [13]. In both cases an M-tree is used to store image descriptors. Garcia-Pèrez et al. motivate the usage of the M-tree with their high-dimensional but non-vector structured descriptor. Another approach is presented by Wong et al. [14] where a Self Organizing Map (SOM) is used to reorganize the 3D models in a hierarchical structure. In [15] a similarity search engine for multimedia data is presented which uses a density-based clustering algorithm for structuring the databases. A clustering algorithm for CAD databases was also topic of research done by Chakraborty [16]. The described systems use a shape similarity measure to classify 3D models. In [17] an early attempt of a shape-based search system is shown which is based on the R-tree, a feature-based index structure for high dimensional data. In [18] Berthold et al. describe the X-Tree as an advancement compared to the R-tree.

All the aforementioned approaches to efficient 3D shape retrieval can be classified into three groups based on their indexing strategy: First, clustering-based approaches grouping similar objects. Clustering-based methods utilize distance functions which can be formulated independent from domain-specifics which are beneficial when considering datasets from various domains. But retrieving 3D shapes based on a clustered dataset implies a large computational effort necessary to update the clusters each time the dataset is modified (e.g. computing the $n \times n$ similarity matrix for a dataset with n 3D shapes). In a practical use case with large datasets, this would lead to inacceptable response times.

The second group contains multi-dimensional feature-based trees like the R- or the X-tree. These are only applicable to feature-based datasets, a prerequisite which cannot be guaranteed in the context of shape-based retrieval.

Third, M-tree based approaches can be identified as being capable of handling high-dimensional descriptors and flexible in terms of the descriptors which can be indexed. Because of the fact that different types of shape descriptors perform different on datasets consisting of different, domain-specific 3D shapes [19] it is necessary to evaluate and apply an indexing method which can be applied to different types of

descriptors from the case bases containing cases with shapes from probably different domains.

3 Requirements for Indexing Methods on Datasets containing Complex 3D Shape Descriptors

When thinking about any kind of information retrieval e.g. for documents, images or 3D shapes, the two main requirements are certainly **efficiency** and **accuracy**. To validate the accuracy of a selected descriptor method, various benchmarks have been developed during the last years (s. e.g. [19, 20, 21]). Most recent approaches have focused on analyzing the efficiency of a single distance computation and the accuracy of the retrieval but the efficiency analysis of the whole retrieval process and the development of a specialized index structure were not included.

The majority of methods which are used to numerically represent the characteristics of a 3D model are based on feature vectors [3, 4]. Considering this it is not surprising that most attempts to organize shape descriptors in an index structure are also feature-based. The usage of feature vectors is a standard technique for multimedia retrieval in case it is not possible to compare two objects (e.g. 3D models) directly because of their complexity [5]. Besides the feature-based approaches there exist shape matching methods which are based on graph structures, view-based descriptors or histograms [3, 4, 22]. The variety of domains and applications using 3D models results in various techniques to summarize and compute the domain-specific similarity. Hence, an index structure is required which is suitable to index datasets containing probably **different types of descriptors** from different application domains. This requirement makes feature-based approaches which map the feature values into a multidimensional space almost inapplicable in our context.

In contrast to these approaches, a technique is preferred allowing the distance function to be treated as a black box to create the index structure without regarding domain-specific features. In addition to this requirement, the index structure should be based on the **overall similarity** and not only on similarity considering a subset of all available attributes. The usage of **range queries** and ***k* nearest-neighbour queries** has to be supported and it should be able to handle large amounts of data. This means it should be able to deal with a dynamically changing dataset as it is the case when considering the retrieval of 3D shapes from a product database. Efficient insert, delete and retrieval operations should be available.

One index technique which satisfies these requirements is the previously discussed M-tree.

4 3D Shape Indexing using the M-tree

The most common strategy for 3D shape retrieval is to start a request with a query or an example. As already mentioned in most existing system this approach is distance- and not index-based. An innovative and effective index structure is the M-Tree [23],

which efficiently supports proximity queries on very large data sets by employing a distance function only.

The M-tree can be viewed as generalization of the B^+-tree for metric data, where only a metric distance function *dist* has to be provided by the user. The data objects are kept in pages (of fixed size) that represent the leaves of a balanced tree with logarithmic height. The index nodes contain routing entries, which consists of a routing element and a pointer to a subtree. Each of the routing elements itself is represented as a triple (*C, r, dP*), where *C* is an object and *r* is the radius of a ball with center *O*. The invariant of the M-tree guarantees that the data objects in the associated subtree are in the ball, i. e., $dist(C,O) \leq r$ for all objects *O* that are in the leaves of the subtree. The parameter *dP* represents the distance from *C* to the center stored in the corresponding routing element of the node. This parameter is used for efficient processing of queries, which will be explained in the next paragraph.

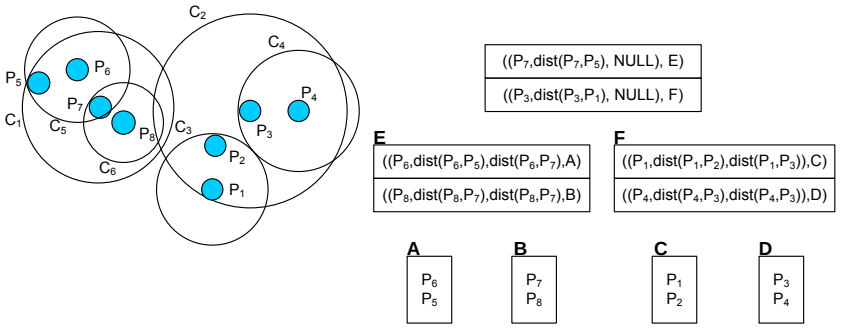


Fig. 1. M-tree Example

An example of an M-tree for a set of 8 points $\{P_1,...,P_8\}$ is illustrated in Fig. 1. On the left hand side, the points and their enclosing circles $C_1,...,C_6$ are plotted, whereas on the right hand side the tree-structure is depicted. For sake of simplicity we set the minimum node capacity to 2, which is generally much higher for M-trees. Note that circle C_1 is represented by its center P_7 and its radius $dist(P_7,P_5)$. The associated subtree is rooted at node E and consists of leaves *A* and *B*. The circle is actually the minimum bounding circle with center P_7 that contains the points in the corresponding leaves.

M-trees support distance queries and nearest neighbor queries on metric data sets, see [24]. Let us illustrate the basic idea for processing distance queries. Given a query object Q and a distance ε, a distance query returns all objects *O* with $dist(Q,O) \leq \varepsilon$. Note that the query region can be viewed as a ball with radius ε and center *Q*. The processing of the query starts by examining the entries in the root of the M-tree. We use the following important property to prune entire subtrees: a data object of a subtree does not qualify for the query, when the corresponding ball does not have a common intersection with the query ball. Therefore, we check the balls of the entries whether they intersect with query ball. Only if this is true, the query is forwarded to

the associated subtree in a recursive fashion. From this description of the distance query, we can derive an important design goal for creating an M-tree: The balls within the routing entries should be as small as possible. This design goal is difficult to achieve, but clever heuristics have been proposed [24].

The major cost factor for processing queries is actually to minimize expensive distance calculations. The parameter *dP*, which is kept in the routing information of an entry, is actually used for pruning the associated subtree without any kind of distance calculations. This can be achieved by making use of the triangle quality.

5 Experimental Results

For our experiments we use the Purdue Engineering Shape Benchmark (ESB) [20], an established dataset consisting of engineering models. The dataset contains about 866 models divided into three main classes and 45 subclasses. To measure the similarity between two shapes an image-based descriptor is used which is based on a combination of image descriptors from the MPEG-7 standard together with a 2D Fourier descriptor.

5.1 The Image-based Shape Descriptor Used in the Experiments

The method used to determine the distance between two 3D models is based on visual similarity (view-based). The main idea is: "If two 3D models are similar, they also look similar from all viewing angles." [22] To measure the similarity between different 3D models, we calculate the similarity between twenty different images taken of the product model. The feature extraction process (s. Fig. 2) starts with a normalization algorithm. This is essential to avoid problems resulting from translation, rotation and scaling invariances.

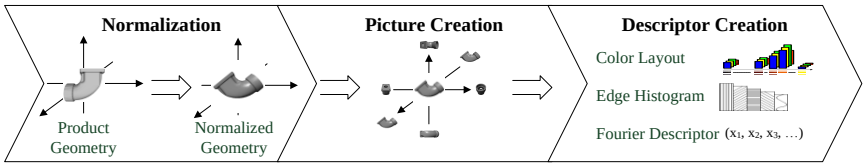


Fig. 2. Descriptor Creation Process

The method to translate and rotate the 3D model is based on the Principal Component Analysis (PCA) [25] but is extended by a scaling procedure. After the normalization phase, the feature extraction starts with the image creation. Chen uses in [21] a dodecahedron to create twenty images for the LFD. In our approach we use the same dodecahedron to create images of the 3D model but we use not only black and white pictures we use grayscale pictures to be able to use texture and color based descriptors. The last two steps of the process consist of feature extraction and storing. The descriptor used to summarize an image and to represent a 3D model is a

combination of three different descriptors: Edge Histogram [26, 27] a texture based image descriptor, Color Layout [27, 28] a color based image descriptor and a Fourier descriptor which is used to describe the contour of an image [29]. Each one of these descriptors has already been tested in previously published papers [26-30]. In our earlier work two of the proposed methods Color Layout and Edge Histogram were well tested on the Princeton Shape Benchmark [30]. The global descriptor consists of the average similarity of the three similarities.

5.2 Experimental Results for the M-tree-based Approach

In order to evaluate the M-tree indexing algorithm we used the ESB dataset. Linear scanning of this dataset (i.e. comparing one query object against all the descriptors in the dataset) needs 866 comparisons with an average duration of *2.18 msec* per element. This time includes the time spent on I/O access[1]. We tested our M-Tree index structure with a maximum leaf size of 100 elements before a node has to be split. The height of the created tree was five and the used shape descriptor was the one described in section 5.1. For each level of the tree, two comparisons are necessary to determine the branch to be chosen. If the 100 most similar elements should be retrieved from the database then two times the height plus 100 comparisons are required. In Fig. 3 the four 3D models selected as queries from the dataset are displayed. Table 1 contains the results of four different search runs (given response time is the average of ten independent runs of the search engine with and without M-tree-based indexing). The achieved reduction of response time is in all of the four tested cases well over 70%. Compared to the previously used linear search in the dataset this reduction was expected but nevertheless validates the M-tree based approach to be very useful also in this context. With an increasing size of the database the reduction is expected to be even higher than observed in this case.

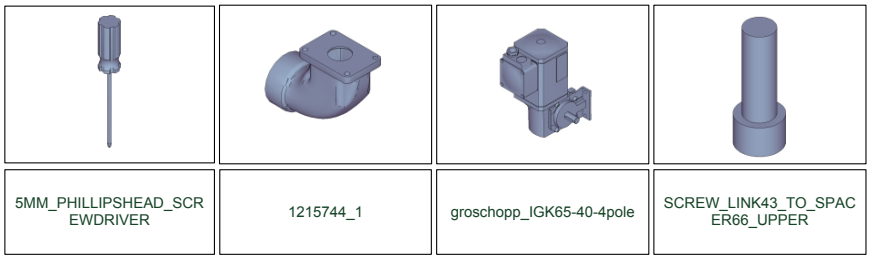


Fig. 3. 3D models used for the evaluation.

[1] The tests were done on an Intel Core 2 machine at 2.17GHz, 2GB of Ram and Java 6.0 Virtual Machine

Table 1. Response time for 3D shape retrieval with and without the M-tree index.

Query	Time without M-tree [ms]	Time with M-tree [ms]	Time reduction [%]
groschopp_IGK65-40-4pole	1929,70	424,10	78,02
1215744_1	1921,70	441,60	77,02
5MM_PHILLIPSHEAD_SCREWDRIVER	1799,20	451,90	74,88
SCREW_LINK43_TO_SPACER66_UPPER	1894,60	511,00	73,03

6 Conclusions and Future Work

In this paper we presented certain requirements and an adequate indexing method to improve the performance of a similarity search engine for 3D models. This similarity search for 3D models is one essential part when supporting knowledge-intensive processes in engineering. Since the similarity analysis of 3D shapes is based on high-dimensional descriptors, the indexing structure is in the focus of research. Due to the fact that conventional index structures like the R-tree or the R^*-tree are not adequate for retrieval dealing with high-dimensional 3D shape descriptors, special index structures are required. An M-tree based index structure for 3D shape descriptors is identified as one promising approach to answer the requirements for indexing structures of 3D shape descriptors. It was validated that the response time can be substantially reduced using the M-tree indexing structure for indexing the Engineering Shape Benchmark dataset.

For the future it is planned to further improve the M-tree based index structure itself and to combine the presented method with similar techniques for numerical and alphanumerical product data retrieval. One reason against utilizing a cluster-based strategy – as already briefly discussed in this paper – is the high effort necessary to perform a single cluster analysis as a prerequisite for each search process. To reduce this effort it is planned to evaluate different techniques for supervised and unsupervised classification like e.g. neural networks. For improving the system's usability, in a next step it will be possible to use a clustering algorithm for result set categorization as it is shown for visualizing search results of e.g. internet searches in [31]. Since the 3D shapes are coming from different domains (e.g. metal forming, casting) different similarity measures have to be used. Hence, one future challenge is the automatic adjustment of generic similarity measures enabling an improved adaptation of our search engine to datasets from different domains or even multi-domain datasets. The search in graph-structured data like product structures, bills of material, or working plans is also in focus of future research.

References

1. Aamodt, A., Plaza, E.: Case Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. J. AI Communications 7, pp. 39-59. (1994)
2. Lütke Entrup, C., Barth, T., Schäfer, W.: Towards a Process Model for Identifying Knowledge-Related Structures in Product Data. In: Reimer, U., Karagiannis, D. (eds.) PAKM 2006. LNCS, vol. 4333, pp. 189-200. Springer, Berlin (2006)
3. Iyer, N., Jayanti, S., Lou, K., Kalyanaraman, Y., Ramani, K.: Three Dimensional Shape Searching: State-of-the-art Review and Future Trends. J. Computer-Aided Design 37, pp. 509-530. (2005)
4. Tangelder, J. W., Veltkamp, R., C.: A survey of content based 3D shape retrieval methods. J. Multimedia Tools and Applications 39, pp. 441-471. (2008)
5. Bustos, B., Keim, D. A., Saupe, D., Schreck, T., Vranic, D. V.: Feature-Based Similarity Search in 3D Object Databases. J. ACM Computing Surveys 37, pp. 345-387. (2005)
6. Azuaje, F.; Dubitzky, W., Black, N., Adamson, K.: Retrieval strategies for case-based reasoning: a categorized bibliography. J. The Knowledge Engineering Review 15, pp. 371-379 (2000)
7. 3D model retrieval system, http://3d.csie.ntu.edu.tw/~dynamic/cgi-bin/DatabaseII_v1.8/
8. 3D model search engine, http://shape.cs.princeton.edu
9. 3D model similarity search engine, http://merkur01.inf.uni-konstanz.de/CCCC
10. 3D search tool, http://3d-search.iti.gr/3DSearch
11. Victory Project, http://www.victory-eu.org:8080/victory/
12. Mahmoudi, S., Daoudi, M.: Retrieval by shape using CSS and M-tree. In: 3rd International Workshop on Content-Based Multimedia Indexing, pp. 297-302. (2003)
13. Garcia-Pèrez, D., Mosquera, A., Berretti, ST., Del Bimbo, A.: Evaluation of a M-Tree in a Content-Based Image Retrieval System. In: Workshop on Efficiency Issues in Information Retrieval (2008)
14. Wong, H.-S., Cheung, K.K.T., Sha, Y., Ho-Shing Ip, H.: Indexing and retrieval of 3D models by unsupervised clustering with hierarchical SOM. In: 17th International Conference on Pattern Recognition, pp. 613-616. (2004)
15. Brecheisen, S., Kriegel, H.-P., Kröger, P., Pfeifle, M., Schubert, M., Zimek, A.: Density-Based Data Analysis and Similarity Search. In: Petrushin V. A., Khan L. (eds.) Multimedia Data Mining and Knowledge Discovery, pp. 94-115. Springer, London (2007)
16. Chakraborty, T.: Shape-Based Clustering Of Enterprise CAD Databases. J. Computer-Aided Design and Applications 2, pp. 145-154 (2005)
17. Iyer, N., Kalyanaraman, Y., Lou, K., Jayanti, S., and Ramani, K.: Early results with a 3D Engineering Shape Search System. In: International Symposium on Product Lifecycle Management (2003)

18. Berchtold S., Keim D. A., Kriegel H.-P.: The X-Tree: An Index Structure for High-Dimensional Data. In: 22nd International Conference on Very Large Data Bases, pp. 28-39. (1996)
19. Jayanti, S., Kalyanaraman, K., Iyer, N., Ramani, K.: Developing an Engineering Shape Benchmark for CAD Models. J. Computer-Aided Design 38, 939-953. (2006)
20. Shilane, P., Min, P., Kazhdan, M., Funkhouser, M.: The Princeton Shape Benchmark. In: Shape Modeling International, pp. 167-178. (2004)
21. SHREC - Shape Retrieval, http://www.aimatshape.net/event/SHREC
22. Chen, D.-Y.: Three-Dimensional Model Shape Description and Retrieval Based on LightField Descriptors. (Ph.D. Thesis), NTU CSIE (2003)
23. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces, In: 23rd International Conference on Very Large Data Bases, pp. 426-435. (1997)
24. Hjaltason, G. R., Samet, H.: Index-driven similarity search in metric spaces. J. ACM Transactions on Database Systems 28, pp. 517-580. (2003)
25. Vranić, D., V.: 3D Model Retrieval. (Ph.D. Thesis), University of Leipzig (2004)
26. Won, C. S., Park, D. K., Park S.-J.: Efficient Use of MPEG-7 Edge Histogram Descriptor, J. ETRI Journal 24, pp.23-30 (2002)
27. Manjunath, B.S.; Salembier, P.; Sikora, T.: Introduction to MPEG-7: Multimedia Content Description Interface. John Wiley (2002)
28. Manjunath, B. S., Ohm, J.-R., Vasudevan, V., Yamada, A.: Color and texture descriptors. J. IEEE Transactions on Circuits and Systems for Video Technology 11, pp.703-715. (2001)
29. Zhang, D., Lu. G.: A Comparative Study of Fourier Descriptors for Shape Representation and Retrieval. In: 5th Asian Conference on Computer Vision, pp. pp. 646-651. (2002)
30. Müller, U., Lütke Entrup, C., Barth, T., Grauer, M.: Applying Image-based Retrieval for Knowledge Reuse in Supporting Product-Process Design in Industry. In: 8th International Conference on Application of Fuzzy Systems and Soft Computing, pp. 396-404. (2008)
31. Kules, W., Wilson, M. L., Schraefel, M., Shneiderman, B.: From Keyword Search to Exploration: How Result Visualization Aids Discovery on the Web. Technical Report, School of Electronics and Computer Science, University of Southampton (2008)

Memory Based Reasoning: A Dynamical Systems Perspective

Sutanu Chakraborti

Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Chennai – 36
Email: sutanuc@iitm.ac.in

Abstract While CBR has been successful in several domains, it has failed to measure up to the cognitive and psychological models of human reminding, that inspired it in the first place. This paper explores the potential of Dynamical Systems theory in arriving at more cognitively sound realizations of Memory Based Reasoning.

1 Introduction

Case-Based Reasoning (CBR) was inspired by seminal work on computational models of human memory by Roger Schank in the late seventies and early eighties [8]. However, with the passage of time, many of his observations which are interesting from a cognitive standpoint got watered down, making room for implementations that are strongly dictated by pragmatics of real-world application needs. In this short paper, I revisit some of the cognitive premises of Memory-Based Reasoning (MBR) and examine in that light why traditional case representations and similarity models are inadequate in simulating dynamics of human memories. I then present a model of memory based on Nonlinear Dynamics, and present parallels between parameter-learning in this approach and evolution of knowledge containers [9] in CBR. Strengths and weaknesses of the proposed approach are then identified, along with scope for future work. A note regarding the style of presentation: the intention of this paper is to trigger discussion in the CBR community, rather than present finished work. So mathematical rigour has been eschewed when an example or illustration suffices to convey the central idea.

The debate around whether it is at all desirable to try and build models that emulate psychological models of our memories demands a mention at the outset. While aeroplanes do not flap their wings as birds do, Jurafsky and Martin [10] rightly observe that aeroplanes do have wings like birds, in the first place. Building computational analogs of human memories can lead us to better models (our parallel to mechanics of flight) ,which on their own, or in conjunction with existing MBR formalisms, can potentially lead to better working implementations.

The paper is organized as follows. In Section 2, we motivate the need for a fresh approach to MBR by identifying aspects of human memory that are not realized by most CBR implementations. Section 3 presents the key idea behind using Dynamical Systems Approach for MBR, and shows how the CBR notions of cases,

similarities and nearest neighbour retrieval can be modelled using this new formalism. Section 4 discusses the strengths and weaknesses of this new model, especially with reference to the observations in Section 2. Section 5 concludes the paper.

2 Human Memory vs. CBR

Most CBR systems use a feature value representation for storing cases. Given a new problem situation, *similar* cases are retrieved. Similarity between cases is evaluated by computing feature-specific (local) similarities which are weighted and combined to yield global similarities. Some CBR implementations also involve an adaptation step to repair the solution suggested based on differences between the current problem and the retrieved case(s). It needs to be emphasized that similarity is used as a surrogate for the *utility* of the retrieved case in solving the new problem, since utility is harder to estimate [4].

Run-of-the-mill CBR systems differ significantly from psychological models of human memory in at least the following aspects:

1. Human memory proactively generalizes specific instances, in a way that makes it more effective for anticipated problem solving tasks. While individual visits to restaurants may be forgotten, these episodes are generalized to help us in creating expectations. Any episode that significantly defies these expectations will also be indexed. Thus episodes contribute to generalizing; generalizing leads to expecting; expectations failures need to be explained and can potentially lead to revision of our generalizations, which in turn changes our expectations. This is the central concept of learning in Schank's thesis [8]. Generalization also has a favourable influence on storage requirements, sometimes referred to as cognitive economy. Generalization of cases is often ignored in CBR, and is not regarded as a key step in the Retrieve, Ruse, Revise, Retain cycle of CBR [4].
2. Human memory is known to be strongly associative [5]. In contrast cases in most CBR systems exist irrespective of each other. Thus, case-base maintenance tasks like addition or deletion of cases [11] can be carried out without affecting the other cases of the case-base[1]. A cognitively sound case addition should potentially (though not necessarily) generalize or alter existing cases, and consequently lead to reorganization of the casebase.
3. Unlike most CBR systems, we are particularly bad at enumerating items (cases) in our memory, rather descriptions of episodes are constructed accessing the entire episode [5]

[1] Addition or deletion of cases obviously has an impact on cases retrieved in response to a query, and thus affect the CBR performance implicitly. However, no reorganization of the casebase takes place and the existing cases remain unchanged.

4. The same structures are used by our memories for both processing (throw up expectations and retrieve) and for storage. This is true for many neural network architectures, which embed learning, recall and storage in the same architecture. In contrast, most CBR systems currently maintain indices (with associated access mechanisms) separately from the cases. Lenz [5] notes that while discrimination networks come closer to human memories in this respect, they are restricted by imposing an order in which descriptions have to be entered.
5. Not all human reminding can be explicitized. Schank identifies two categories of knowledge that we are rationally unaware of possessing -- nonconscious and physical knowledge. These knowledge forms are involved when one is reminded of a musical note or when one chooses strokes in playing tennis respectively. In contrast, the ability to explain retrieval is regarded a strength of CBR vis a vis black box approaches like neural networks, though this arguably narrows down the ability of CBR to model human reminding across diverse tasks.

3 The Dynamical Systems Idea

A dynamical system is characterized by a set of variables that evolve over time. Dynamical systems are described using differential equations for continuous time evolution, or using difference equations for discrete time evolution. The equation below describe a dynamical systems, that could be linear or non-linear based on the nature of the function *v(x):*

$$\frac{dx}{dt} = v(x) \tag{1}$$

The complexity of a dynamical system is determined mainly by two factors: (a) the number of variables and (b) the degree of nonlinearity. An excellent introduction to the field is the textbook by Strogatz [7]; below I summarize some key ideas and show how they are relevant to building computational models of memory.

It is useful to picture a dynamical system geometrically; this helps us gain qualitative insights into the system behaviour without solving the equation(s) analytically. For example, the dynamical system

$$\frac{dx}{dt} = \sin(x) \tag{2}$$

can be represented as a *phase portrait* shown in Figure 1.

Imagine a fluid that flows along x axis with the velocity $\frac{dx}{dt}$ given by Equation 1. When $\frac{dx}{dt}$ is positive, *x* must be increasing and hence the flow is to the right. When $\frac{dx}{dt}$ is negative, the flow is to the left. When $\frac{dx}{dt}$ = 0, we have *fixed points* where there is no flow. Points A and C are called *stable* fixed points or

attractors since all flows around them are directed towards them; point B is called an *unstable* fixed points or repeller since particles move away from it. We also get qualitative pictures as follows: A particle originating at E tends to speed up as it moves right till it reaches D when it starts slowing down, eventually coming to rest at A.

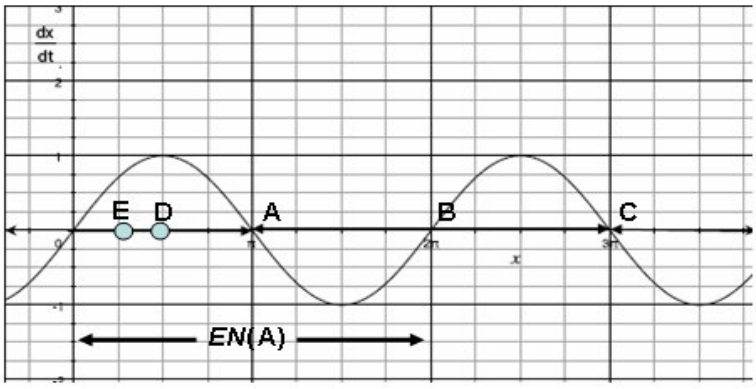


Fig.1. Cases and Effective Neighbourhood in a Phase Portrait (a modified version of phase portrait in [1]

The notion of attractors is important for our discussion since cases in CBR can be treated as attractors in an *n*-dimensional space, over which a vector field is defined as a differential (or difference) equation. Any case *c* defines a region around it, such that all flows originating from any point in that region will be directed towards that case. We call this the *effective neighbourhood* of that case and denote it by *EN*(c). In effect, any query that falls in *EN*(c) will lead to retrieval of case *c* after certain time steps; the convergence time is governed by the nature of the vector field and the distance of the query from *c*. Figure 1 illustrates *EN*(A). A steeper gradient around an attractor leads to a faster convergence and hence models higher similarities of neigbouring points to the attractor. Figure 2 below shows neighbourhood regions of cases as defined by a 1-NN retrieval scheme in the original decision theoretic space; these regions correspond closely to our idea of effective neighbourhood.

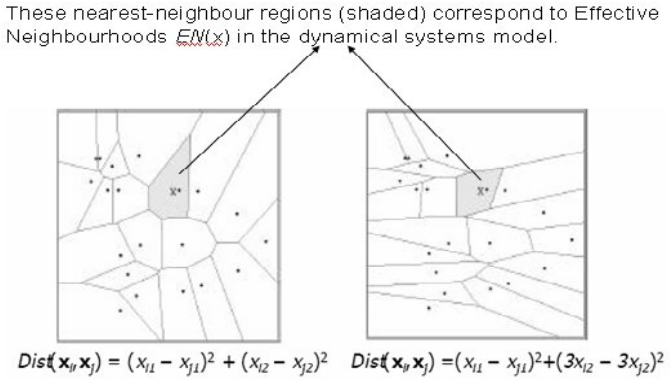


Fig. 2. Nearest neighbour regions of cases according to two distance measures (adapted from [2])

Learning in the dynamical systems model involves changing parameters defining the differential equation. *Bifurcations* refer to the qualitative changes in dynamics based on parameters. Of particular interest is the *saddle node bifurcation* which can be used to model case addition and deletion in terms of creation or destruction of stable fixed points. This is illustrated in Figure 3, where different values of the parameter r can lead to very different vector fields. Also, one or more attractors can merge to form a new attractor; this process can model generalization as explained in Section 2.

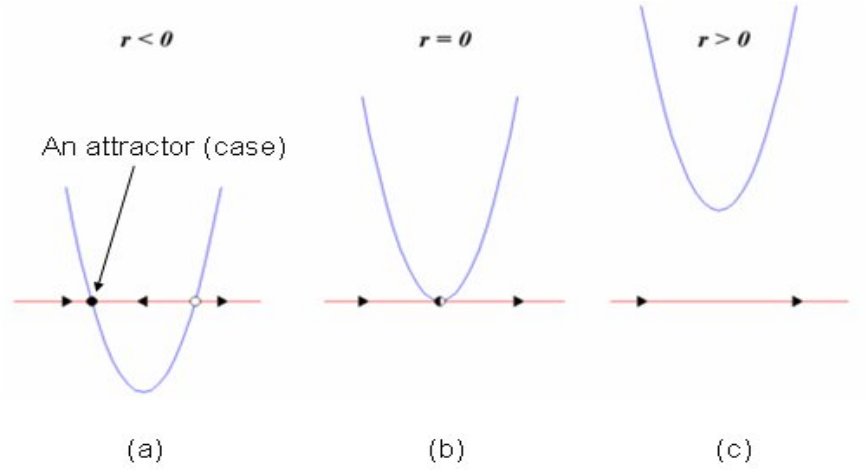


Fig. 3. Saddle node bifurcation (a modified version of figure in [3]) can be used to model case deletion (as we move from (a) to (c)) or case addition (from (c) to (a)). The system involved is $\frac{dx}{dt} = r + x^2$

Choice of parameters also implicitly capture the knowledge of case similarities. Parameter changes lead to updating similarities; it can lead to a change in the effective neighbourhood of the case, or in the rate of convergence of flows originating at queries to neighbouring cases.

Given a differential system, the forward problem is to find a set of cases that correspond to the attractors of the system. From a practical standpoint, the reverse problem is more interesting: given a set of cases, determine the equations defining a dynamical system whose attractors correspond to the given cases. Of several possible systems that satisfy this primary criterion, we need to identify those that best satisfy the domain needs as characterized by the knowledge of similarities between cases. More generally, any kind of domain knowledge has the effect of constraining the set of dynamical systems which qualify as potential candidates. We start with a dynamical system defined by a set of parameters, which are updated based on the feedback received from the environment that the system interacts with. As a model of memory, the system allows creation and deletion of cases, relocation of cases to

nearby locations, generalization of existing cases resulting in compaction, and evolution of similarity knowledge as described above, all aimed at improving the effectiveness of the system in solving unseen problems based on reminding.

4 Discussion

The idea of using attractors to model memories can be traced back to a class of neural networks called attractor networks, of which the Hopfield Network [6] is an instance. To my knowledge, parallels of these approaches with CBR have not been explored so far. Unlike the approaches presented in this paper, memories in attractor networks are static and pre-specified, and hence these approaches do not lend to evolution of similarity functions or case transformations. However such works lay foundations for analytic techniques involved in designing dynamical systems given a set of attractors, which fall outside the scope of the current paper.

Till now we have discussed retrieval of single cases in response to a query. An issue that merits some discussion is that of retrieval of multiple cases relevant to a query. This is possible by generating several pseudo-queries such that each of these pseudo-queries is a slight perturbation in the query vector. The set of cases to which the pseudo-queries independently converge refines the retrieval set. A similar approach can be used to model similarities between cases, which are otherwise not reachable from one another since each defines a stable fixed point. The cost of a perturbation, along with the time steps for convergence to the case as dictated by the gradient (refer Section 3) defines the similarity between the cases.

The dynamical systems approach can be used to model reminding across diverse topics in the course of conversations. For this, we allow each attractor to define an entry point for a new dynamical system with its own set of attractors. Thus, we have a set of dynamical systems each having two sets of attractors: the first that define memory within context, and the second that allow, with non-zero probability, a drift into another context. Thus we arrive at a networked set of dynamical systems participating in cross-contextual reminding, with the latter category of attractors facilitating jumps from one context (alternately one system) to another. Figure 4 shows an example with 4 contexts, each having its own set of cases as stable fixed points shown as solid circles. The arrows in this figure show how a chain of remindings can be modelled across several contexts. For example, cases A, C and E in Context 1 allow, with non-zero probabilities, a drift into contexts 2, 3 and 4 respectively.

A majority of similarity models used in CBR are based on decision theoretic or syntactic formalisms. The first involve positioning objects in a (weighted) feature space, and using the distance between objects to model similarity. The second involve computing the cost of transforming one object to the other using a set of fixed-cost primitive operators; an example is the use of edit distances. A strength of these approaches is their ability to explain retrieval. In contrast, the dynamical systems approach aims at a cognitively sound retrieval scheme, while apparently losing out on transparency. However, cognitive studies have revealed that generating explanations is often a posterior process, and can have very little to do with the central process involved in reminding. Humans are reminded of related episodes or descriptions without being aware of the processes and knowledge that took part in reminding.

When asked for an explanation, the reminding is attributed to one of several *possible* correspondences between the problem situation and the retrieved case. It appears possible to generate similar explanations for remindings in the dynamical memory model, based on query perturbations and qualitative rendering of the underlying mechanics, e.g. steep slope in the phase portrait leading to a case can map to higher similarity of that case with cases in its effective neighbourhood.

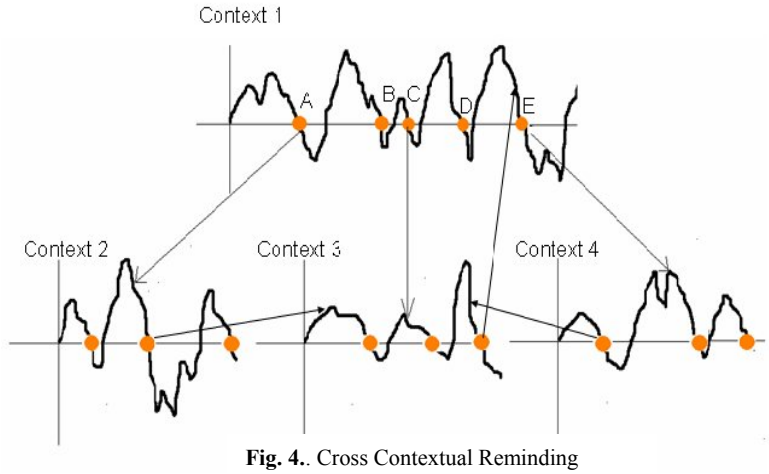


Fig. 4.. Cross Contextual Reminding

The proposed approach overcomes several limitations of current CBR models, as identified in Section 2. Firstly, processing and storage use the same mechanisms. The knowledge of cases and similarities is implicitly embodied in the phase portrait. The sequence of steps involved in retrieval is governed by the vector field (phase portrait). Generalization of cases and incremental learning of similarity knowledge can be realized using parameter learning. Generalization happens when stable fixed points corresponding to two or more cases get merged into one stable fixed point. The nuances corresponding to each specific case is lost. As we observed before, similarity knowledge is captured in the parameters of the equations defining the vector field. Case additions and deletions involve changing the parameters as well, leading to changes in the vector field, and thus have local or global repercussions. The system is good at reconstructive reminding; a partial description which acts as a trigger converges onto the attractor defined by the relevant memory. This is sometimes referred to as case completion [5]. As with humans, not all reminding is transparent, though partial explanations can be generated a posteriori. As discussed earlier in this section, the possibility of modeling cross-contextual reminding, a distinguishing feature of human memory [8], is another interesting fallout. However, realizing a non trivial memory model over sizeable number of cases using the dynamical systems approach would require us to develop analytic tools that scale up to real world needs. The multiple-systems approach described above for facilitating cross-contextual reminding point to the possibility of partitioning the problem into smaller sub-problems at design time.

5. Conclusion

The paper positions the dynamical systems approach as a non-conventional model of memory based reasoning. This line of research can potentially lead to new computational models of human reminding, with a more well-founded psychological and cognitive basis.

References

1. http://www.learner.org/courses/mathilluminated/images/units/13/dxdt.png
2. Moore A., Tutorial Slides on Instance Based Learning, at http://www.autonlab.org/tutorials/mbl.html
3. http://upload.wikimedia.org/wikipedia/commons/thumb/6/61/Saddle-node.png/500px-Saddle-node.png
4. Bergmann R., Experience Management: Foundations, Development Methodology, and Internet-Based Applications Springer 2002
5. Lenz M., Case Retrieval Nets as a Model for Building Flexible Information. Systems. PhD Thesis, Humboldt Universitat zu Berlin, 1999.
6. Haykin, S., Neural Networks: A comprehensive foundation, 2nd Edition, Prentice Hall, 1998.
7. Strogatz, S.H., Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering, Perseus Books, Cambridge, 1994
8. Schank, R.C, Dynamic Memory Revisited, 2nd edition New York, Cambridge University Press, 1999
9. Richter M., Introduction. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (Ed.), Case-Based Technology. SNLAI 1400
10. Jurafsky, D., and Martin J.H.. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. Prentice-Hall. 2000
11. Massie, S., Craw, S., and Wiratanga, N.. Complexity profiling for informed case-base editing. In Proc. of ECCBR-06. Springer, 2006.

Workflow Monitoring and Diagnosis Using Case Based Reasoning on Incomplete Temporal Log Data

Stelios Kapetanakis, Miltos Petridis, Jixin Ma, Liz Bacon

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

Abstract. This paper presents an approach for intelligent diagnosis and monitoring of workflows based on incomplete operation data in the form of temporal log data. The representation of workflows in this research using graphs is explained. The workflow process is orchestrated by a software system using BPEL technologies in a service oriented architecture. Episodic cases are represented in terms of events and their corresponding temporal relationships. The matching and CBR retrieval mechanisms used in this research are explained and the architecture of an integrated intelligent monitoring system is shown. The paper contains a simple evaluation of the approach based on a university quality assurance exam moderation system. Finally, further work on the system and the extension to an intelligent monitoring and process optimisation system is presented.

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

1 Introduction

Modern business processes are increasingly being monitored and managed using computer systems. In order for this to happen effectively, business processes need to be more formally defined and structured events relating to their operation are should also be captured and reported to the various business process stakeholders and managers.

Business processes are typically defined and represented in terms of a series of workflows and temporal relationships and constraints between them. Business processes can be defined using UML diagrams such as activity diagrams and represented formally using newly emerged business process representation standards. The Business Process Modelling Notation (BPMN) developed by the Business Process Management Initiative (BPMI) and Object Management Group (OMG) provides a standard for the graphical representation of workflow based business processes[1]. Workflow based business process representation is possible with standards covering the definition, orchestration and choreography of business processes.

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

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

Modern enterprise systems are able to separate the definition of workflow based business processes from the software implementing the operation of these workflows, offering much more flexibility and agility than was possible in older systems. This allows enterprise computer systems to monitor and control business processes and workflows within an organisation. Additionally, this allows for the agile change of workflows to adapt to the changing business needs of an organisation.

Case Based Reasoning (CBR) has been proposed as a natural approach to the recall, reuse and adaptation of workflows and knowledge associated with their structure. Minor et al [4] proposed a CBR approach to the reuse and adaptation of agile workflows based on a graph representation of workflows and structural similarity measures. The definition of similarity measures for structured representations of cases in CBR has been proposed [5] and applied to many real life applications requiring reuse of domain knowledge associated with rich structure based cases [6],[7].

A key issue associated with the monitoring and control of workflows is that these are very often adapted and overridden to deal with unanticipated problems and changes in the operating environment. This is particularly the case in the aspects of workflows that directly interact with human roles. Most business process management systems have override options allowing managers to bypass or adapt workflows to deal with operational problems and priorities. Additionally, workflows are liable to change as the business requirements change and in many case workflows involving processes from different parts of an organisation, or between collaborating organisations can “tangle”, requiring the need for synchronisation and mutual adaptation to allow for compatible synergy.

The flexibility and adaptability of workflows provides challenges in the effective monitoring of a business process. Typically, workflow management systems provide outputs in terms of event logs of actions occurring during the execution of a workflow. These could refer to an action (such as a sign-off action or uploading a document), or a communication (such as a transaction initiation or email being initiated and sent). The challenge in monitoring workflows using event information is that even where the workflow structure is well defined and understood, the trace of events/actions does not usually contain the context behind any decisions that caused these events/actions to occur. Additionally, there are often a lot of contextual information and communications that are not captured by the system. For example, some actions can be performed manually and informal communications/meetings between workflow workers may not be captured by the system. Knowledge of the workflow structure and orchestration of workflows does not necessarily define uniquely the choreography and operation of the workflows.

The effective monitoring of workflows is therefore required to deal with uncertainty stemming from these issues.

The approach proposed in this paper is based on a CBR process requiring similarity measures informed from knowledge discovery of norms and problems from past operation. The CBR approach proposed uses a graph based representation of cases based on events, actions,intervals and their temporal relationships.

