ICCBR 2010

The Eighteenth International Conference on

Case-Based Reasoning

Alessandria, Italy
July 19–22, 2010

Workshop Proceedings

Volume Editor

Cindy Marling
School of Electrical Engineering and Computer Science
Ohio University
Athens, Ohio, USA
marling@ohio.edu

Preface

Welcome to the Workshop Proceedings of the Eighteenth International Conference on Case-Based Reasoning (ICCBR 2010)! This year's conference, which consolidates ICCBR with the European Conference on Case-Based Reasoning (ECCBR), is held in Alessandria, Italy, from July 19 through July 22, 2010. The Workshop Program enables dissemination, demonstration, and discussion of research in progress, facilitating interaction, feedback, and collaboration at the forefront of CBR research and development. Five workshops are included in this year's program.

Case-Based Reasoning for Computer Games: This is the third in a series of highly successful workshops focusing on CBR in computer gaming environments. The six papers included in this volume present work in meta-reasoning for adaptive behavior, case extraction from game replay repositories, automated feature selection, reinforcement learning, generic intelligent gaming frameworks, and domain-independent learning.

Provenance-Aware CBR: Applications to Reasoning, Metareasoning, Maintenance and Explanation: New this year is an exploration of case provenance and its roles in trust and reputation, reasoning and metareasoning, and explanation. Provenance encompasses the sources of cases, case acquisition contexts, derivation traces, and other meta-data. Five papers illustrate the use of provenance-aware CBR for exploiting traces as knowledge sources and experience containers, supporting explanation, and enriching the CBR process.

WebCBR: Reasoning from Experiences on the Web: This second WebCBR workshop promotes CBR as a way to capture, represent and reuse unstructured experiential Web content and Web usage data. Six papers represent new research directions and applications. Technological issues addressed include similarity and retrieval of textual content, case vocabulary mining, and the interoperation of CBR with the Web of Data. WebCBR applications described include support for authoring new product reviews, making proactive recommendations, and adapting travel itineraries based on past knowledge searches.

CBR Startups: This novel workshop elucidates the opportunities and challenges of commercializing cutting-edge CBR research. In true CBR fashion, CBR researchers who have created startup companies will share their own past experiences with those who could benefit from retrieving and reusing the experiences retained. While peer-reviewed technical papers would be out of place at this venue, the organizers have provided a workshop overview for inclusion in this volume.

Computer Cooking Contest: This will be the third workshop held in conjunction with the Computer Cooking Contest (CCC), a competitive, entertaining and lively arena for fostering the novel development of AI approaches, techniques and integrations. Five teams will field running systems, completing diverse cooking challenges, in a live competition. Technical papers describing the associated research and development will be presented at this workshop.

Many thanks are due to the organizers of these workshops, the individual workshop program committees, and the contributing authors, who together have created this vibrant Workshop Program. I would also like to express my appreciation to last year's Workshops Chair, Sarah Jane Delany, for her derivation traces, LaTeX templates, and advice. The provenance, of course, extends back through the first seventeen International and European Conferences on Case-Based Reasoning. As always, it is both an honor and a delight to work with ICCBR Conference Chairs Isabelle Bichindaritz and Stefania Montani.

I hope that participants will find this year's Workshop Program to be a fruitful and stimulating experience, and that these proceedings will help to extend the experience beyond ICCBR 2010.

Cindy Marling July 2010

Welcoming you, from left to right, are Isabelle Bichindaritz, Cindy Marling, and Stefania Montani. *Photo by Peter Funk*

Table of Contents

Case-Based Reasoning for Computer Games

Workshop at the
Eighteenth International Conference on
Case-Based Reasoning

Alessandria, Italy
July, 2010

ICCBR 2010

Manish Mehta, Santiago Ontañón and
Antonio A. Sánchez-Ruiz (Eds.)

Co-Chairs

Manish Mehta
Georgia Institute of Technology, USA
mehtama1@cc.gatech.edu

Santiago Ontañón
IIIA-CSIC, Spain
santi@iiia.csic.es

Antonio A. Sánchez-Ruiz
Universidad Complutense de Madrid, Spain
antonio.sanchez@fdi.ucm.es

Program Committee

Preface

Computer games are an excellent testbed for artificial intelligence techniques for two main reasons. First, they are exceptionally challenging domains that require real-time reasoning, uncertainty management, on-line learning, and strategic reasoning among others. Second, the computer game industry is growing to be larger every year, being closely matched with the movie industry. Therefore, solving problems related to games has a direct strong impact in industry. For those two reasons, in the last few years the community of artificial intelligence and specially CBR researchers working in the domain of games is steadily increasing.

Since the presence of the CBR community in the AI and games related conferences such as AIIDE or CIG is ever increasing, the motivation for this workshop is to encourage the exchange of ideas among the CBR and computer games community. This workshop can be seen as the third in a series of successful workshops, the first one held at ICCBR 2005 in Chicago, USA, and the second one, last year at ICCBR 2009 in Seattle, USA.

Six papers are to be presented at this workshop. Various gaming environments are represented in these papers, from real-time strategy games and robotic soccer. Notable of mention is the Starcraft domain, that is attracting a lot of attention due to the Starcraft AI competition to be held at the AIIDE conference this year. The technical contributions of these papers range from generic frameworks to connect AI to games, to meta-reasoning, feature extraction and reinforcement learning.

Mehta et al. present advances on the use of meta-reasoning to achieve adaptive behavior in the context of real-time strategy games. They use Warcraft II as their evaluation domain.

Weber and Ontañón present an approach for automatically extracting cases from large collections of already existing replays for the real-time strategy game Starcraft to be used in the context of a case-based planner. One of the most interesting contributions of this paper is raising the idea that the enormous amount of game replays available online for some games can be used as an automatic source of cases.

Acosta et al. focus on the domain of the RoboCup to evaluate techniques for automated feature selection. They compare a binary search-based technique to a rough set-based technique, showing that rough sets, although achieving slightly worse performance, perform feature selection in a much less computationally expensive way, enabling the use of automatic feature selection when large collections of features are available.

Mowery et al. combine CBR and reinforcement learning for learning how to micro-manage in the domain of Starcraft. They study on-line learning techniques that automatically learn cases while playing, and study the trade-offs resulting from learning too many cases.

Gómez et al. present a generic framework to connect AI bots to computer games, which has specifically been designed with CBR in mind. The main features of this framework with respect to alternatives are specific support for storing game traces and a complex action representation which can be used to monitor the execution of actions at run-time.

Floyd and Esfandiari focus on domain-independent learning by observation, and present case studies in the domains of RoboCup, a space combat computer game and on physical robots.

Overall, these papers represent a good sample of the recent trends in CBR and computer games. Finally, we would like to thank everyone who contributed to the success of this workshop, especially the authors, the program committee members, and the organizers of the ICCBR 2010 conference.

Manish Mehta July 2010
Santiago Ontañón
Antonio A. Sánchez-Ruiz

Meta-Level Behavior Adaptation in Real-Time Strategy Games

Manish Mehta[1], Santiago Ontañón[2], and Ashwin Ram[1]

[1] CCL, Cognitive Computing Lab Georgia Institute of Technology,
Atlanta, GA 303322/0280,
{mehtama1,ashwin}@cc.gatech.edu
[2] IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra (Spain),
santi@iiia.csic.es

Abstract. AI agents designed for real-time settings need to adapt themselves to changing circumstances to improve their performance and remedy their faults. Agents typically designed for computer games, however, lack this ability. The lack of adaptivity causes a break in player experience when they repeatedly fail to behave properly in circumstances unforeseen by the game designers. In this paper, we focus on an AI technique for game playing agents that helps them adapt to changing game circumstances. The agents carry out runtime adaptation of their behavior sets by monitoring and reasoning about their behavior execution and use this reasoning to dynamically carry out revisions on the behaviors. The evaluation of the behavior adaptation approach in a complex real-time strategy game shows that the agents adapt themselves and improve their performance by revising their behavior sets appropriately.

1 Introduction

Faults during a problem solving episode provide both humans and artificial agents strong cues on what needs to be learned [4]. AI agents can thus benefit from approaches that provide them the ability to learn from their failed encounters in the world. The need for adaptivity for agents that play modern computer games has been emphasized as well [11]. This paper presents an approach that addresses this need for AI game playing agents to be adaptive, learning from their experiences to improve their performance.

AI agents for games are typically created using hand authored behaviors that are static in nature [10]. This results in agents that cannot dynamically adapt themselves to changing scenarios within the game. Incorporating knowledge representation and planning techniques can enable an agent's behavior to become more flexible in novel situations, but it does not guarantee success: when an agent's behavior fails to achieve its desired purpose, most agents are unable to identify such failure and will continue executing the ineffective behavior. Ideally we need self-adapting AI agents for games that can learn from their own experience and adapt themselves. The problem is harder for state of the art real time games. Most of them involve complex strategic behaviors, such as real-time strategy (RTS) games, or involve characters that must behave human-like in a believable way. Both kind of games have huge decision spaces and require real time performance which complicate the creation of adaptive AI approaches.

In this paper, we address the problem of lack of adaptivity in RTS game playing agents. Our proposal involves the use of meta-reasoning [3] to identify the execution failures and their causes and carrying out appropriate revisions in response. We have developed the meta-reasoning approach to address two classes of failures. The first class of failures are based on deliberating over the analyzed differences across successful and failed executions and appropriately addressing the important differences. The second class of failures represent anomalous situations that can be identified from a game playing episode and need to be appropriately addressed. In our previous work [7], we focused explicitly on addressing the first class of failures and fixes needed to resolve the analyzed differences (across successful and failed executions) and the corresponding experimental evaluation looked at the performance of a specific game playing system, Darmok [13], with and without the meta-reasoning layer.

In the work presented in this paper, our focus is on the second class of failures. Our approach is based on using a collection of author-defined *failure patterns*, which are used to represent explicit descriptions of anomalous situations. Failure patterns provide a case-based solution for detecting failures in the execution of the base reasoner. After failures have been detected, fixes are made in the system to prevent those failures from happening again. These failure patterns explain what went wrong with the executed actions in the world (and not just differences across successful and failed system executions). This paper has three main contributions. First, we present a meta-reasoning approach that can be used in real-time game domains with limited number of trials to improve the performance. Second, instead of directly modifying the behaviors being run by the game playing agent, our meta-reasoning module creates *daemons*, which operate over the executing behaviors in real time to detect and prevent the anomalous situations from happening again. Third, and most important, we propose to use *failure patterns*. Instead of designing behavior sets which can generate correct behavior in all situations, failure patterns allow the game designer to specify a set of anomalous situations, and associate possible fixes for them (failure patterns). We argue that this approach alleviates the behavior creation process, and, coupled with the proposed meta-reasoning module, achieves adaptive behavior for real-time games.

Our test results show that our proposal improves the performance over different game playing sessions. Meta-reasoning systems are composed of a base reasoner that is in charge of the performance task (in this case playing an RTS game), and a meta-reasoner that observes and modifies the base reasoner's behavior. As in our previous work [7], in this paper we are going to use Darmok [9] as the base reasoning system, that is a case-based planning system designed to play RTS games. The resulting system by applying our meta-reasoning approach to Darmok is called Meta-Darmok.

The rest of the paper is organized as follows. First we present the related work. Then, we introduce Darmok and the RTS game used in our experiments. After that, we present our meta-adaptation approach, the Meta-Darmok system, and the empirical evaluation of it. Finally we conclude the paper and discuss future steps.

2 Related Work

Meta-reasoning [3] is the process of monitoring and control of reasoning. The control part involves the meta-level control of the computational resources spend in specific tasks (attention). The monitoring part involves figuring out what went wrong and why through *introspective reasoning*. Our system explores the later aspect of meta-reasoning. The Meta-Aqua system, for example, uses a library of pre-defined patterns of erroneous interactions among reasoning steps to recognize failures in the story understanding task [4]. The Autognostic system [12] uses a model of the system to localize the error in the system's element and and uses the model to construct a possible alternative trace which could lead to the desired but unaccomplished solution. IULIAN [8] uses questions about its own reasoning and knowledge to re-index its memory and to regulate its processing. These approaches typically assume the agent is the sole source of change, actions are deterministic, take unit time, and their effects well defined, and that the world is fully observable. In RTS games all these assumptions are violated. Furthermore, these approaches are generally applied to detect failures in plans consisting of STRIPS operators or minor extensions of STRIPS; using it for revising complex behavior sets for game domains requires novel failure detection techniques, a vocabulary of failure patterns pertinent to game domains and new behavior modification strategies.

One of the approaches to planning that deal best with exogenous events and non-determinism are those based on Markov decision processes (MDP), focusing on learning a policy. These approaches, however, require a large number of iterations to converge and only do so if certain conditions are met. In complex game domains, these techniques are intractable (MDP learning algorithms require a polynomial time in the number of states of a problem [5], which in the case of complex games is prohibitively large). Further, these approaches generalize poorly. The playing style of a human player can significantly change the game dynamics; the learned static policy might have to be retrained to accommodate such changes.

More recent work has relaxed these assumptions and has been applied to more complex game domains [2, 14]. Another body of related work applied to complex game domains is in Adaptive AI [11]. Dynamic Scripting is a technique based on reinforcement learning, that is able to generate "scripts" by drawing subsets of rules from a pre-authored large collection of rules. If the rules are properly authored, different subsets of those rules provide meaningful and different behaviors. Dynamic scripting is a technique for learning which subsets of those rules work better under different circumstances, by learning which rules work good or bad, altering their probability of being selected again.

3 Application Domain: WARGUS and Darmok

We have used WARGUS, a real time strategy game, as our game domain. Each player's goal in WARGUS is to survive and destroy the other players. Each player has a number of troops, buildings, and workers (who gather resources such as gold, wood and oil in order to produce more units). Buildings are required to produce more advanced troops, and troops are required to attack the enemy. The calculations inherent in the combat system make the game non-deterministic. The game involves complex strategic reasoning, such as terrain analysis, resource handling, planning, and scheduling, all of them under tight time constraints. WARGUS (sometimes referred to as "stratagus") is a well known domain, and has been used by several researchers as a test bed for AI techniques [1, 6].

Darmok is a real time planning and execution system that has been designed to play games such as WARGUS and is capable of dealing with both the vast decision spaces and the real-time component of RTS games. [9]. Darmok combines learning from demonstration with case-based planning:

– *Learning from Demonstration*: Darmok learns behaviors (stored as cases to be later used using case-based planning), by analyzing annotated game traces. An annotated game trace is a log of the actions executed by a human expert to play and win a game, which includes annotations of which goals were pursued with each action.
– *Case-based Planning*: Darmok uses the behaviors learnt from demonstration through case-based planning. Cases are retrieved, executed and adapted on real time. When plans fail, they are retracted, and new adaptations or new cases are attempted.

Darmok plays the complete game without any abstraction, and it takes every single decision in the game, beginning to end. Moreover, Darmok's degree of proficiency depends highly on the demonstrations being provided. If good demonstrations are provided, Darmok can learn to defeat the built-in AI of WARGUS in a variety of maps. Darmok is capable of taking behaviors observed from a human in a demonstration and adapt them for different situations. However, due to the complexity of the domain, this adaptation is a complex procedure, and Darmok does not always succeed in producing a good enough adaptation. Moreover, Darmok has to decide which behaviors, from the set of behaviors observed from humans, to use in each map. This selection is performed by both assessing game state similarity and goal similarity, however, sometimes Darmok fails to assess the current situation, and a suboptimal behavior is selected. The meta-level behavior adaptation approach presented in this paper uses Darmok as the base reasoning system to execute the behaviors that play the game, and monitors the behavior of Darmok, in order to fix it when failures are detected.

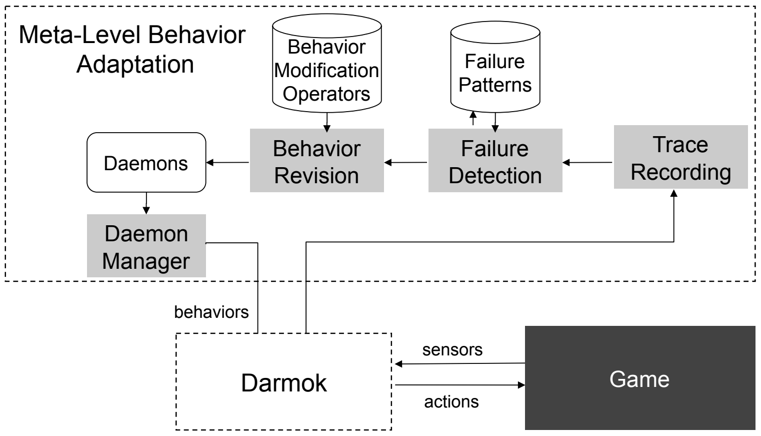


Fig. 1. The proposed behavior modification architecture.

4 Meta-Level Behavior Adaptation

Once it has learnt, the Darmok system is not able to modify the behaviors it has learnt. Although Darmok has the capability of learning from experience (by remembering which behaviors succeeded and which ones failed in different situations), it does not have any capability to fix the behaviors in its case base. If the expert that Darmok learnt from made a mistake in one of the behaviors, Darmok will repeat that mistake again and again each time Darmok retrieves that behavior. The meta-reasoning approach presented in this paper provides Darmok exactly with that capability, resulting in a system called *Meta-Darmok*, shown in Figure 1. By analyzing past performances, Meta-Darmok can fix the behavior resulting from the behaviors in its case base.

Our meta-level behavior adaptation approach consists of four parts: *Trace Recording*, *Failure Detection*, *Behavior Modification*, and the *Daemon Manager*. During trace recording, a trace holding important events happening during the game is recorded. Failure detection involves analyzing the execution trace to find possible issues with the executing behaviors by using a set of *failure patterns*. These failure patterns represent a set of pre-compiled patterns that can identify the cause of each particular failure by identifying instances of these patterns in the trace. Once a set of failures has been identified, the failed conditions can be resolved by appropriately revising the behavior using a set of *behavior modification routines*. These behavior modification routines are created using a combination of basic modification operators (called *modops*, as explained later). The modifications are inserted as *daemons*, which monitor for failure conditions to happen during execution when Darmok retrieves some particular behaviors. A daemon manager triggers the execution of such daemons when required.

4.1 Trace Recording

The behavior execution system during execution records a trace that contains information related to basic events including the name of the behavior that was being executed, the corresponding game state when the event occurred, the time at which the behavior started, failed or succeeded, and the delay from the moment the behavior became ready for execution to the time when it actually started executing. All this information is recorded in the execution trace, which the system updates as events occur at runtime. The trace provides a considerable advantage in performing behavior adaptation with respect to only analyzing the instant in which the failure occurred, since the trace can help localize portions that could possibly have been responsible for the failure. Once a game finishes, an *abstracted trace* is created from the execution trace that Darmok generates. The abstracted trace is the one used by the rest of components of the meta-reasoning module. The abstracted trace, consists of various relevant domain-dependent features: like information regarding units such as hit points, or location, information related to units that were idle, killed or attacked and the cycles at which this happens. The abstracted trace also contains basic behavior failure data that consists of the apparent reason for behavior failures, such as whether it was due to insufficient resources or not having a particular unit available to carry out a behavior. The abstracted trace is used to find occurrences of the failure patterns.

4.2 Failure Detection

Failure detection involves localizing the fault points. Although the abstracted execution trace defines the space of possible causes for the failure, it does not provide any help in localizing the cause of the failure. Traces can be extremely large, especially in the case of complex RTS games on which the system may spend a lot of effort attempting to achieve a particular goal. In order to avoid this potentially very expensive search, a set of pre-compiled patterns of failures can be used to help identify the cause of each particular failure by identifying instances of these patterns in the trace [4]. The failure patterns essentially provide an abstraction mechanism to look for typical patterns of failure conditions over the execution trace. The failure patterns simplify the blame-assignment process into a search for instances of the particular problematic patterns.

Failure patterns are defined as finite state machines (FSMs) that look for generic patterns in the abstracted trace. An example of a failure pattern represented as FSM is *Very Close Resource Gathering Location failure* (VCRGLfail) (shown in Figure 2) that detects whether a peasant is gathering resources at a location that is too close to the enemy compared to other possible resource gathering locations. This could lead to an opening for enemy units to attack early. Other examples of failure patterns and their corresponding behavior modification operators are given in Table 1. Each failure pattern is associated with modification routines. When a failure pattern generates a match in the abstracted trace, an instantiation of the failure pattern is created. Each instantiation contains which were the particular events in the abstracted trace that matched with the

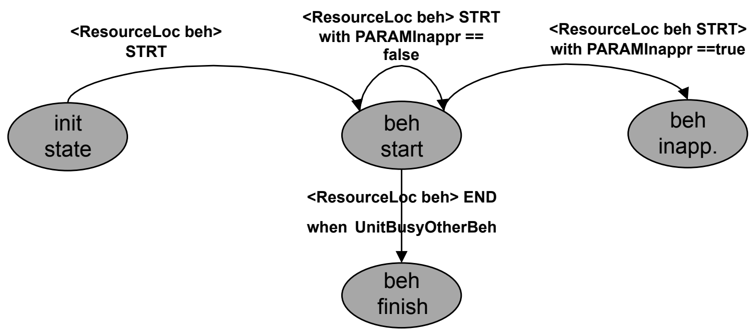

<ResourceLoc beh> STRT
<ResourceLoc beh> STRT with PARAMInappr == false
<ResourceLoc beh STRT> with PARAMInappr ==true
init state
beh start
beh inapp.
<ResourceLoc beh> END when UnitBusyOtherBeh
beh finish

Fig. 2. The figure shows the FSM corresponding to the failure pattern VCRGLfail. STRT, END represent the status of the behavior. PARAMInappr. represents the routine that checks whether the location where peasant is gathering resources is at a location that is too close to the enemy compared to other resource gathering locations.

pattern. This is used to instantiate particular behavior modification routines that are targeted to the particular behaviors that were to blame for the failure.

4.3 Behavior Modification

Once the cause of the failure is identified, it needs to be addressed through appropriate modification. These modifications are in the form of inserting or removing steps at the correct position in the failed behavior, or changing some parameter of an executing behavior. Once the failure patterns are detected from the execution trace, the corresponding behavior modification routines and the failed conditions are inserted as daemons for the map in which these failed conditions are detected. The daemons act as a meta-level reactive plan that operates over the executing behavior at runtime. The conditions for the failure pattern become the preconditions of the daemon and the behavior modification routine consisting of basic modops become the steps to execute when the daemon executes. The daemons operate over the executing behavior, monitor their execution, detect whether a failure is about to happen and repair the behavior according to the defined plan modification routines. Notice that Meta-Darmok does not directly modify the behaviors in the case-base of Darmok, but reactively modifies those behaviors when Darmok is executing them.

In the current system, we have defined 20 failure patterns and behavior modification routines for WARGUS. The way Meta-Darmok improves over time is by accumulating the daemons that the meta-reasoner generates (which are associated to particular maps). Thus, over time, Meta-Darmok improves performance by learning which combination of daemons improves the performance of Darmok for each map. The adaptation system can be easily extended by writing other patterns of failure (as described in [15]) that could be detected from the abstracted trace and the appropriate behavior modifications to the corresponding behaviors that need to be carried out in order to correct the failed situation.

Failure Pattern	**Behavior Modification Operator**
Resource Idle failure (e.g., resource like peasant, building, enemy units could be idle)	Utilize the resource in a more productive manner (for example, send peasant to gather more resources or use the peasant to create a building that needed later on)
Very Close Resource Gathering Location Failure	Change the location for resource gathering to a more appropriate one
Inappropriate Enemy Attacked failure	Direct the attack towards the more dangerous enemy unit
Inappropriate Attack Location failure	Change the attack location to a more appropriate one

Table 1. Some example failure patterns and their associated behavior modification operators in WARGUS

5 Empirical Evaluation

To evaluate our behavior adaptation approach, we conducted two different experiments turning the behavior adaptation on and off respectively. The experiments were conducted on 8 different variations of the 2-player version of the classical map “Nowhere to run, Nowhere to hide”(NWTR), one of the maps from *Battlenet* regularly player by human players, characterized by a wall of trees that separates the players. This map leads to complex strategic reasoning, such as building long range units (such as catapults or ballistas) to attack the other player before the wall of trees has been destroyed, tunneling early in the game through the wall of trees trying to catch the enemy by surprise, or other strategies. The focus of the experiments is the ability of meta-reasoning layer to detect anomalous situations and revise the behavior sets. The results are reported for 24 games in 6 different scenarios. These 6 scenarios correspond to 6 different expert demonstrations from NWTR maps. As Darmok can learn from more than one demonstration, we evaluate results when Darmok learns from one upto six demonstrations. Each one of the expert demonstrations exemplified different techniques with which the game can be played: fighters rush, knights rush, ranged attacks using ballistas, or blocking the enemy using towers.

Figure 3 shows the average results for all the 144 games in terms of number of wins, draw and losses. The figure also shows average player and opponent score (where the “score” is a number that WARGUS itself calculates and assigns to each player at the end of each game). Finally, WP shows the win percentage, presenting the improvement in win percentage comparing adaptation with respect to no adaptation. The results show that behavior adaptation leads to an improvement of the percentage of wins as well as the player score to opponent score ratio. Figure 4 shows the overall system improvement plotted against the number of traces. An improvement occurs in all cases irrespective of the number of traces used. Overall system performance improved considerably (overall increase of 62.3% on average).

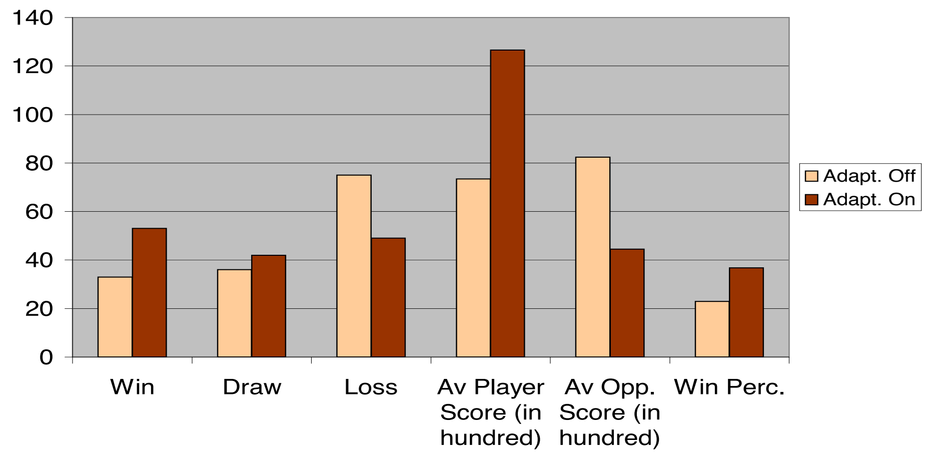


Fig. 3. The figure shows the average results from experiments turning the behavior adaptation on and off

6 Conclusion and Future Work

We aim to create adaptive agents for complex real-time games that learn from their experience and avoid their mistakes. In this paper, we have presented an approach based on meta-reasoning to behavior adaptation that achieves this. It is based on the use of failure patterns in order to ease blame assignment, a vocabulary of failure patterns to detect anomalous situations pertinent to game domains and behavior modification strategies to figure out what went wrong and the reasoning behind it. We have shown that *daemons* can be used to introduce reactive elements in the execution of the system that will adapt the behavior if failures are detected in real time. Our approach has been implemented and tested in a real time strategy game WARGUS. Our experimentation results indicate that overall performance of the system improves with the introduced behavior adaptations. One of the issues in the current behavior modification system is introduction of conflicting changes which can possibly result in degradation of performance. Some of the introduced changes cause unwanted revisions of the behavior causing a degradation in system performance. We plan to address this issue in the future. We intend to run more experiments on different maps to further test our approach and identify other potential issues in the future.

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 Lecture Notes in Artificial Intelligence, pages 5–20, 2005.
2. Tristan Cazenave. Metarules to improve tactical go knowledge. *Inf. Sci. Inf. Comput. Sci.*, 154(3-4):173–188, 2003.

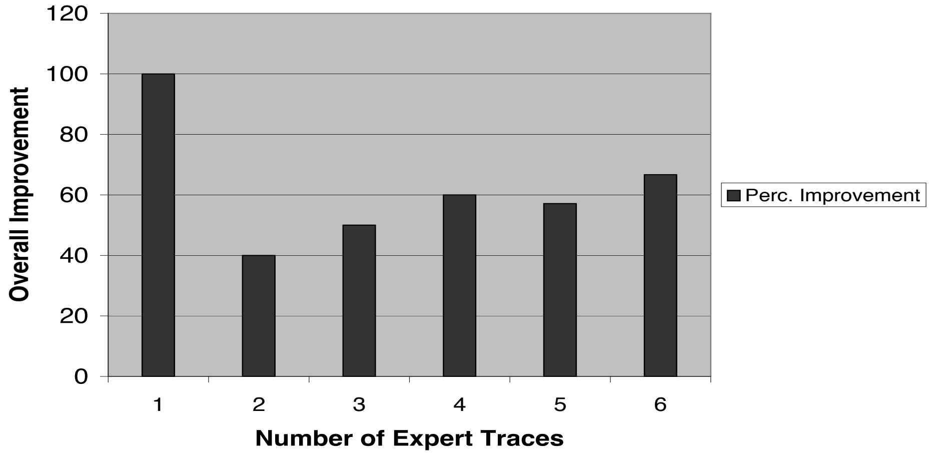


Fig. 4. The figure shows the overall percentage improvement using the behavior adaptation approach

3. Michael T. Cox. Metacognition in computation: a selected research review. *Artif. Intell.*, 169(2):104–141, 2005.
4. Michael T. Cox and Ashwin Ram. Introspective multistrategy learning: On the construction of learning strategies. Technical report, 1996.
5. Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.
6. Josh McCoy and Michael Mateas. An integrated agent for playing real-time strategy games. In *AAAI*, pages 1313–1318. AAAI Press, 2008.
7. Manish Mehta, Santiago Ontañón, and Ashwin Ram. Using meta-reasoning to improve the performance of case-based planning. In *ICCBR 2009*, 2009.
8. Rudiger Oehlmann. Metacognitive adaptation: Regulating the plan transformation process. In *Proceedings of the Fall Symposium on Adaptation of Knowledge for Reuse*, 1995.
9. Santiago Ontañón, Kinshuk Mishra, Neha Sugandh, and Ashwin Ram. On-line case-based planning. *Computational Intelligence Journal*, 26(1):84–119, 2010.
10. S. Rabin. *AI Game Programming Wisdom.* Charles River Media, 2002.
11. Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen Kuyper, and Eric Postma. Adaptive game ai with dynamic scripting. *Machine Learning*, 63(3):217–248, 2006.
12. Eleni Stroulia and Ashok K. Goel. Functional representation and reasoning in reflective systems. *Journal of Applied Intelligence*, 9:101–124, 1995.
13. Neha Sugandh, Santiago Ontañón, and Ashwin Ram. Real-time plan adaptation for case-based planning in real-time strategy games. In *ECCBR 2008*, pages 533–547, Berlin, Heidelberg, 2008. Springer-Verlag.
14. Patrick Ulam, Joshua Jones, and Ashok K. Goel. Combining model-based meta-reasoning and reinforcement learning for adapting game-playing agents. In *AIIDE*, 2008.
15. 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.