Section 2 discusses the exam moderation business process application domain that is used to evaluate the approach. Section 3 presents the proposed workflow and event log case representation and similarity measures used. Section 4 presents the architecture of the workflow intelligent monitoring system CBR-WIMS that has been developed to evaluate this work. Section 5 presents an evaluation based on two workflow monitoring experiments.

2 The Exam Moderation Business Process Workflows

In order to evaluate the approach proposed in this research, it was decided to use the University of Greenwich, School of Computing and Mathematical Science exam moderation system. This is an automated web enabled secure system that allows course (module) coordinators, course moderators, exam drafters (typically senior managers), admin staff and external examiners to upload, modify, approve and lock student exam papers. The system automates the whole process and provides an audit trail of events generated by workflow stakeholders and the system. The system orchestrates a formal process made up of workflows. The process can be defined and displayed formally in terms of a UML activity diagram (Fig. 1). The system tracks most workflow actions in terms of timed events. Mostof these generate targeted email communications to workflow stakeholders, some for information and others requiring specific further actions from these stakeholders.

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

2.1 Uncertainty in Workflows

The overall exam moderation workflow process is formally defined and constrained by the system operation. There are also some limited facilities for manual override by the system administrator. However, the overall process in conjunction with the actions and communications audit trail do not uniquely explain the exact cause of individual actions and cannot predict reliably what the next event/action will be and when this is likely to occur. Most of the uncertainty stems from the problem that a significant part of the workflow occurs in isolation from the system. The system does not capture all of the contextual knowledge associated with workflows. A lot of the communications between workflow stakeholders can occur outside the system e.g. direct emails, physical meetings and phone calls adding to the uncertainty associated with past or anticipated events and the clear definition of the current state.

Discussions with workflow monitoring managers showed that patterns of events indicated, but not defined uniquely the current context and state of a workflow. Managers

were able to guess from looking at the workflow events and communications audit what the context and current state of a workflow was and point to possible problems. Most problems occur due to human misunderstanding of the current state and confusion with roles and responsibilities and usually result to the stalling of a workflow. Managers will then try to restart the process by adding comments to the system, or initiate new actions and communications. However, this depends on managers realizing that such a problem has occurred.

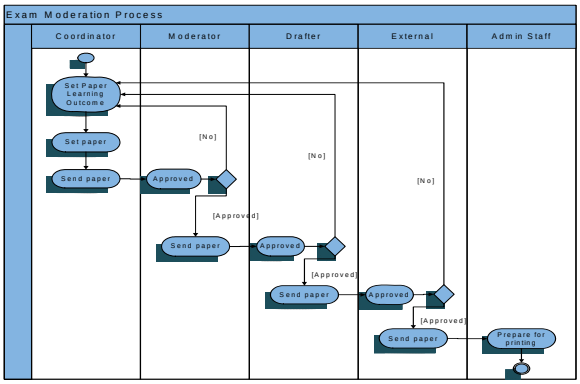


Fig. 1. The exam moderation process activities and workflows (simplified)

A typical problem series of event could be one where a stakeholder has missed reading an email requiring an action. In that case, the workflow would stall until a manager or another stakeholder spots the problem and produces a manual action (such as sending an email) to get the workflow moving again. For example, a course coordinator upload notification may have been missed by a moderator who would then not read the new version and either approve or try to amend by a new upload as s/he needs to do. In that case, the coordinator may take no further action and other stakeholders will not act expecting an action from the moderator to occur.

A key problem with uncertainty about the current status of a workflow is that due to the expected normal delay between workflow events/actions, it may not be clear at a given point in time whether the workflow has stalled or the moderator is just slow at responding to the original action of the coordinator upload. This can only be resolved in a stochastic way based on retrieved knowledge from a similar series of events in past workflows for that moderator in addition to norms?.

Discussions with system managers indicated that some of the uncertainty associated with expected response delays can be reduced by using past experience about response profiles and norms for individual stakeholders. Data mining or statistical analysis of the information obtained from past workflows for individual system users, in a particular workflow role, can provide the most likely response and likely response time for the user in a new workflow context. This can then be used to provide a more reliable similarity measure for the effective comparison between a new, unknown workflow state and past cases as part of a case-based reasoning retrieval process.

2.2 The CBR Workflow Monitoring System

The aim of the CBR Workflow Intelligent Monitoring System (CBR-WIMS) is to provide an automatic monitoring system that will notify managers and stakeholders of potential problems with the workflow and provide advice on actions that can remedy a perceived problem.

The monitoring system is designed to work based on experience of past event/action temporal sequences and the associated contextual knowledge and classification in a Case-Based Reasoning system. Similarity measures allow the retrieval of close matches and their associated workflow knowledge. This allows the classification of a sequence as a particular type of problem that needs to be reported to the monitoring system. Additionally, it is intended that any associated knowledge or plan of action can be retrieved, adapted and reused in terms of a recommendation for remedial action on the workflow.

The CBR monitoring system uses similarity measures based on a linear graph representation of temporal events in a workflow normalized by experience from past behaviour on individual user workflow participation patterns.

3 Workflow and Event Log Representation and Similarity Measures

In CBR-WIMS workflows are defined using UML activity diagrams and mapped through Business Process Management Notation (BPMN)[1] into Web-Services Business Process Execution Language (WS-BPEL) [2] and stored within the system. The storage of workflows is temporal as a number of workflow versions can be stored to allow for modifications of the workflow following business process changes and their application to different contexts of use for a particular process. For example, variants of the exam process workflows can be defined to allow for specific types of exams, such as ones that require external validation or collaboration for courses delivered in partnership with other institutions. Similarity measures between workflow representations can be defined on a graph representation of workflow processes using an exhaustive graph similarity search algorithm based on the Maximum Common Subgraph [7]. This allows the reuse of knowledge about workflows between different workflow processes and variants however, it is beyond the scope of the work presented in this paper.

The workflows stored in WS-BPEL are used by CBR-WIMS to automatically orchestrate the execution of workflows in the system.

The representation of events in the workflow event log uses a general time theory, based on intervals [8]. In the theory used here, the temporal relationships have been reduced from the ones proposed by Allen [9] to just one, the “meets” relationship.

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

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

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

This approach has been shown to be suitable for defining temporal similarity measures in the context of a CBR system based on the graph representation of events and intervals and their temporal relationships and similarity measures based on graph matching techniques such as the Maximum Common Subgraph (MCSG)[11][7]. Additionally, such a graph can be checked for consistency of temporal references using linear programming techniques [11].

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

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

$M = \{Meets(t_1, t_2), Meets(t_1, t_3), Meets(t_2, t_5),$
$Meets(t_2, t_6), Meets(t_3, t_4), Meets(t_4, t_7),$
$Meets(t_5, t_8), Meets(t_6, t_7), Meets(t_7, t_8)$;

$D = \{Dur(t_2) = 1, Dur(t_4) = 0.5, Dur(t_6) = 0, Dur(t_8) = 0.3\}$

The graphical representation of temporal reference (T, M, D) is shown in Fig. 2:

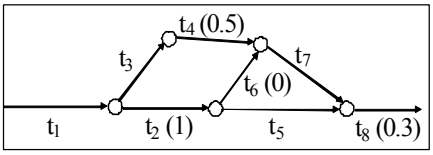


Fig. 2. Graph representation of temporal relationships

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

$$S(G,G') = \frac{(\sum_{\substack{matches\\ C,C'\\ in\\ MCG}} \sigma(C,C'))^2}{count(G).count(G')} \qquad (1)$$

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

In the case of time stamped events produced by the workflow event log, the duration of each interval can be calculated, so the graphs are collapsed into a single timeline. In this case, the similarity measure is easier to calculate as the MCS is a common segment made up of events and intervals in a given order in each of the compared workflow logs. In this common graph segment each edge (event or interval) has a similarity measure to its counterpart in the other log that exceeds a given threshold value ε. Eq. 1 above can still be used to provide the overall similarity between the two workflows.

4 The Architecture of the Workflow Intelligent Monitoring System

CBR-WIMS is an Intelligent Workflow Monitoring System incorporating a CBR component. The role of the system is to assist the transparent management of workflows in a business process and to orchestrate, choreograph, operate, monitor and adapt the

workflows to meet changing business processes and unanticipated operational problems and inconsistencies. Fig. 3 below shows the overall architecture and components of CBR-WIMS. The system allows process managers to create, modify and adapt workflows to suit the changing business needs, and/or to allow for variations related to special business requirements. Workflow descriptions are stored in a temporal repository and can be used for looking up past business processes and to provide historical context for past event logs of operations.

The main part of the system controls the operation of the workflows. It responds to actions of various actors to the system and communicates messages about the operation of the system to them. The control system has a workflow orchestrator component that looks up the current workflow definition and orchestrates responses by invoking specific Web Services. The control component also manages and updates the data stored and current state of the workflow operation and provides an event audit log of the key events and actions that occur within the operation of the workflow.

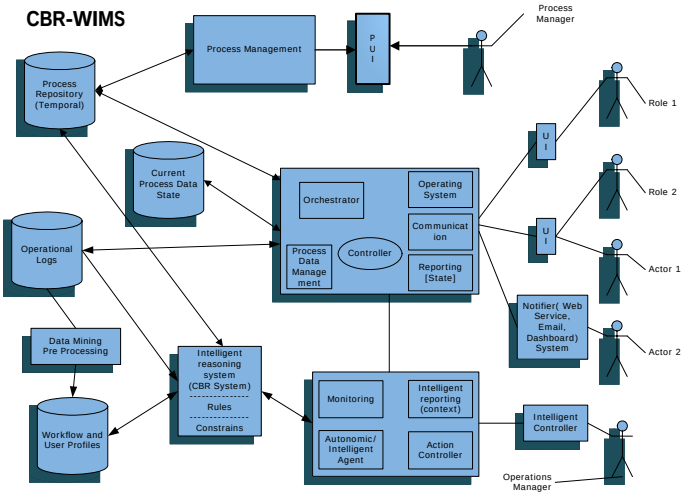


Fig. 3. The Intelligent Workflow Management System Architecture

The workflow monitoring and intervention controller monitors, reports, and proposes possible remedial actions to the workflow operation manager. The monitoring system uses a CBR system to retrieve past useful experience about workflow problems occurred in the past by retrieving similar sequences of events/actions in the events log for a given workflow (or workflow part) compared to the current state and recent sequence of events/actions in the operation of the workflow. If a fault or possible problem pattern is detected, this is reported to the workflow operations manager together with the retrieved similar cases and associated recorded experience of any known remedy/course of action.

In order to deal with the uncertain and contextual dimension of workflow similarity, the CBR system relies on knowledge discovered from past cases about workflow norms and user profiles created by statistical and data mining pre-processing. The pre-processing

component analyses operational logs and attempts to discover knowledge about norms and patterns of operation that can be used in the calculation of the similarity measures for the CBR process. This is particularly important for the monitoring process as any "interesting" or "abnormal" states need to be seen in the context of what has been normal or abnormal behaviour in past event sequence cases.

5 Workflow Monitoring Experiments and Evaluation

In order to evaluate the suitability of the approach proposed in this paper, a number of simple experiments were conducted using the CBR-WIMS system. A simplified workflow process based on the exam moderation problem was constructed and a simulation was used to produce a series of workflow case studies. 320 simple event logs of workflows were produced to serve as cases in the case base. Each case was labelled as either "stalled" or "not stalled" to indicate the presence or not of a problem in the workflow execution. Only exam upload actions were considered and only the last 3 such uploads in a series of workflow events were used to represent each case. A workflow event log audit trace is represented as:

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

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

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

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

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

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

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

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

C= (CoordUpload,John,3, ModUpload, Phil, 0,CoordUpload, John, 5) and

C'=(ModUpload,Phil,4, ModUpload, Phil, 0,CoordUpload, Mary, 3)

Assembling the MCSG:

1. •(5, 3)=1-2/8=0.75
2. •(CoordUpload, CoordUpload) = 1
3. •(0, 0)=1
4. •(ModUpload, ModUpload) = 1
5. •(4, 3)=1-1/7=0.857
6. •(CoordUpload, ModUpload) = 0 .. MCSG Matching stops

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

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

The 320 cases were split randomly into a case base of 300 cases and 20 test target cases. Using the KNN algorithm for K=3, the three nearest neighbours to every target case were

used to classify the target case as “stalled” or “not stalled” using simple voting. The results were compared against the known classification for the target cases. This evaluation run was repeated 10 times and the results of the classification were averaged over the 10 runs.

Table 1. below shows the results of the evaluation runs:

	Average number of cases / 20	**%**
Target Cases Correctly classified	13.8	69
Missed positives	5	25
False positives	1.2	6

Table 1. First Evaluation results – no normalisation for person profiles

For the second set of experiments, the interval similarity measures were normalised to take into account the different rates of responses expected from different workflow actors. A Data analysis of the cases classified workflow actors into:

Fast responders: 0-2 days / Medium responders: 2-4 days / Slow responders: over 4 days

For these cases, the interval duration I for each interval was replaced by the difference of the actual duration minus the nominal duration for the relevant type of workflow actor:

Fast responders: 1 day / Medium responders: 3 days / Slow responders: over 5 days

So assuming that if in the example above analysis of past behaviour has shown that John is a fast responder and Phil is a slow responder, the case is represented as:

C= (CoordUpload,John,2, ModUpload, Phil, 5,CoordUpload, John, 4)

This way the similarity measure is modified to provide a context based on knowledge discovered from past cases. The results of running a similar set of experiments as in the first iteration are summarised in Table 2.

	Average number of cases / 20	**%**
Target Cases Correctly classified	15.3	76.5
Missed positives	3.8	19
False positives	0.9	4.5

Table 2. Second Evaluation results – normalised for person profiles

It can be seen that the overall number of target cases correctly classified has increased, mainly by the corresponding reduction of missed positives.

This preliminary evaluation is encouraging. Further evaluation using a larger dataset from actual (not simulated) workflow event audit logs is planned to evaluate this approach further. In the planned work, larger segments of event log will be used in the case represenation involving the full set of possible exam moderation actions and events to predict the exact type of workflow disruption.

6 Conclusions

This paper discussed an approach for intelligent diagnosis and monitoring of workflows based on incomplete operation data in the form of temporal log data. This was based on a graph representation of workflows using temporal relationships. The workflow process is orchestrated by a software system using BPEL technologies in service oriented architecture in the CBR-WINS system. The matching and similarity measures presented here showed in

a preliminary evaluation that they are capable of classifying problems correctly in a simplified workflow process. In particular it was shown that an analysis of past workflow event logs can provide norms and context that can reduce the uncertainty in similarity based matching and improve the efficiency of the reasoning process.

Further work will concentrate on further and more realistic evaluation of the approach based on more complex case representation and similarity matching. Work on further building and automating the CBR-WINS system will allow the extension to provide intelligent advice to operators in addition to the existing simple monitoring action. Other work direction will cover the challenge of explaining the reasoning results and advice to the workflow operation managers, the combination of constraints and temporal consistency checking and the combination of workflow event log temporal knowledge with other uncertain temporal knowledge available about a workflow. Finally, the reuse of knowledge across different workflows, concentrating on changed workflows and variants will be investigated.

References

1. Business Process Management Initiative (BPMI): BPMN 1.1: OMG Specification, February, 2008, http://www.bpmn.org/, accessed April 2009.
2. OASIS: BPEL, The Web Services Business Process Execution Language Version **2.0.** http://www.oasis-open.org/apps/org/workgroup/wsbpel, May 2006.
3. Workflow Management Coalition (WfMC): XPDL 2.1 Complete Specification (Updated Oct 10, 2008), http://www.wfmc.org/xpdl.html, accessed April 2009.
4. Minor, M., Tartakovski, A. and Bergmann, R.: Representation and Structure-Based Similarity Assessment for Agile Workflows, in Weber, R., O. and Richter, M., M.(Eds) Case-Based Reasoning Research and Development, Proceedings of the 7th international conference on Case-Based Reasoning, ICCBR 2007, Belfast, NI, UK, August 2007, LNAI 4626, pp 224-238, Springer-Verlag, 2007
5. Bunke, H., Messmer, B.T.: Similarity Measures for Structured Representations. In: Wess, S., Richter, M., Althoff, K.-D. (eds) Topics is Case-Based Reasoning. LNCS, vol. 837, pp 106-118, Springer, Heidelberg (1994)
6. Mileman, T., Knight, B., Petridis, M., Cowell, D., Ewer, J. (2002): Case-Based Retrieval of 3-D shapes for the design of metal castings in Journal of Intelligent Manufacturing, Kluwer. 13(1): 39-45; Feb 2002.
7. Wolf, M., Petridis, M.: Measuring Similarity of Software Designs using Graph Matching for CBR, In workshop proceedings of AISEW 2008 at ECAI 2008, Patras, Greece (2008)
8. Ma, J., Knight, B.: A General Temporal Theory, the Computer Journal, 37(2), 114-123 (1994).
9. Allen, J., Hayes, P.: Moments and Points in an Interval-based Temporal-based Logic, Computational Intelligence, 5, 225-238 (1989).
10. Ma, J., Knight, B.: A Framework for Historical Case-Based Reasoning, in K.D. Ashley and D.G. Bridge (Eds.): ICCBR 2003, LNAI 2689, pp. 246–260 (2003)
11. Ma, J., Knight, B., Petridis, M.: Deriving Explanations from Partial Temporal Information, In workshop proceedings of ExACT-08 at ECAI 2008, Patras, Greece (2008)

Sampling with Confidence: Using *k*-NN Confidence Measures in Active Learning

Rong Hu, Sarah Jane Delany and Brian Mac Namee

Dublin Institute of Technology, Dublin, Ireland
rong.hu@dit.ie,sarahjane.delany@dit.ie,brian.macnamee@dit.ie

Abstract. Active learning is a process through which classifiers can be built from collections of unlabelled examples through the cooperation of a human oracle who can label a small number of examples selected as *most informative*. Typically the most informative examples are selected through *uncertainty sampling* based on classification scores. However, previous work has shown that, contrary to expectations, there is not a direct relationship between classification scores and classification confidence. Fortunately, there exists a collection of particularly effective techniques for building measures of classification confidence from the similarity information generated by *k*-NN classifiers. This paper investigates using these confidence measures in a new active learning sampling selection strategy, and shows how the performance of this strategy is better than one based on uncertainty sampling using classification scores.

1 Introduction

Active Learning (AL) [1] attempts to overcome the problem that in supervised learning labelled datasets can be difficult or expensive to obtain. AL attempts to build labelled datasets by selecting only the *most informative* examples in a larger unlabelled example set for labelling by an *oracle*, typically a human expert. The most common selection strategy for picking these most informative examples is *uncertainty sampling* [2] in which examples are selected based on the certainty with which a classifier can classify them.

The typical approach to uncertainty sampling is to use the output of a ranking classifier that produces numeric *classification scores* (e.g. *k*-Nearest Neighbour, Naïve Bayes or Support Vector Machines) as a measure of *classification confidence*. However, Delany et al. [3] have shown that there is not a direct relationship between classification scores and classification confidence. This suggests that AL selection strategies that measure certainty using factors other than classification scores would be more effective. Delany et al. [3] show that an aggregate of five basic confidence measures used with *k*-NN classifiers are particularly effective in estimating classification confidence. In this paper we investigate an AL selection strategy based on these confidence measures, and evaluate whether this performs better than a selection strategy based on classification scores.

Section 2 will discuss AL in more detail and provide examples of how AL has been used in Case-Based Reasoning (CBR). Section 3 will then discuss the

confidence measures that will be used in our selection strategy. Section 4 will describe our overall AL approach including the details of how the confidence measures are integrated into the selection process. This *confidence-based selection strategy* has been evaluated against a strategy based on classification scores using a number of text datasets and the results of these evaluations will be presented and discussed in Section 5. Finally, we conclude and outline our intended directions for future work in Section 6.

2 AL and CBR

The principle aim of AL is to build quality classifiers using as few labelled training examples as possible. The most common AL scenario is *pool-based* AL [2, 4] which assumes that the learner has access to a large pool of unlabelled examples from the beginning of the process and this is the scenario considered in this work.

The pool-based AL process begins by selecting a small number of examples from the pool, that the oracle is asked to label to form the initial *labelled set*, or *case base*. The labelled set is used to build a classifier which in turn is used to calculate the *informativeness* of each example remaining in the pool. The informativeness of an example is a measure of how useful to the training process it would be to solicit the oracle for a label for that example. The most informative examples from the pool are then labelled by the oracle, removed from the pool, and added to the labelled set. A new classifier is then built using the labelled set and the process iterates until a stopping criteria is reached — for example the oracle exceeds a label budget, or labelling further examples is not deemed sufficiently informative.

The predominant research issue in pool-based AL is determining the best selection strategy for choosing those examples most informative to the training process. Uncertainty sampling, first proposed by Lewis and Gale [2], is the most widely used approach. Uncertainty sampling uses ranking classifiers that associate a certainty score with each classification. The certainty score, $P(C|e)$, indicates the certainty of the system that example e belongs to class C. Certainty scores fall into the range $[0, 1]$ where 0 indicates that the system is certain that the example does not belong to the class in question, and 1 indicates that it is certain that it does. At each iteration of the AL process the certainty scores of each example are computed and those for which classifications are least certain (i.e. those with scores closest to 0.5) are selected for labelling. The philosophy behind this approach is that a better classifier can be built by reducing the uncertainty in the dataset. The advantages of the uncertainty sampling approach include its simplicity and fast execution speed.

Other selection strategies include *version space reduction* [1] in which examples that best reduce the version space associated with a classifier are selected; Query-By-Committee (QBC) [5] in which the examples that give rise to the most disagreement in an ensemble of classifiers are selected; the use of Expectation-

Maximization (EM) [6]; and the inclusion of density information to select those examples in most densely populated regions of the example space [7].

Although just about any classifier can be used in the AL process, the CBR approach to classification is particularly attractive as certainty scores are easily calculated, and the repeated classifier retraining required in AL is especially efficient — new examples are simply added to the case base. Two of the earliest examples of using CBR and AL together were by Hasenjager & Ritter [8] who contrasted local learning approaches against global ones; and Lindenbaun et al. [9] who developed AL strategies for nearest neighbour classifiers. More recent examples of the use of CBR and AL together include their combination for the semantic labelling of text [10]; solving problems in drug development [11]; creating case retention strategies for CBR [12]; and supervised network intrusion detection [13].

Earlier work by Li et al. [14] proposed a *confidence-based AL* approach to image segmentation which calibrates the classification scores of SVM classifiers to classification confidence [15]. The overall benefits of using classifiers properly calibrated to produce class-membership probabilities is discussed in [16].

3 Confidence Measures

To attach confidence to classification scores Delany et al. [3] proposed five basic confidence measures that can be used with k-NN classifiers and showed that an aggregate of these is particularly effective. The use of aggregate measures is also supported by the work of Cheetham & Price [17] who presented a similar result, using different measures.

The objective of the k-NN measures is to assign higher confidence to those examples that are 'close' (i.e. with high similarity) to examples of its predicted class, and are 'far' (i.e. low similarity) from examples of a different class. The closer a target example is to examples of a different class, the higher the chance that the target example is lying near or at the decision surface. Whereas the closer an example is to other examples of the same class, the higher the likelihood that it is further from the decision surface. All the k-NN measures perform some calculation on a ranked list of neighbours of a target example using a combination of:

- the distance between an example and its nearest neighbours ($NN_i(t)$ denotes the ith nearest neighbour of example t),
- the distance between the target example t and its nearest like neighbours ($NLN_i(t)$ denotes the ith nearest *like* neighbour to example t),
- the distance between an example and its nearest unlike neighbours ($NUN_i(t)$ denotes the ith nearest *unlike* neighbour to example t).

Preliminary experiments using the five measures proposed in [3] showed a high correlation between three of them, and so we chose to use the three of the five that are least correlated in our evaluations. Full details on each measure can be found in [3].

Average NUN Index (M1) The Average Nearest Unlike Neighbour Index (Avg NUN Index) is a measure of how close the first k NUNs are to the target example t as given in Equation 1.

$$AvgNUNIndex(t,k) = \frac{\sum_{i=1}^{k} IndexOfNUN_i(t)}{k} \tag{1}$$

where $IndexOfNUN_i(t)$ is the index of the ith nearest unlike neighbour of target example t, the index being the ordinal ranking of the example in the list of NNs.

Similarity Ratio (M2) The Similarity Ratio measure calculates the ratio of the similarity between the target example t and its k NLNs to the similarity between the target example and its k NUNs, as given in Equation 2.

$$SimRatio(t,k) = \frac{\sum_{i=1}^{k} Sim(t, NLN_i(t)) + \epsilon}{\sum_{i=1}^{k} Sim(t, NUN_i(t)) + \epsilon} \tag{2}$$

where $Sim(a,b)$ is the similarity between examples a and b and ϵ is a smoothing value to allow for situations where an example may have no NLNs or NUNs ($\epsilon = 0.0001$ is used in all of our evaluations).

Similarity Ratio Within K (M3) The Similarity Ratio Within K is similar to the Similarity Ratio as described above except that, rather than consider the first k NLNs and the first k NUNs of a target example t, it uses only the NLNs and NUNs from the first k neighbours. It is defined in Equation 3.

$$SimRatioK(t,k) = \frac{\sum_{i=1}^{k} Sim(t, NN_i(t))\delta_{t,NN_i(t)}}{\epsilon + \sum_{i=1}^{k} Sim(t, NN_i(t))(1 - \delta_{t,NN_i(t)})} \tag{3}$$

where $Sim(a,b)$ is as above, δ_{ab} is Kronecker's delta where $\delta_{ab} = 1$ if the class of a is the same as the class of b and 0 otherwise, and ϵ is a smoothing value to allow for situations where an example may have no NUNs ($\epsilon = 0.0001$ is used).

4 Approach

The important aspects of the AL process are: forming the initial case base, building a classifier to label all examples in the pool, and selecting examples for labelling by the oracle. This section will describe our approach to each of these (further details are available in [18]).

4.1 Initial Case Base Selection and Classifier

The AL process begins with a small set of examples labelled by the oracle which is the initial case base. While this selection can be performed at random, it offers an opportunity to prime the AL process through informed selection. Previous work has shown that using clustering to select the initial case base gives better

results than random selection [19] . However, this can lead to highly inconsistent results over many trials as clustering is quite unstable, especially when dealing with high dimensional textual data. For this reason, we use the *furthest-first* initialisation algorithm [20] which is deterministic and will always return the same initial case base for a given dataset.

At every iteration of the AL process all of the unlabelled examples remaining in the pool are classified using a classifier trained on the examples labelled by the oracle so far. In our system the classifier used to do this is a k-NN classifier using distance weighted voting [21] with $k = 5$.

4.2 The Confidence-Based Selection Strategy

Before any of the confidence measures described in Section 3 can be used to calculate classification confidence it is necessary to identify for each measure a confidence threshold value for each of the possible classes. Predictions with confidence values higher than the predicted class's threshold are considered *confident*, while those with values below are considered *non-confident*. The threshold value for a particular class is that value that results in the highest proportion of correctly predicted examples of a particular class when there were no incorrect predictions. The confidence thresholds are referred to as $thres_{ij}$ for each confidence measure M_i $(i = 1 \ldots n)$, and each class $j = 1 \ldots c$. Specific details on the approach used for setting the threshold level for a class are described in [3].

Our ACM Selection (ACMS) strategy aggregates the three confidence measures used into a new selection strategy. First each example e_k in the pool is classified using the initial case base and the value for each confidence measure m_{ik} is calculated. Based on the predicted class of the example the appropriate threshold value is checked for each of the measures. If any one of the measures indicates confidence, i.e. $m_{ik} > thres_{ij}$ for any $i = 1 \ldots n$ and $j =$ *the predicted class*, then we consider that the example has been classified with confidence, and it gets added to the *confident set*. Otherwise, it gets added to the *non-confident set*.

A single $rank(e_k)$ value is associated with each e_k example. For an example e_k classified with confidence, $rank(e_k)$ is assigned the value that indicates most confidence, i.e. $rank(e_k) = max(m_{ik})$ for those M_i's that indicate confidence; while the one used for an example in the non-confident set should be the m_{ik} that indicates least confidence (i.e. $rank(e_k) = min(m_{ik})$ for those M_i's that do not indicate confidence). Different strategies for combining confidence measures were considered in preliminary experiments which showed the min/max combination to be consistently best.

In order to be able to compare m_{ik} across different confidence measures, the values of m_{ik} for each M_i are normalised using statistical normalisation after performing a log transformation to correct those with skewed distributions.

Once all pool examples have been classified, the one that the classifier is least confident of is the example in the non-confident set that has the smallest $rank(e_k)$ value. If the non-confident set is empty, the least confident example is the one in the confident set with the smallest $rank(e_k)$ value. This is the example

that is presented to the oracle for labelling before the process repeats until the stopping criteria is met. The algorithm for our ACMS strategy is presented in Algorithm 1.

5 Evaluation

The two objectives to the evaluations described here were to confirm the superiority of using an aggregate confidence measure over using single confidence measures; and to compare the performance of our ACMS approach with an uncertainty sampling approach based on classification scores.

In order to conduct a comprehensive analysis, we tested various algorithms on seven datasets: a spam dataset [22]; four binary classification textual datasets derived from the 20-Newsgroup collection[1]; and two binary classification datasets from the Reuters collection[2]. The properties of each dataset and the average accuracy achieved in five iterations of 10-fold cross validation using a 5-NN classifier are shown in Table 1 (accuracies are included as an indication of the difficulty of each classification problem). Each dataset was pre-processed to remove stop-words and stemmed using Porter stemming.

To evaluate the system, we simulated the labelling process and compared the results with the actual labels in each dataset. The accuracy of the labelling is used to evaluate the performance of the system, calculated as $Accuracy = C/N$ where N is the number of examples in the dataset (including the examples in the initial case base) and C is the number of correctly labelled examples. Both manually and automatically labelled examples are included in this calculation to avoid the accuracy figure becoming unstable in the latter stages of the process. The accuracy is recorded after each manual labelling.

At present we use a simple stopping criterion that allows the human oracle to only provide a specified number of labels, a *label budget*. We set the label budget to 110 which includes 10 initial labels and 100 during the AL process.

We evaluated the performance of sampling selection strategies using each individual confidence measure and using the aggregation of the measures on all of the datasets. Illustrative results on two datasets are shown in Figure 1. The results indicate that ACMS is at least as good as but generally dominates the individual measures. Furthermore, we found that the ACMS strategy is more stable than using individual measures.

Figure 2 shows the results of comparing the ACMS strategy with the more typical Uncertainty Sampling (US) strategy using classification scores. A Random Sampling (RS) strategy, which randomly picks the example to label, is also included as a baseline. The accuracy graph for the ACMS strategy dominates the graph for the RS strategy in all cases, and the graph for the US strategy for five (WinXwin, Comp, Vehicle, Reuters, Spam) of the seven datasets. Interestingly, across all ACMS experiments the average *effectiveness* — how often

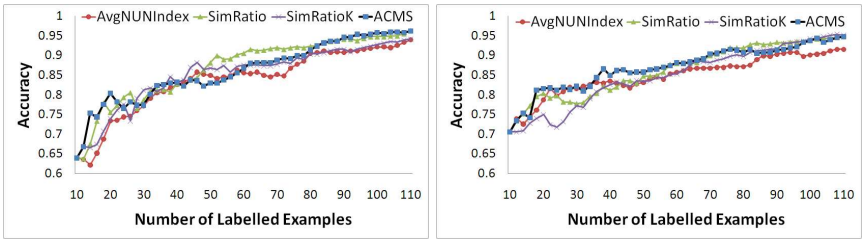
[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html
[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Input: An initial labelled case base $\mathcal{CB}$, an unlabelled pool $\mathcal{P}$ of p examples, a k-NN classifier $\mathcal{C}$ for classes $1 \dots c$, a stopping criterion $\mathcal{S}$, a batch size b, a set of confidence measures M_i, $i = 1 \dots n$
Output: A labelled case base
while *S is not met* **do**
  foreach *confidence measure* $M_i, i = 1 \dots n$ **do**
    Identify the threshold: find $thres_{ij}$ and k_{ij}, for $j = 1 \dots c$;
  end
  foreach *example* $e_k \in \mathcal{P}$ **do**
    $ConfSet = \emptyset$, $NonConfSet = \emptyset$, $Selected = \emptyset$;
    Classify e_k using the classifier $\mathcal{C}$;
    Calculate m_{ik} using k_{ij} for $i = 1 \dots n$ and j = predicted class of e_k;
    if $m_{ik} > thres_{ij}$ *for any* $i = 1 \dots n$ *and* $j =$ *predicted class of* e_k **then**
      $ConfSet = ConfSet + e_k$;
      Set the ranking score: $rank(e_k) = max(m_{ik})$;
    else
      $NonConfSet = NonConfSet + e_k$;
      Set the ranking score: $rank(e_k) = min(m_{ik})$;
    end
  end
  foreach $l, l = 1 \dots b$ **do**
    if $NonConfSet == \emptyset$ **then**
      $Selected = Selected + e$ where
      $rank(e) = min(rank(e_k)), e_k \in ConfSet$;
    else
      $Selected = Selected + e$ where
      $rank(e) = min(rank(e_k)), e_k \in NonConfSet$;
    end
  end
  Label each $e_l \in Selected$;
  $\mathcal{CB} = \mathcal{CB} \cup Selected$, $\mathcal{P} = \mathcal{P}/Selected$;
end

Algorithm 1: The algorithm for the Aggregated Confidence Measure Selection (ACMS) strategy

Table 1. Benchmark Datasets.

Dataset	**Task**	**Examples**	**Features**	**Accuracy**
WinXwin	comp.os.ms-windows.misc vs. comp.windows.x	496	8557	91.14%
Comp	comp.sys.ibm.pc.hardware vs. comp.sys.mac.hardware	500	7044	85.56%
Talk	talk.religion.misc vs. alt.atheism	500	9000	93.92%
Vehicle	rec.autos vs. rec.motorcycles	500	8059	92.96%
Reuters	acq vs. earn	500	3692	89.56%
RCV1	g151 vs. g158	500	6135	95.36%
Spam	spam vs. non-spam	500	18888	96.80%



(a) Talk Dataset (b) Vehicle Dataset

Fig. 1. Comparison of Individual Confidence Measures and the ACM as the Sampling Selection Strategy

the $rank(e_k)$ given to a case by ACMS is determined by a particular confidence measure — of M1, M2 and M3 are 38.87% 34.57% and 26.56% respectively.

6 Conclusions and Future Work

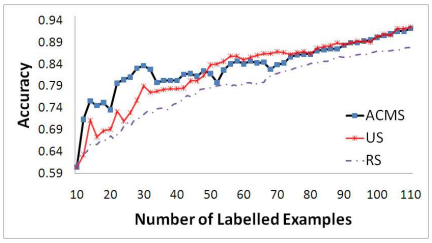
In this paper, we propose a new selection strategy for active learning using k-NN based confidence measures. The experimental results show that an aggregated confidence measure is more effective than single confidence measures. We also show that ACMS generally outperforms the more typical uncertainty sampling approach using classification scores. Although the algorithm is computationally expensive, the use of k-NN classifier makes it possible to cache and re-use case similarities making ACMS computationally feasible, even for large datasets. Furthermore, a larger batch size b can be used to reduce the computational load.

There are three main areas we intend to explore in the future. Firstly, the furthest-first method may include outliers in the initial case base which may limit the exploitation capability of the AL process. To solve this problem, more sophisticated initial case base selection strategies will be considered. However, the stability problems with clustering textual data must be overcome.

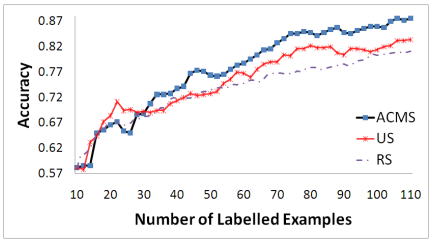
Secondly, ACMS focuses on refining the decision boundary. However, there is a balance to be achieved between this and the exploration of new regions in the decision space that the current classifier may not perform well on. We will consider using additional information, such as density information to allow our AL process to explore more while maintaining good performance.

Finally, the work described here has focussed on binary classification, but we intend to extend this to multi-class situations in the near future.

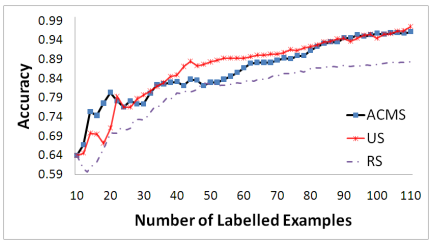
Acknowledgments. This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/RFP/CMSF718.



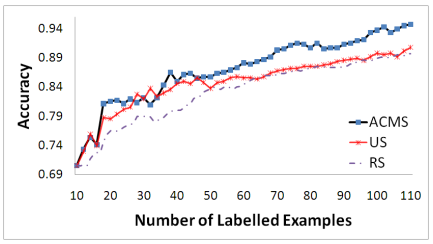
(a) WinXwin Dataset



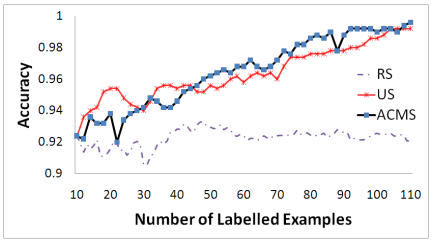
(b) Comp Dataset



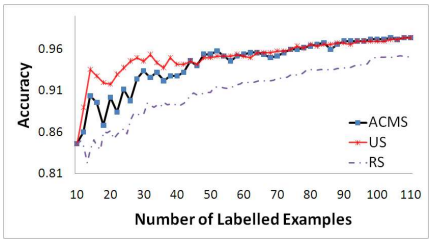
(c) Talk Dataset



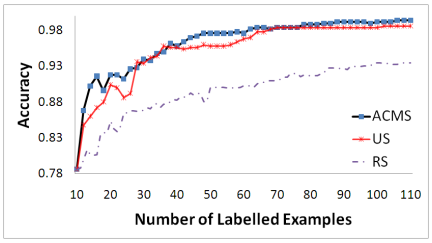
(d) Vehicle Dataset



(e) Reuters Dataset



(f) RCV1 Dataset



(g) Spam Dataset

Fig. 2. Comparison of ACMS, US and RS selection strategies

References

1. Tong, S.: Active Learning:Theory and applications. PhD thesis, Computer science department, Stanford University (August 2001)
2. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: Proc 17th annual International ACM SIGIR conference on Research and Development in Information Retrieval, Springer-Verlag NY (1994) 3–12
3. Delany, S.J., Cunningham, P., Doyle, D.: Generating estimates of classification confidence for a case-based spam filter. In: Proc of ICCBR '05. Volume 3620 of LNAI., Springer (2005) 170–190
4. McCallum, A.K., Nigam, K.: Employing EM and pool-based active learning for text classification. In: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann (1998)
5. H.S.Seung, M.Opper, H.Sompolinsky: Query by committee. In: In Proceedings of the Fifth Workshop on Computational Learning Theory. (1992) 287–294
6. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Learning to classify text from labeled and unlabeled documents. In: Proc of AAAI '98. (1998) 792–799
7. Xu, Z., Yu, K., Tresp, V., Xu, X., Wang, J. In: Representative Sampling for Text Classification Using Support Vector Machines. Springer (2003) 11
8. Hasenjager, M., Ritter, H.: Active learning with local models. In: Neural Processing Letters. Volume 7. (1998)
9. Lindenbaum, M., Markovitch, S., Rusakov, D.: Selective sampling for nearest neighbor classifiers. In: Proceedings of AAAI '99. (1999) 366–371
10. Mustafaraj, E., Hoof, M., Freisleben, B.: Learning semantic annotations for textual cases. In: In Textual Case-based Reasoning Workshop at the 6th ICCBR. (2005)
11. Cebron, N., Berthold, M.R.: An adaptive multi objective selection strategy for active learning. Technical report, Universität Konstanz (2007)
12. Ontañón, S., Plaza, E.: Collaborative Case Retention Strategies for CBR Agents. In: Case-Based Reasoning Research and Development. Springer (2003)
13. Li, Y., Guo, L.: An active learning based TCM-KNN algorithm for supervised network intrusion detection. Computers and Security **26** (2007) 459–467
14. Li, M., Sethi, I.K.: Confidence-based active learning. IEEE Trans. Pattern Anal. Mach. Intell. **28** (2006) 1251–1261
15. Ma, A., Patel, N., Li, M., Sethi, I. In: Confidence Based Active Learning for Whole Object Image Segmentation. Springer Berlin / Heidelberg (2006) 753–760
16. Cohen, I., Goldszmidt, M.: Properties and benefits of calibrated classifiers. In: 8th European Conference on Principles and Practice of Knowledge Discovery in Databases. (2004) 125–136
17. Cheetham, W., Price, J.: Measures of solution accuracy in case-based reasoning systems. In: Proc of ECCBR '04. (2004) 106–118
18. Hu, R., Mac Namee, B., Delany, S.J.: Sweetening the dataset: Using active learning to label unlabelled datasets. In: Proceedings of the the 19th Irish Conference on Artificial Intelligence and Cognitive Science (AICS '08). (2008)
19. Kang, J., Ryu, K., Kwon, H.: Using Cluster-Based Sampling to Select Initial Training Set for Active Learning in Text Classification. In: Advances in Knowledge Discovery and Data Mining. Volume 3056. Springer (2004) 384–388
20. Greene, D.: A State-of-the-Art Toolkit for Document Clustering. PhD thesis, University of Dublin, Trinity College (2006)
21. Mitchell, T.: Machine Learning. McGraw Hill (1997)
22. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L. In: A Case-Based Technique for Tracking Concept Drift in Spam Filtering. Springer London (2005) 3–16

Case-Based Reasoning in the Health Sciences

Workshop at the
Eighth International Conference on
Case-Based Reasoning
(ICCBR 2009)

Seattle, Washington, USA
July, 2009

Cindy Marling and Stefania Montani (Eds.)

Co-Chairs

Cindy Marling
Ohio University, USA
marling@ohio.edu

Stefania Montani
University of Piemonte Orientale, Italy
stefania.montani@unipmn.it

Programme Committee

Klaus-Dieter Althoff, University of Hildesheim, Germany
Isabelle Bichindaritz, University of Washington, USA
Peter Funk, Malardalen University, Sweden
Alec Holt, University of Otago, New Zealand
Jean Lieber, Loria, France
Lakhmi Jain, University of South Australia, Australia
Stefan Pantazi, Conestoga College Institute of Technology, Canada
Petra Perner, Institute of Computer Vision & Applied Computer Sciences, Germany
Luigi Portinale, University of Piemonte Orientale, Italy
Rainer Schmidt, University of Rostock, Germany
Olga Vorobieva, I. M. Sechenov Institute of Evolutionary Physiology and Biochemistry, Russia

Preface

The research community working on health sciences applications of case-based reasoning (CBR) has been very active recently, as evidenced by special issues hosted by first class Artificial Intelligence (AI) journals, as well as by books currently being edited on the topic. It is now a tradition that the community meets yearly at the workshops on CBR in the health sciences, held during ECCBR/ICCBR conferences. As a matter of fact, this workshop is the seventh in a series of exciting workshops, the first six of which were held at ICCBR-03, in Trondheim, Norway, at ECCBR-04, in Madrid, Spain, at ICCBR-05, in Chicago, USA, at ECCBR-06, in Oludeniz, Turkey, at ICCBR-07, in Belfast, Northern Ireland, and at ECCBR-08, in Trier, Germany.

Five papers are to be presented at this seventh workshop on CBR in the health sciences. These papers represent the research and experience of 18 authors working in four different countries on a wide range of problems and projects, and illustrate some of the major trends of current research in the area.

The first paper (by Bichindaritz et al.) is motivated precisely by the significant number of recently published research contributions to CBR in the health sciences, which requires a classification and literature analysis effort. To this end, the authors have proposed a proper classification and indexing system, and have then applied it to discover research trends in the field.

The large scientific production in this field clearly demonstrates how well suited CBR is for medical applications. As a matter of fact, some medical CBR tools are already in routine clinical use, while other research prototypes aim to move beyond the laboratory into clinical practice and commercialization. This is the aim of the project described in the second paper (by Marling et al.), which reports on a research study being realized to evaluate a case-based decision support system for managing diabetes patients, as a pre-liminary step for designing a clinical trial.

The success of a CBR tool for supporting medical decision making can be further in-creased by defining a close synergy between CBR itself and other AI methodologies. This claim is supported by the other papers to be presented at the workshop. In the third paper (by Ahmed et al.), CBR is combined with several techniques, such as textual information retrieval, rule-based reasoning and fuzzy logic, to enable a more reliable and efficient diagnosis and treatment of stress. In the fourth paper (by Nicolas et al.), classification rules are adopted to combine the results of two independent CBR systems, in order to improve the correctness of skin cancer diagnosis and classification. Finally, in the fifth paper (by Houeland et al.), CBR as well as rule-based and probabilistic model-based reasoning are integrated in a meta-level reasoning architecture for clinical decision support, in which the reasoning process can be automatically and continuously improved in time.

Overall, these papers represent an excellent sample of the most recent advances of CBR in the health sciences, and promise very interesting discussions and interaction among the major contributors in this niche of CBR research.

Cindy Marling July 2009
Stefania Montani

Classification and Characterization of Case-Based Reasoning Research in the Health Sciences

Isabelle Bichindaritz and John C. Reed Jr.

University of Washington Tacoma, Institute of Technology
1900 Commerce Street
Tacoma, Washington 98402, USA
ibichind@u.washington.edu

Abstract. Research in case-based reasoning in the health sciences started about 20 years ago and has been steadily expanding during these years. This paper describes the state of the research through an analysis of its mainstream literature. The methodology followed involves first the definition of a classification and indexing scheme for this research area using a tiered approach to paper categorization based on application domain, purpose of the research, memory organization, reasoning characteristics, system design, and research theme. This paper analyzes the literature trends in terms of research themes, application domains, application purposes, evolution of number of papers and authors.

Keywords: case-based reasoning in the health sciences, classification

1 Introduction

The field of Case-Based Reasoning in the Health Sciences (CBR-HS) [1] has seen a tremendous growth in the last decade. Six special conference workshops have been held consecutively from 2003 focused solely on this topic and are accessible through Cbr-biomed.org Web portal [2]. A special issue on CBR-HS was published in the Journal of Artificial Intelligence in Medicine [3] and a second one in the Computational Intelligence journal [4]. Most recently, yet another special issue has been published in the Applied Intelligence journal [5]. Moreover CBR-HS papers are often published in different artificial intelligence and health informatics journals and conferences. We developed a classification and indexing scheme for CBR research in the Health Sciences to make possible the meta-analysis of this interdisciplinary research area [1]. This paper details knowledge of CBR-HS gained by using this classification scheme and the research trends identified in terms of application domains, application purposes, evolution of number of papers, citations, and research themes.

2 Methods

The specific application of CBR to the health sciences has been discussed in several surveys [6, 7, 8, 9, 10, 11]. However recent trend analyzes in CBR as a whole failed

to identify CBR-HS as a sub-research area through automatic methods [12]. This may in particular be due to the variety of application domains comprising the health sciences, which prompts for the need for indexing systems capable in particular to group documents related to, for example, oncology, diabetology, phrenology and so forth. Therefore we developed a classification and indexing system capable of drilling down and rolling up in its different components and presented in detail elsewhere [1]. This domain-specific indexing is enabled by the use of one of the most used classification schemes in the health sciences: the Medical Subject Headings (MeSH) [13]. Like most other classifications, it uses a tree like structure where broader categories are narrowed down with each branch and branches are represented by dots.

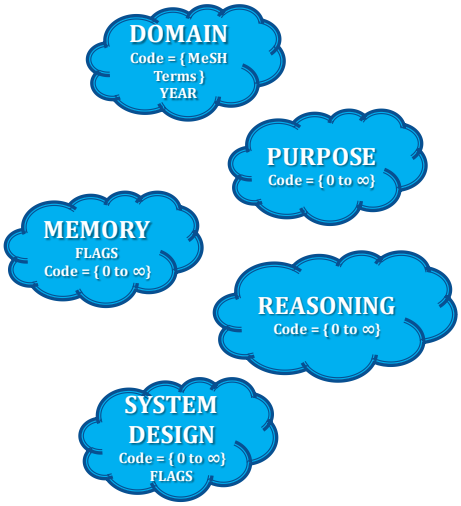


Fig. 1. CBR Health Sciences tiered classification scheme

The papers selected for CBR-HS cover all the 16 ECCBR and ICCBR conferences until 2008, the 6 Workshops on CBR in Health Sciences, the 4 special issues on CBR in the Health Sciences, the 2 DARPA workshops of 1989 and 1991, and the survey papers on CBR in the Health Sciences. 117 papers were indexed with the CBR-HS classification scheme, presented in the next section. Therefore the tables below showing the terminology learned have been refined on these 117 papers.

3 Classification System

Figure 1 presents the tiered architecture of the CBR-HS classification scheme. There are five distinct categories (domain, purpose, memory and case management, reasoning, and system design) defined in this section. In addition, a research theme is selected to characterize the main research hypothesis and findings of the paper.

Table 1. Sample Purpose Classifications

Code	Purpose	Code	Purpose
10	Medical Purpose	20.2	Evaluation
10.1	Decision Support	20.2.1	System Level Testing
10.1.1	Diagnosis	20.2.2	Pilot Testing
10.1.2	Treatment/therapy	20.2.3	Clinical Trial
10.1.3	Prognosis	20.2.4	Routine Clinical Use
10.1.4	Follow-up	20.3	Concept
10.1.5	Classification	20.4	Method
10.2	Tutoring	20.5	Survey
10.3	Epidemiology	30	Bioinformatics Purpose
10.4	Research support	30.1	Proteomics
10.5	Image interpretation	30.2	Phylogenetics
20	Research Purpose	30.3	Genomics
20.1	Formalization	40	Research Theme

Domain

The range of domains in the health sciences fields is vast and, as a result, it was chosen as the first level of classification. However, rather than creating a new set of descriptors, it is proposed to use the MeSH descriptors, of which there are over 24,000 that cover just about every aspect of the health sciences. Along with the domain, another primary means of discriminating the relevance of an article is its publication date. Since the date plays no real role in classifying an article, the date has no field of its own, but instead is combined with the Domain.

Purpose

The purposes, or tasks, of CBR systems have been thoroughly discussed in many articles summarizing the CBR-HS domain. One of the first papers to survey the field in 1998, by Gierl et al., used the purpose as the primary means to subdivide the different systems [6]. In their paper, Gierl et al. specified four main purposes: diagnosis, classification, planning, and tutoring. Later, both Holt et al. 2006 and Nilsson and Sollenborn 2004 used the same four descriptors. In the early years the majority of systems were diagnostic in nature, but in recent years more therapeutic and treatment systems have been developed [14]. Table 1 presents examples of purpose classifications. Planning has been replaced here by treatment since most of the time planning refers to treatment planning. However, planning tasks may involve not only treatment but also other aspects such as diagnosis assessment, which often consists in a series of exams and labs orchestrated in a plan. Planning is a classical major task performed by artificial intelligence systems. Therefore planning is listed in our system as a design option and thus can be added to the treatment choice in the purpose dimension.

Table 2. Sample Memory and Case Management Classifications

Code	Memory Organization	Code	Memory Organization
10	Flat	T	Time Series
20	Hierarchical	A	Text
20.1	Decision Tree	M	Microarray
20.2	Concept Lattice	V	Attribute/Values
20.3	Conceptual Clustering Tree	N	Plans
30	Network	Memory Structures Flag	
40	Inverted Index	G	Ground Cases
Case Representation Flag			P
I	Images	L	Clusters
S	Signals	O	Concepts

CBR systems generally support either medical clinical work, research, or bioinformatics. Therefore we have added these as top level purpose categories. In the clinic, decision support systems support mostly diagnosis, treatment, prognosis, follow-up, and/or classification, such as in image interpretation. Well known diagnostic systems include CARE-PARTNER [15], and AUGUSTE [16]. Well known classification systems include PROTOS [17] and IMAGECREEK [18]. Well known treatment planning systems include T-IDDM [19]. Several systems provide multi-expertise, such as CARE-PARTNER [15] ensuring diagnosis and treatment planning. Well known tutoring systems include ICONS [20].

More recent articles require to differentiate between the purpose of the system developed, which is generally a clinical purpose, from the purpose of the research paper, which can be, among others, a survey paper or a classification paper like this one. Some papers focus on formalization such as KASIMIR [21]. Among these, the evaluation of a system can be performed more or less thoroughly. This is an important dimension to note about a research paper: whether the system was tested only at the system level, which is the most frequent, at the pilot testing level, at the clinical trial level, or finally whether the system is in routine clinical use.

Table 3. Sample Reasoning Classifications

Code	Reasoning
10	Retrieve
10.1	Index
10.2	Similarity measure
20	Reuse
20.1	Adaptation
20.2	Interpretation
30	Revise
40	Retain
40.1	Maintenance

Finally, a paper is generally identified by a research theme by its authors. By indexing a set of 117 papers currently in our database, we have identified major research themes, such as CBR and electronic medical records (EMR), knowledge morphing, CBR and clinical guidelines, or application of a generic CBR framework.

Table 4. System Design Classifications

Code	Construction	Code	Construction
10	Pure CBR	60	Information Retrieval Combination
20	Rule Based Combination	70	Explanation Combination
30	Model Based Combination	CBR Role Flag	
40	Data Mining Combination	P	Primary Technology
40.1	Conceptual Clustering	S	Secondary Technology
40.2	Neural Networks	E	Equivalent Role Technology
40.3	Nearest Neighbor	CBR Additional Technology Flag	
40.4	Decision Tree	T	CBR is Separate
40.5	Bayesian Networks	F	CBR is Combined
50	Planning Combination		

Memory and Case Management

This is a very broad category and could easily be subdivided. It encompasses both how the cases are represented and also how they are organized in memory for retrieval purposes and more (see Table 2). As a result, it is made up of more than one code. The first part of the code represents the format of the cases. The primary types being images, signals, mass spectrometry, microarray, time series data and regular attribute/values pairs, which is used by the majority of the systems. Similar to the different formats of data are the flags that represent what kinds of memory structures the CBR system uses to represent the data, such as ground cases (G), prototypical cases (P), clusters (L), or concepts (O). Lastly, when it comes to memory management there are potentially an infinite number of possibilities, some of which may never have been used before. The main types, however, represent how the memory is organized, whether it is flat or hierarchical, what kind of hierarchical structure, such as decision tree, concept lattice, conceptual clustering tree, or others.

Reasoning

This category regroups the inferential aspects of the CBR. Classically, retrieve, reuse, revise, and retain have been described. Nevertheless, researchers have often added

many more aspects to the inferences, such that it is best to keep this category open to important variations (see table 3). Each of these parts of the reasoning cycle can be hierarchically refined so that a tree is formed here also.

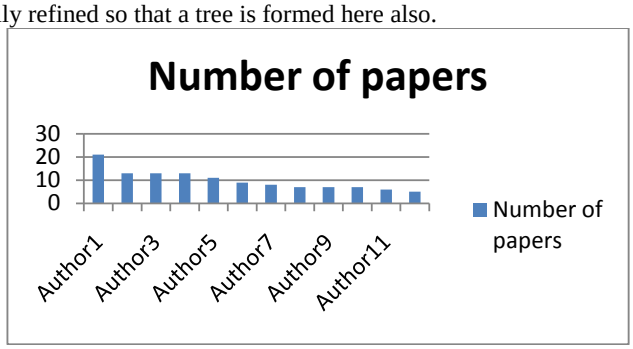


Fig. 2. CBR Health Sciences tiered classification scheme

System Design

The construction of the CBR system specifies what technologies it uses. This area of classification may not seem intuitive at first, but upon the examination of CBR systems it can be seen that many use a combination of technologies, not just case-based reasoning. The most common technology used in conjunction with CBR is rule-based reasoning; however some systems combine CBR with information retrieval, data mining, or other artificial intelligence methods. See table 4 for an example of different possible construction classifications. If the construction of the system does use additional technologies, a flag should be appended to the end of the code to denote whether the case-based reasoning is executed separately. Also, an additional flag is used to designate CBR's role in the system, whether primary, secondary, or equivalent.

5 Research Trends

Trend tracking comprises domains, purposes, memory, reasoning, and system design. First some figures are provided about this domain.

Statistics

The 117 selected papers were written by 132 different authors from over the world. However, a grouping of papers by author clearly identifies a group of researchers contributing particularly actively to CBR-HS research (see Figure 2). The average number of papers per author is 2.19, and the range 1 to 21.

The number of papers by year has seen a rapid increase after 2003 – corresponding to the first workshop on CBR-HS (see Figure 3).

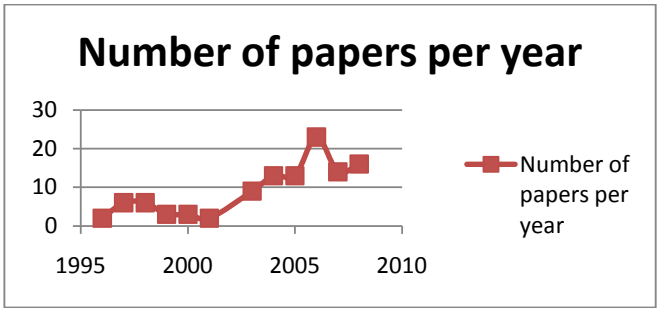


Fig. 3. CBR Health Sciences evolution of the number of papers per year

Domains

The 117 papers cover 41 domains all together. Although the domains of application all belong to the Health Sciences, some domains are more represented than the other ones. The most represented domain is medicine with 26 papers as a whole, which corresponds to either survey papers, or general frameworks and concepts applicable to any health sciences domains. Close second comes oncology (24 papers), then further come stress medicine (13 papers), transplantation (8 papers), diabetology (7 papers), fungi detection (6 papers), breast cancer (6 papers), nephrology (6 papers), genomics (5 papers), and infectious diseases (4 papers). All the other domains count less than 4 papers. It is interesting to note in particular that cancer, being a very prominent disease, is studied by several CBR-HS teams in the world.

Purpose

Among the 24 purposes listed for these papers, 30 papers propose treatments / therapies, 23 papers refer globally to decision support, 21 papers perform diagnoses, 18 papers perform classification tasks, and 11 papers are survey papers. Image interpretation also accounts for 9 papers, and research support for 6 papers.

Memory

Memory structures and organization refers to 17 different concepts. The most represented is prototypes (24 papers), closely followed by time series (20 papers). Further come images (12 papers), text (6 papers), microarray data (5 papers), clusters (3 papers), generalized cases (3), inverted indexes (2 papers), networks (2 papers), and genomic sequences (2 papers). The other listed memory structures are graphs, multimedia data, plans, structured cases, and visio-spatial cases.

Reasoning

Surprisingly, for a methodology like CBR where research is very active in all reasoning cycles phases, in CBR-HS the vast majority of systems refer to retrieval only (61 papers). Maintenance is also well represented (11 papers), and adaptation (8 papers). Further are represented: indexing (3 papers), retain (3 papers), similarity measures (2 papers), and revision (2 papers). However many systems also perform additional reasoning steps, even though the papers studied did not detail these steps, focusing on retrieval aspects instead.

System design

Main characteristics of developed systems include temporal abstraction (22 papers), knowledge-based systems combination (18 papers), prototype and generalized case mining (10 papers), temporal reasoning (7 papers), fuzzy logic (6 papers), information retrieval combination (6 papers), knowledge acquisition combination (5 papers), and in 3 papers: clinical guidelines integration, feature selection, distributed cases, neural networks combination, and planning combination. 38 different types of system design were identified, most of them dealing with some data mining / knowledge discovery combination, and as seen above temporal abstraction / temporal reasoning. However two categories are specific to medical domains: clinical guidelines integration, and electronic medical records integration.

Research themes

So far, the research themes are coded in free text format. More work remains to be done to categorize this part of each paper. One possibility to simplify the problem is to select one particular research focus among the dimensions of the paper: domain, purpose, memory, reasoning, and system design. With this simplification, the research themes are in majority oriented toward design (61 papers), purpose (38 papers), including 11 survey papers, and reasoning (14 papers). It is not surprising that no paper actually focuses exclusively on the application domain – this kind of publication would be better suited for a medical journal in a medical specialty. However it is surprising that so many papers focus on design aspects most of the time dealing with some combination with another methodology of artificial intelligence (data mining, knowledge discovery, temporal reasoning and abstraction, knowledge based systems) or beyond, such as information retrieval and databases. Only 14 papers focus on some reasoning step – in practice only adaptation and retrieval – using methods intrinsic to CBR. The group of papers focusing on the paper purpose regroup such large subgroups as decision-support capability, classification, image understanding, survey papers, papers exploring different roles of CBR, and research support. The memory dimension is almost always connected with either some design aspect such as prototype learning, some reasoning aspect such as using prototypes to guide retrieval, or some purpose aspect, such as decision support. Indeed the memory needs to prove useful to the system in some way, since these papers are applied.

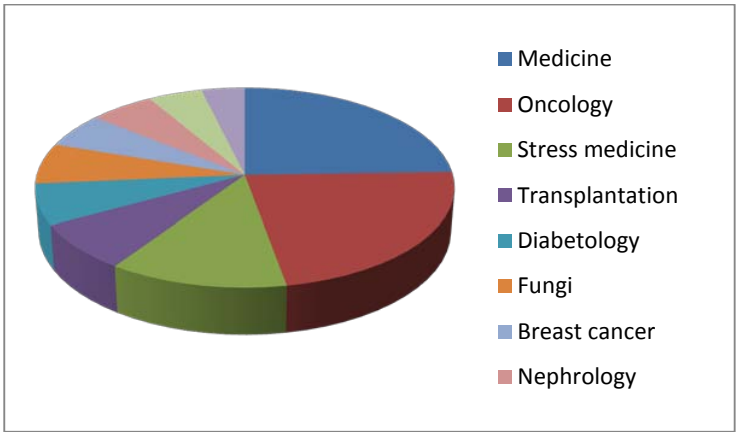


Fig. 4. CBR Health Sciences domains

6 Conclusion

The CBR-HS classification system is being incrementally built. The different categories and each category's list of descriptors is by no means exhaustive. However it proved useful for indexing and tracking CBR-HS research literature. With its systems of tiers, some of which may be omitted, this system is very flexible and can index either fielded applications, frameworks, or survey papers. This study has identified interesting trends characteristic of applied domains such as health sciences domains. Our next goals are to use this classification to index the numerous CBR-HS papers published outside of the venues selected for this pilot study, to perform a citation analysis on this exhaustive pool of papers, as well as to cluster the research themes identified. This analysis of CBR-HS literature will also permit to identify potential future research directions.

References

1. Bichindaritz, I., Reed, J.: Methodology for Classifying and Indexing Case-Based Reasoning Systems in the Health Sciences. In: Florida Artificial Intelligence Research Society Conference, North America, mar. 2009. Available at: http://www.aaai.org/ocs/index.php/FLAIRS/2009/paper/view/144 (2009)
2. Bichindaritz, I., Reed, J.: Cbr in the health sciences workshops. http://www.cbrbiomed.org/workshops/ (2008) This is an electronic document. Date of publication: October 1, 2007. Date retrieved: February 1, 2008. Date last modified: October 17 (2007)
3. Bichindaritz, I.: Case-based reasoning in the health sciences. Artificial Intelligence in Medicine 36(2) 121–125 (2006)
4. Bichindaritz, I., Marling, C.: Introduction to the special issue on case-based reasoning in the health sciences. Computational Intelligence 22(3–4) 143–147 (2006)

5. Bichindaritz, I., Montani, S., Portinale, L.: Special issue on case-based reasoning in the health sciences. Applied Intelligence 1-3 (2008)
6. Gierl, L., Bull, M., Schmidt, R.: CBR in Medicine. In: Case-based reasoning technology. From foundations to applications. Springer-Verlag, Berlin, Germany 273–297 (1998)
7. Schmidt, R., Montani, S., Bellazzi, R., Portinale, L., Grierl, L.: Case-based reasoning for medical knowledge-based systems. International Journal of Medical Informatics 64(2-3) 355–367 (2001)
8. Nilsson, M., Sollenborn, M.: Advancements and trends in medical case-based reasoning: An overview of systems and system development. In: Proceedings of the 17th International FLAIRS Conference, Special Track on Case-Based Reasoning, AAAI 178–183 (2004)
9. Bichindaritz, I.: Case-based reasoning in the health sciences. Artificial Intelligence in Medicine 36(2) 121–125 (2006)
10. Holt, A., Bichindaritz, I., Schmidt, R., Perner, P.: Medical applications in casebased reasoning. The Knowledge Engineering Review 20(3) 289–292 (2007)
11. Bichindaritz, I. An Analysis of Research Themes in the CBR Conference Literature. In *Proceedings of the 9th European Conference, ECCBR 2008*. Berlin: Springer-Verlag 1-17 (2008)
12. Greene, D.; Freyne, J.; Smyth, B.; and Cunningham, P. An Analysis of Research Themes in the CBR Conference Literature. In *Proceedings of the 9th European Conference, ECCBR 2008*. Berlin: Springer-Verlag 18-43 (2008)
13. Lipscomb, C.E.: Medical subject headings (mesh). Bulletin of the Medical Library Association 88(3) 265–266 (2000)
14. Schmidt, R.: Case-Based Reasoning in Medicine Especially an Obituary on Lothar Gierl. In: Advanced Computational Intelligence Paradigms in Healthcare. Volume 48 of Studies in Computational Intelligence. Springer-Verlag, Berlin Heidelberg New York 63–87 (2007)
15. Bichindaritz, I., Kansu, E., Sullivan, K.M.: Case-based reasoning in care-partner: gathering evidence for evidence-based medical practice. In: Smyth, B and Cunningham, P (eds) Proceedings of the 4th European Workshop on CBR, Berlin, Germany, Springer-Verlag 334–345 (1998)
16. Marling, C.R., Whitehouse, P.J.: Case-based reasoning in the care of alzheimer's disease patients. In: Proceedings of the International Conference of Case-Base Reasoning, ICCBR-01 (Lecture Notes in Artificial Intelligence, vol. 2080), Berlin, Germany, Springer-Verlag 702–715 (2001)
17. Bareiss, E.R., Porter, B.W., Wier, C.C.: Protos: an exemplar-based learning apprentice. In: Proceedings of the Fourth International Workshop on Machine Learning, Los Altos, CA, USA, Morgan Kaufmann 12–23 (1987)
18. Grimnes, M., Aamodt, A.: A two layer case-based reasoning architecture for medical image understanding. In: Smith, I and Faltings, B (eds) Proceedings of the Advances in Case-Based Reasoning. Third European Workshop, EWCBR-96, Berlin, Germany, Springer-Verlag 164–178 (1996)
19. Montani, S., Bellazzi, R., Portinale, L., Stefanelli, M.: A multi-modal reasoning methodology for managing iddm patients. International Journal of Medical Informatics 243–256 (2000)
20. Gierl, L.: Icons: cognitive basic functions in a case-based consultation system for intensive care. In: Andreassen, S et al. (eds) Proceedings of the 4th Conference on Artificial Intelligence in Medicine Europe, Amsterdam, Neth, Elsevier Science Publishers B.V. 230–236 (1993)
21. Lieber, J., dAquin, M., Badra, F., Napoli, A.: Modeling adaptation of breast cancer treatment decision protocols in the kasimir project. Applied Intelligence Vol 28 N 3 261-274 (2008)

Towards Evaluation of a Medical Case-Based Decision Support System

Cindy Marling[1], Stan Vernier[1], Tessa Cooper[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@ohio.edu, sv193702@ohio.edu, tc273702@ohio.edu

[2] Appalachian Rural Health Institute, Diabetes and Endocrine Center
College of Osteopathic Medicine
Ohio University, Athens, Ohio 45701, USA
shubrook@ohio.edu, schwartf@ohio.edu

Abstract. A clinical research study is underway to evaluate a medical case-based decision support system in the domain of diabetes management. Thirty patients with type 1 diabetes on insulin pump therapy are participating in the study to evaluate the 4 Diabetes Support System. System evaluation is especially important in medical domains, because systems must demonstrate positive impact on patient outcomes if they are to be used in practice. This study follows a preliminary system evaluation and precedes a full randomized clinical trial to quantify clinical outcomes. An overview of the 4 Diabetes Support System, the evaluation study protocol, and preliminary results of the evaluation are presented.

1 Introduction

In a medical domain, the ultimate success of a case-based decision support system is determined by its impact on patient outcomes. Patients who use a system should have some measurable advantage over those who do not, in terms of health, longevity, quality of life and/or cost of health care. As explained in [1], measuring the clinical outcomes of a system requires a randomized clinical trial, in which some patients use the system and others do not. However, because such a trial is expensive, in terms of time and human resources, the evaluation process may be phased. In the first phase, it is important to thoroughly evaluate the system in terms of its accuracy and usability. This is essential not only to maximize the success of the later trial, but also to minimize any potential negative impact on patients participating in the trial.

The Data-Driven Diabetes Decision Support (4 Diabetes Support) System aims to help physicians manage patients with type 1 diabetes on insulin pump therapy. These patients need to vigilantly monitor their blood glucose levels,

keeping them as close to normal as possible, to avoid serious diabetic complications. Helping these patients make therapy adjustments to combat problems in blood glucose control is a data intensive, time-consuming task for physicians. A preliminary clinical study was conducted in which a prototype of the 4 Diabetes Support System was constructed [2–4]. Because preliminary results were encouraging, a second clinical study is underway to evaluate and enhance this prototype, prior to the conduct of a randomized clinical trial. This paper briefly reviews the 4 Diabetes Support System, describes the evaluation protocol, and presents preliminary findings of the evaluation.

2 The 4 Diabetes Support System

This section gives a brief overview of the 4 Diabetes Support System. The 4 Diabetes Support System prototype was built during a preliminary clinical study in which 20 patients with type 1 diabetes on insulin pump therapy participated. Fifty cases of problems in blood glucose control, with their associated solutions and clinical outcomes, were compiled into a central case base. Specific problems are rich in context and vary widely. However, they revolve around hyperglycemia, hypoglycemia, and fluctuations between the two. Hyperglycemia, or high blood glucose, is responsible for serious long-term complications of diabetes, including blindness, neuropathy, and heart failure. Hypoglycemia, or low blood glucose, may result from insulin treatment to control hyperglycemia. Its effects are more immediate, including weakness, confusion, dizziness, sweating, shaking, and, if not treated promptly, loss of consciousness or seizure. Physicians propose adjustments to therapy as solutions to combat these problems. Adjustments are changes involving insulin, food and/or exercise.

An overview of the system operation can be seen in Figure 1. Patients enter daily glucose, insulin and life event data into an Oracle database via a Web browser. Data from a continuous glucose monitoring system (CGMS) is uploaded directly from the patient's monitoring device to the database. The situation assessment module analyzes the data to detect the problems in glucose control that a patient is experiencing. This module contains 12 problem detection routines, as listed in Figure 2. After detection, these problems are presented to the physician for review. The physician selects the most critical problem or problems for the patient. The next step performed by the system is to retrieve the closest matching case for each selected problem. A standard two-step retrieval method, with a nearest neighbor algorithm, is used, as reported in [3].

Results of the preliminary evaluation were encouraging, but problems were also identified. Participating patients completed an exit survey, which indicated acceptance of the concept of automated decision support. The time required for data entry was noted as a potential impediment to use, however. It took patients between 15 and 60 minutes per day to enter data, and some patients who did not complete the entire protocol cited the time required for data entry as a reason. A panel of three physicians and one advance practice nurse specializing in diabetes reviewed a random sample of problems detected by the situation assessment

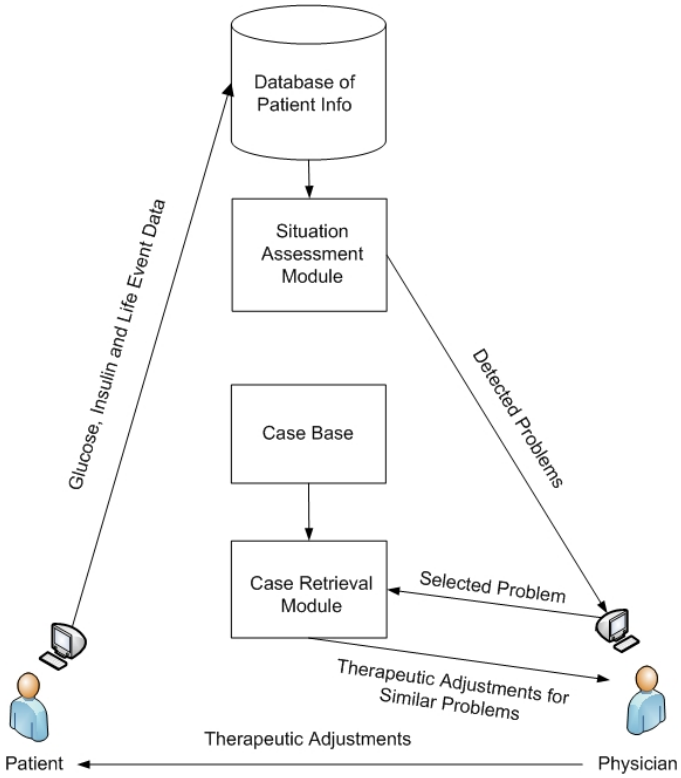


Fig. 1. Overview of System Operation

1. Over-correction for hypoglycemia
2. Post-exercise hypoglycemia
3. Possible pump or infusion set malfunction
4. Over-correction for hyperglycemia
5. Pre-waking hypoglycemia
6. Over-bolus for a meal
7. Hyperglycemia upon awakening
8. Hypoglycemia upon awakening
9. Pre-meal hyperglycemia
10. Pre-meal hypoglycemia
11. Post-meal hyperglycemia
12. Post-meal hypoglycemia

Fig. 2. Situation Assessment Routines

module. They considered the problems to be correctly identified 77.5% of the time, and thought it would be useful to call the problems to the attention of a physician 90% of the time. Leave one out testing was performed to evaluate the case retrieval module. Three diabetes specialists reviewed a random sample of cases with their nearest neighbors. They considered the cases to be similar 80% of the time, and thought that the solutions stored in matching cases would be helpful in solving the original problems 70% of the time. It was noted that not all cases had usefully matching nearest neighbors, and the need to expand the case base was identified. A concern was raised that the test data used was for patients who had contributed data for building the system.

3 The Evaluation Study Protocol

The evaluation study aims to more fully evaluate the performance of the 4 Diabetes Support System prototype, identify future development needs, and reduce the data entry burden for patients. Thirty patients with type 1 diabetes on insulin pump therapy were recruited into the evaluation study. Patients who participated in the initial study were excluded, so that the system could be tested on data for patients whose problems had not been included in the case base.

Each patient visited the Appalachian Rural Health Institute Diabetes and Endocrine Center for an initial visit. At this visit, following informed consent, each patient completed a brief multiple choice inventory designed to gauge the patient's perception of his or her current diabetes control and its impact on quality of life. The HbA1c, a measure of long-term blood glucose control, was obtained. Next, the patient provided background information about his or her current health status, diabetes treatment, and typical daily routines. The times the patient normally awakes, goes to sleep, works or attends school, exercises and eats were recorded.

Each patient was then shown how to automatically transfer data from his or her insulin pump to Medtronic's CareLink data management Web site [5]. CareLink is a commercially available data collection and visualization tool that is free to patients who use Medtronic insulin pumps. It consists of separate, but connected, modules for patients and physicians, CareLink Personal [6] and CareLink Pro [7], respectively. Data is wirelessly transmitted from the patient's pump to CareLink Personal via a USB download device. This data can then be viewed graphically and in log form by the patient in CareLink Personal and by the patient's physician in CareLink Pro. The patient was asked to send data to CareLink once a week for the next four weeks. This data was later extracted, de-identified, and incorporated into the experimental database.

To ease the data entry burden, patients were not asked to input their life events on a daily basis. The simplifying assumption was made that the patient slept, worked, exercised and ate at the times given by their typical daily schedule. Patients were asked to email their physicians with any unusual life events they felt might be impacting their blood glucose levels.

Once a week, data was aggregated and presented for physician review. Problems identified by the situation assessment module were discussed at each weekly review meeting. For each problem detected by the system, the participating patient's physician was asked if it was (a) correct and (b) useful to detect that problem for the patient. Problems that were recognized by the physicians but were not automatically detected by the system were also recorded. These represent future development needs.

Later, the system's ability to retrieve useful cases for the detected problems was tested. This was not done at weekly meetings because solutions are normally sought for only a patient's most critical problems. For thoroughness of evaluation, however, solutions were sought for each type of problem detected for each patient. These were initially reviewed by the knowledge engineering team and are awaiting full review by the physicians.

4 Preliminary Results of the Evaluation

Situation assessment data has been tabulated for the first 20 patients who participated in the evaluation study. Eighteen of these patients completed the full five weeks of observation. Of the two who did not, one was mistakenly enrolled with an incompatible pump type and the other switched pumps midway through the study, losing some of the data. No patients withdrew due to the time required to provide the requested data. This was viewed as a positive finding, because in the preliminary study, 40% of the patients did not complete the entire protocol. Some cited the time required to enter daily data as an impediment to their participation.

The 12 situation assessment routines, listed in Figure 2, found a total of 222 possible problems for the 18 patients. Of the 222 problems detected, 189 were evaluated by the physician of the patient for whom the problem was found. The physicians concluded that 186 of the 189, or 98.4% of the evaluated problem detections were correct and three were incorrect. In regards to whether the detections were useful, physicians rated 181, or 95.8% as useful, six as not useful and two as possibly useful. The details compiled by problem detection routine are seen in Table 1, and by patient in Table 2.

The correctness of the problem detections appears to be better than in the preliminary study. This may be due, in part, to bug fixes, and in part to differences in the way the correctness was evaluated. Only the patient's own physician, who was most familiar with the patient's actual problems, was asked to verify correctness of the problems detected.

It should also be noted that there is no way to accurately determine the number of problems experienced by patients that were not detected by the system. Physicians do not have time to manually detect the large number of problems that could occur in patient data. That is one of the driving factors behind providing automated problem detection in the first place. However, there is reason to believe that many actual problems were missed. The problem detection routines found fewer problems per patient than were found in the preliminary study.

		Correct to Detect			Useful to Detect			
Routine	Total	Yes	No	Maybe	Yes	No	Maybe	Not Evaluated
1	0	0	0	0	0	0	0	0
2	18	15	1	0	15	1	0	2
3	21	10	2	0	10	2	0	9
4	7	7	0	0	7	0	0	0
5	1	0	0	0	0	0	0	1
6	1	1	0	0	0	0	1	0
7	18	18	0	0	18	0	0	0
8	48	40	0	0	40	0	0	8
9	22	18	0	0	15	3	0	4
10	51	46	0	0	45	0	1	5
11	2	2	0	0	2	0	0	0
12	33	28	0	0	28	0	0	5
Totals	222	186	3	0	181	6	2	33

Table 1. Results by Situation Assessment Routine

		Correct to Detect			Useful to Detect			
Patient	Total	Yes	No	Maybe	Yes	No	Maybe	Not Evaluated
1	9	9	0	0	9	0	0	0
3	2	1	0	0	1	0	0	1
4	9	9	0	0	9	0	0	0
5	6	6	0	0	6	0	0	0
7	12	10	0	0	9	0	1	2
8	9	9	0	0	9	0	0	0
9	19	19	0	0	19	0	0	0
10	9	7	0	0	6	0	1	2
11	21	12	0	0	12	0	0	9
12	45	41	3	0	41	3	0	1
13	9	4	0	0	4	0	0	5
14	0	0	0	0	0	0	0	0
15	14	13	0	0	13	0	0	1
16	15	11	0	0	11	0	0	4
17	16	12	0	0	9	3	0	4
18	9	8	0	0	8	0	0	1
19	3	1	0	0	1	0	0	2
20	15	14	0	0	14	0	0	1
Totals	222	186	3	0	181	6	2	33

Table 2. Results by Patient

An average of 12.3 problems per patient were detected in this study versus 29.3 problems per patient in the earlier study. Even pro-rating for the difference in the length of the studies, this study found 50.5% fewer problems per patient [8].

Lack of life-event data appears to account for many missed problem detections. In the preliminary study, patients provided additional life-event data, including their actual daily work and exercise schedules. Because it was time-consuming for patients to provide this data, the current study was designed to determine the impact of using typical daily schedules instead. For all four of the meal related routines and the low after exercise routine, it is disadvantageous not having a patient's actual schedule. Assumptions are made that the patient is exercising every time they plan to, and that recorded carbohydrate intakes within one hour of their usual meal time are actual meals. Carbohydrate intakes recorded at other times are assumed to be snacks. This prevents the system from detecting pre-meal, post-meal, and post-exercise problems when a patient's schedule varies from normal.

While physicians have yet to fully evaluate the case retrieval module of the system, initial observations indicate that the system is not finding matching cases as well as it did in the preliminary study. While CBR, in general, may be robust to missing values, the missing life event data appears to have provided essential context for describing, differentiating, and comparing cases. While generally matching cases are found, more specifically matching cases may be overlooked. Another problem is that the 12 problem detection routines do not account for all of the problems in the case base. Rather, they were developed to account for the most common problems. This effectively makes the case base smaller, as some cases are never good matches for any of the problems detected. While these are negative findings, they do clarify the needs to acquire more life event data, expand the case base, and develop additional problem detection routines.