Using Automated Replay Annotation for Case-Based Planning in Games

Ben G. Weber[1] and Santiago Ontañón[2]

[1] Expressive Intelligence Studio
University of California, Santa Cruz
bweber@soe.ucsc.edu
[2] IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
santi@iiia.csic.es

Abstract. A major challenge in the field of case-based reasoning is building case libraries representative of the state space the system will encounter. We approach this problem by automating the process of converting expert demonstrations, in the form of game replays, into cases. To achieve this, we present a technique for annotating traces with goals that can be used by a case-based planner. We have implemented this technique in the case-based planning system Darmok 2 and applied it to the task of playing complete games of StarCraft. By automating the process of case generation, we enable our system to harness the large number of expert replays available on the web.

1 Introduction

Video games provide an excellent testbed for research in case-based reasoning. Games are increasingly providing data that can be used to automate the process of building game AI, such as replays. Harnessing this data is challenging, because game replays contain only game state and actions performed by players and do not contain a player's goals or intentions. Utilizing this data in agents that reason about goals is problematic, because there is often a gap between the game actions contained in a replay and the agent's goals. This problem becomes apparent in complex games in which players reason about the game at multiple levels of granularity while performing tasks.

Real-time Strategy (RTS) games in particular are an interesting domain for evaluating AI techniques [1], because they provide several challenges for building game AI. The decision complexity of RTS games is huge [2], and requires simultaneously reasoning about both strategic and tactical goals. RTS games are also real-time environments in which agents must react to events including second-by-second world changes at the tactical level, as well as exogenous events at the strategic level, such as an opponent switching strategies midgame.

There has been recent interest in addressing these challenges by building agents that learn from demonstration, such as replays [3, 4]. The goal is to develop techniques that automatically acquire domain knowledge by analyzing examples of gameplay from skilled human players. Building agents that learn from replays poses several challenges: defining a suitable representation for encoding game state, extracting cases automatically from replays, specifying similarity metrics for retrieval of cases, modeling the domain in order for the agent to reason about goals, and supporting real-time execution in the game environment. Previous work has addressed the issue of case extraction by either requiring manual annotation of replays [3], or building hierarchical plans based on dependencies between actions [5].

In this paper, we present a technique for annotating replays for use in case-based planning. Our approach automates the process of extracting cases from replays by labeling the actions performed in replays with the goals being pursued. This enables the use of real-world data for generating huge case libraries. We have implemented this technique within the Darmok 2 framework [5] using the real-time strategy game StarCraft as our application domain.

2 Related Work

Applying case-based planning for building game AI requires formally modeling the domain. Goal oriented action planning (GOAP) is a planning-based approach to building AI for non-player characters in games [6]. In GOAP architectures, a character has a set of goals that become activated based on a set of criteria. Upon activation, a plan is generated to achieve the goal and then executed in the game world. The main challenge in implementing this approach is defining a suitable world representation and operators for building plans in real-time. Our system differs from this approach, because Darmok 2 performs case-based planning, while GOAP architectures utilize generative planning. Another aspect of applying planning to game AI is specifying goals for the agent to pursue. An approach implemented in the RTS game Axis & Allies is triggering goal formulation when specific game events occur, such as capturing an enemy city [7]. Darmok 2 does not explicitly perform goal formulation, but incorporates it in the subgoaling process of plan retrieval.

In the domain of RTS games, two approaches have been utilized to build game AI with case-based reasoning: systems that learn online, and systems that bootstrap the learning process by generating an initial case library from a set of replays. The first approach has been applied to strategy selection in Wargus [2] and micro-management in WarCraft III [8]. The second approach has been applied to building a complete game playing agent for Wargus [3] and build order specifically in Wargus [4]. Our system differs from previous work in that we are using a much larger case library in order to manage the complexity of StarCraft.

There has also been related work on annotating replays for RTS games. Weber and Mateas applied data mining to predicting an opponent's strategy in StarCraft [9]. Their approach labeled replays with specific strategies and represented strategy prediction as a classification problem. Metoyer et al. analyzed how players describe strategies while playing Wargus and identified several patterns [10]. There is also a large online community supporting StarCraft that manually annotates replays by identifying specific strategies [11] and providing high-level commentary[3].

3 StarCraft

StarCraft[4] is a science fiction RTS game in which players manage an economy, produce units and buildings, and vie for control of the map with the goal of destroying all opponents. Real-time strategy games (and StarCraft in particular) provide an excellent environment for AI research, because they involve both low-level tactical decisions and high-level strategic reasoning. At the strategic level, StarCraft requires decision-making about long-term resource and technology management, while at the tactical level, effective gameplay requires both micro-management of individual units in small-scale combat scenarios and squad-based tactics such as formations.

StarCraft is an excellent domain for evaluating decision making agents, because there are many complex tradeoffs. One of the main tradeoffs in StarCraft is selecting a "build order", which defines the initial actions to perform in the game and is analogous to chess openings. There is no dominant strategy in StarCraft and a wide variety of build orders are commonly executed by top players: economic-heavy build orders focus on setting up a strong resource infrastructure early in the game, while rush-based strategies focus on executing a tactical assault as fast as possible. The most effective build order for a given match is based on several factors, including the map, opponent's race, and opponent's predicted strategy. An agent that performs well in this domain needs to include these factors into its strategy selection process.

Another tradeoff in StarCraft is deciding where to focus attention. StarCraft gameplay requires simultaneously managing high-level strategic decisions (macro-management) with highly-reactive actions in tactical scenarios (micro-management). Players have a finite amount of time and must decide where to focus their attention during gameplay. This is also an issue for game-playing agents, because decisions must be made in real time for both macro-management and micro-management actions.

StarCraft has a vast decision complexity, not just because of the game state, but also due to the number of viable strategies in the strategy space. One of the interesting aspects of StarCraft is the level of specialization that is applied in order to manage this complexity. Professional StarCraft players select a specific race and often train for that race exclusively. However, there is large amount of general knowledge about StarCraft gameplay, and it provides an excellent domain for transfer learning research (e.g. adapting strategies learnt for one race to another race).

Our choice of StarCraft as a domain has additional motivating factors: despite being more than 10 years old, the game still has an ardent fanbase, and there is

[3] http://www.gomtv.net

[4] StarCraft and StarCraft: Brood War were developed by Blizzard Entertainment™

even a professional league of StarCraft players in South Korea[5]. This indicates that the game has depth of skill, and makes evaluation against human players not only possible, but interesting.

4 Darmok 2

Darmok 2 (D2) [5] is a real-time case-based planning system designed to play RTS games. D2 implements the *on-line case-based planning* cycle (OLCBP) as introduced in [3]. The OLCBP cycle attempts to provide a high-level framework to develop case-based planning systems that operate on-line, i.e. that interleave planning and execution in real-time domains. The OLCBP cycle extends the traditional CBR cycle by adding two additional processes, namely *plan expansion* and *plan execution*. The main focus of D2 is to explore learning from unannotated human demonstrations, and the use of adversarial planning techniques. The most important characteristics of D2 are:

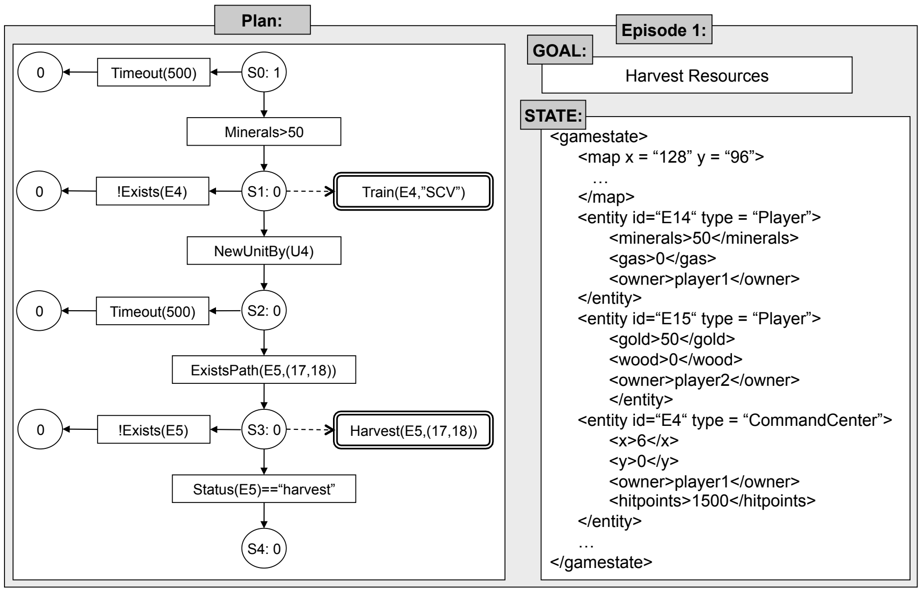
– It acquires cases by analyzing human demonstrations.
– It interleaves planning and execution.
– It uses an efficient transformational plan adaptation algorithm for allowing real-time plan adaptation.
– It can use a *simulator* (if available) to perform adversarial planning.

D2 learns a collection of cases by analyzing human demonstrations (traces). Demonstrations in D2 are represented as a list of triples $[\langle t_1, S_1, A_1\rangle, ..., \langle t_n, S_n, A_n\rangle]$, where each triple contains a time stamp t_i game state S_i and a set of actions A_i (that can be empty). The set of triples represent the evolution of the game and the actions executed by each of the players at different time intervals. The set of actions A_i represent actions that were issued at t_i by any of the players in the game. The game state is stored using an object-oriented representation that captures all the information in the state: map, players and other entities (entities include all the units a player controls in an RTS game: e.g. tanks).

Each case $C = \langle P, G, S\rangle$ consists of a plan P (represented as a *petri-net*), a goal G, and a game state S. A case states that, in a game state S, the plan P managed to achieve the goal G. An example case can be seen in Figure 1. Plans in cases are represented in D2 as *petri-nets* [12]. Petri nets offer an expressive formalism for representing plans that can have conditionals, loops or parallel sequences of actions (in this paper we will not use loops). In short, a petri net is a graph consisting of two types of nodes: *transitions* and *states*. Transitions contain conditions, and link states to each other. Each state might contain *tokens*, which are required to fire transitions. The distribution of tokens in the petri net represent its status. For example, in Figure 1 there is only 1 token in the top-most state, stating that none of the actions has executed yet.

When D2 executes a plan contained in a case, the actions in it are adapted to fit the current situation. Each action contains a series of parameters referring to

[5] Korea e-Sports Association: http://www.e-sports.or.kr/



```
<gamestate>
    <map x = "128" y = "96">
      …
    </map>
    <entity id="E14" type = "Player">
        <minerals>50</minerals>
        <gas>0</gas>
        <owner>player1</owner>
    </entity>
    <entity id="E15" type = "Player">
        <gold>50</gold>
        <wood>0</wood>
        <owner>player2</owner>
        </entity>
    <entity id="E4" type = "CommandCenter">
        <x>6</x>
        <y>0</y>
        <owner>player1</owner>
        <hitpoints>1500</hitpoints>
    </entity>
    …
</gamestate>
```

Fig. 1. A case in D2 consisting of a plan, goal and game state. The snippet contains two actions and the game state representation is not fully included due to space limitations.

locations or units in the map, and other constants. For example, if a parameter refers to a location, then a location in the current map which is the most similar to the location specified in the action is selected. For assessing location similarity, D2 creates a series of potential fields (one for each entity type in the game, in the case of StarCraft: friendly-marines, enemy-marines, friendly-tanks, enemy-tanks, minerals, etc.). Each location in the map, is thus assigned a vector, which has one value for each potential field. This allows D2 to compare map locations and assess which ones are more similar to others. For example, it can detect that a location is similar to another because both are very close to enemy units.

In previous work [5] we explored a technique based on ideas similar to those of the HTN-maker [13] in order to automatically learn cases from expert demonstrations (traces). This strategy lets D2 effectively learn from traces automatically without requiring an expert to annotate the traces as previous work required [3, 14]. However, it relies on the assumption that each action the expert executed was carefully selected, and that each condition that becomes true during the game as an effect of the executed actions was intended by the expert. Since those conditions are too restrictive, sometimes the system learns spurious plans for goals that were achieved only accidentally.

In this paper, we explore an alternative approach which exploits goal recognition techniques [15]. Our approach incorporates an expert provided goal ontology to automate the process of case extraction. While it requires the application of additional domain knowledge, it enables the extraction of cases that achieve high-level goals, rather than grouping action sequences based only on unit dependencies. High quality annotations are important, since they provide a means for D2 to break up a trace into smaller pieces, which will constitute cases.

5 Replay Annotation

Our goal is to make use of real-world StarCraft replays in order to build a case library for D2. Our approach to achieve this goal requires automating the trace annotation process, because manual annotation is impractical for large datasets and presents language barriers for StarCraft, because the majority of professional StarCraft players are non-English speakers. By automating the process of annotating traces, we enable D2 to make use of the large amount of StarCraft traces available on the web.

There are several challenges faced when utilizing real-world examples. First, unlike previous work in Wargus [4, 5], we were unable to specify a trace format for storing examples. The replay format we use is a proprietary binary format specified by Blizzard and we have no control over the data that is persisted. Second, real-world replays are noisy and contain non-relevant actions, such as spamming of actions. For example, in previous work [3] the traces used for learning were carefully created with the purpose of learning from demonstration, and did not have any noise. Third, our StarCraft traces contain large numbers of actions, because players execute hundreds of actions per minute, which can result in traces containing over a thousand actions. To overcome these challenges, we developed a technique for automatically annotating replays, which is used to break up the replays into cases which are usable by D2.

To automatically annotate actions in game traces with goals, we defined a goal ontology for StarCraft and developed a rule set for recognizing when goals are being pursued. To build our case library, we extracted action logs from replays and labeled actions with a set of goals. The result of this approach is annotated game traces that can be used to build a case library for D2.

5.1 Goal Ontology

Our approach utilizes a goal ontology which is used to specify the goals that are being pursued by actions in a game trace. The goal annotations are equivalent to tasks in a HTN planner [5] and are used by the case-based planner to decompose the goal of winning the game into subgoals, such as the goal of setting up a resource infrastructure. The case retrieval process incorporates goal similarity as well as game state similarity when performing retrieval. In order to effectively make use of OLCBP, the goal ontology should model the domain at the level at which players reason about goals.

The goal ontology was formulated based on analysis of professional StarCraft gameplay [11] as well as previous work [16], and is shown in Figure 2. The economy subgoal contains tasks that achieve the goal of managing a resource infrastructure, while the strategy subgoal contains tasks that achieve the goal

- Win StarCraft
    - Economy
        - Produce worker units
        - Harvest resources
        - Build resource facilities
        - Manage supply
    - Strategy
        - Build production facilities
        - Build tech building
        - Produce combat units
        - Research upgrade
    - Tactics
        - Attack opponent
        - Use tech ability or spell

Fig. 2. Our goal ontology for actions in StarCraft

of expanding the tech tree and producing combat units. The ontology is not specific to a single StarCraft race, and could be applied to other games such as Wargus. It decomposes the task of playing StarCraft into the subgoals of managing the resource infrastructure, expanding the tech tree and producing combat units, and performing tactical missions. The main difference between our ontology and previous work is more of a focus on the production of units, rather than the collection of resources, because players tend to follow rules of thumb for resource gathering and spend resources as soon as they are available. Additionally, we grouped the expansion of the tech tree and production of combat units into a shared subgoal, to manage contention of in-game resources.

Each goal in the ontology has a set of parameters that are used to evaluate whether the given goal is applicable to the current game situation. Upon retrieval of a subgoal, D2 first queries for goals that can be activated, and then searches for cases that achieve the selected goal. Each subgoal contains parameters that are relevant to achieving that specific goal. For example, the *economy* goal takes as input the following parameters: the current game time, the number of units controlled by the agent, the maximum number of units the agent can support, the number of worker units, expansions and refineries, and the number of worker units harvesting gas.

5.2 Case Extraction

The system uses a two part process in which cases are extracted from traces and then annotated with goals. In the first part of the process, actions in a trace are placed into different categories based on the subgoal they achieve and then grouped into cases based on temporal locality. For example, an attack command issued in StarCraft would be placed into the *tactics* category and grouped into a case with other attack actions that occur within a specific time period. The second part of the process is the actual annotation phase in which cases are labeled based on the game state of the first action occurring in the case. Our approach currently groups actions into three categories, which correspond to the direct subgoals of the *Win StarCraft* goal in the ontology. Different techniques are used for grouping actions into cases for the different categories.

The *economy* and *strategy* categories group actions into cases using a model based on goal recognition, which exploits the linear structure of gameplay traces [15]. Given an action at time t in a trace, we can compute the game state at time $t+n$ by retrieving it directly from the trace, where n is the number of actions that have occurred since time t. The game state at $t+n$, S_{t+n}, can be inferred as the player's goal at time t, given the player has a plan length of size n. This model of goal recognition relies on the assumption that the player has a linear plan to achieve a specific economic or strategic goal. This assumption is supported by what is referred to as a "build order" in StarCraft, which is a predefined sequence of actions for performing a specific opening.

Cases are built by iterating over the actions in a trace and building a new case every n actions using the following algorithm:

$$\begin{aligned} C.S &= S_t \\ C.P &= petri\_net(A_t, A_{t+1}, \ldots, A_{t+n}) \\ C.G &= category.G \\ C.G.params &= compute\_params(S_{t+n}) \end{aligned}$$

where C is the generated case, S is the game state, P is a petri-net plan, $petri\_net$ is a method for building a petri net from a sequence of actions, A is an action performed by the player in the trace, $category.G$ is the *economy* or *strategy* subgoal, $C.G.params$ is the goal parameters, and $compute\_params$ is a goal specific function for computing the goal parameters given a game state. For *economy* cases, n was set to 5 to allow the agent to react to the opponent, while for *strategy* cases, n was set to 10 to prevent the agent from dithering between strategies [15]. This approach can result in cases with overlapping actions.

The *tactics* category groups actions into cases using a policy similar to the previous approach, but groups actions based on the game time at which they occur, rather than the index in the trace. The motivation for this approach to grouping actions is based on the tendency of players to rapidly perform several attack actions when launching a tactical assault. A new case is created each time an attack issue is ordered to a unit where the distance to the attack location is above a threshold. Given an attack action, A_i, occurring at cycle t, subsequent attack actions are grouped into a case using the following policy:

$$C.P = petri\_net(A_i, \ldots, A_j)$$

$$A_j = \max\{A_n | A_n.gameframe \leq (t + delay)\}$$

where $game\_frame$ is the time at which attack A_n occurred, in game cycles, and $delay$ is a threshold for specifying how much of a delay to allow between the initial attack command and subsequent commands.

6 Experimental Evaluation

We implemented our approach for automated replay annotation in the StarCraft domain. To build a case library, we first collected several replays from professional players. Next, we ran the replays in StarCraft and exported traces with complete game information, i.e. we generated D2 traces from the replay files. Finally, we ran the case extraction code to build a library of cases for D2 to utilize.

D2 was interfaced with StarCraft using BWAPI [6], which enables the querying of game state and the ability to issue orders to units. D2 communicates with StarCraft through BWAPI using sockets. This interface enables D2 to keep a synchronized state of the game world and provides a way for D2 to perform game actions.

We performed an initial evaluation with a case library generated from four traces to explore the viability of the approach. While D2 was able to set up a resource infrastructure and begin expanding the tech tree, the system is currently unable to defeat the built in AI of StarCraft, which performs a strong rush strategy. A full evaluation of this technique is part of our future work.

7 Conclusion and Future Work

We have presented a technique for automating the process of annotating replays for use by a case-based planner. This technique was applied to replays mined from the web and enables the generation of large case libraries. In order to annotate traces with goals, we first defined a goal ontology to capture the intentions of a player during gameplay. We then discussed our approach to breaking up actions in a trace into cases and how we label them. Our system was evaluated by collecting several StarCraft replays and converting them into cases usable by D2. D2 was then applied to the task of playing complete games of StarCraft.

While our system hints at the potential of our approach, there are several research issues to address. For example, the potential field technique used by D2 to adapt examples to the current game situation provides a domain independent mechanism for adaptation. However, there are several specific instances in StarCraft where it may be necessary to hand author the adaptation functionality. Some units in StarCraft have several different uses, and determining the intended result of an action may require the application of additional domain knowledge.

Another aspect of related work is expanding the goal ontology to include a larger number of player goals. For example, a player attacking an opponent may be pursuing one of several goals: destroying an expansion, harassment of worker units, pressuring the opponent, gaining map control, or reducing the opponent's army size. Increasing the granularity of the goal ontology will require more sophisticated techniques for labeling traces.

[6] http://code.google.com/p/bwapi/

Finally, as part of our future work, we plan to study scaling-up issues related to using case-based planning systems in complex domains such as StarCraft with a large collection of complex cases.

References

1. Buro, M.: Real-Time Strategy Games: A New AI Research Challenge. In: Proceedings of the International Joint Conference on Artificial Intelligence. (2003) 1534–1535
2. Aha, D.W., Molineaux, M., Ponsen, M.: Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. Lecture notes in computer science **3620** (2005) 5–20
3. Ontanón, S., Mishra, K., Sugandh, N., Ram, A.: On-Line Case-Based Planning. Computational Intelligence **26**(1) (2010) 84–119
4. Weber, B., Mateas, M.: Case-Based Reasoning for Build Order in Real-Time Strategy Games. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference, AAAI Press (2009) 106–111
5. Ontañón, S., Bonnette, K., Mahindrakar, P., Gómez-Martín, M., Long, K., Radhakrishnan, J., Shah, R., Ram, A.: Learning from Human Demonstrations for Real-Time Case-Based Planning. IJCAI Workshop on Learning Structural Knowledge from Observations (2009)
6. Orkin, J.: Applying Goal-Oriented Action Planning to Games. In: AI Game Programming Wisdom 2. Charles River Media, S. Rabin editor (2003) 217–228
7. Dill, K., Papp, D.: A Goal-Based Architecture for Opposing Player AI. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference, AAAI Press (2005)
8. Szczepański, T., Aamodt, A.: Case-based reasoning for improved micromanagement in real-time strategy games. In: Proceedings of the ICCBR 2009 Workshop on CBR for Computer Games. (2009)
9. Weber, B., Mateas, M.: A Data Mining Approach to Strategy Prediction. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, IEEE Press (2009) 140–147
10. Metoyer, R., Stumpf, S., Neumann, C., Dodge, J., Cao, J., Schnabel, A.: Explaining How to Play Real-Time Strategy Games. Research and Development in Intelligent Systems XXVI (2010) 249–262
11. Team Liquid: Liquipedia: The StarCraft Encyclopedia (April 2010) `http://wiki.teamliquid.net/starcraft`.
12. Murata, T.: Petri nets: Properties, Analysis and Applications. Proceedings of the IEEE **77**(4) (1989) 541–580
13. Hogg, C., Munoz-Avila, H., Kuter, U.: HTN-MAKER: Learning HTNs with Minimal Additional Knowledge Engineering Required. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence. (2008)
14. Könik, T., Laird, J.E.: Learning goal hierarchies from structured observations and expert annotations. Mach. Learn. **64**(1-3) (2006) 263–287
15. Weber, B., Mateas, M., Jhala, A.: Case-Based Goal Formulation. In: Proceedings of the AAAI Workshop on Goal-Driven Autonomy. (2010)
16. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Learning from Demonstration and Case-Based Planning for Real-Time Strategy Games. Soft Computing Applications in Industry (2008) 293–310

Feature Selection for CBR in Imitation of RoboCup Agents: A Comparative Study

Edgar Acosta, Babak Esfandiari, and Michael W. Floyd

Network Management and Artificial Intelligence Laboratory
Carleton University, Ottawa, Canada
`http://www.nmai.ca/research-projects/agent-imitation`

Abstract. We present a comparison of two methods for selecting the features employed in the representation of cases for a case-based reasoning approach to imitation of agents in the RoboCup simulation platform. Feature Selection methods are compared empirically in terms of the imitation performance, the size of the resulting feature set, and execution time. Differences and their implications are discussed.

1 Introduction

One central issue for every artificial intelligence problem is that of knowledge representation. Decisions regarding what to represent and how to do it often facilitate, and likewise constrain, the problem solving process and its success.

The most sensible way to deal with such sensitive decisions involves getting expert advice. However, when there is no expert knowledge available, feature selection methods can help to identify the pieces of information that are more relevant in solving the task [1].

Another aspect in which expert knowledge eases the solution of an artificial intelligence problem regards the processes underlying the task itself. When dealing with a new domain, or when the mechanisms leading to the problem's solution are not well known, case-based reasoning (CBR) provides a powerful learning methodology [2].

Imitation is one such task in which there is no formal model of the behaviour following a sequence of inputs. The RoboCup Simulation Imitation Agent Project (RCIAP)[3] employs both case-based reasoning and feature selection for learning to reproduce the behaviour of soccer playing agents based on the agent inputs.

We compare two feature selection methodologies in order to evaluate which methodology would better help to overcome some of the limitations of the current case recognition framework of the RCIAP.

2 Feature Selection

Feature (subset) selection is a machine learning sub-field that deals with selecting a set of relevant variables (features) from a large set of available features. The aim of feature selection is to reduce the number of features considered by the learning task without deterioration to the learning performance. The employment of feature selection has a number of desirable advantages as a consequence of the reduction in dimensionality of the learning problem, such as faster learning and relatively smaller training sets.

Feature selection methods can be classified according to two parameters: the feature selection model (a.k.a. *feedback* [4]), and search strategies [1].

Three feature selection models can be identified. When feature selection is performed before the learning task in a pre-processing step, the method is called a **filter**. A **wrapper** model is one in which the search for a feature set is guided by a performance measure of the learned model. **Embedded** feature selection occurs when the construction of the set of features is integrated in the learning model (e.g. decision tree algorithms).

Different search strategies can be employed. The most common are **forward search** (starts with a small feature set and larger sets are explored), and **backward search** (starts with a large feature set and explores smaller ones). A simplification of these strategies consists of **sequential search**, in which one feature is included (or removed) at a time without backtracking. Other search methods include random and float search.

3 The RoboCup Simulation Imitation Agent Project

The purpose of the RCIAP is to build Robo Cup agents that imitate other (*target*) agents playing in the Robo Cup Simulation League, with as few domain-specific assumptions as possible.

3.1 The RoboCup Simulation League

The Robo Cup Simulation League is a platform for testing software agents playing on a soccer team. Each agent is a soccer player client that connects to a soccer simulation server. The simulation server keeps track of the game, sends messages to the clients and receives commands from them.

The server provides the agents with different kinds of information. There are two types of messages sent by the server: sensory information (the `see`, the `hear`, and the `sense body` messages), and game state information such as play modes (e.g. offside, goal kick, play on) and the match score. The `see` messages describe an incomplete view of the soccer field in terms of objects such as other players, the ball, field lines and flags. These objects also have properties such as distance and direction.

In response, agents send back one or more commands to the server, such as `dash`, `kick`, `turn`, or `say`. These commands also have properties such as power and angle of a kick.

The Case-Based Reasoning Approach: In RCIAP, imitation of agents is achieved using case-based reasoning. Observations are obtained from log files of the communication between an agent and the server. The captured data is transformed into a spatial representation (called a *scene*) that includes information from the `see` messages (other types of inputs are ignored for now) together with the corresponding commands sent by the agent at each simulation cycle.

The obtained scenes constitute the knowledge of an imitative agent, which is employed in a case-based fashion when playing a soccer game. The agent transforms the current game situation into an scene representation, compares it to the stored scenes, finds the k-nearest scenes ($k = 1$), and then selects the command found on the *1*-nearest scene.

Challenges: Since a RoboCup simulation cycle takes 100ms, the imitative agent has to send a command back to the simulation server in less than 80ms (we impose a smaller than 100 ms limit to account for lag and message processing.) Thus the calculation of distance between the current simulation situation and stored cases has to be performed as fast as possible.

Distance calculation is affected by the number of features used to represent each case. The larger the number of features, the more complex the calculation procedure. Also, the number of cases required to discern between game situations increases factorially with the number of features employed. Therefore, it is convenient to consider as few features as possible.

Current Implementation: In the current RoboCup case-based agent [3], 160 visual input features and 3 output features were considered. The input features correspond to the relative coordinates of up to 80 visible objects (7 object types in bold face): the **Ball**, **flags** (up to 53 field landmarks,) players (up to 21, which are classified as **team mates**, **opponents**, and **not identifiable**,) the visible **goal** if any, and **field boundaries** (up to 4.) The output features correspond to the agent **command**, as well as the **power**, and aimed **direction** of the action.