The system demonstrated its potential benefit when a participating patient was hospitalized with diabetic ketoacidosis. This patient's pump had malfunctioned, so that his insulin was not delivered, and his blood glucose rose. The patient knew that his blood glucose was high, but he did not know that the pump was not delivering the insulin with which he tried to correct his hyperglycemia. He went into diabetic ketoacidosis and experienced an acute coronary event with a silent heart attack. Running retroactively, the system was able to detect the pump problem eight hours before the patient was admitted to the hospital. Had the system been running in real time, it might have been possible to alert the patient to this problem in time for him to correct it.

5 Future Work

The immediate task at hand is to finish the analysis for the current evaluation. Situation assessment data for the remaining ten patients still needs to be tabulated and analyzed. The similarity and usefulness of the cases retrieved needs to be evaluated by a panel of physicians.

Another clinical study is planned to significantly grow the case base and to address the other issues identified during the evaluation. Twenty-eight patients with type 1 diabetes on insulin pump therapy will take part. These patients will automatically upload data from their insulin pumps using Medtronic's CareLink software. However, they will supplement this data with life-event data not stored in their pumps to provide a fuller context for problem solving. They will do this using a shortened and simplified version of the Web browser based interface used in the preliminary study. The longer range goal is for patients to use the technology in their own medical devices and/or cell phones to facilitate data entry.

Once the accuracy and usability of the system is further validated, the next step will be to conduct a multi-site randomized clinical trial. Pre- and post-values of HbA1c, a measure of long-term glucose control, will be used to gauge the efficacy of system use. If measurable improvement in patient outcomes can be demonstrated, the 4 Diabetes Support System may advance beyond the research laboratory into clinical practice.

6 Related Research

This work is anchored within the framework of CBR in the Health Sciences, which is in turn part of a long tradition of research in AI in Medicine [9]. As noted in [1], such work is driven by both the desire to advance the scientific knowledge of AI and CBR and by the real-world needs of patients and health care professionals. For the past six years, workshops on CBR in the Health Sciences have been held at every International and European Conference on Case-Based Reasoning (ICCBR and ECCBR). Several good overview papers have been written on work in the field, including [10–12].

Diabetes management was first identified as a fruitful domain for CBR research by the Telematic Management of Insulin-Dependent Diabetes Mellitus (T-IDDM) project [13–15]. This was a telemedicine project that aimed to remotely monitor and support patients in maintaining good glucose control. T-IDDM was a hybrid system that relied primarily on rule-based reasoning and a probabilistic model of the effects of insulin on blood glucose over time. CBR was integrated to tune rule parameters to optimize advice for patients.

Diabetes shares much in common with other chronic diseases that can not be cured but must be managed or treated over time. Chronic disease management involves consideration of time-varying data, patient variability, and individual patient preferences and needs. Related work has been conducted in the domains of psychiatric eating disorders [16], stem cell transplantation follow-up care [17], end-stage renal disease [18], and Alzheimer's disease [19].

7 Summary and Conclusions

A clinical study of 30 patients with type 1 diabetes on insulin pump therapy was designed to evaluate the 4 Diabetes Support System prototype. This is an

intermediate step between the preliminary evaluation conducted when the prototype was originally built and the randomized clinical trial needed to measure its impact on patient outcomes. Test data was collected from patients who were not involved in the original study in which the case base was built. This helped to ensure that the system was not overfit to the individual test cases, a potential criticism of leave one out testing. User interface issues identified during the original study were addressed. While results are still being tabulated and analyzed, preliminary results have already identified system strengths, weaknesses, and development needs. Evaluation studies are especially crucial for systems developed in medical domains, because positive impact must be demonstrated before systems can move beyond the research laboratory into clinical use. Conducting evaluations that demonstrate impact is difficult for many reasons, including financial constraints, time constraints, and the rapidly evolving nature of software systems [20]. Additional work is needed to define and document practical system evaluation methodologies suitable for CBR in the Health Sciences.

8 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, 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. Bichindaritz, I.: Case-based reasoning in the health sciences: Why it matters for the health sciences and for CBR. In Althof, K.D., Bergmann, R., Minor, M., Hanft, A., eds.: Advances in Case-Based Reasoning: 9th European Conference, ECCBR, Berlin, Springer (2008) 1–17
2. Marling, C., Shubrook, J., Schwartz, F.: Towards case-based reasoning for diabetes management. In Wilson, D.C., Khemani, D., eds.: Workshop Proceedings of the Seventh International Conference on Case-Based Reasoning (ICCBR-07), Belfast, Northern Ireland (2007) 305–314
3. Marling, C., Shubrook, J., Schwartz, F.: Case-based decision support for patients with type 1 diabetes on insulin pump therapy. In Althof, K.D., Bergmann, R., Minor, M., Hanft, A., eds.: Advances in Case-Based Reasoning: 9th European Conference, ECCBR, Berlin, Springer (2008) 325–339
4. 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) (2008) 603–611
5. Medtronic MiniMed: CareLink personal therapy management software for diabetes (2009) https://carelink.minimed.com/patient/, accessed April, 2009.
6. Medtronic MiniMed: CareLink Personal Therapy Management Software for Diabetes: User Guide. Medtronic MiniMed, Northridge, California (2008) Available at http://www.minimed.com/pdf/carelink_user_guide.pdf.

7. Medtronic MiniMed: CareLink Pro Therapy Management Software for Diabetes: User Guide. Medtronic MiniMed, Northridge, California (2007) Available at http://www.minimed.com/pdf/UserGuide7335_ENGLISH.pdf.
8. Miller, W.: Problem Detection for Situation Assessment in Case-Based Reasoning for Diabetes Management. Master's thesis, Ohio University, Athens, Ohio (2009)
9. Altman, R.B.: AI in medicine: The spectrum of challenges from managed care to molecular medicine. AI Magazine **20**(3) (1999) 67–77
10. Nilsson, M., Sollenborn, M.: Advancements and trends in medical case-based reasoning: An overview of systems and system development. In: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference – Special Track on Case-Based Reasoning, Menlo Park, California, AAAI Press (2004) 178–183
11. Holt, A., Bichindaritz, I., Schmidt, R., Perner, P.: Medical applications in case-based reasoning. The Knowledge Engineering Review **20** (2005) 289–292
12. Bichindaritz, I., Marling, C.: Case-based reasoning in the health sciences: What's next? Artificial Intelligence in Medicine **36** (2006) 127–135
13. Bellazzi, R., Larizza, C., Montani, S., Riva, A., d'Annunzio, M.S.G., Lorini, R., Gómez, E.J., Hernando, E., Brugués, E., Cermeno, J., Corcoy, R., de Leiva, A., Cobelli, C., Nucci, G., Prato, S.D., Maran, A., Kilkki, E., Tuominen, J.: A telemedicine support for diabetes management: the T-IDDM project. Computer Methods and Programs in Biomedicine **69** (2002) 147–161
14. Montani, S., Bellazzi, R.: Supporting decisions in medical applications: The knowledge management perspective. International Journal of Medical Informatics **68** (2002) 79–90
15. Montani, S., Magni, P., Bellazzi, R., Larizza, C., Roudsari, A.V., Carson, E.R.: Integrating model-based decision support in a multi-modal reasoning system for managing type 1 diabetic patients. Artificial Intelligence in Medicine **29** (2003) 131–151
16. Bichindaritz, I.: MNAOMIA: Improving case-based reasoning for an application in psychiatry. In: Artificial Intelligence in Medicine: Applications of Current Technologies, Stanford, CA, Working Notes of the AAAI-96 Spring Symposium (1996)
17. Bichindaritz, I., Kansu, E., Sullivan, K.M.: Case-based reasoning in CAREPARTNER: Gathering evidence for evidence-based medical practice. In Smyth, B., Cunningham, P., eds.: Advances in Case-Based Reasoning: 4th European Workshop, Proceedings EWCBR-98, Berlin, Springer-Verlag (1998) 334–345
18. Montani, S., Portinale, L., Leonardi, G., Bellazzi, R.: Applying case-based retrieval to hemodialysis treatment. In McGinty, L., ed.: Workshop Proceedings of the Fifth International Conference on Case-Based Reasoning, Trondheim, Norway (2003)
19. Marling, C., Whitehouse, P.: Case-based reasoning in the care of Alzheimer's disease patients. In Aha, D.W., Watson, I., eds.: Proceedings of the Fourth International Conference on Case-Based Reasoning, ICCBR-01, Berlin, Springer (2001) 702–715
20. Moehr, J.R.: Evaluation: Salvation or nemesis of medical informatics? Computers in Biology and Medicine **32** (2002) 113–125

A Multi-Modal Case-Based System for Clinical Diagnosis and Treatment in Stress Management

Mobyen Uddin Ahmed, Shahina Begum, Peter Funk

School Of Innovation, Design and Engineering,
Mälardalen University, PO Box 883 SE-721 23, Västerås, Sweden
{firstname.lastname}@mdh.se

Abstract. A difficult issue in stress management is to use biomedical sensor signal in the diagnosis and treatment of stress. Clinicians often base their diagnosis and decision on manual inspection of signals such as, ECG, heart rate, finger temperature etc. However, the complexity associated with the manual analysis and interpretation of the signals makes it difficult even for experienced clinicians. A computer system, classifying the sensor signals is one valuable property assisting a clinician. This paper presents a case-based system that assist a clinician in diagnosis and treatment of stress. The system uses a finger temperature sensor and the variation in the finger temperature is one of the key features in the system. Several artificial intelligence techniques such as textual information retrieval, rule-based reasoning, and fuzzy logic have been combined together with case-based reasoning to enable more reliable and efficient diagnosis and treatment of stress. The performance has been validated implementing a research prototype and close collaboration with the experts. The experimental results suggest that such a system is valuable both for the less experienced clinicians and for experts where the system may be seen as a second option.

1 Introduction

Medical knowledge is today expanding rapidly to the extent that even experts have difficulties to follow all new results, changes and new treatments. Computers surpass humans in the ability to remember. This property is very valuable for clinician work and computer-aided system enable improvements in both diagnosis and treatment. Different methods have proven to be valuable in different diagnosis and treatment situation. Especially methods and techniques from Artificial Intelligence (AI) such as case-based reasoning, textual case based reasoning and fuzzy logic have drawn much attention and proven to be useful in solving tasks previously difficult to solve with traditional computer-based methods. Recent advances show that by combining more than one AI methods and techniques increase the potential for clinical decision support systems. The multi-faceted and complex nature of the medical domain often leads to designing of multi-modal systems [11] [13].

Diagnosis and treatment of stress is such an example of a complex application domain. It is well known that increased stress level can lead to serious health problems. During stress the sympathetic nervous system of our body activates causing a decrease in peripheral circulation which in turn decreases the skin temperature and reverse effect (i.e. parasympathetic nervous systems activates) occurs during the relaxation. Thus the finger skin temperature responds to stress [17]. Since there are large individual variations when looking

at the FT, this is a worthy challenge to find a computational solution to apply it in a computer-based system. Case-based reasoning (CBR) is especially suitable for domains with a weak domain theory, i.e. when the domain is difficult to formalize and is empirical. The advantages of CBR in medical domain have been identified in several research works i.e. in [1, 8, 4, 12, 20]. For some applications the integration of CBR and rule-based reasoning have been explored, e.g. in systems like [6, 21]. Cases comprised textual features or textual cases and introducing ontology into the CBR system, to get the advantages, are also implemented in systems such as, in [14, 19]. The use of fuzzy logic in medical informatics has begun in the early 1970s. In fuzzy CBR, fuzzy sets can be used in similarity measure e.g. [5, 7, and 18].

This paper presents a multi-modal and multipurpose-oriented clinical decision support system for stress management. Our previous work in [10] provides a solution for the diagnosis of stress based only on the finger temperature using CBR and fuzzy similarity matching. The system presented in this paper, for stress management, is not only based on the FT sensor data but also considers contextual information i.e. human perception and feelings in textual format. The system applies CBR as a core technique to facilitate experience reuse. Moreover, an effort has been made in this research work to improve the performance of the stress diagnosis task when there are limited numbers of initial cases introducing a fuzzy rule-based classification scheme. Reliability of the performance for diagnosis and decision making is further enhanced using textual information retrieval with ontology. Finally, a three phase CBR framework is also included into the system to assist in treatment i.e. biofeedback training.

2 CBR system for stress management

The construction of multi-purposed and multi-modal medical systems is also becoming a hot topic in the current applied CBR research. Fig. 1 presents the steps to develop a hybrid multi-purpose CBR system to support in diagnosis and treatment of stress-related disorder.

Step 1: Clinical studies show that FT in general decreases with stress and helps to determine stress-related disorders [17]. Analyzing/interpreting finger temperature and understanding large variations of measurements from diverse patients requires knowledge and experience and, without adequate support, erroneous judgment could be made by a less experienced staff.

Step 2 and 3: The measurement is taken from 31 subjects using a temperature sensor in six steps (i.e. Baseline, Deep-breath, Verbal-stress, Relax with positive thinking, Math-stress and Relax) in the calibration phase [10]. Eight woman and twenty three men within the age range of 24 to 51 are participated in this study. The numbers of individual parameters identified and features extracted from the complex data format are briefly presented in section 2.1.

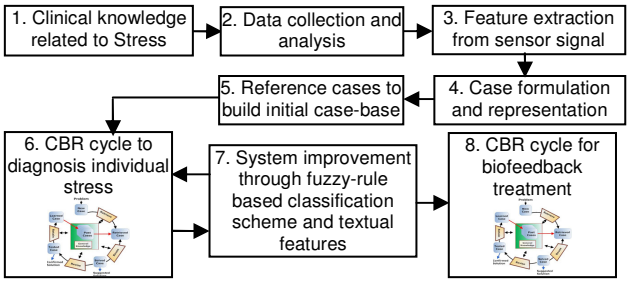


Fig. 1. Schematic diagram of the stress management system in the IPOS

Step 4 and 5: A new problem case is formulated with 19 features in total. The problem description part of a case contains a vector of the features extracted from the FT measurements and the solution part provides a level of stress. The levels of stress are denoted as Very Relaxed, Relaxed, Normal/Stable, Stressed and Very Stressed and the initial case base, with 53 reference cases from 31 subjects, is classified by the domain expert.

Step 6: To diagnose individual stress level, a new FT measurement (formulated as a problem case) is inputted into the CBR cycle. The new problem case is then matched using different matching algorithms including *modified distance function*; *similarity matrix* and *fuzzy similarity matching*. A *modified distance* function uses Euclidean distance to calculate distance between the features of two cases. Hence, all the symbolic features are converted into numeric values before calculating the distance for example, for a feature 'gender' male is converted to one (1) and female is two (2). The function *similarity matrix* is represented as a table where the similarity value between two features is determined by the domain expert. For example, the similarity between same genders is defined as 1 otherwise 0.5. In *fuzzy similarity,* a triangular membership function (mf) replaces a crisp value of the features for new and old cases with a membership grade of 1. In both the cases, the width of the membership function is fuzzified by 50% in each side. Fuzzy intersection is employed between the two fuzzy sets to get a new fuzzy set which represents the overlapping area between them.

$$sim(C_f, S_f) = s_f(m1, m2) = \max(om/m1, om/m2) \quad (\mathbf{1})$$

Similarity between the old case (S_f) and the new case (C_f) is now calculated using equation 1 where *m1*, *m2* and *om* is the area of each fuzzy set. The system can provide matching outcome in a sorted list of best matching cases according to their similarity values in three circumstances: when a new problem case is matched with all the solved cases in a case base (between subject and class), within a class where the class information is provided by the user and also within a subject.

Step 7: A fuzzy rule-based classification scheme [2] and textual features [4] are introduced to provide improved performance in the stress diagnosis task. Detailed information of the system improvement is presented in section 2.2 and 2.3

Step 8: The last step in fig 1 focuses on the CBR system in biofeedback treatment. A three phase CBR framework [3] is deployed to classify a patient, estimate initial parameters and to

make recommendations for biofeedback training. A detailed description on the three phases is given in section 2.4.

2.1 Feature mining from the biomedical sensor signal

An experienced clinician often classify FT signal manually without being pointed out intentionally all the features he/she uses in the classification. However, identifying appropriate features is of great importance in developing a computer-based system. To determine important features the system uses 15 minutes measurements (time, temperature) in 1800 samples, together with other numeric (age, room-temperature, hours since meal, etc) and symbolic (gender, food and drink taken, sleep at night, etc) parameters. After analyzing a number of sample measurements it was found that the FT decreases during the *verbal stress condition* and increases in *relax condition*. In our opinion, either mean value or standard deviation of the FT measurement might not be an indicative for stress. For instance, consider two signals one is increasing from 20 to 30, the other decreasing from 30 to 20, and then both have same mean/standard deviation value in the duration, but indicate opposite for stress levels. As an alternative way, we guess that the mean of the slope value might be a feasible feature to convey relation with stress. If the mean slope is sufficiently positive, it will be an indication of relax, otherwise an indication for stress. But if the mean slope is around zero, it shows a situation with high uncertainty for decision or weak decision. According to closer discussion with clinicians, the derivative of each step of FT measurement (from calibration phase) is used to introduce "degree of changes" as an indication of the FT changes. A low angle value, e.g. zero or close to zero indicates no change or stable in finger temperature. A high positive angle value indicates rising FT, while a negative angle, e.g. -20° indicates falling FT.

Total signal, except the baseline, is divided into 12 parts with one minute time interval and 12 features (i.e. *Step2_Part1*, *Step2_Part2*, *Step3_Part1*, …, *Step6_Part1*, *Step6_Part2*) are extracted. The system thereafter formulates a new problem case combining these generated features and other features namely *start temperature, end temperature, minimum temperature, maximum temperature* and *difference between ceiling and floor.* Also we consider human defined features such as, sex, hours since last meal etc. This new formulated case is then applied in diagnosing and treatment plan of stress by using the CBR cycle.

2.2 Fuzzy rule-base reasoning for creating artificial cases

The cases stored in the case library should be both representative and comprehensive to cover a wide spectrum of possible situations. The composition of the case library is one of the key factors that decide the ultimate performance of a CBR system. Initially, this CBR system has a limited number of available cases which reduces the performance of the system. Therefore, a fuzzy rule-based classification scheme is introduced into the CBR system to initiate the case library, providing improved performance in the stress diagnosis task.

In fuzzy logic, exact reasoning is treated as a special case of approximate reasoning. Everything in fuzzy logic appears as a matter of some degree i.e. degrees of membership function or degrees of truth.

Table 1. Rules for the fuzzy inference system

Fuzzy rules for classification		
Rule no.	**Antecedent**	**Consequent**
	Percentage_Negative_Slope	***Stress_Class***
1.	VeryHigh	VeryStress
2.	High	Stress
3.	Medium	Normal/Stable
4.	Low	Relax
5.	VeryLow	VeryRelax

A single-input single-output Mamdani fuzzy model is implemented in which the *percentage of negative slope* features is taken as the input variable and the corresponding *stress class* as the output. The parameters of the IF–THEN rules (known as antecedents or premise in fuzzy modeling) define a fuzzy region of the input space, and the output parameters (known as consequent in fuzzy modeling) specify a corresponding output as shown in table 1.

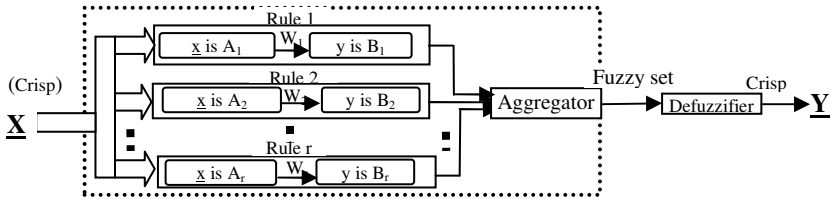


Fig. 2. Block diagram of a fuzzy inference system [9]

The basic structure of fuzzy logic expert systems, commonly known as fuzzy inference system (FIS) shown in Fig. 2, is a rule-based system consisting of three conceptual components: a rule base that consists of a collection of fuzzy IF–THEN rules; a database that defines the membership functions (*mf*) used in the fuzzy rules; and a reasoning mechanism that combines these rules into a mapping routine from the inputs to the outputs of the system, to derive a reasonable conclusion as output. *Percentage_Negative_Slope* and *Stress_Class* are the linguistic variables with the universe of discourse {0, 100} and {1, 5} respectively. VeryHigh, High, Medium, Low and VeryLow are the linguistic values determined by the fuzzy sets "TriangleFuzzySet" on the universe of discourse of *Percentage_Negative_slope*; VeryStress, Stress, Normal/Stable, Relax and VeryRelax are the linguistic values determined by the fuzzy sets "SingletonFuzzySet" on the universe of discourse of *Stress_Class*.

2.3 Textual information retrieval

Clinicians are also considering other factors such as patients feelings, behaviours, social facts, working environments, lifestyle and so on in diagnosing individual stress levels. Such information can be presented by a patient using natural text format and visual analogue scale. Textual data of patients capture important indication not contained in measurements and also provide useful supplementary information. Therefore the system added textual features in the

case vector which helps to better interpret and understand the sensor readings and transferring valuable experience between clinicians [3]. For the textual cases, the *tf-idf* (term frequency–inverse document frequency) [15] weighting scheme is used in the vector space model [16] together with cosine similarity to determine the similarity between two cases. Additional domain information often improves results, i.e., a list of words and their synonyms or a dictionary provides comparable words and relationships within the words using class and subclass. It uses domain specific ontology that represents specific knowledge, i.e., relation between words. The different steps in retrieval of similar case(s) in the system are described in Fig. 3.

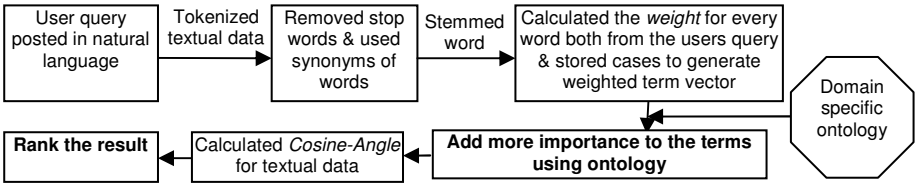


Fig. 3. Different steps for the case retrieval

The text tokenizer algorithm decomposes the whole textual information into sentences, and then into individual words. A filtering step is required to improve retrieval effectiveness due to the huge amount of words. The following three steps are applied to extract the important textual features:

1. Remove the stop-words and special characters blacklist both from the users' query and patients' record.
2. A list of synonyms of the words is used to reduce the number of terms and Porter stemming algorithm helps stemming the words that provide the ways of finding morphological variants of search term. After calculating the weight for each word, these words are represented as terms in a vector space.
3. Improve the importance assessments for candidate terms before measuring the cosine similarity value for the textual information between the stored case and user's query case by using domain specific ontology.

2.4 Biofeedback treatment

The basics of biofeedback is that a patient gets feedback in a clear way (patient observes the graph and knows from preceding education how it should change) and with this feedback can behaviourally train the body and/or mind to biologically respond in a different better way. Biofeedback often focused on relaxation and how the patient can practice relaxation while observing, e.g. the changes in skin temperature. The intention of the system is to enable a patient to train himself/herself without particular supervision. After discussion with clinicians it has been figured out that most of the sensor based biofeedback applications comprised of three phases, 1) analyze and classify patient and make a risk assessment, 2) determine

individual levels and parameters, and finally 3) adapt and start the biofeedback training. If the clinician only uses sensor readings shown on a screen then the classification is highly experience based.

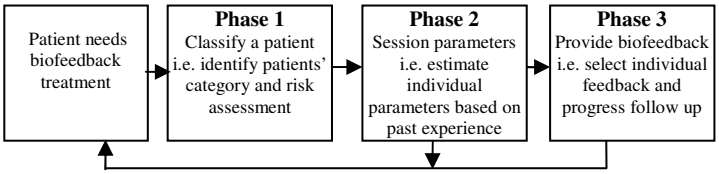


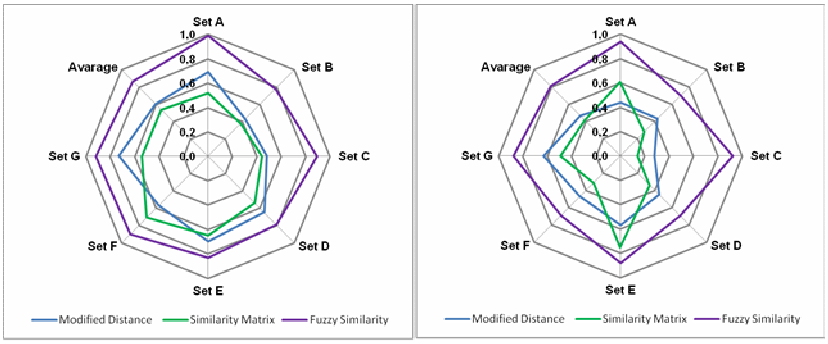
Fig. 4. General architecture of a three-phase biofeedback system

In the first phase as shown in fig 4, a clinician normally asks a number of questions and makes a number of more or less systematic measurements/calculations and then classify a patient depends on the risk and risk-reduction (e.g. stress reactivity and recovery/capacity) of stress. In the second phase, a number of measurements have been done to find out parameters such as, baseline, ceiling, floor temperature etc. needed to tailor the biofeedback session to a patient in order to achieve as good results as possible. Finally, the third phase generates recommendations for a biofeedback training session.

3 Experimental results

The performance of the system has been validated in a prototypical system and close collaboration with experts. Note that, in our previous study, the experiment has been conducted considering 39 cases within the 24 subjects but this paper presents experimental work considering 53 cases from 31 subjects. Moreover, more than one test data sets i.e. for the similarity matching evaluation 7 test sets and for the rest of the evaluation 2 test sets have been considered.

Similarity matching in CBR: Seven subsets of cases and seven query cases, for example: Set A: {7 cases} with query case id 4, Set B: {11 cases} with query case id 16, Set C: {10 cases} with query case id 28, and so on are chosen randomly. All the test sets have been sorted according to the similarity with a query case decided by a domain expert (human reasoning). The sorted cases are then converted to the rank numbers, i.e., the position of a case in the ranking. The top six cases from each set according to the expert's ranking use as standard for the evaluation process where both the similarity values and the ranking numbers are considered. Main goal of this experiment is to investigate the best similarity algorithm for the CBR system compare to the expert's opinion.



a) R^2 value in ranking b) R^2 value in similarity

Fig. 5. Goodness-of-fit both in ranking and similarity to compare three algorithms

Figure 5 illustrates the comparison between the three algorithms (i.e. modified distance, similarity matrix and fuzzy similarity) for the seven test subsets. The goodness-of-fit (R^2), to evaluate both the ranking numbers and the similarity values compare to an expert, is calculated for all the subsets. The value of R^2 lies between the 0 and 1. A value close to 1 indicates that both the algorithm's and expert's propose a similar decision value. As can be seen from figure 5, both in "ranking number" and "similarity value" criteria, fuzzy similarity algorithm is more reliable than the other algorithms. Fuzzy similarity algorithm performs better in all the seven test subsets.

Fuzzy rule-based classification to CBR: the performance of the CBR system depends on the number of available cases in a case library. So, the goal of this evaluation is to see the improvement of the CBR system adding these artificial cases into the CBR library. Experiment has been done by defining two different case libraries as: LibraryA, with real cases only, classified by the expert and LibraryB being twice as big as LibraryA with hybrid cases, classified by the expert and the fuzzy rule-based classification. As shown in table 2, for the two tests (test1 and test2) on an average the LibraryB indicates the classification accuracy 87% while the LibraryA reaches 74% of fitness compared to expert classification. So, there is 13% increase in the R^2 value and 22% (Mean absolute difference) decrease in the error rate when the system uses the LibraryB (hybrid cases) i.e. case library containing enough cases. For the two tests (using two case libraries) the number of correctly classified cases on average is presented in percentage (see 4th column) in table 2. Here, the CBR system can correctly classify 83% using LibraryB whereas using LibaryA the system can only correctly classify 61% of the cases.

Table 2. Comparison results among the case libraries

Average result for test1 and test2	**Goodness-of-fit (R^2)**	**Mean Absolute Difference**	**Correctly classified cases**
LibraryA	0.74	0.38	61%
LibraryB	0.87	0.16	83%

System performance vs. junior clinicians: for the testing purpose an experienced clinician and two junior clinicians (*JC1* and *JC2*) are involved, 2 subsets of cases (setA and setB) are created randomly with the 11 and 14 expert approved cases. The cases of both subsets are classified by the two junior clinicians who have less experience in this domain. The main goal is to see how good the system can classify compare to the junior clinicians i.e. whether the system can be useful to assist the junior clinician in the classification task.

Table 3. Comparison results between the system and junior clinicians for the two test data sets.

Evaluation Method	Test setA			Test setB		
	JC1	*JC2*	*The System*	*JC1*	*JC2*	*The System*
Correctly Classified Cases	64%	55%	81%	57%	57%	79%
Goodness-of-fit (R^2)	0.86	0.88	0.92	0.80	0.81	0.83
Absolute Mean Difference	0.36	0.45	0.18	0.43	0.43	0.28

From the table 3 it can be seen that the system using fuzzy similarity matching algorithm can classify correctly better than all the junior clinicians. The test group SetA with 11 cases, the system classifies correctly 81% and the junior clinicians classify correctly 64% and 55% respectively. The number of the correctly classified cases for setB with 14 cases in percentage is 79 by the system whereas the junior clinicians have succeeded to classify correctly as 57 in percentage. The Goodness-of-fit (R^2) value for both the test groups (setA and setB) are 92% and 83% by the system against the senior clinician, the R^2 values are almost the same or little better than the junior clinicians as 86% and 88% for setA, 80% and 81% respectively for setB. The absolute mean difference or error rates in classification for both the test groups are comparatively lower (0.18 and 0.28) than the junior clinicians.

4 Conclusions

Clinical systems have proven to be able to extend the capability of clinicians in their decision making task. But reliability is often a concern in clinical applications. The system presented in this paper supports a clinician in a number of complex tasks in stress management by combining more than one artificial intelligence techniques where CBR is applied as the core technique. Reliability of clinical systems based on sensor readings could certainly be increased by providing contextual information supporting the reasoning tasks. Therefore, the system considers additional information in textual format applying textual information retrieval with ontology. Here, it is also illustrated that it is possible to increased accuracy in the classification task, by extending the case library with artificial cases. A case study also shows that the system provides results close to a human expert. Today the system is based on one physiological parameter i.e. finger temperature sensor signal, in future several other parameters such as heart rate variability, breathing rate etc. could be investigated as a reference of the work for more reliable and efficient decision support in stress management.

References

1. Holt, A., Bichindaritz, I., Schmidt, R., Perner, P.: Medical applications in case-based reasoning, The Knowledge Engineering Review, Vol. 20:3, (2005) 289 – 292

2. Ahmed, M. U., Begum, S, Funk, P., Xiong, N., Schéele, B. V.: A Three Phase Computer Assisted Biofeedback Training System Using Case-Based Reasoning, In 9th European Conference on Case-based Reasoning workshop proceedings. Trier, Germany (2008)
3. Ahmed, M. U., Begum, S, Funk, P., Xiong, N., Schéele, B. V,: Case-based Reasoning for Diagnosis of Stress using Enhanced Cosine and Fuzzy Similarity, Transactions on Case-Based Reasoning on Multimedia Data, vol Volume 1, nr Number 1, IBaI Publishing, (2008a), ISSN: 1864-9734
4. Bichindaritz, I., Marling, C.: Case-based reasoning in the health sciences: What's next? In Artificial Intelligence in Medicine. 36(2), (2006), pp 127-135
5. Bonissone, P., Cheetham, W.,: Fuzzy Case-Based Reasoning for Residential Property Valuation, Handbook on Fuzzy Computing (G 15.1), Oxford University Press. (1998).
6. Bradburn, J., Zeleznikow: The application of case-based reasoning to the tasks of health care planning. In proceedings of topics in case-based reasoning: 1st European workshop, EWCBR-93. Springer, Berlin, (1994), pp 365–378
7. Dvir, G., Langholz, G, Schneider, M: Matching attributes in a fuzzy case based reasoning. Fuzzy Information Processing Society, pp. 33–36. (1999).
8. Gierl, Schmidt, R: CBR in Medicine. In Case-Based Reasoning Technology, From Foundations to Applications. Springer-verlag. (1998), pp. 273 – 298. ISBN:3-540-64572-1
9. Jang, S, R, Sun, C.T, Mizutani, E: Neuro-fuzzy and Soft Computing. A computional approach to learning and machine intelligence. Prentice Hall, NJ. ISBN 0-13261066-3. (1997)
10. Begum, S., Ahmed, M.U., Funk, P., Xiong, N., Schéele, B.V: A case-based decision support system for individual stress diagnosis using fuzzy similarity matching, In Computational Intelligence (CI), in press, (2008). Blackwell.
11. Montani, S: Exploring new roles for case-based reasoning in heterogeneous AI systems for medical decision support. In Applied Intelligence. 2007, pp 275–285
12. Montani, S., Portinale, L., Leonardi, G., Bellazzi, R., Bellazzi, Case-based retrieval to support the treatment of end stage renal failure patients, In Artificial Intelligence in Medicine 37 , (2006), pp 31-42
13. Nilsson, M., Sollenborn: Advancements and Trends in medical case-based reasoning: An overview of systems and system development. In proceedings of the 17th International FLAIRS Conference, (2004), pp. 178-183.
14. O'Sullivan, D., Bertolotto, M., Wilson, D., McLoghlin, E.: Fusing Mobile Case-Based Decision Support with Intelligent Patient Knowledge Management. In Workshop on CBR in the Health Sciences, 151-160. (2006). ECCBR'06
15. Salton, C., Buckley, Term Weighting Approaches in Automatic Text Retrieval, Technical Report. UMI Order Number: TR87-881., Cornell University 1987.
16. Salton, A. Wong, C. S. Yang, A Vector Space Model for Automatic Indexing, Communications of the ACM, vol.18, nr. 11, 1975. pp 613–620.
17. http://www.iworx.com/LabExercises/lockedexercises/LockedSkinTempNL.pdf, last referred April 2009
18. Wang, W. J: New similarity measures on fuzzy sets and on elements. Fuzzy Sets and Systems, 1997, pp. 305–309.
19. Wu, D.D., Weber, R., Abramson, F. D. A Case-Based Framework for Leveraging NutriGenomics Knowledge and Personalized Nutrition Counseling. In Workshop on CBR in the Health Sciences, 2004,71-80. ECCBR'04
20. Perner, P.: Introduction to Case-Based Reasoning for Signals and Images, In: P. Perner (Ed.) Case-Based Reasoning on Signals and Images, Springer Verlag (2007)
21. Marling, C., Shubrook, J., Schwartz F.: Case-Based Decision Support for Patients with Type 1 Diabetes on Insulin Pump Therapy. In Advances in Case-Based Reasoning: 9th European Conference, ECCBR 2008 Proceedings, Springer, Berlin. 2(008).

Improving the Combination of CBR Systems with Preprocessing Rules in Melanoma Domain

Ruben Nicolas[1], David Vernet[1], Elisabet Golobardes[1], Albert Fornells[1], Susana Puig[2], and Josep Malvehy[2]

[1] Grup de Recerca en Sistemes Intel.ligents
La Salle - Universitat Ramon Llull
Quatre Camins, 2 - 08022 Barcelona
{rnicolas,dave,elisabet,afornells}@salle.url.edu

[2] Melanoma Unit, Dermatology Department
Hospital Clinic i Provincial de Barcelona, IDIBAPS
{spuig, jmalvehy}@clinic.ub.es

Abstract. Nowadays solar exposure habits have caused an important increase of melanoma cancer in the last few years. Mortality rates caused by this illness are the most important ones in dermatological cancers. Despite of it, recent studies demonstrate that early diagnosis improves drastically life expectancy. This work introduces a way to combine different kinds of diagnostic techniques to help experts in early detection. The approach is an improvement of a previous work that combines information of two of the most important non-invasive image techniques: Reflectance Confocal Microscopy and Dermatoscopy. Current work beats the results of the previous system by the support of a set of rules obtained from data preprocessing. This improvement increases the reliability of diagnosis.

1 Introduction

Sun rays and its artificial substitutes are exceedingly appreciated in our society. Actually, they could be healthy with an appropriate protection against excessive tanning. In spite of it, new social habits in solar exposure have increased the appearance of melanoma and other skin cancers (according to the American Cancer Society data). This information and the fact that melanoma holds the highest percentage of death faced with more common dermatological cancers, approximately twenty percent of non early prognosticated cases, makes even more important the early diagnosis. In order to deal with this problem, the most important way is to use non-invasive techniques based on images. In this field stands out two kinds of image analysis: Dermatoscopy and Reflectance Confocal Microscopy [12]. The former is based on the microscopical image created with epiluminiscence and the latter makes the image with the reflectance of a coherent laser with a cell resolution.

This work presents a computer aided system for medical experts in melanoma diagnosis. To catch this aim up we based our effort on solving the problems observed in our previous implemented solution [11]. Our prior approach for this

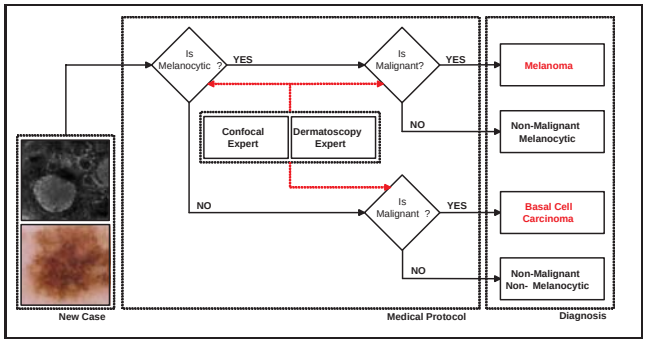


Fig. 1. Protocol followed by medical experts for melanoma diagnosis.

problem focused on the importance of the work in combination of different diagnostic criteria to ascertain the stand out of confocal in the medical protocol. We propose the use of Case-Based Reasoning (CBR) [1] techniques in order to assist the diagnosis. The CBR is a suitable approach because it uses past experiences to solve new cases. It is exactly the same procedure used by experts. Then, we use two independent CBR systems with different types of information in each one in order to obtain a shared prognostic. Moreover, we follow the medical protocol in this kind of decisions which is: to analyze whether the new case is melanocytic and, afterwards, to assess about its malignancy. Thus, the combination of both diagnosis allows experts to determine if the new case is Melanoma, Basal Cell Carcinoma (BCC) or a non-malignant tumor as figure 1 shows. With this first solution we denote that due to its high precision, Confocal microscope allows medical experts to improve its prognostic capacity making up its high economical and temporal cost and the combination of this technique with dermatoscopy allows even a better diagnostic. But this work stresses one difficulty which is that the available attributes are discrete. Considering this characteristic of the data, in our current proposal we try to tackle this property using a preprocessing technique of the data domain that is independent from the medical experts. Eventually, both techniques are combined to create a computer aided system that uses the two CBR modules to classify new cases and fetching obtained rules to combine the classification results. The whole process follows the medical protocol in this kind of decisions and the introduction of the rules guarantees more reliability in the integrated system because all rarities of the data are detected, as we will explain in following sections.

The paper is organized as follows. Section 2 describes some related work. Section 3 describes the new tool obtained by the combination of the different modules. In section 4, experimentation is presented and the performance is analyzed. Finally, section 5 summarizes the conclusions and further work.

2 Related Work

In this work, we would like to use the combination of different decisions in order to classify new melanoma cases. Although there are works focused on studying

the melanoma domain from individual approaches such as in [5], the application of ensemble methods and combination systems has increased in the last years. A line is to improve clustering using ensembles [3]. There are also works to allow the classification using data of different complexity [2] and with different types of medical information [6]. In contrast to these approaches we would like to classify melanoma following the medical diagnosis protocol using different CBR independent classifiers.

In our framework we have two available different points of view (confocal and dermatoscopic) and we select the best classification (the one prognosticated for one of the systems) depending on different criteria. These characteristics do not allow to talk about an Ensemble Learning system but the combination idea is pretty similar to the one used in this kind of methods. Ensemble methods combine the decisions from different systems to build a more reliable solution using the individual ones [9].The combination of approaches can be summarized [4] in: 1) Bagging, 2) Boosting, and 3) Stacking. Bagging and Boosting are based on the combination of the outputs using votes. Particularly, Bagging replicates N systems of the same approach with different data sources. In opposition Boosting defines complementary models. On the other hand Stacking is based on heuristics that combine the outputs of several approaches. As voting methods the most common ones [8] are: 1) Plurality, 2) Contra-Plurality, 3) Borda-Count, and 4) Plurality with Delete. All these methods are based on the number of votes of a class (plurality) but with multiple types of plurality addition and decision of better class.

Attending to the medical necessities and the existing data, it would be interesting to create a combination model with an expert for each kind of data and a final diagnosis opinion. Note that we are not using any of the classical systems of ensemble learning. As we have said, we are not formally using that technique but we are doing an adaptation of different ideas from these models. We modify the use of the data because we are using different attributes of the same data in each system, then the independence of the data is guaranteed, in contrast to the standard Bagging. Analyzing that the classification attributes are boolean, the vote method should be based on plurality but with some arranges requested by medical researchers, who weight more the information from Confocal Microscopy. Once we have experimented the protocol used by experts, we would like to improve it by using rules to do a better system combination.