Distance calculation (between the current input and a case input) is performed in the following way:

1. Seven lists of objects are generated, each ordered by nearest to farthest, and from left to right.
2. The objects from each list are matched in order. Non-matching objects are ignored. If only n objects can be matched, then let i be the set of matched objects from the current input, and c_m the corresponding set of matches in case m. i.e.

$$\begin{aligned} i &= \{\mathrm{obj}_1, \mathrm{obj}_2, \cdots, \mathrm{obj}_n\} \\ c_m &= \{\mathit{match}(\mathrm{obj}_1), \mathit{match}(\mathrm{obj}_2), \cdots, \mathit{match}(\mathrm{obj}_n)\} \end{aligned}$$

3. The euclidean distance between matched objects is obtained: $d\left(\mathrm{obj}_k, \mathit{match}(\mathrm{obj}_k)\right)$
4. All these distances are weighted and added up:

$$d(i, c_m) = \sum_{j=1}^{n} w_j \times d\left(\mathrm{obj}_j, \mathit{match}(\mathrm{obj}_j)\right)$$

Weights are determined offline in a pre-processing step. The weight calculation method used in the current implementation is a backward sequential search binary wrapper algorithm [5, page 56]. Weights are 0 or 1 depending on the object type. The idea is to assign weight 1 only to types of objects that are highly relevant, and ignore object types with low relevance.

This approach is a feature selection algorithm that considers only 7 features, one per object type. It starts with the 7 object types (all weights are 1) and evaluates the CBR performance; this is the current best set of features. Starting with the current best features set, the algorithm considers all the subsets of features that obtain by removing only one feature (switching its weight to 0), and evaluates the CBR performance of each of these sets. If one of those subsets has the best CBR performance, and is at least 0.5% better than the current best, then it becomes the new current best set of features and the process is started again. Otherwise the algorithm returns the current best features set.

Limitations: One important limitation of this approach is that by bundling together all the objects of the same type into a single feature we are imposing an overgeneralizing and strong constraint to distance weights, which also affects the inductive bias of the *k*-nearest case search. By doing so, we end up using many features that may not be very relevant, and ignoring others that may be relevant.

For example, it may well be the case that the most relevant object is the player that is closer to the ball, and that the output command depends on whether that player is a team mate or an opponent.

If we performed feature selection over the original 160 features we would have better chances of capturing such distinctions. However, this feature selection algorithm is extremely slow, so it cannot scale to many more than 7 features. Time is of the essence because one of our future goals involves being able to build the case base and compute weights on line.

Improving Feature Selection: It would be useful to find a feature selection algorithm that is faster and can be scaled to at least 160 input features. We must, however, be systematic and attack each of these desiderata at a time. First we need to find a feature selection method that is faster and obtains similar CBR performance when we apply it to the same 7 features. Once we achieve that goal, we can try to use it with larger feature sets.

Zhong, et al. [6] report a rough set based filter approach for feature selection that is fast and effective in large databases. In the next section we propose a similar algorithm. Our hypothesis is that this algorithm will achieve the first goal, namely to perform feature selection faster without sacrificing CBR performance. Then, in Section 5 we test our hypothesis by comparing both algorithms in terms of CBR performance and feature selection performance.

Salamo and Golobardes [7] discuss two other (continuous) weighting methods for CBR based on rough sets. Other applications of rough set theory for CBR include case base reduction [8] and case representation [9].

4 Rough Set Theory Approach to Feature Selection

Rough set theory (RST) [10] deals with the representational properties of sets of finite discrete valued features.

In RST, a *decision system* is a tuple (U, C, D) where U is the universe of considered objects, and C and D are two disjoint collections of features. Both C and D induce equivalence classes in U, let us call their sets of equivalence classes $\mathcal{C}$ and $\mathcal{D}$ respectively. The idea in a decision system is to approximate classes in $\mathcal{D}$ in terms of the classes in $\mathcal{C}$. In other words, we try to characterize the different classes in $\mathcal{D}$ using only the feature set C.

This is relevant to machine learning if we think of C as the features used to describe observations, and of D as the features of those objects that we want to learn. Hence, we call them *condition features* and *decision features* respectively.

The *positive region* $POS_C(\mathcal{D})$ of $\mathcal{D}$ in terms of C is the set of objects in U that can be correctly classified as belonging to one class in $\mathcal{D}$ using only the information about features C.

4.1 Rough Set Theory methods for Feature Selection

Given a decision system $S = (U, C, D)$, the feature subset selection problem consists of finding a minimal family of features $C' \subseteq C$ such that $POS_{C'}(\mathcal{D}) = POS_C(\mathcal{D})$.

A solution C' to the feature subset selection problem is a D-reduct of C.
If f is a feature in C such that $POS_{C-f}(\mathcal{D}) \subsetneq POS_C(\mathcal{D})$, then f is D-indispensable in C.
The union of all D-indispensable features in C is the D-core of C, $CORE_D(C)$.

Some interesting properties of reducts are worth noticing:

multiplicity of solutions: There may be more than one D-reduct of C, i.e. the feature selection problem may have more than one solution.
core and reducts: Every D-reduct of C contains the $CORE_D(C)$.

These observations suggest a basic method for feature selection: we can start by finding the core, and then add features to the core until we find a subset of features with a positive region that matches that of the full set of condition features (see Algorithm 3). Finding the core is as simple as testing every feature in C for the indispensability property (see Algorithm 2). Each of these procedures depend on calculating the positive region of $\mathcal{D}$ (as the feature set evaluation metric) in terms of different subsets of condition features (see Algorithm 1).

Algorithm 3 returns the first solution it finds, thus, the order (`rank()` procedure) in which dispensable features are added to the core is one source of inductive bias. The other source of inductive bias comes from the feature set evaluation measure $|POS_{G \subseteq C}(\mathcal{D})|$ (`PR()` procedure.)

Algorithm 3 does not guarantee finding a D-reduct of C. Nevertheless, it finds a feature set A such that A contains a D-reduct of C, and $A \subseteq C$,i.e. no

Algorithm 1 Positive Region

Inputs: observations, conditionFeatures, decisionFeatures

Outputs: positiveRegion: number of discernible observations

```
PR(observations, conditionFeatures, decisionFeatures)
    conditionClasses = { }
    positiveRegion = 0
    for(each observation Obs in observations)
        cClass = determineConditionClass(Obs[Input],conditionFeatures)
        dClass = determineDecisionClass(Obs[Output],decisionFeatures)
        if(cClass does not exist in conditionClasses)
            elementCounter = 1
            outputClasses = set{dClass}
        else
            elementCounter = conditionClasses[cClass][0] + 1
            outputClasses = conditionClasses[cClass][1]
            outputClasses.set_add(dClass)
        conditionClasses[cClass]= {elementCounter,outputClasses}
    for(each condition class condClass in conditionClasses)
        elements = conditionClasses[condClass][0]
        outputs = conditionClasses[condClass][1]
        if(number of different outputs == 1)
            positiveRegion += elements
    return positiveRegion
```

information is lost by using this algorithm. In most cases Algorithm 3 will return a smaller set than C. Running the basic algorithm again on the first run solution may help to further reduce the set of condition features.

If n is the number of training cases, $c = |C|$, and m is the maximum number of values that a feature in C can take, then Algorithm 3 is of order $O(nmc^2)$.

4.2 Advantages of Rough Set Theory Methods

One consequence of RST is that the cardinality of the positive region $|POS_C(\mathcal{D})|$ is the maximum number of objects in U that can be correctly categorized with the information provided by any subset of the set of condition features. Thus, the positive region is an upper bound for the learning performance of any machine learning approach.

4.3 Proposed Feature Selection Method

Despite the fact that Algorithm 3 does not always find a reduct, we propose to compare it with the current weight calculation employed in RCIAP.

Hypothesis: Algorithm 3 implies a filter feature selection model, which has the potential for a faster execution time than the current wrapper approach. Thus, we expect an important reduction in execution time.

Algorithm 2 Core

Inputs: observations, conditionFeatures, decisionFeatures, maxPR

Outputs: coreFeatures

```
CORE(observations, conditionFeatures, decisionFeatures, maxPR)
    coreFeatures = { }
    for(each feature Feature in conditionFeatures)
        complement = conditionFeatures
        complement.remove(Feature)
        complementPR=PR(observations, complement, decisionFeatures)
        if(complementPR < maxPR)
            coreFeatures.add(feature)
    return coreFeatures
```

As for CBR performance, we do not expect statistically significant differences, or important differences otherwise. The same hypothesis holds for the number of selected features.

5 Feature Selection Methods Comparison

This section describes the comparisons performed between the current backward sequential search binary wrapper algorithm and a rough set based forward sequential search filter algorithm (Algorithm 3 with no ranking of dispensable features).

5.1 Materials

Target Agents: Data from 3 RoboCup agents was used. In order of complexity: *Krislet* a simple team that chases the ball and kicks it toward the opponents goal, *A1* based on Krislet (this agent will not follow the ball if it sees a team mate closer to it,) and *CMUnited 2000* RoboCup World Champions developed by Carnegie Mellon University.
Case Base: The case base consisted of 15, 000 cases per agent. Each case encoded the 160 input features, as well as the 3 output features described in Section 3. For feature selection, only 7 seven features (the object types) were used.
Independent variables: The experiment consisted of 6 experimental conditions with 2 independent variables: target agent (3 conditions), and feature selection algorithm (2 conditions).
Performance metrics:

Feature set size: The number of features in the solution set.
Execution time: This is the execution time of the feature selection procedure.
Accuracy: The rate of CBR outputs that matched the output in the testing case. This only takes into account the type of action.

Algorithm 3 Forward Sequential Feature Selection
Inputs: observations, conditionFs, decisionFs
Outputs: reduct

```
RSFS(observations, conditionFs, decisionFs)
    maxPR = PR(observations, conditionFs, decisionFs)
    coreFeatures = CORE(observations,conditionFs,decisionFs,maxPR)
    dispensableFeatures = conditionFs - coreFeatures
    despensableRanked = rank(dispensableFeatures)
    currentBest = coreFeatures
    bestPR = PR(observations, currentBest, decisionFs)
    for(each feature Disp in dispensableRanked)
        testSet = currentBest
        testSet.add(Disp)
        testPR = PR(observations, testSet, decisionFs)
        if(testPR > bestPR)
            currentBest = testSet
            bestPR = testPR
        if(bestPR == maxPR)
            return currentBest
    return currentBest
```

Recall per action type: For each action type, this is the rate of testing cases for which CBR selected the correct action.

Precision per action type: For each action type, this is the rate of CBR outputs that correctly selected this action type.

f-measure per action type: The f-measure combine recall and precision into a single metric

$$f_{\text{action}} = \frac{2 \times Recall_{\text{action}} \times Precision_{\text{action}}}{Recall_{\text{action}} + Precision_{\text{action}}}$$

Global f: The global f-measure is the average of the collected f-measures.

The first two metrics regard to the feature selection performance, and the remaining metrics regard to CBR performance with the resulting set of features. The experiment, and CBR, were performed with the tools described in [5, section 9.5.].

5.2 Methodology

30 trials per condition were performed. On each trial, $1,500$ random cases were used during feature selection, and $6,000$ random cases were used during CBR performance evaluation.

Statistical analysis: In order to determine significance of the observed differences between the two feature selection approaches, we performed a 1-factor repeated measures analysis of variance per metric collected for each agent condition.

5.3 Results

The next two tables show the averaged metrics for CBR performance and Feature Selection performance, as well as the result of the ANOVA performed for the effect of the feature selection algorithm on the collected metrics.

CBR Performance:

	Krislet		**A1**		**CMU**	
Algorithm	Global f	Accuracy	Global f	Accuracy	Global f	Accuracy
BS wrapper	0.84	0.98	0.81	0.89	0.70	0.74
RS filter	0.66	0.86	0.77	0.88	0.64	0.73
$F(1, 29)$	$F = 88.77$ $p < 0.01$ $\eta^2 = 0.75$	$F = 77.67$ $p < 0.01$ $\eta^2 = 0.73$	$F = 39.00$ $p < 0.01$ $\eta^2 = 0.57$	$F = 5.36$ $p < 0.05$ $\eta^2 = 0.16$	$F = 62.67$ $p < 0.01$ $\eta^2 = 0.68$	$F = 0.56$ $p = 0.46$ $\eta^2 = 0.02$

Feature Selection Performance: Execution times are in minutes.

	Krislet		**A1**		**CMU**	
Algorithm	time	set size	time	set size	time	set size
BS wrapper	66.29	1.13	77.59	2.90	56.57	3.93
RS filter	1.89	1.90	2.70	2.57	1.61	3.43
$F(1, 29)$	$F = 1030$ $p < 0.01$ $\eta^2 = 0.97$	$F = 69.41$ $p < 0.01$ $\eta^2 = 0.71$	$F = 759$ $p < 0.01$ $\eta^2 = 0.96$	$F = 4.26$ $p < 0.05$ $\eta^2 = 0.13$	$F = 572$ $p < 0.01$ $\eta^2 = 0.95$	$F = 4.58$ $p < 0.05$ $\eta^2 = 0.14$

6 Discussion

All the metrics, except CBR accuracy for the CMU agent, had statistically significant differences at the indicated significance levels (p value). As it was to expect, CBR performance degraded with an increase of complexity of the target agents, regardless of the feature selection method.

Although the current Backward Sequential Search binary wrapper algorithm produces superior CBR performance than the proposed Rough Set based Forward Sequential Search filter algorithm, the gap in CBR performance diminishes as the target agent complexity increases. The difference in CBR performance is likely due to the feature selection model: the wrapper approach finds a solution that best works with the particular learning algorithm.

The most impressive difference between the two feature selection approaches is observed in the execution time of feature selection, being the proposed algorithm almost 30 times faster than the current one. This difference is due to the cost of the feature set evaluation employed; whereas the current approach evaluates by observing the CBR performance that the feature set yields, the proposed algorithm computes the positive region of the feature set. Notice that any other wrapper approach (including most evolutionary approaches) would suffer from the same problem.

7 Conclusion

Although the proposed algorithm produced slightly, but significant, worse CBR performance than the current algorithm, it impressively improved the feature selection time. This makes the proposed algorithm a good candidate for performing feature selection over more than the current 7 features.

Further work should focus on trying to use this algorithm with more features, ideally the 160 employed in case distance calculation. Such improvement is likely to lead to better imitation performance. Also, the proposed algorithm can be improved both in speed, and in better approximating a D-reduct of the condition features.