In previous works in which clustering was used in order to discover new patterns on the medical domain [13] we detected a particular behavior on the data. This characteristic guarantees a correct classification of the new patients when certain conditions in the attributes of the case were detected. Thus, bearing in mind that the main goals are the improvement of the classification and to minimize the false negatives situations, we created the D-Rule module. This module preprocesses the input data and creates a set of rules to help the whole classifier. The module details are explained later on.

A similar idea is shown in [7]. In that work, the attributes of a domain are studied using overlapped intervals and computing the distance between these

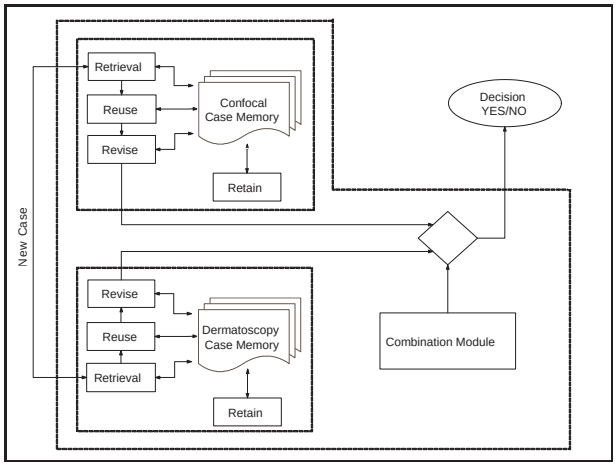


Fig. 2. Combined Decision Schema.

intervals. In this way, a technique called *overlapping of binding box* is presented and it is based on the creation of delimited intervals to define the attributes domain. However, in this paper the preprocessing module is not based on intervals. Concrete values (with useful classification) are detected and encapsulated in a rule. Moreover, our rules do not depend ones on each other, and each attribute is analyzed independently.

3 Improving a Combination of CBR Systems

In this section, we want to describe how a basic system evolves to a reliable tool for medical experts in an action framework in melanoma diagnosis. First of all, the simple combination of CBR systems is presented. Then, we introduce the D-Rule module which is capable to extract a set of characteristics of training data set using a preprocessing algorithm. Finally, we show the result of the combination of both parts.

3.1 Using the Combination of Case-Based Reasoning Systems in Dermatological Cancer

We have developed a computer aided system for melanoma diagnosis based on the medical protocol described in the first section. For each one of the decision points, a CBR system is used to answer the medical question using the knowledge extracted from the Dermatoscopy and the Reflectance Confocal Microscopy image data as Fig. 2 shows. In this first approach [11], we implement as combination module the same protocol used by medical experts.

As we can observe, the global system combines the output of two Case-Based Reasoning (CBR) systems [1]. This is because CBR performs the same resolution procedure that experts: solving new cases through the comparison of previously solved ones. In a general way, the CBR life cycle can be summarized in the next four steps: 1) Retrieving the most similar cases from the case memory with

the assistance of a similarity function; 2) Adapting the retrieved solutions to build a new solution for the new case; 3) Revising the proposed solution and 4) Retaining the useful knowledge generated in the solving process if it is necessary. Thus, the explanation capability is highly appreciated by experts because they are able to understand how decisions are made. Each one of the CBR systems feed from two different case memories which store all the previously diagnosed injuries through the confocal and the dermatoscopy studies respectively. These two parts are completely independent and at the end of its work they put on its vote for the best classification according to their specific data. With this separate ballots, the system creates the final diagnosis (Solution) and, if proceeds, saves the new case in one of the case memories or in both.

The first option tested, as a decision process to perform a diagnosis, is described in figure 3. This approach represents exactly the logical scheme used by the experts. In spite of using a collaborative scheme, where both diagnosis are combined, experts mainly focus on confocal diagnosis and, only if the diagnosis is non conclusive they use the dermatological one. Therefore, the selection of the threshold values used to perform this decision are crucial to achieve a good performance. Both values need to be defined by experts.

3.2 The D-Rule Module

One of the most important problems detected in medical environments is that the majority of attributes are discrete. This characterization of the domain produces a complicated situation when a CBR system is used in these cases. From this idea, we decide to improve our combination of classifiers with an additional tool which allows to evaluate the solution proposed in a reliable way using a set of specialized rules in the discrete attributes. So, in order to achieve this goal, the D-Rule module was designed. Analyzing the training data, a set of several rules are generated by this module. These rules summarize the data complexity in the case memory. The main goal is to represent the existing gaps in the data space with no information associated, in order to advise the classifier in this sense. On the same way, when the correct classification is guaranteed with a high reliability,

Let c_{new} be the new input case
Let $best_{confocal}$ be the most similar case using the confocal CBR
Let $best_{dermatoscopical}$ be the most similar case using the dermatoscopical CBR
Let $distance(c_i, c_j)$ be the distance between two cases c_i and c_j performed by the normalized Euclidean distance
Let $threshold_{confocal}$ be the minimal value to accept two cases as similar from the confocal point of view
Let $threshold_{dermatoscopical}$ be the minimal value to accept two cases as similar from the dermatoscopical point of view
Let *class(c)* be the class of the case c
if $distance(c_{new}, best_{confocal}) < threshold_{confocal}$ **then**
  return $class(best_{confocal})$
else
  if $distance(c_{new}, best_{dermatoscopical}) < threshold_{dermatoscopical}$ **then**
    return $class(best_{dermatoscopical})$
  return $class(best_{confocal})$

Fig. 3. Algorithm to diagnose a new case using the confocal and dermatoscopical criteria with plain combination.

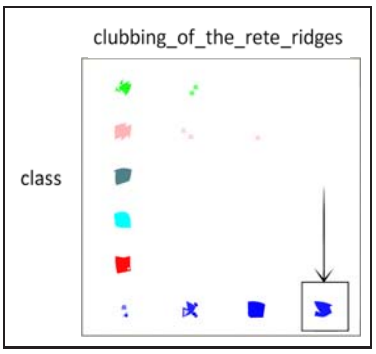


Fig. 4. Example of an attribute considered by the D-Rule module.

an alert to the following classifiers is sent, if a set of characteristics in the input data is detected.

In the example of Fig.4, we can observe a situation where a new rule is generated by the system. Looking at the *clubbing_of_the_rete_ridges* attribute, we notice that a discrete value is defined with all the cases belonging always to the same class. In this case, a new rule could be generated in order to help the CBR systems.

In our domain, it is quite usual that from a certain discrete value of an attribute, the variation of the final decision on the classification does not change. So, the intervals proposed in [7] have been eliminated and clear-cut zones affected always in the same way have been created. These zones are summarized in one or more rules.

The pseudocode used to generate the rules is described in Fig. 5. As we can observe, in the algorithm we use all possible values of an attribute to analyze the input data. One of the advantages found in the medical domain is that the attributes are well delimited, so it is not possible to find a new case with a different value of the predefined ones in the domain.

On the D-Rule module output, a set of $if - then - else$ rules are created to improve the CBR classifiers (see fig 6). The number of defined rules depends on the data complexity and it varies with the different types of classification (Melanoma, Melanocytic or BCC).

3.3 Using Rules in the Combination of CBR Systems

The final approach used in this paper is to implement the combination module based on preprocessing generated rules. Mainly, we follow the same scheme as in the first approach using the best retrieved cases but adding information

```
Let A be the set of all attributes of the medical domain
forall attributes in A do
    Let A_i be the attribute i of the set A
    Let V be the all possible values of attribute A_i
    forall values in V do
        if ∃V_j || class is unique for all cases in training set then
            CreateRule (A,i,V,j,class)
```

Fig. 5. Algorithm to generate rules in the D-Rule system.

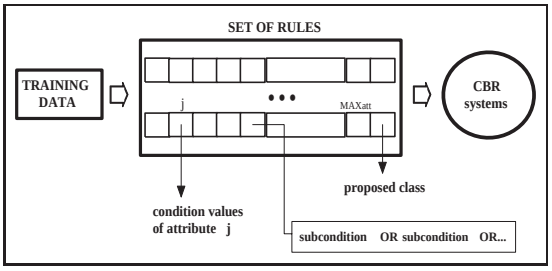


Fig. 6. Representation of the set of rules.

provided by rules. The module works with independent rules obtained from a processing technique applied to training cases. It allows the possibility of automatic and better selection of the best diagnosis. The logic process followed, as a combination, is the one described in Fig.7.

4 Experimentation

This section describes the data extracted from images and analyzes the results of the experiments performed through sensitivity and specificity rates.

4.1 Testbed

The classification of injuries in melanoma domain is not trivial. One of the main difficulties is the huge amount of information that new technologies are able to collect and the ignorance about how they are interrelated [10]. The most used techniques to gather information from tissue are the dermoscopic and the confocal analysis. Experts want to evaluate if confocal analysis detects cases impossible to assess with dermatoscope and if the usage of both techniques can improve the individual analysis.

Nevertheless, a negative point is that the confocal analysis is a long and expensive test, so the number of available cases is limited. Due to this situation,

Let c_{new} be the new input case
Let $best_{confocal}$ be the most similar case using the confocal CBR
Let $best_{dermatoscopical}$ be the most similar case using the dermatoscopical CBR
Let $distance(c_i, c_j)$ be the distance between two cases c_i and c_j performed by the normalized Euclidean distance
Let $numRules_{confocal}$ be the number of rules carried out by $best_{confocal}$
Let $numRules_{dermatoscopical}$ be the number of rules carried out by $best_{dermatoscopical}$
Let *class(c)* be the class of the case c
if $numRules_{confocal} > numRules_{dermatoscopical}$ **then**
  return $class(best_{confocal})$
else
  if $numRules_{confocal} < numRules_{dermatoscopical}$ **then**
    return $class(best_{dermatoscopical})$
  else
    if $distance(c_{new}, best_{dermatoscopical}) < distance(c_{new}, best_{confocal})$ **then**
      return $class(best_{dermatoscopical})$
    else
      return $class(best_{confocal})$

Fig. 7. Algorithm to diagnose a new case using the confocal and dermatoscopical criteria and combining the results by rules.

Table 1. Classification accuracy using only confocal images, only dermoscopic images, both images with plain combination, and both images with rules combination.

	Melanoma	Melanocytic	BCC
Only Confocal Images	87%	90%	96%
Only Dermatoscopy Images	90%	98%	95%
Confocal and Dermoscopic Images with Plain Combination	89%	96%	95%
Confocal and Dermoscopic Images with Rules Combination	94%	99%	99%

the data set used in this work is composed only by 150 instances of suspicious lesions. All instances contain information related to confocal and dermatoscopic images and the histology corroborated diagnosis. Attending to the considerations of the medical experts that have created this set, it includes enough cases from each kind of illness to be representative of the domain. Then, in medical terms it is an appropriated case memory for this study. Detailing the instances, dermatological information has forty-one fields and confocal microscopy, due to its higher resolution, contributes with data from eighty-three different attributes.

4.2 Experimentation Framework

We have tested the classification accuracy of the platform proposed with a basic decision combination and with the use of rules obtained through the use of preprocessing algorithms (as has been explained). In addition, we tested the accuracy of the two independent CBR systems (one for confocal data and another for dermatoscopy). This information is complemented with the analysis of the sensitivity and the specificity for the different cases. In the case of the plain combination platform, we have tested different decision thresholds according to medical experts criteria. The final consensus is to use 0.5 as confocal threshold and double of the distance between new case and best confocal case as dermatological one. All the CBR systems used in experimentation are configured with one-nearest neighbor algorithm with normalized Euclidean distance as retrieve function. This experiment framework has been tested applying a leave-one-out to the original data to obtain the average accuracy of those systems.

4.3 Results and Discussion

Analyzing these results, table 1 shows the accuracy rate to classify new injuries. This analysis is done in the three possible classes (Melanoma, Melanocytic and BCC) and using only confocal image, only dermoscopic image and both images with and without rules. The results obtained highlight two points: firstly, the combination of systems (even without rules) obtain a significatively better classification results, or at least an equivalent one (tested with t-test at 95% confidence level) than the use of only one kind of data. Secondly, the results are even better if we apply the preprocessing obtained rules to the combination module. In this way, we could see that the use of rules improve a minimum of 3% the accuracy in comparison with the plain combination. In all kind of classifications

Table 2. Sensitivity results using only confocal images, only dermoscopic images, both images with plain combination, and both images with rules combination.

	Melanoma	Melanocytic	BCC
Only Confocal Images	73%	94%	81%
Only Dermatoscopy Images	73%	99%	92%
Confocal and Dermoscopic Images with Plain Combination	70%	96%	92%
Confocal and Dermoscopic Images with Rules Combination	81%	100%	92%

Table 3. Specificity results using only confocal images, only dermoscopic images, both images with plain combination, and both images with rules combination.

	Melanoma	Melanocytic	BCC
Only Confocal Images	92%	81%	99%
Only Dermatoscopy Images	96%	98%	95%
Confocal and Dermoscopic Images with Plain Combination	95%	96%	96%
Confocal and Dermoscopic Images with Rules Combination	98%	98%	100%

analyzed, the increase of accuracy using rules is significant.On the other hand, tables 2 and 3 summarize the results of analyzing the statistics from the point of view of specificity and sensitivity. They show that is more reliable to do a prognostic of real negative cases than the positive ones. This happens because data sets are unbalanced, what means that, they have different number of cases of each type because data sets represent a real situation: there are more healthy people than sick people. Despite of it, the use of the combination of both types of images with the help of preprocessing obtained rules allows an important increase of sensitivity and specificity rates (in addition to improve the accuracy). It is important to highlight that, in melanoma classification, we reduce from 10 to 2 the false negatives cases and from 10 to 7 in the case of false positives. It is the most important result that medical experts would like to obtain in this kind of classification. In addition, in Basal Cell Carcinoma classification we reduce a 60% the rate of false negatives and a 100% in the case of melanocytic class. This decrease of the false negative cases (even the decrease of false positives produced too in all cases) using the combination with rules is extremely interesting and valued by medical experts.

5 Conclusions and Further Work

Melanoma early diagnosis is one of the main goals in dermatology. The diagnosis process with non-invasive techniques is complex because of the data typology. We propose a platform for automatizing the medical protocol followed in order to diagnose dermatological cancer. The proposal combines information from two promising techniques based on images through a combination algorithm based on experts' experiences. In addition, we use preprocessed rules in order to improve this combination. After the results analysis of testing melanoma data set, we can

conclude that the combination of both images improves the individual results and that by using rules we obtain even better accuracy.

The further work is focused on two lines. The first one is the improvement of the systems combination with new types of rules or other combination techniques. Secondly, the use of preprocessed rules to extract useful diagnostic patterns for medical experts.

Acknowledgments

We would like to thank the Spanish Government for supporting the MID-CBR project under grant TIN2006-15140-C03, the Generalitat de Catalunya for the support under grants 2005SGR-302 and 2008FI_B 00499, and La Salle for supporting our research group. The work performed by S. Puig and J. Malvehy is partially supported by: FIS, under grant 0019/03 and 06/0265; Network of Excellence, 018702 GenoMel from CE.

References

1. A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.
2. R. Abdel-Aal. Abductive network committees for improved classification of medical data. In *Methods of Information in Medicine*, 2004.
3. R. Avogadri and G. Valentini. Fuzzy ensemble clustering for DNA microarray data analysis. In *Lecture Notes in Computer Science 4578*, 2007.
4. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.
5. A. Fornells, E. Armengol, E. Golobardes, S. Puig, and J. Malvehy. Experiences using clustering and generalizations for knowledge discovery in melanomas domain. In *Advances in Data Mining. Medical Applications,E-Commerce, Marketing, and Theoretical Aspects*, volume 5077, pages 57–71. Springer, 2008.
6. A. Fornells, E. Golobardes, E. Bernadó, and J. M. Bonmatí. Decision support system for breast cancer diagnosis by a meta-learning approach based on grammar evolution. In *Eighth International Conference on EIS*, pages 222–229, 2006.
7. T. Ho, M. Basu, and M. Law. Measures of complexity in classification problems. In *Data Complexity in Pattern Recognition, pp. 1-23, Springer*, 2006.
8. K. T. Leung and D. S. Parker. Empirical comparisons of various voting methods in bagging. In *Ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 595–600. ACM Press, 2003.
9. P. Melville and R. J. Mooney. Diverse ensembles for active learning. In *ICML 04: Twenty-first international conference on Machine learning*, 2004.
10. R. Nicolas, E. Golobardes, A. Fornells, S. Puig, C. Carrera, and J. Malvehy. Identification of relevant knowledge for characterizing the melanoma domain. In *Advances in Soft Computing*, volume 49 2009, pages 55–59. Springer Berlin-Heidelberg, 2008.
11. R. Nicolas, E. Golobardes, A. Fornells, S. Segura, S. Puig, C. Carrera, J. Palou, and J. Malvehy. Using ensemble-based reasoning to help experts in melanoma diagnosis. In *Frontiers in AI and App., vol 184*, pages 178–185. IOS Press, 2008.
12. A. Scope and al. In vivo reflectance confocal microscopy imaging of melanocytic skin lesions. *J Am Acad Dermatol*, 57:644–658, 2007.
13. D. Vernet, R. Nicolas, E. Golobardes, A. Fornells, C. Garriga, S. Puig, and J. Malvehy. Pattern Discovery in Melanoma Domain Using Partitional Clustering. In *Frontiers in AI and App., 184, IOSPress*, pages 323–330, 2008.

Towards an Introspective Architecture for Meta-level Reasoning in Clinical Decision Support Systems

Tor Gunnar Houeland and Agnar Aamodt

Department of Computer and Information Science,
Norwegian University of Science and Technology,
NO-7491 Trondheim, Norway
{houeland,agnar}@idi.ntnu.no

Abstract. The paper presents core elements of a meta-level architecture for clinical decision support, in the domain of palliative care. The goal of the reported research is to develop an architecture and an integrated set of methods for an introspective meta-level reasoner. Within the architecture a system is under development that addresses the identification and utilization of clinical guidelines for the assessment and treatment of cancer pain. Case-based reasoning is a core component of the architecture, which also incorporates rule-based and probabilistic model-based methods. The paper presents the overall architectural constraints and exemplifies parts of it through structured component descriptions.

1 Introduction

A clinical decision support system that covers several clinical tasks, such as patient examination, disease hypotetization, diagnosis determination, treatment planning, and drug administration, would typically need to combine several types of knowledge and several reasoning methods to provide good advisory support. There has recently been a renewed interest in meta-level reasoning as a means for a system to effectively combine several reasoning methods [1].

The focus of the research presented here is on meta-level reasoning for clinical decision support. A component-based architecture and an integrated set of methods for a meta-level reasoner to improve its reasoning and learning abilities are currently being developed. The architecture allows for the integration of all three major reasoning paradigms in symbolic AI, i.e. rule-based, case-based, and (deeper) model-based reasoning.

We are addressing this problem in the domain of clinical decision support for palliative care. In a cooperation with the Palliative Medicine Unit, Cancer Department, St. Olavs University Hospital in Trondheim, we are studying the potential for proactive, advice-giving systems to improve treatment of pain in cancer patients. In our current project, short-named TLCPC, the focus is on lung cancer. This research is tightly linked to a larger EU project called EPCRC [2], which covers all forms of cancer pain as well as other problems related to

palliative care for long-term cancer patients. A motivation for that project is also to standardize procedures and to unify clinical practice in palliative care [3], a goal for which computerized decision support systems have a strong potential. The system under development will in particular address the identification and utilization of clinical guidelines.

In the next section we review some of the related research in metareasoning with case-based components, and CBR used for guideline supported clinical decision-making. In section 3 our metareasoning approach is introduced. Essential components of the architecture are discussed in section 4. The status of the clinical guideline application is presented in section 5. Concluding remarks end the paper.

2 Related Research

There is a significant amount of research that has addressed metareasoning in relation to CBR systems. In Meta-AQUA [4, 5] introspection is used in the retain phase to learn from mistakes. What is referred to as *introspective meta-XPs* are used to represent failures encountered while the system operates. The system constructs learning plans that consist of calling various learning algorithms.

Early accounts of meta-level architectures involving CBR also include BOLERO [6] and ANALOG [7]. In BOLERO, a case-based meta-level planner controls the execution of a rule-based reasoner designed to accomplish different medical diagnosis tasks. New plans are constructed during the problem solving process when needed, and captured as new cases by the meta-level learner. In ANALOG a selected method runs until an impasse situation is encountered, at which time a new method selection process is run. The meta-level learner remembers failed and successful instantiations of methods.

Christodoulou and Keravnou describe a meta-level architecture [8] in the domain of breast cancer histopathology. Each problem solver is associated with a task, an inference mechanism, and domain knowledge constraints. A set of meta-parameters is used by the metareasoner to characterize knowledge types (e.g. experience-based, causal), and desired solution properties (e.g. level of detail, accuracy, efficiency). The metareasoner is a case-based reasoner that captures and stores problem-solving paths as strategy cases incrementally.

In the ADAPtER system [9], an architecture that combines model-based and case-based reasoning for medical diagnosis, the CBR component is the primary object level reasoner. The model-based method is triggered either when the retrieval method fails to find a matching case, or the case adaptation module fails. Cases are captured as a compilation of the model-based process.

The principles of evidence-based medicine, in which systematic research evaluation regimes are used to assess and justify results from medical research, form a well-established basis for medical practice [10]. One manifestation of this is the specification of clinical guidelines, i.e. operational procedures for how to conduct a particular type of patient examination, make a diagnosis, administer a type of drug, or perform other types of treatment. Another source of information used

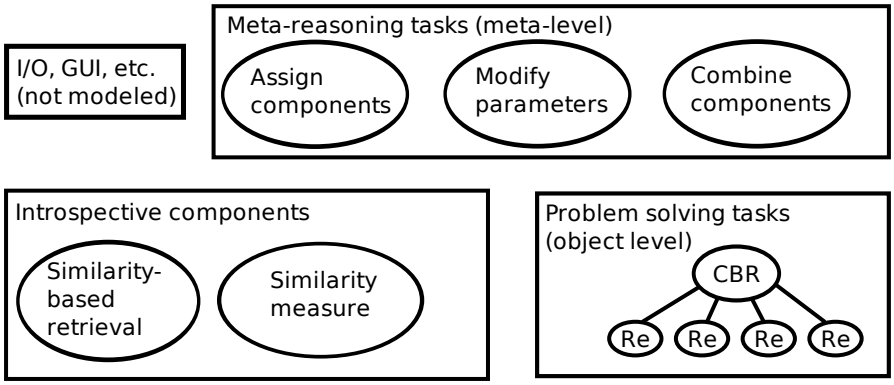


Fig. 1. A CBR system based on the introspective architecture

by clinicians in their daily practice is, of course, the set of experiences a clinician has from earlier patients. Hence, a combination of general guidelines with past experience cases is potentially a strong combination. Identifying the strengths and weaknesses of a case-based method vs. a generalization-based one can be done analytically or experimentally. Marling et. al. [11] reports on an experiment in the domain of nutritional menu planning, in which a hybrid system was developed by combing the strengths of separate rule-based and case-based reasoners. The hybrid system outperformed both separate systems.

In the CARE-PARTNER system [12] medical guidelines are realized in the form of problem solving pathways, implemented as a rule-based system combined with information retrieval, and with CBR as the main problem solving method. GLARE [13] is a general decision-support system for managing and utilizing clinical guidelines, in which guidelines are represented as a hierarchical structure of decision paths. A CBR system was incorporated as additional knowledge to handle situations that are not covered by the guidelines, so-called non-compliances [14]. Even if several medical guideline systems combine different reasoning paradigms, there has been very little work on moving the combined reasoning up to the metareasoning level.

3 Metareasoning Approach

Recently, Cox and Raja [1] presented a general high-level framework of metareasoning. It relates three different levels, the ground level (physical perception and action), the object level (reasoning about action), and the meta-level (reasoning about reasoning). Leake and Wilson [15] address the learning part of metareasoning within this framework, and present a set of challenging issues, such as learning for self-understanding and self-explanation.

A system using our introspective architecture with CBR reasoning methods which follows this general framework is shown in figure 1. The ground level is represented as I/O for the system, which is not covered in our architecture but

must necessarily exist in some form for a complete CBR system. In the shown system reasoning tasks are explicitly represented, with a "CBR-based problem solving task" at the object level and several metareasoning tasks.

In our architecture there is no fundamental difference in how reasoning tasks for the two levels must be represented, but they are shown as seperate boxes in the figure to ease understanding and fit the metareasoning framework of Cox and Raja. The importance of our introspective architecture lies in the components, each of which implements a specific functionality, and is modified and assigned to a particular reasoning task by meta-level reasoning components.

Many meta-level reasoning systems start by building a partial solution and then invoking meta-level reasoning functionality to determine whether the object-level reasoning processes are working satisfactorily or not. Such systems then either proceed as normal, or classify the reasoning process as a failure and create a new metareasoning goal to learn from this failure.

This is in contrast to the meta-level control agent in our approach, which operates on the object-level reasoning components directly without a specific reasoning failure to address. This allows the system to have a clearer broad focus on performing changes that affect the entire reasoning system instead of correcting single failures, and this broadening is also identified by Leake and Wilson [15] as an important opportunity for a more flexible learning focus. Future planned meta-level components in our approach such as a competence-evaluator for problem-solving methods can also assist in providing the system with a level of self-understanding, which is another identified opportunity for introspective learning systems.

However, this illustrates an interesting trade-off compared to other approaches. In our approach the meta-level components are explicitly meant to affect the system's reasoning processes, and the architectural design has been created to make this feasible and as easy as possible. But we do not have a so-called narrow focus on repairing specific portions of the system [15], and in practice this means that our approach is not currently as well developed for making these repairs as some other systems. Due to our focus we do not yet have a meta-level component dedicated to detecting and repairing failures during object-level reasoning, and as such the individual problem solving processes are only improved indirectly by selecting the best available components. Unlike several other systems, if this competence evaluation is incorrect it will not be detected by our current set of meta-level components during the problem solving process itself, and the lessons learned from this failure will only be available for subsequent problem solving instances.

Richter [16] introduced the knowledge container model, where pieces of knowledge can be categorized into four categories: case vocabulary, cases, similarity assessment and adaptation knowledge. Given that a set of precise and consistent terms is important for reasoning at the meta-level, we explicitly include the CBR system vocabulary as part of the case vocabulary category.

To make sure that the vocabulary contains semantics that are useful for human designers, we suggest that the vocabulary should be created and updated

manually, unlike the other knowledge sources which will be changed and influenced by the meta-level reasoner. In our architecture the case knowledge is stored in cases, similarity assessment as part of the components providing methods for object-level retrieval and the adaptation knowledge is shared between object level reuse components and a more profound system-level adaptation in meta-level components.

For the meta-level control agent to be able to examine and calibrate the system's components, it's important that the settings available can be interpreted by the control program, and that the components expose interfaces at an appropriate level of abstraction. To facilitate this, we are developing a vocabulary of CBR-related terms and the intended semantics as part of our approach.

An important aspect of the vocabulary is that it describes what the terms mean at a semantic level without relying on the specific realizations in any particular CBR system implementation. An example of this is a case base, which is simply defined as a set of cases. While in practice many CBR systems store the cases as a form of ordered lists, their reasoning processes typically do not rely on the particular order the cases are listed in within computer memory or on disk. The purpose of the vocabulary is to abstract away the particular specifics in implementations and generalize the terms to a semantic level where it describes the actual requirements for a term without imposing a needlessly specific design.

Our vocabulary is being developed to be conceptually compatible with the terms used in CBROnto [17], a modeling framework that has already examined the meanings and relations between terms from an ontological perspective. As an added benefit the CBROnto ontology is compatible with the "4 REs" CBR process cycle [18] which is widely quoted and referred to in the CBR literature.

4 Introspective Architecture

A particularly important part of our semantic task description is the type system for component inputs and outputs. These types represent at the knowledge level the essential meaningful content that is to be processed by each component.

In our conceptual framework we consider the traditional core case-based reasoning process as one possible problem solving method at the object level. We consider this to be one part of the combined reasoning system, with specific responsibilities, and there can be potentially many other problem solving methods implemented in the same architecture.

A high-level task decomposition of the CBR process for our architecture is shown in figure 2, where the task "Case-based reasoning" is split into the 4 RE subtasks, and each of these are further split into smaller subtasks. Any components that are assigned to parts of this process must match the semantic input and output types for the corresponding tasks, which are also included in the figure.

To elaborate on the case-based retrieval task, it starts from a **Problem-Input** and retrieves a set of cases **Set**<*CB*> where *CB* is a type adhering to the **Case** semantics, and is further subdivided into subtasks that further specify

Task	Input	Output
Case-based reasoning	**ProblemInput**	**ProblemSolution**
- Retrieve	**ProblemInput**	**Set**<*CB*>
– Problem characterization	**ProblemInput**	*PC*
– Case retrieval	*PC*	**Set**<*CB*>
– Focus	**Set**<*CB*>	**Set**<*CB*>
- Reuse	(*PC*, **Set**<*CB*>)	**ProblemSolution**
– Adapt solution method	(*PC*, **Set**<*CB*>)	**SolutionMethod**
– Adapt solution	(*PC*, **SolutionMethod**, **Set**<*CB*>)	**ProblemSolution**
- Revise	**ProblemSolution**	**(PS, SE)**
– Evaluate solution	**ProblemSolution**	**SolutionEvaluation**
– Repair solution	**(PS, SE)**	**ProblemSolution**
- Retain	**(PS, SE)**	
– Update general knowledge	**(PS, SE)**	
– Add to case base	**(PS, SE)**	

(PS, SE) is an abbreviation for **(ProblemSolution, SolutionEvaluation)** due to space concerns.

Fig. 2. Inputs and output for the CBR tasks

how this is performed. It starts with a subtask identifying the important aspects of the problem and characterizing it as a query, transforming the **ProblemInput** into an intermediate form *PC*. This intermediate form is not restricted by the architecture, and the only requirement is that the methods performing the subsequent CBR subtasks can accept the characterization form *PC* as input.

This characterization is used to retrieve a number of previous cases from the case base, and then this is further narrowed to just focus on the most relevant information by e.g. filtering out a subset of cases or generalizing cases. The other subtasks are formalized in a similar way.

In our approach, the metareasoning components exist separately from the object-level CBR reasoning components, and in fact influence and controls how the CBR problem solving method is performed, which corresponds to the aforementioned metareasoning cycle [1].

By evaluating how the CBR method performs while solving actual new problem instances, the meta-level control agent can identify the strengths and weaknesses of the current system and attempt to use this to improve the system's competence or use alternative reasoning methods. For our meta-level component-combiner this is achieved by attempting to re-solve problems using different assignments of methods for each of the tasks and subtasks in the architecture. Whether the the newly combined reasoning process is an improvement is then evaluated based on whether the solutions produced for individual problem queries are correct for more problem instances than before.

4.1 Architectural components

One of the most important features of this architecture is that each component contains extra structures that semantically describe the component semantically using our CBR vocabulary. It is on the basis of this added information that the meta-level control component can automatically assign components to perform the system's reasoning tasks. Figure 3 shows such a self-describing semantic structure for an example object-level retrieval component.

Each component contains a list of types that apply for the component, which is listed first in the figures. This can either just be a single *name* to indicate that any type is supported, or a statement of the form "*name* isa *supertype*", which indicates that the *name* type must support the same operations as *supertype*. This is useful when a component performs a generic operation that can apply to many different types of input, and e.g. only requires that the input and output types are the same or that two inputs are comparable. This section is empty for simpler components that only refer directly to specific types in the vocabulary, such as the illustrated example retrieval component.

After that the input and output variables are listed. Each variable consists of a line of the format "*name*: *type*", where *name* is an identifier that is used to refer to the input or output throughout the component specification and *type* is the semantic type, which can either be a specific from the vocabulary or one of the types specified in the component's Types section. An example of this is the line "*query*: **AttributeCase**" from the example component, which means that the component receives an input of type **AttributeCase** which is referred to as *query* in the component description.

The following section lists Conditions that have to be fulfilled for the component to produce the expected results. As long as the input variables conform to the specified conditions, the output variables are guaranteed to follow the specifications in the Guarantees section. While the type specifications are semantic restrictions on what kind of operation the component can perform, the conditions specify for which values of inputs the component will actually behave as intended, and these conditions are not necessarily checked by the component.

Because of this the component's operation can usually be performed for non-conforming inputs, but this can produce undefined results and should be avoided. By listing the conditions in the self-describing structure, the metareasoning component can make sure that only compatible components are combined, or that the conditions are checked on-demand before the operation is performed where this cannot be guaranteed.

The final section is a short semantic description of the core functionality performed by the component. This has the format "x based on y", where x and y are statements composed using the input and output variables as well as a set of pre-specified terms representing important concepts related to the reasoning system and the application domain.

By combining the example component in figure 3 with a component providing similarity assessments, the meta-level control agent can create a new component that accepts a query and casebase of type **AttributeCase** and returns a ranked

Types:	
Input:	*query*: **AttributeCase** *casebase*: **Set<AttributeCase>** *similarityfunction*: (**AttributeCase** *source*, **AttributeCase** *target*) → **Similarity**
Output:	*ordered*: **List<AttributeCase>**
Conditions:	$5 \leq \text{sizeof}(casebase) < 100000$ $0 \leq similarityfunction(x, y) \leq 1$
Guarantees:	$\text{sizeof}(ordered) = 5$
Approach:	order(*casebase*) based on *query*

Fig. 3. An example component that orders a case base based on similarity to a query using a specified similarity function.

list of matching cases. This is done by verifying that it fulfills the semantic type requirements, and any conditions that cannot be guaranteed to be always be fulfilled can be verified at run-time in an implemented system.

If the similarity measure is based on e.g. feature weights, the meta-level learning component can also further refine this into a component that learns the feature weights automatically while it is being used. This can be done by matching it with an appropriate learning method that takes a case base (**Set<AttributeCase>**) as input and produces a similarity importance value for each feature among cases in the casebase.

5 Palliative Care Application

Although there has been a lot of focus on developing clinical guidelines within the medical community, and several guideline systems have been developed by medical professional organizations, there is not a consensus as to what is a good guideline system. Further, the active use of guidelines in a clinical setting is far from the level desired, both from the perspective of quality of treatment and the perspective of unified treatment across hospitals and countries.

In our partner project EPCRC, being a collaboration involving many highly influential medical groups in across Europe, the aim is to reach a consensus on a set of high-quality and operational guidelines that will be used in practice. While awaiting the results from EPCRC, we currently work with an existing set of guidelines defined by NCCN (National Comprehensive Cancer Network) in the US. An ontology is currently being built based on a combination of generic UMLS terms combined with terms from the SNOMED and NCI (National Cancer Institute in the US) ontologies.

The top-down design process is combined with bottom-up experimental system building, starting from simple system components that will be combined. Currently a simple a rule-based reasoner is being implemented at the object level. Example guidelines link patient data related to pain level, pain history, and history of treatment, to the next treatment. Type of treatment considered

is the administration of different types of analgesics, with opioids as the largest subclass. Based on the results from the initial model, the system will be extended with case-based and model-based components. The model-based component will a be Bayesian network for reasoning about causality under uncertainty.

Acquiring the necessary medical knowledge needed for our experiments is a continuing process. To advance the method development we are developing a toy example system in the domain of advice-giving for film selection, in parallel to the more complex clinical guidelines system. In that system CBR methods are currently focused on both the meta and object levels. Cases, problems and solutions are currently represented as feature vectors, and there are no explicitly represented general knowledge structures outside of the CBR reasoning components. To predict a movie rating for a user, a retrieval component retrieves a number of similar other users that have seen the movie, where similarity is determined as the average difference in ratings for movies both users have rated. A simple reuse component then copies the majority rating among the retrieved cases. The system has a meta-level control agent that adheres to the principles of the metareasoning approach described earlier. Component combinations are tried out based on matching the input and output type descriptions. Although simple and different from the clinical application, the system assists in the bottom-up specification of the introspective architecture by providing a testbed for experiments.

In a clinical guideline support system past cases may be utilized in several ways. The role we have intended for the cases in our guideline system is twofold. Having arrived at a leaf node in the guideline structure, CBR will be used to continue from there by providing a more specific and detailed advice, based on adapting a past result. On the other hand, if the guideline system cannot provide reasonable advice, CBR is triggered as a complementary method. The latter approach is similar to the non-compliance method [14] referred to earlier.

6 Conclusions

In this paper we have presented an introspective approach to meta-level learning and outlined a component-based architecture for designing reasoning systems which supports our introspective methods. The main contribution of our approach is the way our architecture allows for gradual additions of metareasoning methods that focus on broad, system-wide improvements.

We are working on further developing this architecture, and adding new components directed towards both object-level and meta-level reasoning methods for a clinical decision support system based on medical treatment guidelines.

Acknowledgements

This research is supported by Norwegian Research Council (NFR) grant, Contract no 183362, Translational Research in Lung Cancer and Palliative Care (TLCPC), together with funds from the Norwegian University of Science and

Technology (NTNU). Thanks to Tore Bruland and Helge Langseth who contributed on the AI method side through a series of discussions, Sunil Raja together with Heidi Knobel who have provided insight into clinical pain assessment and treatment, and Stein Kaasa who is the initiator and project leader of both the TLCPC and EPCRC projects.

References

1. Cox, M.T., Raja, A.: Metareasoning: A manifesto. Technical report, BBN TM-2028, BBN Technologies (2007)
2. Haugen, D.F., Kaasa, S.: The EPCRC. Eur J Palliat Care (2007; 14(3):130)
3. Kaasa, S.: Palliative care research – time to intensify international collaboration. Palliat Med (2008; 22:301-2.)
4. Cox, M.T., Eiselt, K., Kolodner, J., Nersessian, N., Recker, M., Simon, T.: Introspective multistrategy learning: On the construction of learning strategies. Artificial Intelligence **112** (1999) 1–55
5. Cox, M.T.: Multistrategy learning with introspective meta-explanations. In: Machine Learning: Proceedings of the Ninth International Conference, Morgan Kaufmann (1992) 123–128
6. Lopez, B., Plaza, E.: Case-based planning for medical diagnosis. In Ras., Z. (ed.), Proceedings of ISMIS-93. LNAI 837, Springer, 1993: 96-105
7. Arcos, J., Plaza, E.: A reflective architecture for integrated memory-based learning and reasoning. In Weiss, S. et. al. (eds.): LNAI 873 , Springer, 1994: 289-300
8. Christodoulou, E., Keravnou, E.: Metareasoning and meta-level learning in a hybrid knowledge-based architecture. Artificial Intelligence in Medicine 14 (1998): 53-81
9. Torasso, P.: Multiple representations and multi-modal reasoning in medical diagnostic systems. Artificial Intelligence in Medicine, 23 (2001): 49-69.
10. Friedland, D.: Evidence-based medicine: A framework for clinical practice. (1998)
11. Marling, C., Petot, G., Sterling, L.: Integrating case-based and rule-based reasoning to meet multiple design constraints. Comp. Intell., 15 (3), 1999, 308-332
12. Bichindaritz, I., Kansu, E., Sullivan, K.M.: Case-Based Reasoning in CARE-PARTNER: Gathering Evidence for Evidence-Based Medical Practice. EWCBR'98, LNAI 1488, pp. 334-345 (1998)
13. Terenziani, P., Molino, G., Torchio, M.: A modular approach for representing and executing clinical guidelines. Artificial Intelligence in Medicine, 23(3):249–276 (2001)
14. S. Montani, S.: Case-based reasoning for managing non-compliance with clinical guidelines. ICCBR 2007, 5th Workshop on CBR in the Health Sciences (2007)
15. Leake, D., Wilson, M.: Extending introspective learning from self-models. Proceedings of the AAAI 2008 Workshop on Metareasoning: Thinking About Thinking (2008)
16. Richter, M.M.: The knowledge contained in similarity measures. Invited Talk at ICCBR-95 (1995)
17. Díaz-Agudo, B., González-Calero, P.A.: CBROnto: A Task/Method Ontology for CBR. In: Procs. of the 15th International FLAIRS'02 Conference, AAAI Press (2002) 101–105
18. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications **7**(1) (March 1994) 39–59

Computer Cooking Contest

Workshop at the
Eighth International Conference on
Case-Based Reasoning
(ICCBR 2009)

Seattle, Washington, USA
July, 2009

Mirjam Minor, Armin Stahl, David Leake (Eds.)