References

1. Liu, H., Motoda, H.: Less is more. In: Computational Methods of Feature Selection. Chapman & Hall/CRC, Boca Raton (2008)
2. Pal, S.K., Shiu, S.C.K.: Introduction. In: Foundations of soft case-based reasoning. Wiley-Interscience, Hoboken, N.J. (2004)
3. Floyd, M.W., Esfandiari, B., Lam, K.: A case-based reasoning approach to imitating RoboCup players. In: Proceedings of the Twenty-first International Florida Artificial Intelligence Research Society Conference, AAAI Press (2008) 251–256
4. Wettschereck, D., Aha, D.: Weighting features. In: Proceedings of the First International Conference on Case-Based Reasoning, Springer (1995) 347–358
5. Floyd, M.W.: Improving the performance of a RoboCup case-based imitation agent through preprocessing of the case base. Master's thesis, Carleton University, Ottawa, Ontario, Canada (December 2008)
6. Zhong, N., Dong, J.Z., Ohsuga, S.: Using rough sets with heuristics for feature selection. Journal of Intelligent Information Systems **16**(3) (August 2001)
7. Salamo, M., Golobardes, E.: Analysing rough sets weighting methods for case-based reasoning systems. Inteligencia Artificial. Revista Iberoamericana de IA (15) (2002) 10–18
8. Salamó, M., Golobardes, E.: Rough sets reduction techniques for case-based reasoning. In: Case-Based Reasoning Research and Development. Springer-Verlag (2001) 467–482
9. Wierzbicki, J.: Rough set approach to CBR. In Ziarko, W., Yao, Y., eds.: Rough Sets and Current Trends in Computing. Volume 2005 of Lecture Notes in Artificial Intelligence., Berlin, Germany, Springer-Verlag (2001) 503–510
10. Pawlak, Z.: Rough sets : Theoretical aspects of reasoning about data. Volume 9. Kluwer Academic Publishers, Dordrecht; Boston (1991)

Online Micro-Level Decision Making in Real-Time Strategy Games: A Case-Based Reasoning and Reinforcement Learning Approach

Chad Mowery, Nathan Spencer, Isabelle Bichindaritz

Department of Computing and Software Systems
University of Washington, Tacoma
{mowerych, nathanps, ibichind}@u.washington.edu

Abstract. The specialization of artificial intelligence (AI) techniques within the gaming industry is becoming a fast growing field of study for academia and industry developers alike. Many of the AI techniques used in modern video games are unique to a gaming problem domain as game developers often face the challenge of implementing their game AI with limited resources (game frames, memory, and processor cycles), more so than traditional AI projects and studies. However, due to the technological advancement of modern video game consoles and PCs, previous AI techniques used in industry are quickly becoming outdated and uninteresting to players. In this paper we outline a complete AI system that utilizes Case-based Reasoning and Reinforcement Learning components to provide new levels of player immersion. By utilizing real-time learning with Case-based Reasoning we can give a new outlook on games artificial intelligence as challenging and adaptive opponents for players.

1 Introduction

The competitive industry of video games has seen little slowdown of growth and as players continually demand new levels of challenging game play game developers have their hands full trying to keep up. Real-Time Strategy (RTS) games in particular are considered to be one of the most challenging game domains for building intelligent AI agents as they are faced with decision making operations with limited information about their opponents and the game world often contains more states than can be computed during a short game frame time period.

RTS games require a focus on high-level strategies which generally govern some sort of resource and build management. Such macro-level strategies are important to the player as these decisions drive the in-game economy and allow the player to build and research structures or unit abilities that can be used to win during small and large scale unit combat. Additionally, focus must also be maintained at a tactical level involving individual and group level micro-management skills. It is not uncommon for an expert player to exceed 300 actions per minute in order to command hundreds of units while playing an RTS game, thus, the micromanagement of game units constitutes the defining aspect of an expert player over a novice. In this paper we present a game AI system that utilizes Case-based Reasoning for the selection and execution of unit-specific micro-behaviors and Reinforcement Learning for the online adaptation of such unit behaviors in the RTS game Starcraft (Blizzard Entertainment™) in order to provide expert players a more challenging AI opponent.

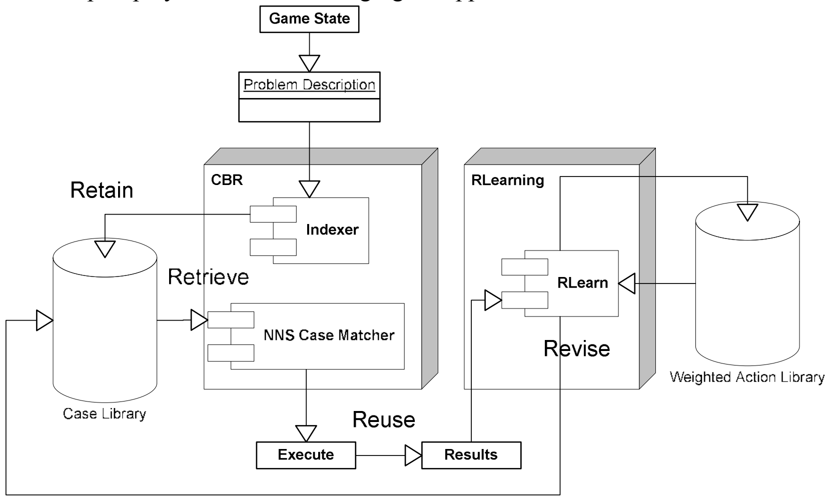


Figure 1. System Architecture and Overview

2 Background

The exploration of integrated Cased-based Reasoning (CBR) systems for RTS and other strategy-style games has recently seen a sharp increase in interest. The work of Aha et al. utilizes a sophisticated form of reactionary CBR that focuses on the spatial information derived from the formations of football teams in the game Rush 2008 [1]. Mateas and Weber have also adapted their version of CBR systems to the RTS game Wargus, an open source clone of Warcraft II (Blizzard Entertainment™) that is largely used for academic purposes [2, 3]. Mateas and Weber's CBR implementations feature a build order specific focus which allows their systems to concentrate on macro-level decision making and does not address the micromanagement of individual units during game play. Alternatively, Aamodt and Szczepański have built their CBR system specifically for this micro-level (individual unit) decision making processes for the game Warcraft 3 (Blizzard Entertainment™) [4]. However, Aamodt and Szczepański's CBR relies on an expert player for case creation and adaptation. For example, during a training session in their system an expert must observe a particular game state for which the currently selected case does not cover or was not satisfactory. This expert would then pause the game and manually create a new case and insert it into the case library for further usage [4]. This means that all case adaptation or creation procedures must occur offline and must be constantly monitored by an expert player having the required domain knowledge to make authoritative decisions about case generation. Finally, the work of Manu Sharma et al. uses a very simliar approach to combining the use of CBR and Reinforcement Learning which focuses on the transfer of learning game states across both the strategic and tactical layers of RTS games to provide effective case revisions [7].

Our micro-level decision CBR project implementation is an extension of the work performed by Aamodt and Szczepański's micro-level CBR system but is unique in that it integrates Reinforcement Learning (RL) techniques and a fully automated case generation framework meaning case adaptation and creation can occur online during a live game with an opponent. RL is a form of machine learning that can be used in games to allow agents to learn a policy/action to select during every state of the game. Our use of spatial temporal reasoning is fundamentally different from Aha et al.'s implementation (much like Sharma et al.) as our revision steps do not require pre-computed state lattices [1]. Sharma et al.'s implementation is also largely different from ours as they focus on the transference of generalized case learning between multiple layers of a hierarchical implementation which offers delayed online results [7]. The goal of RL in our system is to arrive at an optimal action for each state by maximizing the reward received through a process of both real-time and intermittent trial and error exploration feedback [5]. Using CBR coupled with RL eliminates the need for human feedback loop to learn effective strategies in the game. Additionally, our project integrates concepts from the fully integrated agent manager framework of Mateas and Weber [6]. However, our unit managers are implemented on a smaller scale due to the requirements of small scale combat. Instead of utilizing a large network of managers, our implementation only requires the use of a scout and group manager to perform the basic micro-level tactical behaviors of individual units [6].

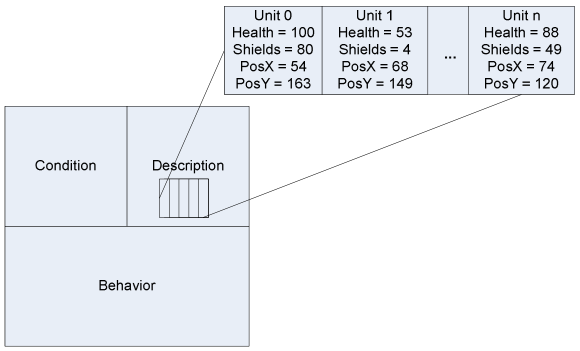


Figure 2. Case Structure

3 System Design

This section outlines our system design in two logical steps: a case-based reasoning component and a reinforcement learning component (see Fig. 1).

3.1 Case-Based Reasoning

3.1.1 Case Structure

Our case structure is similar to the structure outlined by Aamodt and Szczepański [4]. We use three logical separations of the data contained within each case (see Fig. 2). First, there is the conditional data used by the CBR indexer module and containing specific environment conditions that must be met 100% by the problem description case. The second part contains the description of the input problem experience from which the case was generated. In our implementation the description data contains two feature vectors—one that contains all the currently viewable units of the opponent and one that contains all of our units in play. Each element of each of these vectors represents an individual unit description and contains the health, shields, unit type, and positional features for the individual unit. Finally, each case also contains the associated behavior to execute upon case selection. This behavior data can contain an arbitrary number of actions that can either be singular or a sequential list.

3.1.2 Case Retrieval

The case retrieval process happens in three important steps. Our CBR module first transforms the current game state into a problem description object (query case). This is done by isolating specific environmental input features and mapping them to the query case (problem description) and conditional data containers. Next, this new query case is piped to the CBR indexing component which gathers all recorded conditional information and performs a pre-matching operation over each case in the case library generating a set of cases that meet the conditional criteria of the current game state. For example, a Terran Marine unit can be upgraded to have a stimpak ability, that when used, generates more damage to an enemy at the expense of reducing its own health. However, it does not make sense to consider a case that uses this type of micro-behavior if the problem description case does not have a corresponding unit in play that currently has this ability. Thus, our indexer filters out cases that simply do not meet the conditional criteria of the current problem description, thus limiting our search space dramatically saving precious time for more computational operations such as case matching. Finally, the optimized set of cases is then passed to the Nearest Neighbor Selection case matching module which does an in-depth case description comparison using a Minkowski distance-based function for measuring the spatial proximity of each feature vector. The lowest distance-matched case is then selected and executed during the next game frame. Note that after a case has been selected and its action has been executed it is then released back to the case library for future reuse.

3.1.3 Nearest Neighbor Selection Case Matching

Upon completion of the indexer search space optimization, our CBR implementation performs the Nearest Neighbor Selection (NNS) case matching process by creating ordered feature vector distance values. This is done by taking an absolute weighted Minkowski distance of each feature *n* in the Case Description self unit vector and opponent vector for the query case *q* against case returned by the indexing module c_k during a single CBR update (occurs every one second). We then select the smallest distance value as our closest matching experience to the query case problem (1), where c_{ki} and q_i represent values of feature *i* in memorized case c_k and in query case *q*, and w_i the weight of this feature.

$$bestmatch = \min\left(\sum_{k=0}^{m-1}\left(\sum_{i=0}^{n-1} w_i \cdot |\, q_i - c_{ki}\,|^2\right)^{1/2}\right) \qquad (1)$$

A caveat to our NNS case matching algorithm is the bias between the order and lengths of the self and opponent feature vectors in a case description. To address these issues we use a standard sorting measure which arranges the order of each feature vector element by health and shield values from greatest to least in value. In the case when the feature vectors do not match in length we use a simple metric which adds a base value of 100 to the distance for each additional unit in a feature vector that is greater in length. We do this because there are no corresponding features in comparing case and thus, we cannot calculate the feature distance. Although these methods reduce the bias of each feature vector they should still only be considered as minimizations as they cannot fully compensate for large differences in cases.

3.1.4 Case Generation

In order to facilitate the automatic generation of cases, our implementation uses a metric which we call the case-generation threshold and is simply a non-negative integer value that specifies when our CBR system needs to generate a new case. This case generation metric is applied at the end of the NNS case matching process if, and only if, the closest matching case from our case library is greater than the metric value. If this check returns true, a new case is subsequently generated. This metric also controls the active generalization level of each case in relation to the evaluation of problem/solution matching. Thus, a larger case generation value allows each case from the system library to be used over a larger spatial neighborhood of problem situations.

3.2 Reinforcement Learning

Machine learning is often a part of AI systems as it facilitates the ability to adapt and learn conveying a more natural approach to artificial intelligence. Using reinforcement learning, an agent learns by receiving rewards for completing behaviors that result in positive outcomes and is punished for choosing behaviors which result in negative outcomes. We chose to implement an RL technique for dynamic learning because it enabled us to tune an agent's intelligence in real-time (as the game is running) using both goal-oriented task completion and an intermediate performance feedback.

With RL an agent gains information by interacting with the game environment. By (1) sensing the state of the environment, (2) choosing an action, (3) performing the action, and (4) receiving a reward or punishment based on the performance outcome of the action, the agent can perform learning in real-time as the game runs. Thus, RL is tightly coupled to the process of our CBR system as the CBR component chooses the best case to represent a solution to a given problem situation. RL can then evaluate the action (e.g. the solution) that was performed from a case and learn from a delayed outcome. To do so, RL component stores a mapping of these actions and the current fitness score for each unique case that has been generated. In our testing we had six different actions the agent could select for any given case of the world. After a delay of five seconds, the RL system begins the learning process which evaluates the fitness data collected during that time period and then updates the case's action fitness score. This means learning is performed on an action that is executed following each case swap as well as when the same case has been active for over five seconds. Using these intermittent learning updates we are able to quickly adapt and learn on the actions that are being performed after each case selection. Additionally, we use long term rewards to supplement information about actions that typically take longer periods of time to complete. The difference being that short term actions are rewarded on kills and are punished by deaths, while long term rewards affect actions that may cover multiple problem situations in the game world and thus, multiple game updates. For example, moving from one tactical location to another generally results in a much larger reward than an intermittent reward would give.

For intermittent rewards, we take into account all kills and deaths that have occurred in the world since the last reward. If there are no kill and death occurrences (Eq. 2) during this cycle we gradually begin to decrease the desirability of the action currently associated with the case. This limits an agent from selecting actions that do not move the game towards completion and tends to make the agent slightly more aggressive. Alternatively, if there have been kills and deaths during a learning cycle we immediately reward the action by using the number of kills and punish by each death. The death punishment is lowered to balance bias toward the reward-side.

$$kills + deaths \begin{cases} if\ 0\ then\ weight -= decay + decayMod \\ \quad else \quad weight += kills - (losses * bias) \end{cases} \tag{2}$$

After the adjusted reward for an action has been computed we then look at each of the actions the agent could perform from the current case-to-action reward map to determine which action is the most desirable for the current problem situation of the world to the case. After we have selected the new action it is then inserted and replaces the previous case action so that the CBR system can process it in subsequent game cycles. We also make use of a 10% exploration chance to choose any action in the list which provides the agent with a way to experiment with possibly better action outcomes that have not been discovered yet and to avoid runaway fitness values.

4 Testing

The initial testing phase of our AI system begins with adjusting and analyzing of our CBR metric that controls case generation and generalization. Since the testing of different case generation and generalization values is not enough to evaluate the performance of our problem solving procedure, we pair the case generation metric against three different opponent strategies that are very common during standard game play.

This way we can evaluate which levels of case generalization result in the best problem solving performance in different tactical situations. These opponent strategies consist of an aggressive rush in which the opponent performs an immediate strike with little regard to caution, a defensive opponent that attempts to defend a single location with all its units, and an opponent that uses random strategic strike patterns to attack which may include using all or just a subset of its units to attack. Additionally, we make use of an efficiency metric which is simply a weighted ratio of opponent kills over our bots unit losses. We use this metric primarily to judge how efficient each win really is. Higher win efficiency signifies a better use of unit or map resources. Each test game takes place in a medium sized rectangular arena containing multiple ground height levels and narrowed terrain choke-points, both of which can be used to each player's advantage. Each player is also given an equal number of units to play with (in our tests 28 units) which are placed in equal standings on opposite sides of the arena. Currently, we are not allowing the gathering of resources or the building of structures during these small scale combat scenarios.