Co-Chairs

Mirjam Minor
University of Trier, Germany
minor@uni-trier.de

Armin Stahl
DFKI, Germany
armin.stahl@dfki.de

David Leake
Indiana University, USA
leake@cs.indiana.edu

Additional Jury Members

David Aha, Naval Research Laboratory, USA
Mehmet Göker, PriceWaterhouseCoopers, USA
Ralph Traphöner, empolis, Germany

Programme Committee

David Aha, Naval Research Laboratory, USA
Klaus-Dieter Althoff, University of Hildesheim, Germany
Ralph Bergmann, University of Trier, Germany
Amélie Cordier, LIRIS, France
Belén Díaz-Agudo, Complutense University of Madrid, Spain
Mehmet Göker, PriceWaterhouseCoopers, USA
Eyke Hüllermeier, University of Marburg, Germany
Luc Lamontagne, Université Laval, Canada
Enric Plaza, IIIA-CSIC, Catalonia, Spain
David McSherry, University of Ulster, Northern Ireland
Nirmalie Wiratunga, Robert Gordon University, Scotland

Additional Reviewers
Kerstin Bach, Ibrahim Adeyanju

Preface

Computers provide us with assistance in our everyday tasks. Why should they not be able to help us cook healthy and tasty meals? The 2nd International Computer Cooking Contest (CCC) that will take place in conjunction with the ICCBR09 at the University of Washington at Tacoma on July 21, 2009 will hopefully give us a positive, case-based answer.

Are you planning to cook an Asian soup with leek? The CCC software programs are able to suggest a recipe. Do you have to follow a gout or a high cholesterol diet? The software will adapt the recipes to your requirements. Case-based software programs will compete in choosing and adapting recipes for a (yet) human cook. The input is a given database of basic recipes from which appropriate recipes must be selected, modified, or even combined. The queries to the system consist of a number of wanted ingredients and other requirements for the dish or menu. The overall competition is structured into a main compulsory task and two additional challenge tasks.

The **Compulsory Task** involves answering queries that require the selection and modification of a recipe for a single dish. A sample query could be to "cook a main dish with turkey, pistachio, and pasta". An appropriate answer would be to select a recipe for pistachio chicken and to replace chicken by turkey.

The **Adaptation Challenge** is to answer queries that require an adaptation of both the list of ingredients and instructions for preparation of the dish. This challenge will operate on a restricted recipe base.

The **Menu Challenge** requires the composition of a three-course menu based on the available recipes. For example we might ask: *"I do have a filet of beef, carrots, celery, field garlic and cucumber. Potatoes are available, too. For the dessert, we have oranges and mint. A soup would be preferable for the starter."* In this case, a Caldo Verde as a starter, filet steak with baked potatoes and an orange ice cream with mint flavour would be a good solution.

Exercise queries have been defined for the qualifying examination. They describe in an example-based manner the area of competence the developed software is supposed to cover. Hence there is not specified a formal query language, the items that will occur in the queries are restricted to the following:

- ingredients to be included or avoided, restricted to those occurring in the recipes in the database,
- type of ingredients, such as meat, fish, fowl, vegetables, fruits, nuts, alcohol,
- dietary practices, restricted to the following (a seasonal food calendar and the diet recommendations are given): Cholesterol diet, Gout diet, Seasonal food,
- type of meal (for the compulsory task and the menu challenge only), such as starter, salad, soup, ice-cream, cake, sauce, main course, pizza, casserole, risotto, dessert, pancake, three-course menu,
- type of cuisine (for the compulsory task and the menu challenge only), such as Italian, Asian, Mediterranean, ...

Queries will be described in free text but can be transformed manually to an arbitrary input format (structural/formula-based, conversational, text-based) to be processed by the software. The results produced by the software while answering the queries can be either a single or up to five recipes including a note which original recipes from the recipe base have been used for the creation of the result. The software will be evaluated against the competition queries, which are similar to the exercise queries and of the same focus. The competition queries will be kept confidential until the contest while the recipe base is publicly available. Evaluation criteria are scientific quality, technical quality, and culinary quality.

We are happy to present in the following the technical descriptions of the five finalist teams. We would like to thank the jury and the program committee members for their diligent reviews. We are looking forward to having an intriguing and inspiring live competition and hope you will enjoy reading!

Mirjam Minor July 2009
Armin Stahl
David Leake

Knowledge Acquisition and Discovery for the Textual Case-Based Cooking system WIKITAAABLE

Fadi Badra[1], Julien Cojan[1], Amélie Cordier[2], Jean Lieber[1], Thomas Meilender[1], Alain Mille[2], Pascal Molli[1], Emmanuel Nauer[1], Amedeo Napoli[1], Hala Skaf-Molli[1], Yannick Toussaint[1]

[1]LORIA UMR 7503 CNRS, INRIA, Nancy Universities BP 239
54506 Vandœuvre-lès-Nancy, France, First-Name.Last-Name@loria.fr
[2]LIRIS CNRS, UMR 5202, Université Lyon 1, INSA Lyon, Université Lyon 2, ECL
43, bd du 11 Novembre 1918, Villeurbanne Cedex, France,
First-Name.Last-Name@liris.cnrs.fr

Abstract. The textual case-based cooking system WIKITAAABLE participates in the second Computer cooking contest (CCC). It is an extension of the TAAABLE system that has participated in the first CCC. WIKITAAABLE's architecture is composed of a semantic wiki used for the collaborative acquisition of knowledge (recipe, ontology, adaptation knowledge) and of a case-based inference engine using this knowledge for retrieving and adapting recipes. This architecture allows various modes of knowledge acquisition for case-based reasoning that are studied within the TAAABLE project. In particular, opportunistic adaptation knowledge discovery is an approach for interactive and semi-automatic learning of adaptation knowledge triggered by a feedback from the user.

Keywords: textual case-based reasoning, case-based cooking, semantic wiki, opportunistic knowledge acquisition

URL of the system: At `http://taaable.fr`, the reader can find a link to the homepage of the WIKITAAABLE system as well as a report on the results given by the system on the set of training queries.

1 Introduction

As final result from last year did not make us good cooks, we decided to keep on doing research. Hence, for the second edition of the Computer Cooking Contest (CCC), the TAAABLE system has evolved towards a new architecture called WIKITAAABLE. This year, we focused our efforts on two intertwined aspects: knowledge and reasoning. Concerning reasoning, we worked on the inference engine improvement to enhance the adaptation ability of the system. Concerning knowledge we set up advanced knowledge acquisition facilities to support the evolution of the system during its life-cycle.

This paper describes the innovations developed in WIKITAAABLE, whose architecture is described in section 2 and discusses current research issues. One innovation this year is that the system is embedded within a semantic wiki and that the collaborative aspects are also of main concern mainly for document and knowledge edition and update.

The remainder of the paper shows how knowledge is manipulated within the system. Section 3 presents knowledge acquisition and representation within the semantic wiki. Section 4 illustrates how knowledge is used by the inference engine. Section 5 describes an opportunistic acquisition strategy guiding the evolution of the system knowledge. Finally, section 6 draws some conclusions and points out ongoing and future work. For qualification purpose, a restricted interface of the system is available online. This interface only allows users to ask queries to the system. The full application with embedded knowledge acquisition features will be available for the contest.

2 Architecture of WIKITAAABLE

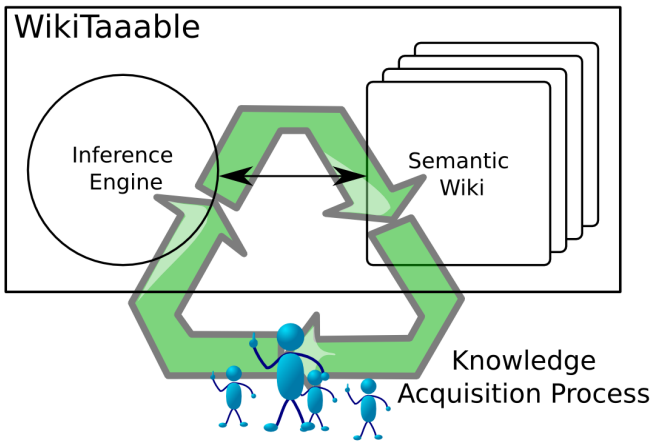


Fig. 1. Overview of the WIKITAAABLE architecture.

In a CBR system, results strongly depend on the quality of available knowledge. As a consequence, continuous improvement of knowledge is required to progressively enhance the obtained results.

The previous version of TAAABLE [1] suffered from different problems making maintenance and evolution of the system knowledge difficult. For example, there was no way to capture user feedback and to reuse it for maintenance. Besides, knowledge was organized in several files of heterogeneous formats that were difficult to update. As a consequence, the evolutions of the system knowledge were tedious tasks, even for the designers of TAAABLE.

In WIKITAAABLE [3] we decided to use a semantic wiki (Semantic Media Wiki [5]) as a central module to manage all data and knowledge used in the system. Making use of a semantic wiki has two major advantages: it enables humans and machines to rely on the same tool for representing and reasoning on shared knowledge and it provides users with user-friendly interfaces for browsing and editing knowledge.

Figure 1 gives an overview of the WIKITAAABLE architecture. Each component has been designed to work with the others and the components are strongly intertwined. For example, knowledge has not been represented at a general level but for reasoning purpose. The *semantic wiki* module manages knowledge of the system. The wiki is accessible for the users through a graphical interface and for the system through bots[1] that automates several tasks. Section 3 details this module. The *inference engine* includes the CBR core and is able to reason on the knowledge available in the wiki. It is described in section 4. In order to facilitate knowledge acquisition, the architecture of WIKITAAABLE is designed in such a way that it enables as much interactions as possible. *Opportunistic knowledge acquisition process* developed in WIKITAAABLE is discussed in section 5.

3 A Semantic Wiki for Collaborative Acquisition of Cooking Knowledge

In [3], Semantic Media Wiki (SMW [5]) is used as a blackboard for WIKITAAABLE knowledge management. WIKITAAABLE gathers the whole knowledge body required by the application.

To import knowledge of the first version of the TAAABLE system [1] into WIKITAAABLE, we wrote several bots that use MediaWiki API. Recipes, ontologies, and specific adaptation knowledge are now represented as a graph of semantic wiki pages. Each page can be freely read and updated by humans and by bots. Hence, TAAABLE is now maintained and improved by a collaboration between users and machines.

3.1 Domain Ontology

The domain ontology contains four hierarchies: *ingredients, dish moments, dish types, and dish origins*. For adapting a recipe by substituting some ingredients by other ingredients, the CBR engine requires knowledge stating similarity between ingredients. Therefore, ingredients are organized in the *ingredient hierarchy*. This hierarchy is used by the CBR engine to compute the cost of a substitution: the closer the ingredients, the lower the cost; e.g., *orange* is closer to *lemon* than *apple*.[2] In order to characterize recipes, three other hierarchies define and organize dish moments (*appetizer*, *dessert*), dish types (*cake*, *pizza*), and dish origins (*Mediterranean*, *Chinese*). The original acquisition of the hierarchies is described in [1].

The four hierarchies are imported into WIKITAAABLE by using the *Category/SubCategory* relation of Semantic MediaWiki [5]. For example, there is a page for orange and another page for citrus and the two pages are linked by this relation. For instance, the figure 2 shows the imported ingredient hierarchy.

[1] A bot is a piece of software for doing automated repetitive tasks.

[2] This closeness can be measured by a weighted length of the shortest path between ingredients in the hierarchy.

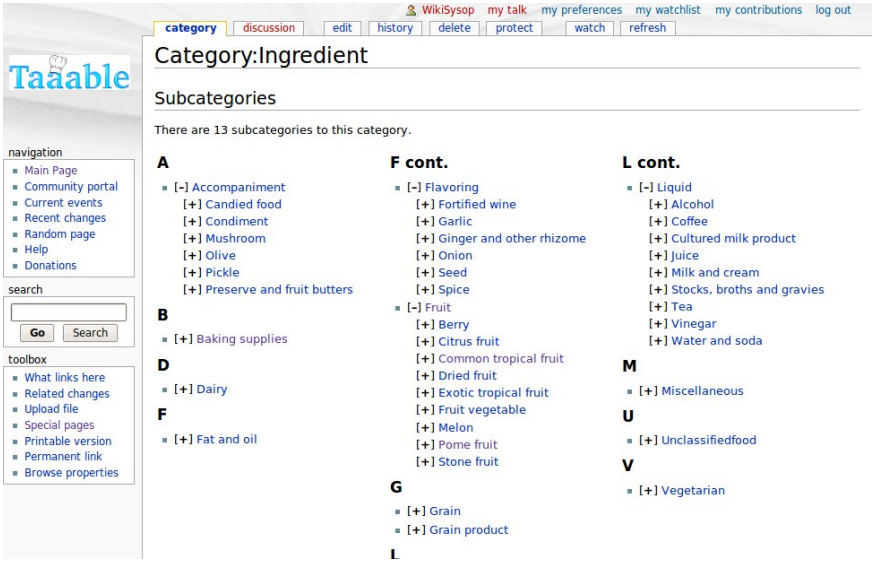


Fig. 2. WIKITAAABLE ingredient ontology.

3.2 Adaptation knowledge

To adapt a recipe, the CBR engine uses the ontology and a set `AK` of substitutions. A substitution $\sigma \in$ `AK` states that, in a given context, some ingredients may be substituted by other ones. For instance, the following substitution states that, for the context of a salad without potato, vinegar may be substituted by lemon juice and salt.

$$\sigma = \begin{array}{|l|l|l|} \hline \textbf{context} \text{ salad} \\ \quad\quad \textbf{no} \text{ potato} & \textbf{from} \text{ vinegar} & \textbf{by} \text{ lemon juice} \\ & & \quad \text{salt} \\ \hline \end{array} \quad (1)$$

Each substitution is represented in a semantic wiki page. For instance, figure 3 shows the wiki page of the above substitution. The acquisition of substitutions is detailed in section 5.

3.3 Recipes

The recipes are also imported into WIKITAAABLE, where a wiki page is created for each recipe. Then, a bot crawls all the recipe pages, parses ingredient information, sets dish types, moments, and origins. It updates recipe pages with this information encoded as semantic annotations. Figure 4 shows a recipe page in WIKITAAABLE. We used the n-ary relationship of Semantic Media Wiki to represent an ingredient line, for example, $(1, \mathrm{c}, \texttt{Category:rice})$ represents `1 c rice`, the first ingredient line in figure 4. The parsing of ingredient information and setting types is described in [1].

Fig. 3. A substitution semantic wiki page.

4 Principles of the CBR Inference

4.1 Knowledge containers

Knowledge in WIKITAAABLE is mainly expressed in a polynomial fragment of propositional logic. The TAAABLE knowledge base is a set of containers

$$\texttt{KB} = \{\mathcal{O}, \texttt{Recipes}, \mathcal{H}_{idx}, \texttt{AK}, \texttt{cost}\}$$

KB is encoded in wiki pages and the CBR engine has access to these pages through SPARQL queries.

$\mathcal{O}$ is the domain ontology represented by a set of axioms of the form $a \Rightarrow b$ where a and b are two variables representing recipe classes. For example, `lemon` (resp., `citrus`) represents the class of recipes with lemon (resp., with citrus) and the axiom $\texttt{lemon} \Rightarrow \texttt{citrus}$ states that any recipe with lemon is a recipe with citrus. In fact, every ingredient name X such as `lemon` is interpreted here as "class of the recipes with ingredient X".

`Recipes` is the set of recipes given by the CCC organizers, and consequently the case base of the CBR system TAAABLE. A recipe $R \in \texttt{Recipes}$ cannot be directly handled by the CBR inference engine: the engine requires a formal representation whereas R is for the largest part written in natural language. Therefore, only the formalized part of the recipe R is used: its index $idx(R)$, which is expressed by a conjunction of literals (the indexing process of the recipes coincides with the annotation process mentioned in section 3.3). For example

$$idx(R) = \texttt{lettuce} \wedge \texttt{vinegar} \wedge \texttt{olive\_oil} \wedge \texttt{tomato} \wedge \textit{Nothing else} \qquad (2)$$

is a formal representation of a recipe R having ingredients lettuce, vinegar, olive oil, and tomato. A closed world assumption is associated to $idx(R)$ stating that if a property cannot be deduced from the recipe description and from the ontology then it is considered as false. Formally, if $idx(R) \not\models_{\mathcal{O}} a$ then "*Nothing else*" contains the

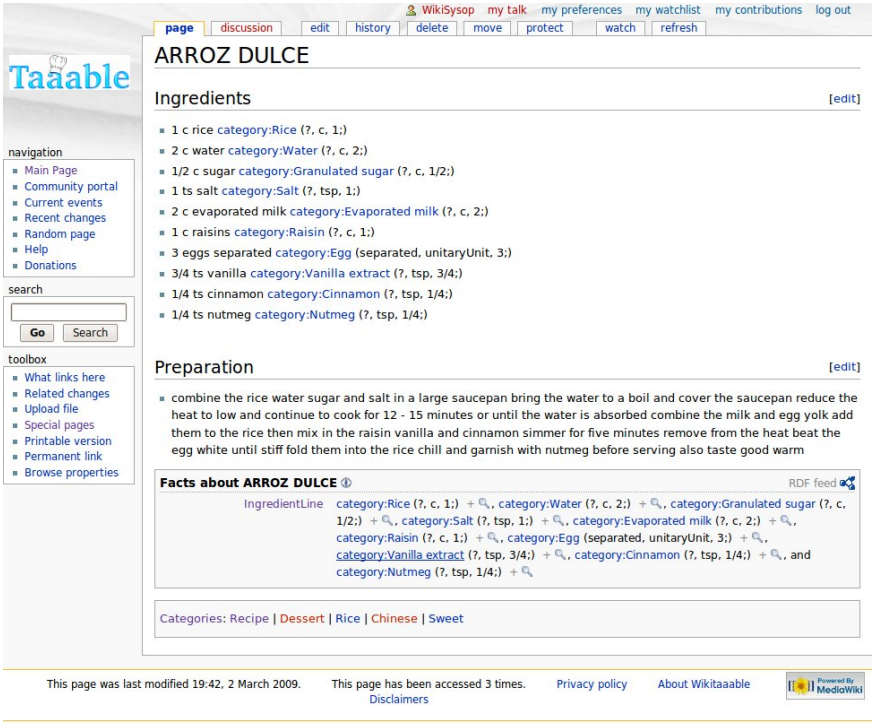


Fig. 4. Indexed recipe of "Arroz dulce".

conjunct $\neg a$.[3] For example, this closed world assumption enables to deduce that $idx(R) \models_{\mathcal{O}} \neg\texttt{meat} \wedge \neg\texttt{fish}$, i.e., that R is a vegetarian recipe.

The indexes $idx(R)$ are used to access recipes through a hierarchy $\mathcal{H}_{idx}$, according to the partial ordering $\models_{\mathcal{O}}$: for $C, D \in \mathcal{H}_{idx}$, $C \models_{\mathcal{O}} D$ iff there is a directed path in $\mathcal{H}_{idx}$ from C to D. The indexes $idx(R)$ are leaves of the $\mathcal{H}_{idx}$.

Adaptation knowledge has two parts. The first part is included in ontology $\mathcal{O}$. The second part is the set $\texttt{AK}$ of substitutions (cf. section 3.2). Any $\sigma \in \texttt{AK}$ may be considered as a domain specific inference rule $\dfrac{R \text{ is a good recipe}}{\sigma(R) \text{ is a good recipe}}$. The substitutions have the form $C \rightsquigarrow D$ where C and D are conjunctions of literals. Applying $C \rightsquigarrow D$ to a conjunction of literals (such as an index or a query) consists in replacing the literals of C by literals of D. For example, the substitution σ described in figure 3 is written as follows

$$\sigma = \texttt{salad} \wedge \neg\texttt{potato} \wedge \texttt{vinegar} \rightsquigarrow \texttt{salad} \wedge \neg\texttt{potato} \wedge \texttt{lemon\_juice} \wedge \texttt{salt} \quad (3)$$

[3] If f and g are two propositional formulas, $f \models_{\mathcal{O}} g$ means that f implies g, given the ontology $\mathcal{O}$. More precisely: if the interpretation $\mathcal{I}$ satisfies both $\mathcal{O}$ and f then $\mathcal{I}$ satisfies g.

It can be noticed that the substitution given by a triple $(\texttt{context}, \texttt{from}, \texttt{by})$ in the wiki pages (cf. equation (1)) are rewritten $\texttt{context} \wedge \texttt{from} \rightsquigarrow \texttt{context} \wedge \texttt{by}$ to suit the CBR engine formalism.

The CBR inference is based on substitutions, either taken from AK or built with the help of ontology $\mathcal{O}$ (see below for details). The choice of substitutions is made according to the problem-solving context and to a cost function $\texttt{cost} : \sigma \mapsto \texttt{cost}(\sigma) > 0$; substitution σ is preferred to substitution τ when $\texttt{cost}(\sigma) < \texttt{cost}(\tau)$. Therefore, the cost function (and its parameters) constitutes an additional knowledge container.

4.2 CBR Inference

Let Q be a query. For example

$$Q = \texttt{endive} \wedge \texttt{lemon\_juice} \wedge \neg\texttt{onion} \tag{4}$$

is a query for a recipe with endive and lemon juice and without onion. The CBR inference consists in (1) retrieving recipes matching exactly or approximately Q and (2) adapting the retrieved recipes.

Retrieval aims at choosing indexes $\mathit{idx}(R)$ matching the query Q. An exact match corresponds to $\mathit{idx}(R) \vDash_{\mathcal{O}} Q$. If no index exactly matches Q, the query Q is progressively relaxed into $\Gamma(Q)$ such that $\mathit{idx}(R) \vDash_{\mathcal{O}} \Gamma(Q)$, for some $\mathit{idx}(R)$. The relaxation of Q is obtained by applying generalizations g_k according to $\mathcal{O}$: $g_k = a_k \rightsquigarrow b_k$ is a substitution such that $(a_k \Rightarrow b_k) \in \mathcal{O}$. Thus $\Gamma(Q) = g_n(\ldots(g_1(Q))\ldots)$. Therefore, retrieval provides a similarity path

$$\mathit{idx}(R) \vDash_{\mathcal{O}} \Gamma(Q) \xleftarrow{g_n} \ldots \xleftarrow{g_1} Q \tag{5}$$

This similarity path is built according to a best-first search minimizing $\sum_k \texttt{cost}(g_k)$. For example, retrieval may give the following result

$$\begin{aligned} Q &= \texttt{endive} \wedge \texttt{lemon\_juice} \wedge \neg\texttt{onion} \\ \Gamma(Q) &= \texttt{green\_salad} \wedge \top \wedge \neg\texttt{onion} \equiv \texttt{green\_salad} \wedge \neg\texttt{onion} \\ \mathit{idx}(R) &= \texttt{lettuce} \wedge \texttt{vinegar} \wedge \texttt{olive\_oil} \wedge \texttt{tomato} \wedge \textit{Nothing else} \end{aligned}$$

(Γ consists in generalizing endive into green_salad and in removing lemon_juice from the query by generalizing it in several steps to $\top$, the hierarchy top).

Adaptation is composed of two sub-steps. Let R be a retrieved recipe, with index $\mathit{idx}(R)$. The first subset of adaptation is matching, that aims at building an adaptation path from $\mathit{idx}(R)$ to Q of the form

$$\mathit{idx}(R) \xmapsto{\sigma_1} \ldots \xmapsto{\sigma_p} \Sigma(\mathit{idx}(R)) \vDash_{\mathcal{O}} \Gamma(Q) \xleftarrow{\gamma_q} \ldots \xleftarrow{\gamma_1} Q \tag{6}$$

where $\sigma_i \in \texttt{AK}$ $(i = 1 \ldots p)$ and substitutions γ_j $(j = 1 \ldots q)$ correspond to axioms of the ontology: $\gamma_j = a_j \rightsquigarrow b_j$ with $(a_j \Rightarrow b_j) \in \mathcal{O}$. Such an adaptation path is built according to a best-first search in a state space minimizing $\sum_i \texttt{cost}(\sigma_i) + \sum_j \texttt{cost}(\gamma_j)$.

The second sub-step of adaptation consists in "following" the adaptation path: first R is adapted successively in $\sigma_1(R)$, $\sigma_2(\sigma_1(R))$, $\ldots\sigma_p(\ldots(\sigma_2(\sigma_1(R)))\ldots) = \Sigma(R)$.

Then, ingredients of $\Sigma(R)$ are substituted by other ingredients according to a generalization-specialization process (generalization corresponds to the relation $\models_{\mathcal{O}}$ and specialization to the substitutions $\gamma_q^{-1}, \ldots, \gamma_1^{-1}$).

For example, let $idx(R)$ and Q be the example presented above. Matching may provide the following adaptation path:

$$idx(R) \overset{\sigma}{\longmapsto} \Sigma(idx(R)) \models_{\mathcal{O}} \Gamma(Q) \overset{\gamma}{\longleftarrow} Q$$

where σ is defined by (3) and $\gamma = \texttt{endive} \rightsquigarrow \texttt{green\_salad}$. Thus, the adaptation of R consists in replacing vinegar by lemon juice and salt (cf. σ) and by substituting `lettuce` by `endive` (cf. $\models_{\mathcal{O}}$ and γ^{-1}).

It can be noticed that retrieval provides a first matching: a similarity path is a kind of adaptation path involving no $\sigma \in \texttt{AK}$. Thus, the retrieved recipe R can be adapted following this similarity/adaptation path. However, during adaptation, some substitutions $\sigma \in \texttt{AK}$ may be used and, if they do, the resulting adaptation requires less effort.[4]

5 Opportunistic Knowledge Acquisition and Discovery

WIKITAAABLE has been designed to facilitate continuous knowledge acquisition through interactions with its users: it is a reflexive and perpetually evolving system. However, due to the heterogeneity of knowledge acquisition situations that can be envisioned, setting up such a process is a complex task. This diversity of situations is explained by several factors:

- The various types of knowledge (ontology, adaptation knowledge, substitutions costs, recipes) that can be acquired.
- The different interaction modalities such as simple user feedback, direct modification on wiki pages, interaction through dedicated interfaces, use of external knowledge discovery methods, etc.
- The provenance of knowledge that is acquired: single users, community of users, experts, other sources of data (web sites), or local knowledge from which new knowledge in inferred.

In the following, a particular scenario of opportunistic knowledge discovery is detailed. In WIKITAAABLE, substitutions $\sigma \in \texttt{AK}$ are acquired at problem-solving time through interactions with the user: they are automatically generated online by the system and are stored in the form of wiki pages. The knowledge discovery process CABAMAKA [4] is used to assist the user in the formulation of new pieces of knowledge. Its role is to generate a set of substitutions $\sigma \in \texttt{AK}$ "on the fly" from the comparison of two sets of recipes of the case base. The generated substitutions can be used by the system to repair a failed adaptation. Each time a substitution is validated by the user, it is stored for future reuse. In the following, the main principles of the approach

[4] If the cost function is an estimation of the adaptation effort, then the adapted recipe should be better by following (6) then by following (5). Indeed, since adding new substitutions (the ones of `AK`) only adds new ways to connect indexes to queries, it comes that $\sum_i \texttt{cost}(\sigma_i) + \sum_j \texttt{cost}(\gamma_j) \leq \sum_k \texttt{cost}(g_k)$.

are illustrated with an example. More details on the proposed approach can be found in [2].

In section 4, the user wanted a salad recipe with lemon juice but without onion, which was modeled by the query Q defined by (4). The substitution $\sigma \in \texttt{AK}$ defined by (3) was used to adapt the retrieved recipe R by replacing vinegar by lemon juice and salt. Such a substitution cannot be obtained from the ontology $\mathcal{O}$, so let us assume in this scenario that σ is not available to the system. Thus, to perform adaptation, the system relies solely on the ontology $\mathcal{O}$ from which it generates the substitution $\texttt{vinegar} \rightsquigarrow \texttt{lemon\_juice}$. Now, in our scenario, the user is not satisfied with the proposed solution and gives this feedback to the system. Therefore, the knowledge discovery process is triggered: a set of substitutions is learned from the case base by comparing salad recipes with vinegar and salad recipes with lemon juice. Among the learned substitutions is the substitution $\sigma_{learned} = \texttt{vinegar} \rightsquigarrow \texttt{lemon\_juice} \wedge \texttt{salt}$, which suggests to replace vinegar by lemon juice in the retrieved recipe R and to add salt. The user is satisfied with the adaptation resulting from the application of this substitution, so the latter is stored for future reuse. At this point, the user is encouraged to specify the condition of application of the substitution $\sigma_{learned}$. The user states that vinegar can be replaced by lemon juice and salt in salad recipes that do not contain potato, which is modeled by the substitution context $\texttt{salad} \wedge \neg \texttt{potato}$. Combining the learned substitution $\sigma_{learned}$ and its application context gives the substitution σ defined by (3).

In WIKITAAABLE, the wiki is used as a gateway enabling to centralize knowledge used in the system. It provides functionalities to facilitate acquisition and maintenance of knowledge and enables to progressively add new acquisition features, allowing the evolution of the whole system. However, setting up a complex knowledge acquisition process raises several issues. For example, tools for ensuring consistency of knowledge used in the system must be developed. Another issue is to handle updates from multiple users. What happens when one believes that an avocado is to be eaten as a starter whereas someone else reckon that it has to be eaten as a dessert? Is the system supposed to converge towards a "commonly accepted" knowledge base or should it be able to deal with user's preferences?

A strength of the architecture of WIKITAAABLE is that it will enable to progressively address these issues. A future work is to allow users to add their own recipes to the system. This functionality requires to be able to dynamically annotate new recipes within WIKITAAABLE in order to make them usable by the CBR inference engine. One of the advantages of such a functionality, combined with the benefits of a wiki, is that communities of users will be able to share their recipes and to collaborate in order to improve the global knowledge of the system. Next, we would like to tackle the multi-user issue which is a prerequisite for envisioning a collaborative building of the knowledge base of TAAABLE.

6 Conclusion

The textual case-based cooking system WIKITAAABLE participates to the second CCC. It is an extension of the TAAABLE system that has participated to the first CCC. WIKITAAABLE's architecture is composed of a semantic wiki used for the collaborative

acquisition of knowledge (recipe, ontology, adaptation knowledge) and of a CBR inference engine using this knowledge for retrieving and adapting recipes. This architecture allows various modes of knowledge acquisition for CBR that are studied within the TAAABLE project. In particular, opportunistic adaptation knowledge discovery is an approach for interactive and semi-automatic learning of adaptation knowledge triggered by a feedback from the users. The WIKITAAABLE system is generic in the sense that it can be applied to another application domain, as soon as the knowledge representation formalism is sufficient.

The first ongoing work is the improvement of the WIKITAAABLE system (user interface, inference engine, knowledge base encoded in wiki pages, and links between these components). Another work planned for the next weeks is the development of tools within WIKITAAABLE for knowledge acquisition triggered by user feedbacks. Such a knowledge acquisition leads to a continuous evolution of the knowledge base and thus, of the behavior of the system. It is important to ensure that these evolutions are improvements and that the integrity of the knowledge is preserved. We plan to use non regression and consistency tests for this purpose.

Currently, wiki pages are accessed and maintained by a limited community: the TAAABLE project members. These pages encode the knowledge that have been acquired on the basis of a consensus. A long term objective is to have several open semantic wikis with cooking knowledge, each of them corresponding to a user community, the consensus being only realized at the level of a community.

Acknowledgments

The participants of the TAAABLE project wish to thank the organizers of the CCC for providing this benchmark, that entails many interesting problems, and the need to collaborate with researchers in various domains on knowledge engineering.

References

1. F. Badra, R. Bendaoud, R. Bentebitel, P.-A. Champin, J. Cojan, A. Cordier, S. Després, S. Jean-Daubias, J. Lieber, T. Meilender, A. Mille, E. Nauer, A. Napoli, and Y. Toussaint. Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In *ECCBR Workshops, Workshop of the First Computer Cooking Contest*, pages 219–228, 2008.
2. F. Badra, A. Cordier, and J. Lieber. Opportunistic adaptation knowledge discovery. In Lorraine McGinty and David C. Wilson, editors, *Case-Based Reasoning Research and Development / ICCBR 2009*, 2009. To appear.
3. A. Cordier, J. Lieber, P. Molli, E. Nauer, H. Skaf-Molli, and Y. Toussaint. Wikitaaable: A semantic wiki as a blackboard for a textual case-based reasoning system. In *SemWiki 2009 – 4th Semantic Wiki Workshop*, Heraklion, Greece, May 2009.
4. M. d'Aquin, F. Badra, S. Lafrogne, J. Lieber, A. Napoli, and L. Szathmary. Case base mining for adaptation knowledge acquisition. In *Proceedings of the International Conference on Artificial Intelligence, IJCAI'07*, pages 750–756, 2007.
5. M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic Wikipedia. *Journal of Web Semantics*, 5(4), 2007.

COOKING CAKE

Christian Fuchs, Christoph Gimmler, Simon Günther, Lukas Holthof, Ralph Bergmann

University of Trier, Germany

{ fuch4102@uni-trier.de, gimm4101@uni-trier.de, guen4101@uni-trier.de, holt4102@uni-trier.de, bergmann@uni-trier.de }

Abstract. In this paper we present "Cooking Cake": a CBR system for ontology-based retrieval and adaptation of recipes. "Cooking Cake" was developed in order to participate in the 2nd Computer Cooking Contest which will be held at the International Conference on Case-Based Reasoning (ICCBR '09) at the University of Washington (USA).

Keywords: Case-Based Reasoning (CBR), retrieval, similarity, ontology, Collaborative Agent-Based Knowledge Engine (CAKE)

1 Introduction

In this paper we present "Cooking Cake", a CBR system based on CAKE (Collaborative Agent-based Knowledge Engine) [1,5,7]. The CAKE system combines workflow, agent and CBR technologies following an ontology approach. "Cooking Cake" is mainly built on CAKE's CBR technology which provides efficient retrieval facilities including sophisticated similarity calculations. CAKE itself was developed at the University of Trier, Germany.

"Cooking Cake" was developed within a two-semester student project in order to participate in the 2nd Computer Cooking Contest at the International Conference on Case-Based Reasoning (ICCBR '09), held at the University at Washington, WA (USA). The Computer Cooking Contest is made up of three different challenges:

- **Compulsory Task**, which involves answering queries that require the selection and, where appropriate, modification of recipe for a single dish.
- **Adaptation Challenge**, which requires an adaptation of the list of ingredients and instructions for preparation of the dish.
- **Menu Challenge**, which requires the composition of a three-course menu based on the recipes provided by the CCC-Jury.

With "Cooking Cake" we mainly want to compete in the Compulsory Task and the Menu Challenge, but all functions needed to take part in the Adaptation Challenge will be implemented as well.

In this paper we want to give a quick overview of the important conceptual aspects of our software. First we will get into the design of our domain-ontology (Section 2). After this we will describe all steps needed to transform the recipe database - given by the Computer Cooking Contest-Jury - to our internal case base (Section 3). This will not only include the pre-processing methods used, but also we give insight on CAKE specific representation of data, the classification process

needed to evaluate a single recipe's "Type of Meal" or "Type of Cuisine" and how the system deals with dietary-practices and explicit seasonal availability of certain ingredients.

In the last part of this paper, we want to give a more detailed view of the similarity and retrieval functions that are provided by CAKE (Section 4). Finally, we give a quick introduction to our interface (Section 5), evaluate the results of given exercise queries (Section 6), draw some conclusions and describe some lines of future work (Section 7).

2 Ontology-Design

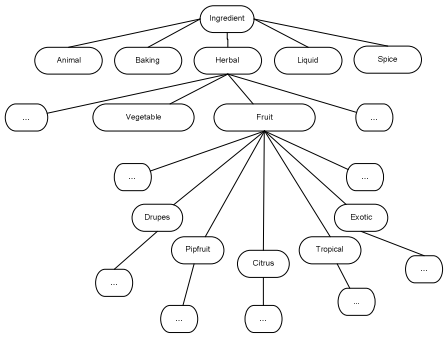
An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base [2]. As the ontology provides nearly all knowledge needed to solve given problems, we put a lot emphasis on reviewing existing resources for gaining information about the domain "cooking". Unfortunately, most available resources – like the available applications of the CCC '08 – were not appropriate for our needs as they followed a different structural approach in the design of the cooking ontology. Therefore we decided to construct a suitable ontology on our own [8] which is represented in a self developed XML-Dialect.

After analyzing the given recipe database we distinguished three different types of ontology-concepts:

- **Category-Nodes**, groups of ingredients (e.g. fish or pork)
- **Ingredient-Nodes**, ingredients (e.g. squash or pumpkin)
- **Synonym-Nodes**, ingredients which are basically identical to another ingredient, but differ in their spelling (American or British English, Typos)

With the support of a local head chef we built a basic ontology with main focus on a skeletal structure of category nodes (see Fig.1). This ontology was consistently expanded and ended up consisting around 140 different category-nodes. It has a hierarchical structure with "ingredient" as the root-node and five main ingredient-origin classes as direct sub-nodes:

- **Animal**, ingredients of animal-origin. Sub-nodes e.g. fish or cheese
- **Herbal**, ingredients of herbal-origin. Sub-nodes e.g. vegetable or corn
- **Baking**, baking related ingredients. Sub-nodes e.g. yeast or gelatin
- **Liquid**, liquid ingredients. Sub-nodes e.g. alcoholic liquors or soft drinks
- **Spices**




```
<node value="ingredient">
   <node value="animal">…</node>
   <node value="baking">…</node>
   <node value="herbal">
     <node value="vegetable">…</node>
     <node value="fruit">…</node>
     …
     </node>
   <node value="liquid">…</node>
   <node value="spice">…</node>
</node>
```

Fig.1. Part of the basic ontology and its representation in XML

3 From Recipe Database To Case Base

The given recipe database is a semi-structured XML document where each recipe consists of three parts (see Fig.2 as an example):

- Title (TI)
- Single/multiple ingredients (IN)
- Preparation (PR)

In order to develop a system which can address all tasks of the Computer Cooking Contest the given recipe database has to be transformed to a fully structured and formalized document.

```
<RECIPE>
      <TI>"Lutheran" Hotdish</TI>
      […]
      <IN>1/2 lb Mild or spicy sausage</IN>
      <IN>1 lg Onion (sliced and quartered) (up to) </IN>
      <IN>1 lb Uncooked pasta (i.e. elbow; twisted; wagon
      wheels, shells; etc) (up to)</IN>
      […]
      <PR> […] Meanwhile, prepare the pasta per pkg
      instructions.    In a large pan, combine all
      ingredients.  Add enough tomato sause until mixture
      is well coated, […] </PR>
</RECIPE>
```

Fig 2. Example Recipe

3.1 Pre-Processing and Ingredient Extraction

In order to provide an accurate retrieval and similarity measurement every <IN>gredient tag needs to be reduced to a basic ingredient. This process was subdivided into four steps:

- **Filter Stopwords**
  Remove all words with no direct relation to an ingredient (e.g. “in”, “the”, “or”).
- **Filter Cooking Units**
  Remove all cooking units (e.g. “1/2 lb”, “1 pkg”, “1 tablespoon”).
- **Filter state of ingredient**
  Remove all ingredient states (e.g. “filtered”, “chopped”, “finely sliced”).
- **Simple Form Reduction**
  Reduce words to their simple form by using the lexicographic rules of Kuhlen [3] (e.g. “Onions” to “Onion”).

“`1 lg Onion (sliced and quartered) (up to)`” (taken from Fig. 2) will be transformed to “`Onion`”.

3.2 Ontology Refinement and Case Generation

To perform an efficient search the constructed basic ontology (See Section 2) had to be extended and enhanced by inserting simple form ingredients which are extracted out of the recipe database. In addition, a link between the automatically extracted ingredient in the recipe and its place in our ontology had to be stored. For this reason we developed a java based application (See Fig. 3 and Fig. 4) which:

1. Performs all the filter methods in Section 3.1.
2. Reduces an ingredient to its simple form.
3. Automatically skips an ingredient which already has been inserted.
4. Creates a link between recipe-ingredient and ontology-ingredient.
5. Supports the manual insertion of a simplified ingredient via GUI into its place in our ontology.
6. Applies “Seasonal Restrictions” and “Dietary Practices” (see Section 3.4).
7. Classifies recipes “Type of Meal” and “Type of Cuisine” (see Section 3.3).
8. Generates CAKE Files: CaseBase, Similarity DataBase and DataModel.

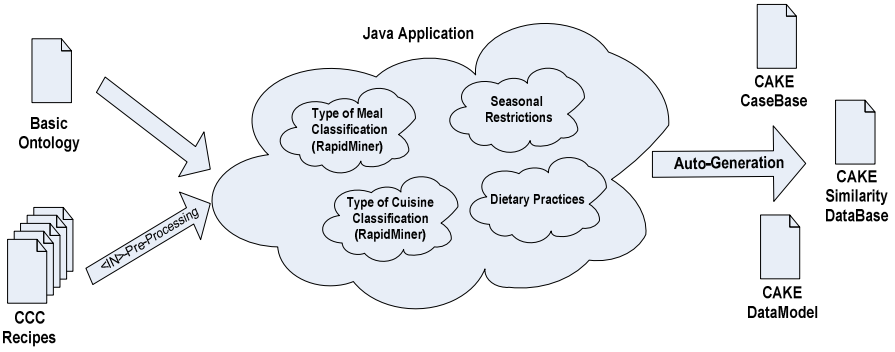


Fig. 3. Process Visualization

The CAKE DataModel consists of an enumeration of all category-nodes of the refined ontology, followed by a taxonomy order of these nodes. The CAKE CaseBase includes all recipes where the ingredients are represented by their simple form values stored in the ontology. This representation is extended by additional information, e.g.

information about “Seasonal Restrictions” and “Dietary Practices” as well as “Type of Meal” and “Type of Cuisine” classification. Finally, the CAKE Similarity DataBase is generated. In this DataBase the similarities between two different ingredients is stored (See Section 4).

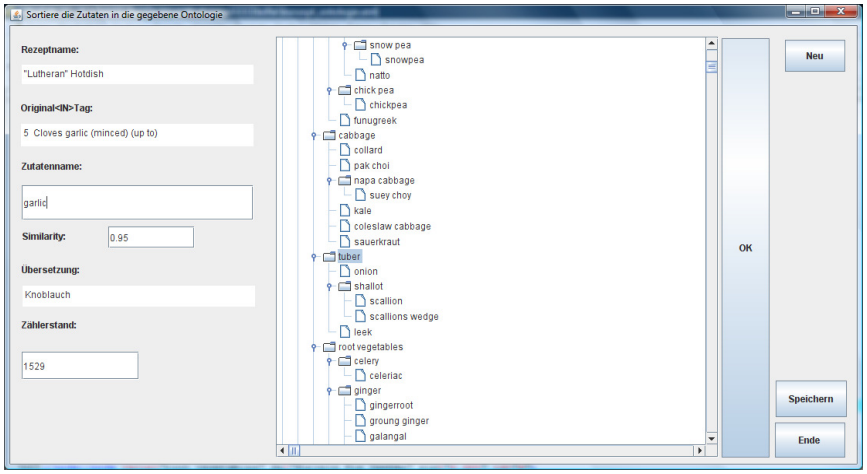


Fig. 4. Developed Application

3.3 Type of Meal / Type of Cuisine

As almost every exercise query includes information about “type of meal/type of cuisine” and “seasonal/dietary practices” the classification of those types of meals or ingredients is a main requirement for all challenges.

Our approach to classify a meal is based on RapidMiner 4.3 (community edition) and it`s implemented classifiers. RapidMiner is a java based open source data mining solution, which addresses a wide range of data mining tasks. It has implemented a bunch of learners, IO-methods and visualization functions [4].

We tested several classifiers supported by RapidMiner and came to the conclusion that the Naive Bayes classification provides the best results for our task. The Naive Bayes algorithm makes the assumptions that all relevant attributes have the same significance and are fully independent of each other. Although these assumptions can’t be fulfilled in our case, it delivers suitable and efficient results for the “type of meal” and “type of cuisine”-classification.

The selection of attributes and recipes is an important step to predict the confidence of the class. The Naive Bayes learner is based on previous calculation of probability values that reflect a link between attribute values and class membership. It determines a confidence for each class and returns the highest value as prediction for each recipe.

First, we had to manually select attributes for the classification. Second, we had to manually pre-classify a set of recipes. We ended up selecting about 70 relevant ingredients for “type of meal” and 80 relevant ingredients for “type of cuisine” which then were the basis of our classification process. Also we had to pre-classify about

130 recipes for “type of meal” and about 90 for “type of cuisine”, e.g. the recipe “Char Siu Gin Doi” which has a typical combination of ingredients used in the asian cuisine like “Bamboo shoots”, “Soy sauce” and “Chinese noodles”. Based on these pre-classifications RapidMiner built a classification model which then could be used to classify all other recipes of our database.

For the purpose of estimating and improving the accuracy of our classification we decided to implement the m-fold cross validation method in RapidMiner. This validation determines the classification accuracy by dividing the pre-classified dataset into m disjunctive subsets, learning a classification concept with m-1 of these subsets and applying the learned concept to the skipped subset. By comparing the pre-classified class of a recipe with the predicted class, the classification error can easily be identified. In our case we used 10 randomized subsets of pre-classified recipes.

Selected Ingredients	**Accuracy**
General Vegetable, Fruit and Meat Ingredients	51%
Sophisticated Herbal and Animal Ingredients	59%
Refined Selection of Herbal, Animal and Spice Ingredients	66%

Fig. 5. Process “Type of Meal”

Although the accuracy percentages in Fig. 5 and Fig. 6 may seem low, most errors are false positives, which derive from intersections between different recipe classes as Naïve Bayes requires fully independent classes. An example for a false positive would be the classification of a pizza recipe as “main course”, which is not a fault at all, but the best classification would be “pizza”. Most errors we are facing are of this kind.

Selected Ingredients	**Accuracy**
All Spice Ingredients	46%
Spices and some Vegetables Ingredients	71%
Spice, Vegetables and Meat Ingredients	91%

Fig. 6. Process “Type of Cuisine”

We were able to improve the accuracy and the selection of relevant attributes (ingredients) by evaluating and analyzing results we achieved with several ontology-subsets. As seen is Fig. 5 and Fig. 6 we were able to increase the accuracy from 51% to 66% for “Type of Meal” and from 46% to 91% for “Type of Cuisine” by refining the ontology as well as the selection of relevant attributes for the classification.
It was not possible to achieve a perfect classification of all recipes concerning “type of meal” or “type cuisine”. Finally we had to correct some fatal error classifications manually.

3.4 Seasonal Restrictions / Dietary Practices

Not only “Type of Meal” and “Type of Cuisine” are possible requirements in a query. “Seasonal Restrictions” and “Dietary Practices” can be desired by a user as well.

"Seasonal Restrictions" occur for 23 different ingredients and can range in their value for each month between "fresh/is in season" and "storable". As an example, "Cabbage" can be stored from January to March, while it is "in season" from September to December.

"Dietary Practices" exist for "Gout Diet" and "Cholesterol Diet". In both cases preferable and restricted ingredients can be found in a chart given by the CCC-Jury. Following a "Cholesterol Diet" the guidelines – which are no medical information – recommend to prefer low-fat milk products and avoid butter, eggs, turkey, or pork.

In order to offer best possible retrieval results we decided to treat all seasonal information and dietary recommendations as extended ingredients. In the current version of "Cooking Cake" we are able to define similarities between each month and even closer between "storable"-months and "fresh"-months. Dietary information for a single recipe is right now only available as a Boolean expression, which means a recipe either is "Gout Diet" or "Cholesterol Diet"-valid or it is not.

4 Retrieval & Similarity

Considering the ontology as the main knowledge base of our software, the weight initialization of every node was a key process in the development of "Cooking Cake". We finally decided to calculate the weight by the formula

$$\mathbf{weight(node)} = \left(\frac{\mathbf{height(node)}}{\mathbf{maxlevel}}\right) \qquad (\mathbf{1})$$

where "maxlevel" stands for the depth of the path the node is on. In some evident cases manual enhancement of these weights were necessary.

Choosing CAKE as framework for developing "Cooking Cake" allowed us to use the CAKE Data Model and the already implemented CBR technology. The CAKE Data Model describes all kinds of data that can occur in the system. It is an object-oriented model using specialization and aggregation to define the data classes. Available data classes are atomic classes like boolean, integer, double, date or time as well as compound data classes like aggregates, collections and intervals. These system classes are used to define a cooking specific data model.

Each data class of this data model can be instantiated as a CAKE data object. The main function of the CBR technology is the similarity based retrieval of above described data objects. For this purpose a similarity model is defined on the top of the data model combining and configuring similarity functions predefined in a similarity library [5].

The CAKE's library provides about 30 similarity measures. Similarities in the provided version of "Cooking Cake" are calculated by determining the first common father of query and case object. This solution is meant to be an initial solution as we are working on the development of an independent calculation method in order to get the best possible and differentiated results for the cooking-domain.
Right now the retrieved similarity between a case object and a query object is calculated as

$$\sum_{i=0}^{\#AggAttribute} AggAttributeWeight(i) * weight(CommonFather(Query, Case)) \quad (2)$$

where aggregate (Agg) contains for example the five main categories (see Section 2) like “animal” or “herbal”. For each of these aggregate-attributes a weight is defined which represents the importance of this attribute for the retrieval. This weight is multiplied by the weight of the common father-node of the query and the case object.
The possibility to avoid ingredients is implemented as an initial solution as a filter on the retrieved cases, but will be reworked and re-implemented in the similarity measurement.

5 “Cooking Cake” / Graphical User Interface

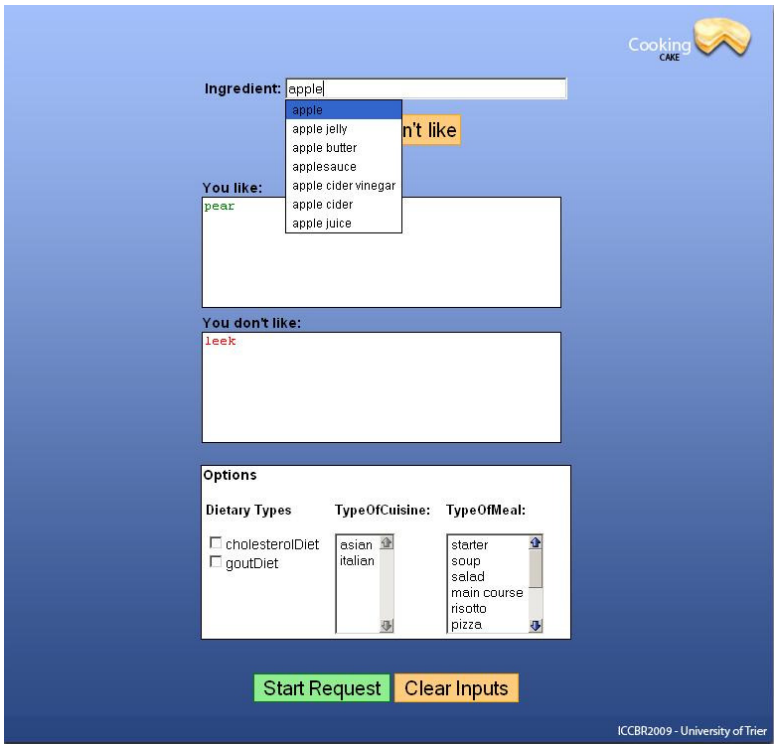


Fig. 7. Interface

The current version of “Cooking Cake“ comes as a MySQL and PHP-based interface. Hence, in order to work with “Cooking Cake” a webserver with PHP support is required as well as our expanded version of the CAKE Core.
Relating to Figure 7 the interface is divided into two parts. At the top the user can

specify ingredients which are desired ("Do") or ingredients which should be excluded ("Don't"). At the bottom of the interface "Dietary Practices", "Type of Cuisine" or "Type of Meal" can be chosen.
By clicking on the button "Menu Challenge" in the top left corner of the website the user is able to specify each of the described options for three courses which are Starter, Main Dish and Dessert.

6. Exercise Queries

In order to understand how "Cooking Cake" works given Compulsory Task queries - taken from the Computer Cooking Contest website - and their results will be discussed in this section.

Exercise Query #1: Cook an Asian soup with leek.
(Main focus: type of meal, type of cuisine)

As requested the retrieved recipe is an Asian soup with leek. No other restrictions applied.

Exercise Query #2: I would like to have a salad with celery. Please consider that I follow a gout diet.
(Main focus: dietary practice)

Dietary restrictions applied in this query. Therefore only gout diet conform salads with celery were retrieved. No other restriction applied.

Exercise Query #3: Prepare a low-cholesterol dessert with strawberries and avoid citrus fruits.
(Main focus: dietary practice)

The result of this query is a low-cholesterol dessert which includes strawberries and no citrus fruits. Low-Cholesterol and Citrus restrictions applied.

Exercise Query #4: Cook a risotto with carrots.
(Main focus: similarity / modification of recipes)

The result of this query is a risotto with carrots. No other restrictions applied and no modification had to be made.

Exercise Query #5: I would like to cook a pear pancake.
(Main focus: similarity / modification of recipes)

The result of this query is a apple pancake. This result derives from the close similarity between apple and pear. No other restrictions applied.

Exercise Query #9: I do have a filet of beef, carrots, celery, field garlic and cucumber. Potatoes are available, too. For the dessert, we have oranges and mint.

A soup would be preferable for the starter.
(Menu Challenge)

As the result of this query "Cooking Cake" recommends a soup as a starter, a main dish with most of the requested ingredients and a dessert with oranges. No restriction applied and no modification had to be made.

7 Conclusion And Related Work

The main difference towards the systems of the CCC'08 [9] is that "Cooking Cake" is based on the CAKE Framework. Also we want to point out, that the required integration of extended information, like "Type of Cuisine" and "Type of Meal", was implemented by using a Naïve Bayes-Classificator.

On the other hand our system shows similarities in some parts to the COOKIIS'08 which was the only available resource from CCC'08, beside the technical papers provided in the ECCBR 2008 proceeding [10]. For example the ideas behind the existing ontology of this system were partly integrated into the design of our ontology and customized to the needs of our CBR-Software. Therefore our ontology and the COOKIIS' ontology agree in several parts.

With "Cooking Cake" we developed a new CBR-Application for the generic "Collaborative Agent-based Knowledge Engine"-Framework. In this application we were able to implement a highly sophisticated preprocessing method which allows a user to semi-automatically build a large ontology where the manual effort decreases in relation to the increasing amount of ingredients in the ontology.

Currently the focus of our application is to provide a solution for the Competition requirements, but it can be easily extended to a bigger application.

8 References

1. Collaborative Agent-Based Knowledge Engine, Anforderungen - Konzept – Lösungen, http://www.wi2.uni-trier.de/publications/2007_CAKE_Maximini-V2.pdf
2. Swartout, B., Patil, R., Knight, K., Russ, T.: "Toward Distributed Use of Large-Scale Ontologies". In: Proceedings of the Tenth Knowledge Acquisition for Knowledge- Based Systems Workshop. Banff, Alberta (1996)
3. Kuhlen, R.: Experimentelle Morphologie in der Informationswissenschaft, p. 57 Verlag Dokumentation, München (1977)
4. RapidMiner main documentation, http://rapid-i.com/content/view/36/23/
5. CAKE – Technical Report, http://www.wi2.uni-trier.de/publications/2005_technicalReport_cake.pdf
6. Rechernberg, P.: Informatik-Handbuch, p. 987, Hanser-Verlag (2006)
7. Bergmann, R., Fressmann, A., Maximini, K., Maximini, R., Sauer, T., Case-Based Support for Collaborative Business. In: Advances in Case-Based Reasoning ECCBR 2006. pp. 106--120. Springer (2006)
8. Hering, R., Herrmann, F.J.: Herings Lexikon der Küche. Pfanneberg Verlag, Haan-Gruiten (2001)
9. Stahl, A., Minor, M., Traphöner, R. (2008). Call for Participation. Computer Cooking Contest 2008, http://www.wi2.uni-trier.de/eccbr08/index.php?task=ccc
10. Martin Schaaf. Workshop Proceedings. ECCBR 2008. Tharax Verlag (2008)

CookIIS – A Case-Based Recipe Advisor

Norman Ihle, Régis Newo, Alexandre Hanft, Kerstin Bach, Meike Reichle

Intelligent Information Systems Lab
University of Hildesheim
Marienburger Platz 22
31141 Hildesheim, Germany
{lastname}@iis.uni-hildesheim.de

Abstract. The aim of this paper is to present CookIIS[1], a Case-Based Reasoning (CBR) system that provides recipe suggestions. The suggestions are created based on a given set of recipes that the system modifies according to a user's specification. For the adaptation of recipes CookIIS relies on its knowledge model, a set of 342 rules as well as substitution suggestions mined from cooking communities on the WWW. The paper describes how CookIIS processes the competition queries and masters the challenges of the Computer Cooking Contest 2009.

1 Introduction

Retrieving and reusing recipes that are correct and delicious is a difficult task and the participation in last years Computer Cooking Contest (CCC) showed us that the challenges we had to bear are interesting research topics. In this paper we present insights how our system CookIIS is realized and which methodologies and techniques we use to cover the expected areas of competences.

This year the CCC contains three main challenges that we all address with CookIIS. The Compulsory Task focusses on the type of meal and type of cuisine as well as extended dietary practises and the modification of recipes. Therefore we extended and improved last years knowledge models, added new dietary practices and we are now using additional knowledge that has been extracted from cooking communities on the WWW. For participation in the new Adaptation Challenge, CookIIS has an own adaptation component which only contains pasta recipes with more detailed and structured information of the preparation processes. The third task, the Menu Challenge, focusses, like last year, on the composition of a three course menu and we use the findings and further developments made for the Compulsory Task and Adaptation Challenge to also improve the retrieval and adaptation in the Menu Challenge.

Overall CookIIS is a Case-Based Reasoning (CBR), more precisely a structured CBR system that uses techniques like Text Mining and Information Extraction to gather more knowledge and improve the results of our systems. Moreover, we used the experiences we made with last years system to improve

[1] http://cookiis2009.iis.uni-hildesheim.de:8080/ccc

CookIIS, especially its underlying knowledge containers [1]. We realized CookIIS using the industrial strength CBR based tool Information Access Suite (e:IAS) from empolis[2], which employs Case Retrieval Nets [3]. The cases that we use were given by the CCC organizers, so we concentrated on extracting knowledge for recipe transformation, the coverage of our vocabulary and the computation of the similarity measures. Based on last years systems we completely revised the underlying knowledge model: we defined more precise ingredient taxonomies and (semi-)automatically extended the vocabulary aiming that most of the ingredients can be recognized. The more detailed representation of ingredients also results in a better similarity computation.

We decided to use very user-friendly text based graphical interface what makes it easy to use CookIIS. According to the three challenges we have got three components: The *CookIIS Recipe Creator* is developed to compete in the Compulsory Task because its underlying case base contains the according recipes. The *CookIIS Pasta Adaptation* only contains the pasta recipes and focusses in the adaptation task. The *CookIIS Menu Creator* also contains compulsory case base, but has text fields to specify and explicitly exclude ingredients for each of the three courses. CookIIS can be queried in German and in English - however the recipes and the preparation instructions will be in English.

Our paper is structured as follows: Section 2 presents the CookIIS knowledge models and focuses on improvements compared to last year's system. According to the three tasks CookIIS has to perform, section 3 contains the Compulsory Task queries and explains how we extract type of meal and type of cuisine, how we retrieve recipes that fit dietary practises and how we compute the similarity measures. Section 4 presents the adaptation process as well as the integration of community knowledge to provide more reliable substitutions for ingredients. The Menu Challenge queries as well as the creation process of a menu is described in 5. The last section gives a summary of the main features of CookIIS and provides a short outlook on future work.

2 Knowledge Model

Since we use an approach that is based on structured CBR, we implemented a very detailed knowledge model which describes the cooking domain. The model is based on the one of the CCCIIS system that competed in last years Computer Cooking Contest [4]. The main elements of our model are the ingredients that are required to prepare a meal. They are organized in the following eleven classes:

Basic Ingredients	Fish	Meat	Vegetable
Supplement	Fruit	Drinks	Milk
Minor Ingredients	Oil and Fat	Spice and Herb	

Within these classes we modelled about 1000 different ingredients represented as concepts that can be used for describing a meal. The most concepts (177) are contained in the Meat class. Each concept represents one single ingredient and is modeled with synonyms in English and German language. In most cases the

concepts are ordered in one or more taxonomies. These taxonomies are one possibility for the calculation of the similarity between a query and a case as explained in Section 3.4.

For this years contest we manually revised our existing model from last year, cleaned a lot duplicates, added a lot of concepts and synonyms and reworked all of our taxonomies. The remodeling and completion of classes is mainly based on [5]. For example, we split the class 'liquids', which contained drinks as well as non-drinkable fluids into the classes 'Drinks' and 'Oil and Fat' since those two classes have almost nothing in common. Any fluids not fitting in those classes we added to other existing and fitting ones. For example water is now a basic ingredient and only mineral water is a drink. Since there is a special focus on pasta recipes for this years adaptation challenge we also modeled about 40 kinds of different pasta in an additional class.

Besides the ingredients, preparation methods and tools required for the preparation are often mentioned in a recipe. We modeled each in an own class. The preparation methods are not only used to compute the similarity between a query and a case, but it is also used to determine the type of meal (e.g. dessert) as described in the following section. For the Computer Cooking Contest it also important to determine the origin of a recipe. We modeled 48 possible origins and ordered them in a taxonomy to use them for similarity calculation too.

Overall we modelled about 2000 different concepts in 25 different classes. We weighted each class in relation to the importance of the class for finding the right recipe. We built two different case bases using two different aggregates, but the same model: one with the compulsory task recipes and one with the pasta recipes.

3 Compulsory Task

In the Compulsory Task, single recipes have to be retrieved according to automatically computed meta information like the type of ingredients, the type of meal, the type of cuisine while also considering some dietary practices.
The type of ingredients is automatically recognized based on the classes in our knowledge model presented in the previous section. We will explain in the following two sections how we computed other meta information.

3.1 Type of Meal

One type of meta information needed for the recipes is the type of meal, which describes whether a dish is for example a main course or a dessert. We defined a class in which most concepts (i.e. types of meal) are ordered in a taxonomy. We implemented some in order to assign the types of meals to the recipes. We based the implementation on two main aspects:

- indicative keywords in the recipe title,
- indicative ingredients or combination of ingredients.

In the first approach, we extract the type of meal from the title of a recipe. For example, a recipe which contains the indicative word "sherbet" in its title is, with a high probably, a dessert. In the second approach, we analyze the ingredients (and also their types) as well as the preparation methods contained in recipes. The preparation methods are extracted from the given preparation instructions. For example, a frozen meal with milk, fruits and sugar is probably an ice cream. We do not use further information from the preparation instructions yet. We have about 15 rules for the assignment of the type of meal with a precision of about 90 percent.

3.2 Type of cuisine

Another requirement of the CCC is the identification of the type of cuisine (e.g. Italian, Chinese, Mediterranean, etc.), which is not given in the recipe itself. All possible origins are modeled in one class and organized in an taxonomy. For example the concept "Mediterranean" is the parent for "Portuguese", "Spanish" and "Italian", while "Mediterranean" itself is a child node of "Southern European". In order to map the recipes to one of our concepts we implemented three rule-based approaches using rules:

1. identification of the recipe's origin in the recipe title
2. identification of characteristic strings or meals in the recipe title and map them to an origin
3. using the occurrences of spices and herbs and other ingredients to find some elements that are characteristic for a type of cuisine

The first method is based on the finding that many recipes display their origin in their title (e.g. Chinese Chicken Salad). Here the origin is determined right away. The second approach needs some background knowledge. It is based on the fact that some foods or meals are characteristic for a specific type of cuisine (e.g Sauerkraut is typical for the German kitchen). A set of rules maps those occurrences to the according type of cuisine. For the third approach we use ingredients, especially the spices and herbs used in the recipe. For example the use of curry is a strong hint on the Indian cuisine. Overall CookIIS includes about 28 different rules, which are prioritized according to the different approaches and map the recipes with a precision of 80 to 90 percent.

3.3 Dietary Practices

In the CCC tasks and challenges, the participating systems have to be able to retrieve suggestions that are following at least the following two dietary practices (i.e. gout diet and cholesterol diet) and a seasonal food calendar. We also include other dietary practices (i.e. vegetarian, nut free and non alcoholic) which were implemented for our CCCIIS [4] in last year's contest.
To follow the specification of a gout and cholesterol diet, ingredients as well as types of ingredients were given, which should or should not be contained in

the retrieved recipes. For each forbidden ingredient, we set a filter so recipes containing this ingredient are not considered during the retrieval process. We do this by excluding either single ingredients or categories of ingredients (e.g. exclude any kind of meat for a gout diet). We also set filters in order to consider only the recipes that contain at least one of the recommended ingredients. These two kinds of filter ensure that the retrieved recipes always contain some of the recommended ingredients and none of the forbidden ones. We use filters instead of adaptation, because the forbidden ingredients can not be adequately replaced in a recipe while following a diet.
With the seasonal option, the user can specify a month of which the vegetables that are included in the suggested recipe are available. If this option is chosen and some vegetables in the best fitting recipe are not available in the selected month, the system will give a hint. Further we propose, according to the provided seasonal food calendar, adaptation candidates for some unavailable vegetables. The system just proposes candidates that are available in the selected month.

3.4 Similarities (Taxonomies)

As stated before, each recipe is represented as one case consisting of attributes. Each attribute's class is defined by the concepts that we introduced in Section 2. The similarity between two cases is calculated as follows:

- local similarity for each attribute and
- a global similarity measure for recipes.

The taxonomies of our knowledge model are used as a first approach for the computation of the local similarities. By assigning adequate values for the generalization and the specialization step in the taxonomy, similarities between concepts of a class can be computed. In our taxonomies, those values were assigned and adjusted manually. When needed, we defined several taxonomies for given classes. For example, the class meat has two taxonomies. In the first taxonomy the single meat ingredients are ordered according to their kind (e.g. beef, pork, poultry) and in the second taxonomy the ingredients are ordered by the part of the animal they origin from (e.g. fillet, haunch). Since both taxonomies are used for similarity calculation, chicken breast is similar to turkey breast as well as to chicken fillet.
The second approach for the computation of local similarities consists of manually assigning similarity values of some pairs of elements of the class in a similarity matrix. This assignment is only done for pairs of ingredients for which either they are pretty well known to be very similar or one of the ingredients could not be ordered in a taxonomy. We use a combined similarity measure in which the assigned similarity value is the maximal value obtained from both approaches.
The global similarity measure is a weighted sum of the local similarities of the attributes in the case, following the local-global principle for similarity modeling (see [6]). The weights assigned to the local similarities reflects the importance of the corresponding attributes in the cases. The sort of meat used in a meal is, for example, more important than the type of oil used.

4 Adaptation Challenge

Due to the fact that the given recipe base is not large enough to provide recipes for the whole variety of desired and favoured ingredients, an adaptation of the existing recipes to the users needs is necessary. Thereby ingredients which are excluded by a certain diet or explicitly by a user are considered as forbidden ingredients. The intersection of forbidden ingredients and ingredients occurring in a retrieved recipe we call critical and they have to be replaced. Our overall assumption is to replace forbidden ingredients with others from the same ingredient class.

The adaptation of the recipes in CookIIS is mainly done via a set of rules called after the retrieval of similar cases. According to shortcomings of our first adaptation approach used at CCC 2008 we discovered in [7] we restructured our adaptation as a sequence of distinguished steps to pursue different adaptation approaches. At the moment the implemented adaptation steps can be subsumed under two approaches: *community-based adaptation* and *model-based adaptation*. The first executed approach collects concrete pairs of ingredients as adaptation advices from comments inside cooking communities on the WWW and is described in section 4.2. For critical ingredients where no replacements at the first step are found the subsequent adaptation steps are executed (belonging to *model-based adaptation*), which work more general on the designed knowledge model and similarity measures. The reason for using this approach is to avoid handcrafting a single adaptation rule for each ingredient with explicit replacement candidates. It it explained in the next section 4.1. We combine both adaptation approaches and save intermediate results in extra attributes of the case. The main advantage of this sequential adaptation is that we can review and adjust the results of adaptation steps performed before.

4.1 Model-based Adaptation

The adaptation schema bases on a relaxed intersection function from e:IAS [2] Rule Engine which determines similar ingredients for the critical ingredients as replacements. It is explained more in detail in [8].

The replacement of ingredient (concepts) through child concepts of this ingredients is not appropriate (e.g. replacing forbidden tomatoes with cherry tomatoes). Therefore we have to remove the child concepts from the list of replacement candidates. The relaxed intersection function only uses the default similarity measure of a class which is, for CookIIS, the combined similarity measure explained in Section 3.4. It is not possible to explicitly choose one the given approaches used for the computation of similarity values. The relaxed intersection function therefore allows the replacement of forbidden ingredients through parent concepts as well as child concepts, which is not appropriate under certain circumstances. In order to eliminate the child concepts of the forbidden ingredients from the list of replacement candidates, we adjusted some values in the similarity matrices, such that other candidates are more similar than the child concepts. We execute the intersection function twice, once with a threshold equal

to similarity value resulting from a taxonomy and afterwards with a threshold slightly above this first threshold and compare the results to separate and exclude the child concepts.
For desired ingredients which do not appear in the recipe yet we execute an additional adaptation step to replace one of the similar ingredients of the recipe with the desired ingredient. As a last step we count the amount of replacement candidates. If no adequate ingredients remain, we recommend to omit this forbidden ingredient.

4.2 Extraction of Adaptation Knowledge from Cooking Communities

The Internet and the Web 2.0 with its user-generated content is a large source of any kind of knowledge and experience. This includes knowledge and experience about cooking and about adapting recipes, which are discussed in cooking communities. In these communities people upload their favourite recipes for everybody to use and express their opinion about other peoples recipes. Thereby they do not only say what they like or what they do not like about the recipe, they also express the way they adapt the recipe to their needs. This can be for changing the taste, for following a certain diet or just because they did not have an ingredient at hand and took a different one. For this years CCC we implemented an approach that makes this knowledge available for our application.

We collected about 70'000 recipes with more than 280'000 comments from a large German cooking community by crawling the website. This way we got one HTML source-code page for each recipe with the corresponding comments. From this source code we extracted the relevant information entities using filters based on different HTML tags. For the recipe these entities were primarily the recipe title, needed ingredients and the preparation instructions. If users commented the recipe, we extracted the text of the comment, checked if the comment was an answer to another comment and if the comment is marked as a helpful comment or not. All these informations we stored in an database to have an efficient access to the data.

In the next step we used the e:IAS and an extended CookIIS knowledge model to generate two different case bases. One case-base consists of the recipes and one of the comments. Using the e:IAS TextMiner we extracted the mentioned ingredients and stored them as a case for each recipe and each comment. Since our knowledge model is bilingual (English and German) we were able to translate the originally German ingredient names into English terms during this process and also took care of used synonyms. This way had the same terms in the case bases that we use in our CookIIS application.

Having built up the two case bases we first retrieved a recipe, then all of the comments belonging to the recipe, and finally compared the ingredients of the recipe with the ingredients mentioned in the comment. This way we classified the ingredients mentioned in the comments into the following three categories:

- *New*: ingredients that are mentioned in the comment, but not in the recipe

- *Old*: ingredients that are mentioned in the comment as well as in the recipe
- *OldAndNew*: two or more ingredients of one class of our knowledge model, of which at least one was mentioned in the recipe and in the comment and at least one other one was only mentioned in the comment, but not in the recipe

For the CookIIS application the last class is the most interesting. We interpreted ingredients with this classification as either an adaptation (e.g. instead of milk I took cream) or an explanation/specialization (e.g. Gouda is a semi-firm cheese). For each of these ingredients classified as *OldAndNew* we also stored whether it is the new or the old one. We tried to distinguish between adaptation and specialization by looking for hints in the original comment text and by using the taxonomies of our knowledge model. Therefore we tried to find terms in the comment that indicate that an adaptation was described in the comment during the text-mining process (e.g. instead of, alternative, replaced with,...) and stored those terms in the corresponding case. Additionally we looked in the taxonomy of the ingredient class whether the one ingredient is a child of the other (or the other way around). If an ingredient is a child of the other we interpreted this as specialization or explanation, because one ingredient is a more general term than the other. This way we omit to have adaptations like: instead of semi-firm cheese take Gouda.

For each classified ingredient we assigned a specific score, which depends on the following factors:

- the number of ingredients found in the comment text
- whether the comment was marked as helpful or not
- whether a term was found that specifies the classification further or not
- whether a term was found that indicates a different classification or not

After assigning the score we aggregated our classification results. For the CookIIS application we did this in two steps: First we aggregated all classified ingredients of all comments belonging to one recipe. Thereby we counted the number of the same classifications in different comments and subsequently added up the score of the same classifications. Then we aggregated all classifications without regarding the recipe they belong to. This way we can select the most common classifications out of all classifications.

For this years CookIIS application we use the overall aggregation of *OldAndNew*-classified ingredients to generate adaptation suggestions. We look up this ingredient in our "community knowledge DB". If this ingredient is categorized as "old" we use the corresponding "new" ingredient to serve as substitution. If more than one substitution is found, we use the on with the highest score. We usually try to retrieve two adaptation suggestions to be more manifold. Using the approach we got more than 5'300 different adaptation suggestions for about 570 different ingredients of which we only use the most common (regarding the number appearances in the comments and the score). For the future we plan to use the adaptation suggestions with regard to the recipe they belong to. Our idea is to find similar recipes to the recipe the adaptation is done at out of our

pool of 70'000 recipes and find an adaptation suggestion from a similar recipe following the principle that similar recipes need similar adaptations.

The approach described above has a lot of advantages. For finding ingredients we can use our detailed CookIIS knowledge-model. This way we take care of synonyms and have the same terms in the application as well as in the adaptation-database. Since we are not looking for exact semantics but do our own classification we are independent from slang or informal language as often used in bulletin boards in the Internet. By using a large number of recipes and comments we hope to balance wrong classifications out and avoid suggesting false positives. Some more technical details and ideas for the evaluation are described in [9].

5 Menu Challenge

In this challenge CookIIS designs a three-course menu according to given constraints. The user can specify which ingredients should be used, which should be avoided and the *CookIIS Menu Creator* will compose a menu containing a starter, a main dish, and a dessert that fit together. Our underlying assumption is that each dish of a menu should have the same type of cuisine. The technical idea behind the Menu Creator is a two step retrieval [10].

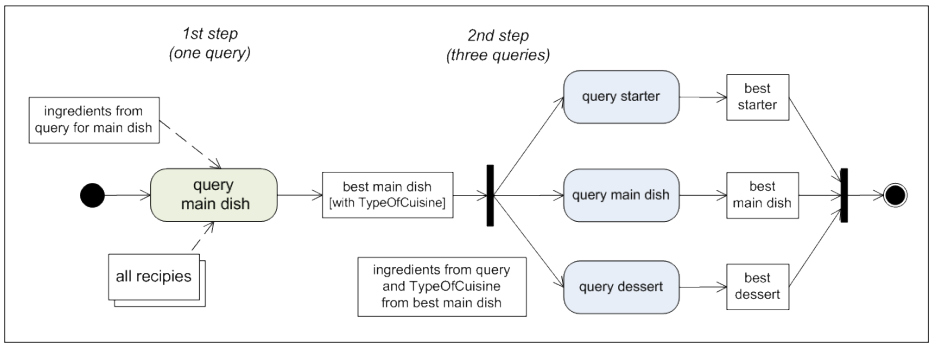


Fig. 1. Menu Design: two-query approach

Figure 1 illustrates our approach, in which a main dish is retrieved first to set the type of cuisine for the whole menu. In the second retrieval step the given ingredients and the type of cuisine are used to retrieve fitting recipes for starter, main dish and dessert. Within the second step we also make sure that the types of meal of the courses differ from each other.

6 Conclusion

In this paper, we presented CookIIS, a case-based system that provides recipe suggestions. We first presented the underlying knowledge model, which contains

25 classes with about 2000 concepts. We improved the knowledge model by taking additional knowledge obtained from cooking literature into account as well as the experience made with our system. In order to meet some of the requirements of the Computer Cooking Contest, various meta information about recipes can be automatically computed in CookIIS. We have a set of rules which is used for the extraction of the needed meta information. We also use a rule based approach to deal with dietary practices by only considering the recipes that can be recommended for a given diet. CookIIS uses a sequential adaptation based on a set of prioritized adaptation rules performing a sequence of steps to pursue two different approaches: on the one hand the model-based adaptation and on the other hand the community-based adaptation.
For the future we still have ideas to improve CookIIS. We are actually trying to consider the amount (i.e. the weight) of ingredients for the retrieval and the computation of meta information. Another improvement we are aspiring consists of learning association rules from the recipes in order to refine the adaptation.

References

1. Richter, M.M.: Introduction. In Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S., eds.: Case-Based Reasoning Technology – From Foundations to Applications. LNAI 1400. Springer-Verlag, Berlin (1998)
2. empolis GmbH: Technical white paper e:information access suite. Technical report, empolis GmbH (January 2008)
3. Lenz, M.: Case Retrieval Nets as a Model for Building Flexible Information Systems. Dissertation, Humboldt University of Berlin, Berlin (1999)
4. Hanft, A., Ihle, N., Bach, K., Newo, R., Mänz, J.: Realising a cbr-based approach for computer cooking contest with e:ias. In Schaaf, M., ed.: ECCBR 2008 Workshop Proceedings, Hildesheim, Tharax (2008) 249–258
5. Reinhard Löbbert, Dietlind Hanrieder, U.B., Beck, J.: Lebensmittel: Waren, Lebensmittel, Trends. Verlag Europa-Lehrmittel (2001)
6. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. Volume 2432 of LNAI. Springer-Verlag (2002) Habilitation.
7. Hanft, A., Ihle, N., Newo, R.: Refinements for retrieval and adaptation of the cookiis application. In Hinkelmann, K., Wache, H., eds.: Fifth Conference Professional Knowledge Management, Solothurn, Switzerland. Volume 145 of LNI., GI (2009)
8. Hanft, A., Ihle, N., Bach, K., Newo, R.: Cookiis – competing in the first computer cooking contest. Künstliche Intelligenz **23**(1) (2009) 30–33
9. Ihle, N., Hanft, A., Althoff, K.D.: Extraction of adaptation knowledge from internet communities. In Delany, S.J., ed.: ICCBR 2009 Workshop Proc., Workshop Reasoning from Experiences on the Web. (july 2009) to appear
10. Reichle, M., Bach, K.: Improving result adaptation through 2-step retrieval. In Nalepa, G.J., Baumeister, J., eds.: Proceedings of the 4th Workshop on Knowledge Engineering and Software Engineering (KESE 2008) at the 31st German Conference on Artificial Intelligence (KI-2008). (sep 2008) 73–84

JaDaCook 2: Cooking Over Ontological Knowledge

P. Javier Herrera, Pablo Iglesias,
Ana Mª García Sánchez, Belén Díaz-Agudo

Department of Software Engineering and Artificial Intelligence,
Universidad Complutense de Madrid, Spain.
{pjherrera@pdi.ucm.es, belend@sip.ucm.es}

Abstract. JaDaCook 2 has been developed to participate in the 2nd *Computer Cooking Contest* 2009 (CCC-09). The system is an improved version of JaDaCook 1.0 that participated in CCC-08. We have reengineered the source code making the code more reusable and extensible. JaDaCook 2 includes general improvements and new functionality. Namely, a new form based interface, a new version of the ontology including more type of ingredients, dietary practices, types of meal type of cuisine, data mining over the ingredients, textual and IE capabilities over the recipes. The system has been developed as the final evaluation assignment for a graduate course in Machine Learning at the Computer Science Faculty (*Complutense University of Madrid*). In this paper we present a brief review of the technical characteristics of the system, describing the knowledge acquisition and reasoning processes, the new functionality and some experimental results.

Keywords: Knowledge Intensive CBR, Ontology, jCOLIBRI 2, Data Mining

1 Introduction

In this paper we describe JaDaCook[1] version 2, a CBR system that solves the task of suggesting a recipe given a restricted set of ingredients as the query. This version of the system has been developed to participate in the Computer Cooking Contest 2009. It is an improved version of the JaDaCook 1 that participated in CCC-08.

The system has been developed by students as the final evaluation assignment for a graduate course in Machine Learning at the *Computer Science Faculty (Complutense University of Madrid)* during the first semester of 2009. They have reengineering the source code of JaDaCook 1 making the code more reusable and extensible. JaDaCook 2 includes general improvements and new functionality. Namely, a new form based interface, a new version of the ontology including more type of ingredients, dietary practices, types of meal, type of cuisine, data mining over the ingredients, textual and IE capabilities to process the text of the given recipes.

JaDaCook 2 reasons using different knowledge sources: (1) a case base of recipes (provided by the CCC-09 organizers and available from textual sources), (2) a cooking ontology, (3) a set of association rules, obtained using data mining techniques, capturing co-occurrences of ingredients in the recipes. These rules are

[1] http://gaia.fdi.ucm.es/grupo/projects/cookingContest/cookingContest.html#jadacook2

used to propose substitute ingredients, (4) a case base of menus built collaboratively using the personal opinion of non-expert users. This case base is used to build menus by the composition of single dishes.

JaDaCook 2 solves the *Compulsory Task* of the CCC-09 that involves answering queries that require the selection and, where appropriate, modification of a recipe for a single dish; and the *Menu Challenge* that requires the composition of a three-course menu based on the available recipes and on collaborative recommending techniques.

JaDaCook 2 adapts the retrieved recipe by substituting ingredients; however it does not aim to solve the *Adaptation Challenge* in CCC-09 as it does not change the preparation directions in a recipe.

JaDaCook 2 has been implemented using the jCOLIBRI [1] framework. jCOLIBRI helps the development as it supports, among other things, textual processing [2] and the use of ontologies to enrich reasoning processes in CBR systems [3].

JaDaCook 2 includes a new form based interface based on the multiplatform framework QT [7]. The new interface replaces the previous natural language interface, simplifies the query formulation process, and minimizes errors in the communication process with the user. The new interface allows navigating the case base, querying it by selecting from the ontology specific ingredients or types to be included or avoided and dietary practices. It also allows querying the system using new ingredients, including new ingredients in the ontology, new menus in the collaborative case base and new recipes that are automatically processed from the given XML file.