Case generation threshold	Rush	Defense	Random
1000	0%	0%	100%
2000	33%	3%	100%
3000	45%	20%	100%

Table 1. Average Wins per Test Set for Three Opponent Strategies

Our testing process consists of running a total of 3 different case generation thresholds against each strategy over 20 game traces for a series of 3 runs and then taking the average values for all recorded statistics resulting in 180 total games per case generation threshold. We save all generated cases in the case library during each test set (20 games) by automatically serializing all cases in the case library to an XML file and begin each new test run with a clean library. During each test set, our AI implementation must learn how to play a game of Starcraft as well as learn what it means to win and lose. This means that initially, our AI does not contain specific knowledge of how to win a game and can only explore these aspects of the game using un-weighted basic sets of micro-level behaviors assigned to six base cases.

AVG Total Cases	Rush	Defense	Random
1000	191	62	204
2000	45	13	16
3000	16	9	15

Table 2. Average Number of Cases Generated per Test Set

Looking at table 1 we can clearly see that our system was able to beat the random opponent strategy 100% of the time during all test runs, but when we switch to the defensive opponent strategy we see a severe drop in wins. This is largely attributed to the fact that an equal number of opposing units can easily defeat an approaching group of units by simply not moving (in Starcraft units cannot attack while moving) and only attacking when the opponent comes in range. As the case generation threshold increases (meaning cases become more generic) we can see that the win percentage increases against all strategies with exception of the random strategy opponent. Our highest win percentage comes from the games against the AI opponent that utilized the rush strategy, however, our implementation was only able to win 45% of the games. The crucial piece of information to take away from this table is that as the case generation thresholds increase our AI walked away with more wins (we will elaborate on possible correlations in the following sections). After playing against the opposition for roughly 8-12 games our AI would learn that it was best to draw a game ending in a stalemate rather than to keep assaulting the heavily defended positions of the enemy. Additionally, as the case generation thresholds reach 3000 our AI becomes aware of this fact much sooner than with lower thresholds.

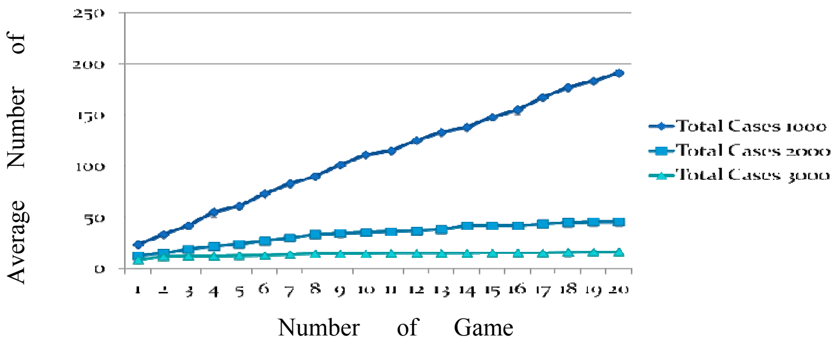


Figure 3. Average Number of Cases Generated per Rush Test Set

In looking at the number of cases that were generated during each test run we can see in table 2 the number of cases created decreases as the case generation threshold increases. This is to be expected because as cases become more generic they are more likely to be reused over a larger set of problem situation instances.

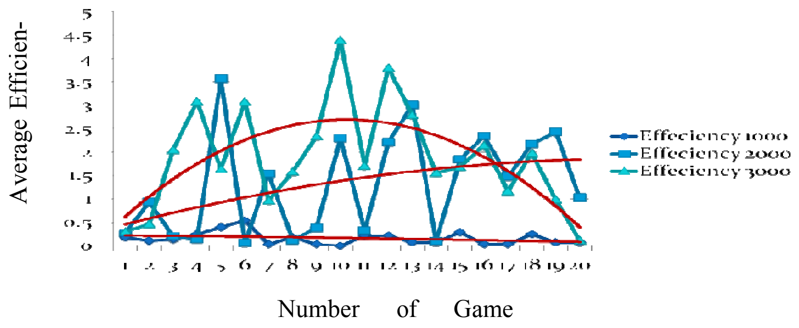


Figure 4. Average Efficiency per Rush Test Set

In figure 3 we can see the total number of cases generated and we observe this case generation process over a period of time. Looking at the cases generated in the 1000 threshold we notice that cases never stop being generated nor do they show any sign of slowing. Conversely, at the 2000 threshold we see a severe tapering of new cases being generated, but still no plateau before reaching the end of the test set. At 3000 thresholds there is a clear end to new case creation roughly around the 7th game trace.

If we now focus on the efficiency values for each case generation threshold in our best performing test runs against the rush opponent strategy we find that in fig. 4 the case 1000 generation threshold level produces an always decreasing efficiency trend line which further makes it less viable for a challenging game opponent. Looking at the 2000 threshold level we notice that the line is always increasing suggesting that further tests should be performed on this threshold level to see the inevitable peak efficiency value. At the 3000 threshold it becomes clear that our AI was actually over trained and that further learning should have been withheld somewhere around the 10th or 11th game as its efficiency peaked very quickly.

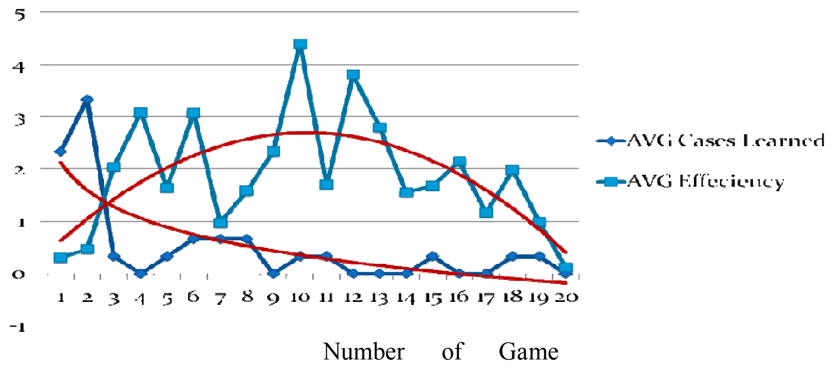


Figure 5. Average Efficiency vs. Learning Comparison in Rush test sets

Our final graph is a simple illustration of a likely correlation between active learning and game play efficiency. During our tests these two values seemed to be inversely related as we can see in fig. 5. Again, we find that the trend line for the average efficiency peaks around 10-12 games while the average number of cases learned begins to slow during this period suggesting that active learning during a game typically has negative effects on the outcome of the match. If we extrapolate our theory to the 1000 case generation threshold where the average number of cases learned is always increasing with each game played, we can also assume that the poor efficiency results and lack of wins in these tests may be largely due to the extreme amount of learning that takes place during these matches. Simply put, the 1000 threshold level generates cases at such a high rate that our RL component cannot effectively identify and reinforce desirable behaviors in the cases, which produces inefficient results.

5 Conclusion

In this paper we have demonstrated that case-based reasoning and reinforcement learning can be used for the selection and adaptation of micro-level decision making processes in real-time strategy games in a dynamic online test environment. Our contributions include the evaluation of the flow and control of online case learning and adaptation, operating within imperfect information domain constraints, and testing against specific strategic opponent AI scripts in challenging multi-dimensional environments. Our results bring about a very interesting property of online dynamic learning in case-based reasoning systems. This property addresses the limits of effective learning during online game play and suggests that for a system such as ours, the amount of learning needs to be carefully moderated using case generalization techniques such as the case generation threshold we use or the techniques suggested by Mateas and Weber as the amount of learning that is performed during play seems to be inversely proportional to the overall problem solving efficiency and therefore, a balance must be struck [2].

Although the current implementation of our project is functionally complete for the problem domain it solves, continued work with CBR regarding macro-level decisions and multi-CBR system integration is the next step. Additionally, we would like to take a second look at the reward/punishment system currently used by the RL component to further refine the metrics used to control the rate of rewards and punishments. We would like to further evaluate higher case generation threshold levels. Finally, we would like to integrate a strategic macro-level CBR system on top of the current system in order to expand the problem solving capabilities and challenge level of our AI.

References

1. Aha, David,. Klenk, Matthew., Laviers, Klenk ., Molineaux, David ,. Sukthankar, Gita. Opponent Modeling and Spatial Similarity to Retrieve and Reuse Superior Plays. In Proceedings of the ICCBR 2009 Workshops (2009) 97-106
2. Mateas, Michael., Weber, Ben. Case-Based Reasoning for Build Order in Real-Time Strategy Games. In Proceedings of the Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2009) [http://www.aaai.org/ocs/index.php/AIIDE/AIIDE09/paper/view/809/1084] (2009)
3. Mateas, Michael., Weber, Ben. Conceptual Neighborhoods for Retrieval in Case-Based Reasoning. In Proceedings of the International Conference on Case-Based Reasoning (ICCBR 2009) (2009) 343-357
4. Aamodt, Agnar., Szczepański, Tomasz. Case-Based Reasoning for Improved Micromanagement in Real-Time Strategy Games. In Proceedings of the ICCBR 2009 Workshops (2009) 139-150
5. Maslow, J., "Using Reinforcement Learning to Solve AI Control Problems."AI Game Programming Wisdom, Vol. 2 Charles River Media, 2003.
6. Mateas, Michael., McCoy, Josh. An Integrated Agent for Playing Real-Time Strategy Games. In Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008) (2008) 1313-1318
7. Sharma, Manu., Holmes, Michael., Santamaria, Juan., Irani, Arya., Isbell, Charles L., Ram, Ashwin. Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL. In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI) (2007) 1041-1046

MMPM: a Generic Platform for Case-Based Planning Research [⋆]

Pedro Pablo Gómez-Martín[1], David Llansó[1],
Marco Antonio Gómez-Martín[1], Santiago Ontañón[2] and Ashwin Ram[3]

[1] Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid, Spain
{pedrop,llanso,marcoa}@fdi.ucm.es
[2] Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
santi@iiia.csic.es
[3] Cognitive Computing Lab
Georgia Institute of Technology
ashwin@cc.gatech.edu

Abstract. Computer games are excellent domains for the evaluation of AI and CBR techniques. The main drawback is the big effort needed to connect AI systems to existing games. This paper presents MMPM, a *middleware* that supports the connection of AI techniques with games. We will describe the features of MMPM, and compare with related systems such as TIELT.

1 Introduction

As several authors have pointed out in the past [1, 5], computer games are excellent domains for the evaluation of AI, machine learning, or Case-Based Reasoning (CBR) techniques; strategy games are well suited for acting as a test bed for all these techniques. The main drawback is the effort needed to connect AI systems to existing games.

A recurrent way used for testing machine learning (ML) techniques is to generate games *traces* containing the complete story of the game sessions played by human experts. Then, the ML algorithms use these traces to infer how to play the game. In a Case-Based Planning (CBP) context, the learning process would extract plans the expert seemed to follow in order to use them in future plays of a game.

These facts were what lead us to the MakeMEPlayME.com idea: an expert plays a game and the game saves a trace that stores every event that has taken place in the session (i.e. actions carried out by game characters) and how the game state has changed afterwards (i.e. changes in entities parameters such as life or position). Traces are the input data for the *learning engine* so it processes these traces and generates a data package which contains the knowledge inferred. This data is what we call a Mind Engine (ME). In this moment the first stage (Make ME) is over. The second stage (Play ME) consists on confronting the ME to play versus a human player or another ME, imitating the behaviours learned from the expert.

[⋆] Supported by the Spanish Ministry of Science and Education (TIN2009-13692-C03-03)

This paper presents MMPM, a *middleware* that supports the MakeMEPlayME.com process. MMPM eases the connection between strategy games and Learning Engines (AI and CBR game playing engines).

In order to prove MMPM, we have connected four different strategy games and a learning engine. The learning engine, known as *Darmok 2* (D2) [9] was based on the MakeMEPlayME.com process, but from an engineering point of view it was tied to the strategy games in such a way that both, games and learning engine, were difficult to reuse. The experience of reengineering them demonstrate to us that MMPM is general enough to support different learning engines and games.

This paper runs as follows: Section 2 is focused on a better explanation of the MakeMEPlayME.com idea and the MMPM middleware architecture. Then, Sections 3 and 4 shows how to define a game domain in a declarative way and how creating Java classes from it to ease the connection between games and Learning Engines. Sections 5 and 6 explained the process of creating a game and a Learning Engine. And the final sections shows some related work and conclusions.

2 General MMPM Architecture

Though theoretically MMPM could be used for different game genres, our main focus is strategy games. Specifically, MMPM expects strategy games with four main components: maps, entities, actions and sensors. Maps are the board where the dynamic elements (or entities) reside. Player and AIs modify the game state using a collection of actions defined in the game rules. When one of the entities performs an action over the map, it may take some time to be completely executed and it can even fail. The other entities are able to inspect the state of the game using some sensors.

From the MMPM point of view, these are the only information it needs to know about the concrete game being played, and it is known as the game domain. Section 3 details how the MMPM users may specify it in a declarative way.

Additionally to games, MMPM assumes the existence of *learning engines* such as D2 [9] (MMPM is well suited for case-based planners, but other learning engines are also possible). As we mentioned above, the essential feature is that they must be able to learn through traces, taken from real player games, in such a way that they should react during a play of a game in the same way as the real player did in the source traces.

From the MMPM point of view, learning engines have two different functionalities: the one who learn from traces (what we call *ME Trainer*) and the one who interprets and executes the previous learned behaviours (*ME Executor*).

With all this in mind, we can state the global experience in MMPM as the one shown in Figure 1. The three steps of a global interaction are:

- An expert plays a game and the game generates a trace that stores every event (*action*), which has taken place during the game, and how the game state has changed due to these actions.
- After the expert has played some plays of a game, he decides to generate a new AI based in his strategies. So, he gives one or more traces to the Learning Engine. Its *ME trainer* processes these traces and generates a data package which contains the knowledge inferred, in the form of a *Mind Engine* (ME).

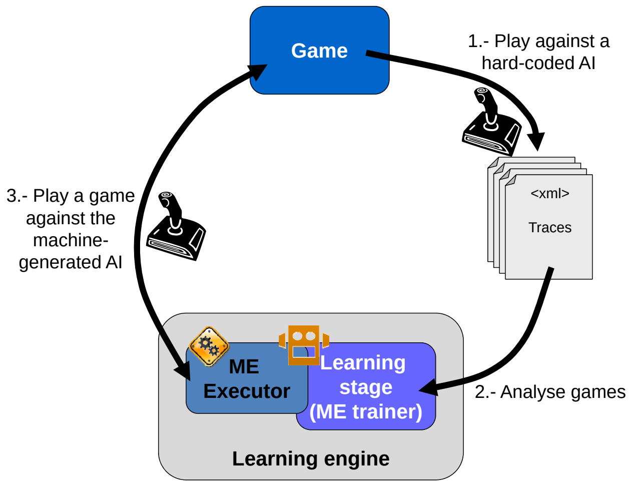

Game
1.- Play against a hard-coded AI
<xml>
Traces
2.- Analyse games
3.- Play a game against the machine-generated AI
ME Executor
Learning stage (ME trainer)
Learning engine

Fig. 1: General Make Me Play Me architecture

– Once the ME has been created, a user may decide to play against this ME. Thus, the game is launched with a *ME executor* that interprets the ME and it can compete against the user imitating the behaviours learned from the traces. Note that it is also possible to configure the game in such a way that we have two MEs generated from different traces or different learning engines playing against each other.