The Menu Challenge asks for the creation of a three course menu. JaDaCook 2 offers a case based collaborative recommender system that bases the menu configuration on the opinion of previous users. When the user queries the menu system, (s)he is asked about her opinion on the result, and his/her answer is recorded and used for future recommendation.

In this paper we present a brief overview of the technical characteristics of the system, describing the knowledge acquisition and reasoning processes, the new functionality and some experimental results with the queries provided in the CCC-09 webpage.

The paper is structured as follows. Section 2 details the main features of the graphical interface of the system. Section 3 describes the knowledge sources of the system and Section 4 briefly explains the CBR processes. Section 5 offers some examples and results and Section 6 concludes the paper.

2 Graphical Interface

The new interface of JaDaCook 2 integrates in the same GUI (see Fig. 1) three tabs (from left to right in Fig.1) that correspond to the following tasks:

1) **Single Dish Challenge**. Given a query, the result will be the retrieval and adaptation of a recipe for a single dish.

2) **Menu Challenge**. This function enables the creation of a three course menu. The collaborative part uses the opinions of previous users to guide future recommendations.

3) **Recipes inspection** where the user can consult the case base.

Next subsections describe these three features of the system.

Fig. 1. JaDaCook Main Window: single dish tab

2.1 Single Dish Tab

In the single dish tab (Fig. 1) the user provides the query information. It is not mandatory to include every component:

- List (WL) of Ingredients that the user would like to include.
- List (WNL) of ingredients that the user would not like to include.
- Dietary practices like Vegetarian, Nut-free or non-alcoholic, plus new dietary practices like gout diet or cholesterol diet.

- Type of cuisine. One of the following: Chinese, Mexican, Mediterranean or Italian.

Selecting corresponding radio button on left and right sides of single dish tab (see Fig. 1) the user can introduce one by one the list of ingredients that the user "would like to" (WL) and "Wouldn't like to" (WNL) include in the recipe, respectively.

There are two ways to add an ingredient to the query lists WL and WNL of ingredients:

1) Select an ingredient from the ontology. If this button is clicked then a new window shows the ontology tree where the user can select the ingredient, either specific, like orange or salmon, or generic, like Fruit or Fish.
2) Write the name of the ingredient in the text box provided to it. If the word entered by the user is a new ingredient, then the user is given the option to add it to the ontology of ingredients for future reference or use instead an ingredient existing in the ontology. A new ingredient entered by the user is ignored in the current query.

The *Suggest ingredient* button can be used to let the system complete the query WL list with other ingredients that are compatible with the ones previously included by the user. These suggested ingredients are obtained from the recipes case base, and are ingredients that typically appear together. This utility is one of the novel features of the new version of the system and it is described in Section 3.3.

The user can remove an ingredient from the query "I would like" or "I wouldn't like" lists with the *Remove* button after selecting the ingredient.

Once the query data have been introduced the system computes similarity and filters the k more similar recipes for the given query. The k value is configurable through the file menu. The resulting recipes are shown and annotated with a label that explains why this recipe has been retrieved. (R) annotates an ingredient of the WL list that appears in the retrieved recipe. (A) annotates an ingredient that is in the WL list but not in the recipe, so it has been substituted by a similar ingredient. And (C) annotates a specific ingredient that appears the WL list.

2.2. Menus Tab

The menus tab allows the user to compose three courses menus. The user starts by using one of the buttons "Starter" "Main Course" or "Desert" to query the system for a single dish. The system uses the same interface described in the previous section. Once the user enters a query for a starter, main course or dessert then the recommender system makes a secondary search to complete the rest of the menu.

The recommender subsystem has a case base of menus that is initialized with a file named "*menus.xml*". The menu recommender system searches in the menu case base for menus that are similar, i.e., they have similar courses, to the query menu, that is partially described menu. High scored menus in the case base are selected. Menu case base has been built collaboratively using the personal scores of non-expert users. The

"punctuation" button let you associate good/bad score to the menu. This score will be used in future recommendation cycles.

In Fig. 2 we see an example. The user queries for a starter with chicken and Chinese as the cuisine type. With the first choice "Asian Chicken Salad W/spicy Peanut Sauce", the system in the menus tab recommends a main course and a dessert.

Fig. 2. JaDaCook Main Window:menus tab

2.3 Recipes Tab

This tab offers a simple inspection utility of the 1484 recipes in the case base. The recipes are organized in 149 pages where the user can easily look for specific recipes.

3 Knowledge Acquisition

In JaDaCook 2, the knowledge of the system has evolved. Apart from the case base, the main source of knowledge of JaDaCook 2 is the ontology where we have conceptualized and formalized cooking knowledge from different sources, including experts, dictionaries and cooking web pages and systems [4-5]. In addition, the CCC-09 organizers provided a large case base of recipes for use by contest participants.

The acquired knowledge has been incrementally structured, conceptualized and formalized in the Web Ontology Language OWL[2] and the new version includes ingredients from the new recipes, new types of ingredients, new types of cuisine and dietary practices, and classification of recipes and ingredients according to these dietary practices, namely gout diet knowledge[3], seasonal food knowledge[4], and cholesterol diet knowledge[5].

3.1 Ontology

Like its predecessor JaDaCook 2 is based on an ontology that captures the terminological knowledge of the cooking domain, organizing objects (individuals) into categories (concepts) to enable inheritance of properties and to create taxonomies. There are animal origin ingredients, grouped as fish, meat, milk, cheese and eggs, each with other subclasess; plant origin ingredients, like cereals, nuts, fruits and vegetables; and other classes like sweeteners, drinks, and basic ingredients like salt and oil. In the new version of the ontology we have included additional knowledge. It has more than 300 ingredients, organized in types and classes that cover the 1484 recipes provided by the CCC-09 organizers. The OWL code of the ontology is available through the web page:
http://gaia.fdi.ucm.es/grupo/projects/cookingContest2/jadacook2/OntologiaIngredientes.owl

3.2 Cases

In the new version of the system we use the case base provided by the CCC-09 organizers. Cases are stored in an XML file, that is processed and the most relevant information is extracted and stored in case base memory structure in the precycle of the application. Processes of loading and storing cases have also been improved in this version using SAX and DOM [8]. Each case in the case base has mainly a text title, a list of ingredients and a textual description of the recipe development process. Each case is linked with the corresponding classes of ingredients in the ontology.

---

[2] http://www.w3.org/2004/OWL/
[3] Gout diet knowledge: http://www.everydiet.org/diet/gout-diet
[4] Seasonal food knowledge:
http://www.comidacasera.com/especiales/dietas/alimentos_temporada.phtml
[5] Cholesterol Diet knowledge: http://gicare.com/Diets/low-cholesterol-diet.aspx

3.3 Dependency Rules between Ingredients

We have applied the Apriori algorithm [9] to mine association rules using WEKA over the case base of recipes. The data set is a csv file obtained from the recipes case base, so it includes 1484 rows, and one Boolean attribute for each ingredient in the ontology. WEKA allows the resulting rules to be sorted according to different metrics such as confidence, leverage, and lift. We have performed different experiments with different values to extract a set of relevant rules. For example, the following rule indicates that if the recipe includes vanilla then it will include sugar with a confidence of 87% and support of 205 recipes.

vanilla=yes 205 ==> sugar=yes 179 conf:(0.87)

These rules allow capturing the degree of compatibility between ingredients. When the user asks for ingredients to include in the recipes, the system uses these rules to suggest ingredients that appear together with the previously selected ingredients in a large percentage of recipes.

3.4 Menu Case Base

The menu case base includes 133 menus acquired in a collaborative way by different users of the system. Each menu is composed of three courses taken from the recipe case base. If a user composes a menu using three single dish queries, then the menu is saved in an XML file to be reused in the future. Each menu has a numeric score that can be modified. This score represents the degree of satisfaction of the non-expert users that have tried this menu.

```
<MENU>
<PR>Fusilli Verde with Broccoli and Red Bell Pepper</PR>
<SE>Fast with Five: Garlic Flank Steak With Onion</SE>
<PO>Cocoa Espresso Cooler</PO>
<NOTA>7</NOTA> </MENU>
```

New menus and their scores will be included in the menu case base and will be taken into account for future menu recommendations.

4 CBR Processes

CBR processes in JaDaCook 2 take advantage of the ontology. The main usage of the cooking ontology has been centered on similarity assessment and ingredient substitution. The assumption here is that two ingredients are more similar if they are located closer in the ontology. And that one ingredient can be substituted for an ingredient that is classified in the same concept. For example, *turkey* and *chicken* are siblings and children of the concept “Fowl” (like *hen*, and *duck*).

The single dish case retrieval process consists of obtaining recipes that are similar to the given query. That means that the retrieved recipe includes the ingredients in the

WL list and it does not include ingredients from the WNL, has the dietary restrictions and type of cuisine specified in the query through the graphical interface (Fig. 1).

The retrieval method is k-nearest neighbor, which compares these characteristics and aggregates them obtaining a ranking similarity number to order the candidates. Then the k most similar are shown to the user. In JaDaCook 2 we have improved the similarity measure, and included more knowledge from the ontology.

Similarity assessment is based on two local similarity functions to compare titles and ingredients of the query and the case. Once all the ingredients have been processed, the overall similarity score is [0,1] normalized. Then the k most similar recipes will be shown to the user.

We use jCOLIBRI similarity functions, more specifically one of the concept based similarity functions that depends on the location of the cases in the ontology. Details about concept based similarity in jCOLIBRI can be found in [3]. If the ingredients in the WL list are concepts of the ontology that represent type of ingredients, then the children are obtained. To that end, we used the library OntoBridge [6] written in Java that provides management ontologies.

The reuse process in JaDaCook 2 is a process based on the substitution of ingredients according to their position in the ontology. The reuse strategy is the same one that was employed in JaDaCook 1 and it is called reinstantiation. It is one of the reuse methods included in the library of jCOLIBRI 2 and it is based on substituting one ingredient by one of its siblings in the ontology structure.

The stage of learning (retain) has as its primary function storing those cases which have been adapted into the knowledge base. Thus, these new cases may be used in future searches to improve retrieval performance. In this system, it is left to the user to decide which cases are retained, so in this way we store the cases that are most useful in problem solving.

5 Results and Examples

Next we provide the preliminary results for the queries in the single dish challenge provided by the CCC-09 organization through the web page. The rest of the queries, captures, the code of the system, and the documentation is available through the web page: http://gaia.fdi.ucm.es/projects/cookingContest2/cookingContest.html#JaDaCook

Q3: Prepare a low-cholesterol dessert with strawberries and avoid citrus fruits. (Main focus: dietary practice)

The WL list includes the 'strawberries' ingredient and we want to avoid 'citrus' so we include it in the WNL list. Dietary practices is set to 'Cholesterol diet'.

Fig. 3 shows the 3rd recipe suggested by the system, 'Fruit Cup#1' with a similarity value of 0.67. It includes strawberries and none of the citrus fruits: orange, lemon, lime and tangerine.

Q9: I do have a filet of beef, carrots, celery, field garlic and cucumber. Potatoes are available, too. For the dessert, we have oranges and mint. A soup would be preferable for the starter.

Q9 belongs to the Menu challenge. The WL list of the main course includes: 'beef', 'carrot', 'celery', 'garlic', 'cucumber' and 'potato'. The system answers with recipe 'Beef Stew with Zucchini' that includes 'beef', 'celery', and 'potato'. It also adapts the recipe

by replacing 'onion' by 'garlic', 'squash' by 'cucumber', and 'celery' by 'carrot'. For the dessert we choose 'orange' and 'mint' as required ingredients. The system answers with the recipe 'Indian Ice Tea'. As the starter the system does not allow looking for 'soup' as it is not available as an ingredient, or a type of ingredient, or dietary practice.

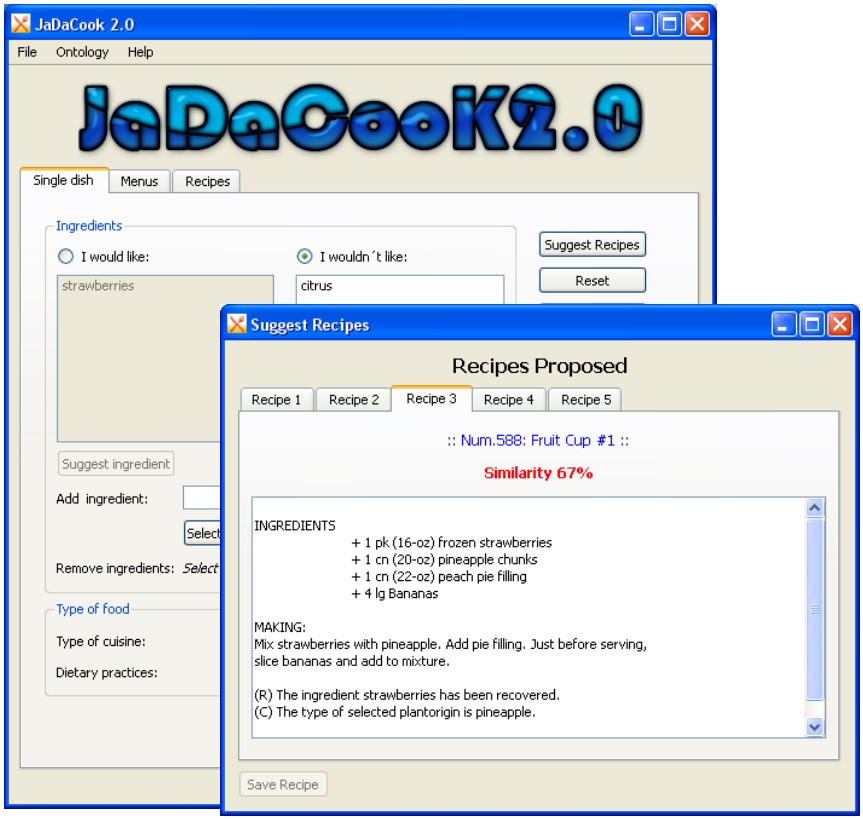


Fig. 3. Results for the query 3 of the single dish challenge provided by the CCC-09

6 Conclusions

In this paper we have described JaDaCook 2, a CBR recipe creation system submitted to the Computer Cooking Contest at ICCBR 2009.

JaDaCook 2 is a new version of JaDaCook 1 that participated in the 1st CCC in 2008. It addresses the main drawbacks of the first version, includes new functionality, improvements on the ontology, a new interface, and data mining capabilities to capture dependencies between ingredients.

JaDaCook 2 reasoning is based on a case base of recipes and an ontology with reusable knowledge about ingredients, types of ingredients, types of cuisine and dietary practices. The ontology is used as background knowledge to measure

similarity between ingredients and single dishes, and to substitute ingredients during adaptation.

The system is able to deal with the single dish and menu challenges, and it is able to learn new ingredients appearing in queries by including them into the hierarchical organization of ingredients. This is a kind of supervised learning. The system also learns new menus, as combinations of existing individual dishes.

The results for the given queries, the code of the system, and the documentation is available through the web page:
http://gaia.fdi.ucm.es/projects/cookingContest2/cookingContest.html#JaDaCook

Acknowledgments

This work has been supported by Spanish Ministry of Science and Education (TIN2006-15140-C03-02) and it has been partially supported by the Comunidad de Madrid Education Council and UCM(consolidated research group 910494).
We also thank the UCM students who participated in the development of JaDaCook 1: David Romero and Ignacio Rubio.

References

1. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J.A., and Sánchez, A. Building CBR systems with jCOLIBRI. Special Issue on Experimental Software and Toolkits of the Journal Science of Computer Programming, (2007). Elsevier Science.
2. Recio-García, J.A., Díaz-Agudo, B., Gómez-Martín, M.A., and Wiratunga, N. Extending jCOLIBRI for Textual CBR. Proceedings of Case-Based Reasoning Research and Development, 6th International Conference on Case-Based Reasoning, ICCBR 2005, Chicago, IL, US, Springer, (2005) 421-435.
3. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A., and Sánchez, A. Ontology based CBR with jCOLIBRI. In R. Ellis, T. Allen, and A. Tuson, editors, Procs. of AI-2006, the Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer. (2006) 149–162
4. Food Classification: http://www.mhlw.go.jp/english/topics/foodsafety/positivelist060228/dl/r04.pdf
5. Hammond, K. CHEF. In: Inside case-based reasoning. Hillsdale, NJ: Erlbaum.(1989).
6. OntoBridge: http://gaia.fdi.ucm.es/grupo/projects/ontobridge/
7. QT Qt Cross-Platform Whitepaper http://www.qtsoftware.com/products/
8. T. Cheung Lam, J. Jason Ding, J-Charn Liu. XML Document Parsing: Operational and Performance Characteristics. IEEE 2008.
9. Agrawal R, Imielinski T, Swami AN.:Mining Association Rules between Sets of Items in Large Databases.In *SIGMOD* **22**(2) (1993) 207-216

Computing the Path to Cooking Outside the Box

Rupali Bodhankar, Olasope Olinatori, John Emerson, Ashish Bindra, Kristoffer Canilang, Seth Galbraith, Glenda Hedden, Isabelle Bichindaritz

Institute of Techonology, University of Washington, Tacoma, WA
{ rupalb, ibby04, famine, abindra, kashiyuk, sethrg, glendh, ibichind }@u.washington.edu

Abstract. In this paper we describe BlueCook web based application developed to fulfill the requirements of the Computer Cooking contest organized by the International Conference on Case-based reasoning (ICCBR). Given a query of ingredients and other requirements, the task is to return the recipe that satisfies these constraints by reasoning from cases of previous cooking recipes. The data provided by the competition are the recipes in xml format. The approach chosen is to mine for cases from recipes, to mine for knowledge from food composition databases, and to facilitate reuse of cases through knowledge-based adaptation. An ontology of foods and cooking know-how, mostly mined from online resources, guides the adaptation tasks.

Keywords: Case-based reasoning, adaptation, case mining

1. Introduction

In this paper we present the BlueCook system developed to fulfill the requirements of the Computer Cooking Contest organized by the International Conference on Case based Reasoning 2009 [1]. The goal of the competition is answering queries by selecting and possibly modifying the recipes given. Queries will be described in free text but can be transformed manually to an arbitrary input format to be processed by the system. The example of a query is, 'Give me an Asian soup with leek'. The results produced by the system while answering the queries can be either a single or up to five recipes including in a note which original recipes from the recipe base have been used for the creation of the result. The contest has three tasks: one compulsory and two additional ones. Compulsory task involves returning a recipe based on the ingredients provided by the user. The additional tasks include adaptation challenge, and menu challenge. The adaptation challenge is to answer queries on pasta recipes that require an adaptation of both the list of ingredients and the preparation directions. The menu challenge requires the composition of a three-course menu based on the available recipes and the user specified constraints.

BlueCook system mines for cooking cases from the XML text-based recipes provided for the competition in order to structure the recipes along their ingredients / utensils / cooking directions. The given transformed recipes are saved in an MySql database. The XML of the recipes contains tags TI, IN, PR, and STEP. Tag <TI> is for title, <IN> for ingredients, <PR> for preparation, and <STEP> contains the preparation steps. A parser was developed to parse the given XML files of recipes

into the database. The parser separates the title, ingredients, ingredient's units, and other characteristics. It then arranges these elements in the appropriate tables in the database. The files do not have any special information about the recipes for example, cuisine type or meal type. An ontology of the cooking domain provides BlueCook with the information about what makes a recipe of specific cuisine type or meal type. This ontology has been created using the Protégé knowledge acquisition system.

The second section explains the BlueCook system architecture and components. In third section we explain the case representation principles and the case mining process is detailed in the fourth section. The fifth section focuses on the ontology built for the system. The sixth section contains the description of retrieval process and ranking mechanism used for the system. In the seventh section are presented some running examples of the system. Finally we provide a discussion, the conclusion and some future works.

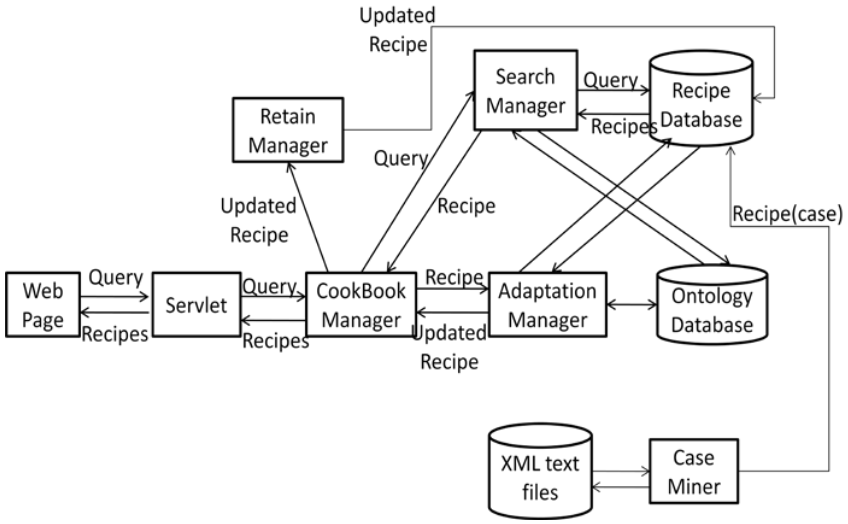


Figure 1. System architecture.

2. Architecture

BlueCook is a distributed Web application. It follows a classical three-tier model, web page-servlet-databases. The web page is developed using HTML and JavaScript. The servlet and the other programs are developed in Java. The architecture of the system is as follows. The user can enter the different parts of a query on the webpage in the form of cuisine type, meal type, ingredients, and dietary practice. These values entered on the web page will be posted on the servlet. Servlet calls the helping class CookBook Manager passing all these values. CookBook Manager calls the Search Manager to look for the recipe's exact match or the closest match that can be sent for adaptation. In the context of case-based reasoning [2], Adaptation Manager adapts the

recipes. CookBook Manager also performs Menu Planning. A more precise description of the main components as represented in Figure 1 is as follows:
CookBook Manager: This module communicates with the Servlet and all other modules after a query has been given by the user. The input of the CookBook Manager is the simple text query sent by the Servlet. The CookBook Manager returns 5 appropriate recipes to the Servlet to display on the webpage. This number was determined empirically by keeping the largest number of recipes required to answer all tested queries. CookBook manager also performs menu planning.
Search Manager: Given the elements in a query, search manager's job is to communicate with the database server to get and return 5 most related recipes i.e. the most similar recipes, represented by their identification (id). The database of the recipes is used by the Search Manager.
Adaptation Manager: Adaptation manager takes recipes passed by Search Manager and important components of the query as an input. It then communicates with the ontology database to see how the recipes can be updated. It returns the updated recipes to the CookBook Manager.
Case Miner: Case miner is used to extract the important parts of the XML recipes and structure recipes as cases, placed into the recipe database or case base.

The components developed for this first entry in the competition are Case Miner, Webpage, Servlet, Cookbook Manager, Search Manager and Adaptation Manager.

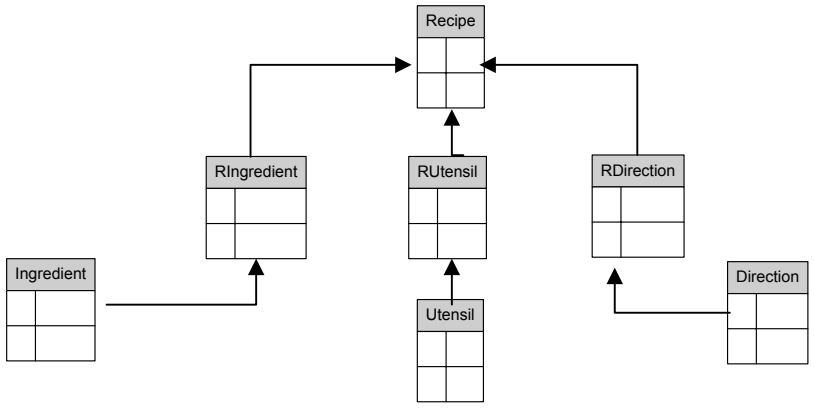


Figure 2. Case representation.

3. Case Representation

In this system we are supposed to use the given recipes to reuse them and adapt them. In the context of case-based reasoning, a case in BlueCook is a recipe. These recipes are provided in XML format. To make it easier for the retrieval and adaptation of the cases, BlueCook separates the parts of recipes using their tags: ingredients of the recipe, utensils, direction steps, in addition to recipe names. These are stored in separate tables in the case base. Each ingredient gets an ingredient-id which is then

connected to the ingredients in the recipes. Each recipe ingredient is stored separately along with the recipe-id. The cases are then retrieved from the case base for ranking and reuse steps. Further the directions and utensils of the recipe are identified by an id and linked with the recipes as well. The case base schema is represented in Figure 2.

4. Case Mining

In the case mining process, a parser extracts the information from the XML data and puts it into the structured relational database as one of the cases (see Figure 2). A case is composed of a list of ingredients, a list of utensils, and a list of directions. Isolating each of these compounds, in particular the ingredients and the directions, allows for the level of adaptation required for the Cooking Competition, since when substituting ingredients, some preparation steps for example need to be changed or removed. As each XML tag is parsed we run a set of rules on the text inside the tags. For example, for the ingredients line, the rule based processing currently provides information such as the amount and units of the ingredient, the cut type, and whether it is optional or not. It also uses a lexicon to identify abbreviations such as lg (for large), sm (for small), kg (for kilogram), etc. Another set of lexicon knowledge is also used for cut types (sliced, diced, peeled, etc). Each concept in the ontology database has a unique identifier. However a table of synonyms allows for mapping different terms to a unique concept identifier. Following are the key data members and methods. The natural language processing (NLP) parser used performs a syntactic analysis on the sentences in the recipe and separates them into sentences stored into the database textual fields. For example, a direction is parsed into the structure shown below:

```
CREATE TABLE RDirection (
      Rid INT NOT NULL,
       Did INT NOT NULL,
      Dnum SMALLINT NOT NULL,
      DConjunction VARCHAR (5),
      DSubject VARCHAR (50),
      DVerb VARCHAR(20),
       DDirectComplement VARCHAR (50),
      DAdverb VARCHAR (50),
      DPreposition VARCHAR(10),
      DIndirectComplement VARCHAR (50),
      DrelPrecondition VARCHAR(10),
      DnumPrecondition SMALLINT,
      DrelPostcondition VARCHAR(10),
      DnumPostcondition SMALLINT,
       CONSTRAINT PKdirection PRIMARY KEY (Rid, Did, Dnum),
      CONSTRAINT FKRDRecipe FOREIGN KEY (Rid) REFERENCES Recipe
                (Rid) ON DELETE NO ACTION ON UPDATE NO ACTION,
      CONSTRAINT FKRDirection FOREIGN KEY (Did) REFERENCES
                 Direction (Did)
                 ON DELETE NO ACTION ON UPDATE NO ACTION
       );
```

5. Ontology

The ontology is composed of an ingredients ontology and additional components for the dietary practices and seasonal foods.

5.1 Ingredient Ontology

The ontology developed for the system contains classical information about foods. On the webpage users can specify the meal type and cuisine type along with the ingredients. The cuisine type can be anything like Asian, Italian, and Mediterranean etc. The meal types are soup, salad, main dish, desert etc. In this ontology the information about the cuisine type, meal type and the ingredient type is stored in hierarchical manner. The information about the cuisine type includes the keywords that represent the particular cuisine type and appear in the title of the recipes. For example if cuisine type in the query is Asian, the keywords that represent an Asian dish would be 'Chinese, Thai, Korean, Asian …'. Similarly the information about the meal type includes the keywords that represent the meal type and can appear in the title of the recipes. For example, if the meal type is 'soup' then the keywords would look like 'soup, chowder'.

The ingredient type is a hierarchical path in the ontology from the root node 'Food' to the leaf nodes of ingredients. The ontology hierarchy for the ingredient is shown in Figure 3.

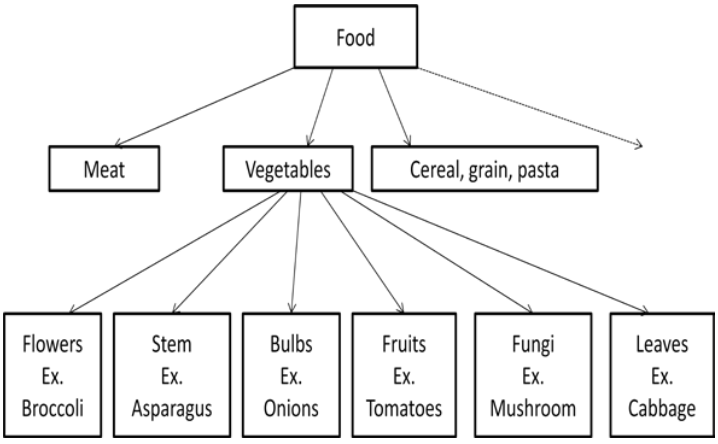


Figure 3. Ontology for Ingredient Type.

If the ingredient at hand is onion then the path returned by the ontology will be <Food, Vegetables, Bulbs>.

This ontology is being incrementally refined to allow more sophisticated adaptation. For example right now the vegetables tomatoes and okra both fall in the same category i.e. fruits, but it is not a good idea to replace them with each other in a recipe. For this reason need a more detailed ontology. The ontology also contains the food composition information about the contents in the ingredients for example sugar level, water level etcetera.

5.2 Dietary Practice

The dietary practices are restricted to Gout diet, Cholesterol diet, and Seasonal food. BlueCook followed the idea of the table of ingredients for dos and don'ts given on the Computer Cooking Contest website. It was expanded using an Internet search for the two dietary restrictions. The list of good and bad ingredients for each diet is stored as well. These lists are then used for ranking the recipes in the ranking step.

For the seasonal food the same data given by the competition is being stored, along with the list of ingredients that are in season and can be storable (kept for a long time) for each month of the year. Users can have a choice of changing the month when they are searching for the recipes with seasonal food. The system first searches for the recipes within season ingredients; if there is no recipe that is a match for in season ingredients then we search for the recipes with storable ingredients, which means ingredients that can be kept for a long time.

6. Retrieval and Ranking of the Recipes

The Search Manager is our retrieval component. The search of the recipes depending on the queries is guided using the ontology of the recipes. The query contains type of cuisine and type of meal. The information of the cuisine type is taken from the ontology database. This information includes the keywords that can be in the title of the recipe. For example if the cuisine type entered is Italian then the keywords include Italian, Pizza, Pasta, etc. The first step is to retrieve the recipes that contain these keywords in their title. Then the information about the meal type is retrieved from the ontology. This information is also a list of keywords that can appear in the title. The recipes that are retrieved from cuisine type are further sorted using this meal information to get the recipes that are of cuisine type and meal type. For example if the menu type is main dish then from all the Italian recipes we pick the recipes that are main dishes. These recipes are then checked for the exact match for query ingredients. If there is no exact match then recipes are checked for adaptation.

Ranking process is twofold. In the first step we eliminate the recipes that have ingredients that user specifically told to avoid or the ingredients that are restricted by the dietary restrictions provided by the user. The remaining recipes are used for ranking. In the second step of ranking process we need to find one recipe that is close to the ingredients given by the user, this recipe is the most adaptable recipe. To find the most adaptable recipe we evaluate the ingredients similarity between the recipe ingredient and the query ingredient. First we take the information about the type of ingredients from the ontology for each recipe and query ingredient. This

information contains the hierarchical path of the ingredient from the root node Food. For example, if we have the ingredient 'cabbage' the ontology will return {Food, Vegetable, Green vegetable, Cabbage family}. We try to match these paths of the ingredients.

If a query ingredient is $ingredient_1$ and a recipe ingredient is $ingredient_2$, then

$$similarity(ingredient_1, ingredient_2) = similarity(ingredient\_type_1, ingredient\_type_2)$$

For the similarity assessment, BlueCook uses the following rules.

1. If the paths of the ingredient types completely match from root node to leaf then recipe ingredient can be replaced by query ingredient and the match value 2 is given to the recipe ingredient.
2. If the two ingredients match only up to the food level, then the match value they get is 0.
3. If the ingredient types match anywhere in between just being food and matching completely then it will get match value of 1.

These 3 levels of match – namely 0, 1, and 2 – suffice in the current system since our hierarchy of ingredients has only two levels below the root. We will extend it to the number of levels matched when the ontology is expanded.

Using this method we look for a replacement for each ingredient from the query with ingredients from a recipe. For example if there are three ingredients provided by the user as carrots, celery and cucumber, then the matching step will look for replacement of each of these ingredients in the list of recipe ingredients.

Later in ranking the system also uses the list of dietary ingredients allowed for the given diet making it the fourth rule in ranking.

4. If one of the ingredients in the recipe is good for the diet requested by the user then the recipe gets extra 2 points in the ranking.

The recipe is ranked by taking the sum of all the match values of recipe ingredients and the extra points of diet ingredients.

$$rank\ of\ recipe = \sum_{i=1}^{n} rank(ingredient_i) + 2 * count(diet\ ingredients)$$

where n is the number of query ingredients provided, $ingredient_1$ to $ingredient_n$ are the selected replacements for query ingredients, and $count(diet\ ingredients)$ provides the number of diet favorable ingredients in the recipe.

All ranked recipes will be stored along with the ranking and ingredient replacements. The top ranked recipe will be selected for the adaptation. In the adaptation the ingredients in the recipe will be replaced by the ingredients in the query and this adapted recipe will be returned to the user.

7. Adaptation

The adaptation process is invoked when the recipe is not the exact match for the query ingredients. In the adaptation of a recipe for compulsory task, the recipe is updated by replacing the selected replacement ingredients in the recipe with the query ingredients. In the ranking process BlueCook keeps track of the replacement ingredient that was selected for the query ingredient. Using this information the adaptation manager changes the appropriate ingredients of the recipe with the query ingredients. The drawback of this is the measurements and the units of the ingredients may not match. However it will give the user an idea of which ingredients in the recipe are replaced by the query ingredients. Following, the system displays two recipes to the user – the original and the adapted. In the adapted recipe the replaced ingredients are highlighted in a different color so that the user will know about the changes.

In the adaptation challenge of the pasta recipes, two main changes are required. First, the units of the ingredients need to be adapted appropriately. For example one cup of some ingredient is not equal to one cup of another ingredient. This requires keeping track of the measurements of ingredients that are of same ingredient type. For example, butter and clarified butter come in the same ingredient type but the quantities that can added up to a recipe are different. So far, the adaptation is performed using procedural knowledge however we plan on changing to a declarative representation in a future version.

Secondly, the direction steps as well as the weights of the ingredients need to be changed. In the direction steps the ingredients are replaced according to the weight comparison of the recipe ingredient and some other factors related to food and culinary knowledge. For example, in some recipes if chicken can be replaced by beef, then BlueCook does not want to say beef wings. So the ontology needs to store this type of knowledge in order for the system to take care of such mistakes in the direction steps adaptation.

8. Example

In the compulsory task the user can provide different elements of the query. The query elements include cuisine type, meal type, ingredients to be in the recipe, ingredients to avoid from the recipe, dietary restrictions, and month for the seasonal food. If the query ‘Give me an Asian soup with leek, please consider that I follow gout diet’ is asked then the parts of the query that are entered are Cuisine type = Asian, Meal type = soup, ingredients to be in the recipe = leek. BlueCook will retrieve all the recipes that are Asian soups. These recipes are then sorted according to the dietary restrictions. If the recipe contains any restricted ingredient that is forbidden for the gout diet then that recipe will not be used. Although this could be an opportunity for using adaptation by substituting for an allowable ingredient, BlueCook in all tests so far could retrieve recipes that met all the requirements of the dietary restrictions. This would depend on the number and variety of recipes in the case base. The remaining recipes are ranked according to the exact match of ingredient ‘leek’ or

substitutability of ‘leek’ with some other ingredient in the recipe. The recipe gets extra points if ingredients good for gout diet are present. All the recipes are then placed in the priority queue according to their ranking. The top recipe is sent for the adaptation. In adaptation if the recipe is not an exact match for the query ingredient leek, BlueCook replaces an appropriate ingredient from the recipe with leek. This adapted recipe is sent to display on the webpage along with the original recipe. When an adapted recipe is displayed on the webpage, the replaced ingredient is highlighted.

9. Discussion

Several papers presented at the Computer Cooking Contest last year used the java based Case-Based Reasoning framework JCOLIBRI to develop their systems [3,4]. As opposed to them we developed our system without using any framework. There are major differences in the ontology of the participants from the last competition and the one we developed. In [3], the authors mentioned that they have various properties of the ingredients is-ingredient-type, is-made-of etc. Also in [5], the authors have developed food categories for the ingredients and engineered knowledge about ingredients. We have relied mostly on the reference food database from the US Department of Agriculture (USDA) [6] since a lot of the properties can be derived from the food composition, such as the texture of an ingredient, its taste, moisture, etc. Therefore our ontology has been mostly automatically learned from the knowledge encoded in this database, then we have refined it manually, which has been efficient in terms of knowledge acquisition. For the similarity previous authors rely on properties different from the ingredient type. Our similarity measure uses the ingredient type to determine how similar two ingredients are and if they can be replaced by each other. To assess the similarity between two ingredients, DeMiguel et al. use fuzzy relationship of two ingredients [3]. They have two different ways of searching for the recipe match ‘At least’ and ‘Just’. In ‘At least’ they look for a recipe with ingredients provided by the user and it does not matter if a few extra ingredients are also present. In ‘just’, they are looking for a recipe that has just the given ingredients [3]. Currently BlueCook has only one way of finding the recipe that is similar to this paper’s ‘at least’. Javier et al. have a different way of storing the ontology in which the recipe is stored as the root of the hierarchical structure and the ingredients are children [4]. In our ontology we stored the ingredient type in a hierarchical form, ‘Food’ being the root node and the ingredients being the leaf nodes. Recipes are stored in a relational database, which allows for fast search and retrieval, and can scale-up efficiently. Like Badra et al., this system uses text mining to mine for important elements in the recipes [7]. However we refer more precisely to case mining in our system since we deeply parse the recipes to transform them into well structured cases. This step allows for isolating for example preparation actions, which facilitates the adaptation for the adaptation challenge. Adeyanju et al. organize cases in feature vectors [8], our system follows a stricter case representation by using a relational database format. In comparison to Zhang et al. [9], our system adopts a knowledge rich approach, which is often followed in case-based reasoning since determining whether ingredients can be substituted requires this knowledge. However

a lot of this knowledge has been learned, or mined, from the USDA database, which has not implied an overload of work.

10. Conclusion

BlueCook's performance is driven by the quality of the ontology. The system we developed works satisfactorily for the current ontology. We have tested the system by generating sample queries and assessing how many of these were satisfactorily answered, from the average opinion of several evaluators. We would like to improve the ontology so that BlueCook will perform even better. The retrieval method will be changed to take into consideration the ingredients of the recipes. The adaptation will be also modified so that it will change the preparation of the ingredients in addition to the directions and ingredients. The ingredients' amount and measurement will be replaced according to their comparison with replacement ingredients. Currently just one adapted recipe is returned to the user. Later we are planning to give user a choice of five recipes.

References

1 Aamodt A, Plaza E. Aamodt: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System. AI Communications, 7(1) (2004) 39-59

2 Richter M.M., Aamodt A.: Case-based Reasoning Foundations. The Knowledge Engineering Review, Volume 20, Issue 03, September 2005 (2005) 203-207

3 DeMiguel J., Plaza L., Diaz-Agudo B.: ColibriCook: A CBR System for Ontology-Based Recipe Retrieval and Adaptation. In: European Conference On Case-Based Reasoning 2008 Workshop Proceedings, Trier, Germany (2008)

4 P. Javier Herrera, Pablo Iglesias, David Romero, Ignacio Rubio, Belen Diaz-Agudo. JaDaCook: Java Application Developed and Cooked Over Ontological Knowledge. In: European Conference On Case-Based Reasoning 2008 Workshop Proceedings, Trier, Germany (2008)

5 Hanft A., Ihle N., Bach K., Newo R., Manz J.: Realizing a CBR-based Approach for Computer Cooking Contest with e:IAS. In: European Conference On Case-Based Reasoning 2008 Workshop Proceedings, Trier, Germany (2008)

6 US Department of Agriculture. USDA National Nutrient Database for Standard Reference Release 18. Online resource available at http://www.nal.usda.gov/fnic/foodcomp/Data/, date retrieved March, 15, 2009

7 Badra F., Bendaoud R., Bendaus R., Bentebitel R., Champin P.-A., Cojan J., Cordier A., Despres S., Jean-Daubias S., Lieber J., Meilender T., Mille A., Nauer E., Napoli A., Toussaint Y.: TAABLE: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In: European Conference On Case-Based Reasoning 2008 Workshop Proceedings, Trier, Germany (2008).

8 Adeyanju I., Craw S., Ghose A., Gray A., Wiratunga N.: RaGoUt: An Arpeggio of Tastes; European Conference On Case-Based Reasoning 2008 Workshop Proceedings, Trier, Germany (2008)

9 Zhang Q., Hu R., Mac Namee B., Delany S.J.: Back to the Future : Knowledge Light Case based Cookery. In: European Conference On Case-Based Reasoning 2008 Workshop Proceedings, Trier, Germany (2008)