This process is in fact what we call the MakeMEPlayME.com, and is not specific to MMPM. The task of MMPM is to ease the communication between games and learning engines. Furthermore the goal of MMPM is to allow the addition of new games and new learning engines transparently.

This goal is accomplished by standardizing the three aspects where game and learning engines communicate with each other:

– The trace format in order for every learning engine can understand traces of every game.
– The way in which learning engines can access the game state. In other words, how a game sends the results of sensors used by learning engines to extract information from game state.
– The way in which games receive actions from ME executors to execute them in the world.

Traces are stored using an XML Schema. Figure 2 shows an XML fragment (adapted for simplicity) of the trace of a strategy game called *Towers* (one of the four games mentioned in Section 1). There, the `player2 builds` a new `TTower`. This example makes clear that the traces depend on the concrete game being traced. In that sense, MMPM XML Schema only specifies the scaffolding of the traces, and it delays the complete trace format until the game domain has been declaratively specified using the ideas described in Section 3.

```
<gametrace>
  <entry time="10">
    <gamestate>
      <entity id="3" type="TBase" owner="player1" coor="16,16,0">
        <hitpoints>20</hitpoints>
      </entity>
      <action type="Build" id="4">
        <parameter name="playerID">player2</parameter>
        <parameter name="type">TTower</parameter>
        <parameter name="coor">{304,304,0}</parameter>
        <parameter name="lifeTime">10</parameter>
      </action>
    </gamestate>
  </entry>
</gameTrace>
```

Fig. 2: Trace example of game Towers

The communication between games and learning engines is also game dependent. As we will see in Sections 4, 5 and 6, MMPM has a set of tools that automatically generates Java classes from the game domain specification that extends games and learning engines implementations. They should be integrated within the game to generate traces and send the game state to learning engines. As well, they should be used in learning engines to interpret traces during the learning stage (ME trainer), and to receive game states from games, inspect them through sensors and emit game actions in a game session (ME executor).

3 Game Domain

The *game domain* represents all the details about the concrete game and, to some extent, its rules. From the MMPM point of view, it is composed of four kind of elements:

- Entities: they are each piece of interactive content in a game [6]. In other words, they are dynamic objects such as non-players characters or items.
- Actions: all that can be done in the virtual world by players and other entities.
- Sensors: all the aspects that can be measured in the virtual world.
- Goals: the player aspirations in the game.

Each game must provide information about its particularities for all these elements as *declarative knowledge*. MMPM specifies an XML schema that allows game developers to create the domain of the specific target game. The resulting XML will be later used by the learning engines to adapt themselves to each game.

Each entity in the game is composed of a name and the name of the parent entity (MMPM assumes that entities are organized in an entity hierarchy). The description may also include a set of *features* (similar to "fields" or "attributes") that are composed of a name, a datatype and an optional default value. This definition provides an

```
<Entity>
  <Name>TBase</Name>
  <Super>PhysicalEntity</Super>
  <Features>
    <Feature Name="hitpoints" DataType="int" DefaultValue="20"/>
  </Features>
</Entity>
```

Fig. 3: `TBase` entity specification for Towers game

```
<Action name="Build">
  <Parameter name="type" type="ENTITY_TYPE"/>
  <Parameter name="coor" type="COORDINATE"/>
  <Parameter name="lifeTime" type="INTEGER"/>
  <...>
</Action>
```

Fig. 4: `Build` action specification for Towers game

archetype of all the entities of the same type, in a similar way to *classes* in object-oriented languages. When the game state is serialized into traces, the concrete feature values for each entity instance is stored. Figure 3 shows the specification of an entity called `TBase` for Towers.

Actions are the *operators* that players can apply in the virtual world. Figure 4 shows a partial specification of the `Build` action that appeared in the trace on Figure 2, including its name and parameters. When one action is *instantiated* in a trace, real values must be provided for all its parameters.

Under the Make ME - Play ME context, learning engines must analyze when certain actions have been performed in the traces to learn when to use them. In order to help guiding this learning process, and also to help the learning engine understand the relations among actions and how to chain them together, actions specify different kind of *conditions*. For instance, the `Build` action above can only be executed when the coordinates where the building will be built are empty, the building is not blocking the path between player and enemy, and he has enough gold.

Actions in MMPM define the following conditions:

– Valid Conditions: which capture the fact that some combinations of parameters in some actions are not acceptable (e.g. coordinates have to be always inside of the playing area).
– Preconditions: capture conditions which have to be true for the action to start (e.g. the player has to have enough gold, etc.). The distinction between preconditions and valid conditions is that if a precondition is not satisfied, the player might be able to do something to correct this (i.e. mine more gold) but if a valid condition is not satisfied, there is nothing the player can do to correct it.

```
<Sensor name="Gold" type="INTEGER">
  <Code>IntAttrib(Entity(Type("TPlayer"), player),"gold")</Code>
</Sensor>
```

Fig. 5: `Gold` sensor specification for Towers game

```
  <PreCondition>
    <![CDATA[ BuildableCoordinates(coor) && NonBlockingPath(coor)
      && (Gold() >= BuildingCost(type)) ]]>
  </PreCondition>
```

Fig. 6: Precondition of the `Build` action specification for Towers game

– Failure Conditions: MMPM does not assume that actions always succeed, nor that they are instantaneous. Once an action is started, learning engines using MMPM can monitor their execution to determine whether they succeeded or failed. If after the execution of an action the failure conditions are satisfied, it means that the action failed.
– Success Conditions: when these conditions are satisfied, the learning engine knows the action succeeded. Success or failure of an action is thus determined by which conditions get satisfied first: failure or success conditions.
– Postconditions: in addition to success conditions, MMPM allows the definition of additional postconditions, which can be considered as side effects of executing an action.

Thus, action definitions in MMPM differ from traditional action definitions in classical planing languages like STRIPS [3], TIELT [8], or other behavior languages such as ABL [7], in that actions do not just specify preconditions and postconditions. But specify a richer set of conditions, specifically devised to allow the learning engine to reason about actions, and to monitor real-time execution of them.

Action conditions are satisfied or not depending on the world state. MMPM provides a set of *native sensors* that can be combined to specify basic queries about the current world state. These sensors can be parameterized, and they return a concrete value used in the condition expression. Sensors use the same basic datatypes available for entities features and action parameters. MMPM also allows games to specify their own complex sensors using the native ones. For example, in the `Build` precondition mentioned previously, "has gold enough" is described using a new sensor created in the game domain combining the MMPM sensors (Figure 5).

Using the new `Gold` sensor (and some others), the `Build` precondition could be specified in the way shown in Figure 6.

Keep in mind that while entities and actions are serialized into traces, sensors and conditions are static information, useful for the learning engines but that are not stored in traces in any case. Sensors can be also used for communicating the game and the ME executor while playing a game, in a similar way to actions.

```
<WinGoal name="WinGoal">
  <Code>
    FLOAT(NumberOfEntities(Type("TBase"), player)) /
        FLOAT(NumberOfEntities(Type("TBase")))
  </Code>
</WinGoal>
```

Fig. 7: `WinGoal` for Towers game

Finally, the domain is completed with *goals*. They are *conditions* specified by sensors, boolean and relational operators, that are used by learning engines for guiding their training. As sensors and actions, goals are also static information that is not serialized.

Games must at least specify one `WinGoal` which, when true, indicates the end of the game and the victory of the player (Figure 7). They can also provide non-final goals that act like desired milestones in the game session.

4 Supporting Tools

The communication process between games and ME Executors is not resolved yet. Moreover, to promote developers work adapting games and Learning Engines to MMPM, tools that eases the trace generation in games and trace reading in Learning Engines should be provided.

A generic communication model (e.g. Web Services) could be used to connect games with ME Executors, but this would make the integration quite complex and would slow down the execution of games. To avoid these problems, we assume games and Learning Engines will be implemented using Java is done in other research projects such as Weka [4] and jColibri [2]. Furthermore the Java imposition still allows connections with other languages if the needed bindings were programmed.

Once we assume Java as implementing language, MMPM provides a a tool that automatically generates classes from the domain specification mentioned in Section 3. For instance, Figure 8 shows part of the `TBase` entity class automatically generated from the XML domain (Section 3). Apart from translating information described in the domain, the tool adds methods for serializing and deserializing traces in the XML format mentioned in Section 2.

All these generated classes are based on several general classes that conform the MMPM middleware. There are superclasses for entities, actions and sensors and support for MMPM native sensors and types. Binding our middleware to Java has an additional advantage: when there is not a way to define a new game sensor using the MMPM native sensors, it can be defined using plain Java in the XML domain. Although `Gold` sensor (Figure 5) was able to be specified using a native expresion, `NonBlockingPath` sensor that `Build` precondition used (Figure 6) cannot be specified in this way. Nevertheless it can be defined in the XML domain in Java code as in Figure 9.

```
public class TBase extends PhysicalEntity {
  public TBase(String entityID, String owner) {
    super(entityID, owner);
    _hitpoints = 20;
  } // Constructor
  ...
  protected int _hitpoints;
  ...
} // class TBase
```

Fig. 8: Fragment of the java code generated for the `TBase` entity

```
<Sensor name="NonBlockingPath" type="BOOLEAN">
  <Parameter name="coor" type="COORDINATE"/>
  <Code language="java">
    <![CDATA[
      boolean[] blocked = new ...
    ]]>
  </Code>
</Sensor>
```

Fig. 9: Java sensor example

As we will see in Section 6, this alternative and non declarative way for defining sensors is not a problem for Learning Engines. They are written in Java too, so they can evaluate these sensors executing them.

5 Developing a Game for MMPM

First of all, game developers must specify the game domain as explained in Section 3 and they should create and generate the Java classes (Section 4). These classes will be very useful to generate game traces (the first stage in the MakeMePlayMe.com process shown in Figure 1) and to let learning engines execute actions within the game (the third stage).

If game developers wanted to adapt a previously created game, they would need to write methods to convert the original game state representation into objects of the MMPM classes and vice versa.

On the other hand, in case the game was not developed yet, the auto generated classes could be even used as the skeleton of the logic of the game. Every entity, action and sensor class would be generated with its features and ready to be serialized, so game developers should save a lot of time using them as part of the game itself. They would have to implement how actions modify the game but a lot of work would be already done.

During a play of the game, traces must be generated. So, the game must store its game state and produced actions every certain period of time and then, the trace has to be sent where the one who launched the game specified. MMPM provide a *tracer*, a set of Java classes that the game invoke periodically with the MMPM game state objects to be stored.

The other connection that must be made is between the game and a ME executor. First of all, MMPM has a ME executor factory that games may use to create the machine-generated AIs. Afterwards, the game invoke them using MMPM game state objects to produce the MMPM actions according to the strategy inferred by the Learning Engine. Game code will translate these game-domain actions into the implementation-dependent actions that are used to perform them into the game.

6 Developing a Learning Engine for MMPM

A Learning Engine is compound of two different systems: a ME Trainer and a ME Executor. A ME Trainer is an off-line system that does not interact with games but with their game traces. From the MMPM point of view, and to ensure independency between systems, the Learning Engine is a black box that receives game traces and creates a ME from them. In order to manipulate game traces easily, Learning Engines should use the auto-generated domain classes. Furthermore, if the Learning Engine uses MMPM super classes as native classes, it can make use of their serialization to store MEs in the ME Trainer system and to load them again in the ME Executor.

On the other hand, a ME Executor is bound to games and it must react quickly. It receives game states and reacts returning actions to be executed in the game. The machine-generated classes must be used because game states are given, and actions must be returned to games, as objects of these classes. Moreover, using these classes is the only way that Learning Engines have to evaluate sensors and action conditions, which are, somehow, the rules of the game. Action conditions should be checked before returning them to the game in order to know if their execution in the actual game state makes sense.

7 Related Work

The need for a system such as MMPM that ease the complexity of developing Learning Engines and connecting them to games was identified in the past by Molineaux and Aha [8], who designed the TIELT system to address this problem.

Our goal is to encourage progress in this area and offer an alternative to TIELT, which is more suitable for certain kind of games. The main differences between TIELT and MMPM are the problems they try to solve and the formalism to define some of the game domain knowledge. For instance, while TIELT also attempts at solving the problem of performing experiments, MMPM only attempts at bridging AI components with games. Another big difference between TIELT and MMPM is the way actions are specified. For TIELT, an action is defined by a set of capable roles (who can execute this action), a set of preconditions, and a set of state changes (that define how the state will change after executing this action). The state changes are defined as a script, which defines the state transformation done by the action. MMPM has a richer action definition language, allowing: valid conditions (restrictions on the input parameters), preconditions, success conditions (conditions that if satisfied we can consider the action succeeded), failure conditions (conditions that if satisfied after the action was initiated we know that the action will never succeed), and additional postconditions (with a similar meaning to the state changes in TIELT). This action definition was defined with the goal in mind of allowing the AI systems to easily monitor the execution of actions at run-time, which is hard to achieve using TIELT.

8 Conclusions

This paper has presented MMPM, a *middleware* that supports the connection between games and learning engines, promoting the development of machine learning techniques in games. In that sense, MMPM recalls some other research projects such as TIELT.

We have also adapted an existing Learning Engine known as *Darmok 2* (D2) to be used with MMPM. D2, that uses Case-Based Planning to create machine-generated AIs from expert traces, has been successfully used with up to four different games that were also adapted to be used with MMPM.

Currently we are working on extending the game domain to allow more complex actions (or operators).

References

1. M. Buro. Real-time strategy games: A new AI research challenge. In *IJCAI'2003*, pages 1534–1535. Morgan Kaufmann, 2003.
2. B. Díaz-Agudo, P. A. González-Calero, J. A. Recio-García, and A. A. Sánchez-Ruiz. Building CBR systems with jCOLIBRI. *Special Issue on Experimental Software and Toolkits of the Journal Science of Computer Programming*, 69(1-3):68–75, 2007.
3. R. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intellicence*, 2(3/4):189–208, 1971.
4. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, June 2009.
5. J. E. Laird and M. van Lent. Human-level AI's killer application: Interactive computer games. In *AAAI 2000*, pages 1171–1178, 2000.
6. N. Llopis. *Introduction to Game Development*, chapter Game Architecture, pages 267–296. Charles River Media, 2005.
7. M. Mateas and A. Stern. A behavior language for story-based believable agents. *IEEE intelligent systems and their applications*, 17(4):39–47, 2002.
8. M. Molineaux and D. W. Aha. Tielt: A testbed for gaming environments. In *AAAI*, pages 1690–1691, 2005.
9. S. Ontañón, K. Bonnette, P. Mahindrakar, M. Gómez-Martín, K. Long, J. Radhakrishnan, R. Shah, and A. Ram. Learning from human demonstrations for real-time case-based planning. *IJCAI-09 Workshop on Learning Structural Knowledge From Observations (STRUCK-09)*, 2009.

Toward a Domain-independent Case-based Reasoning Approach for Imitation: Three Case Studies in Gaming

Michael W. Floyd and Babak Esfandiari

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

Abstract. We describe a domain-independent approach to learning by observation that uses case-based reasoning to allow a software agent to behave similarly to an expert when presented with a similar set of input stimuli. Case studies have been performed in two simulated domains, soccer and space combat, and a physical robotics domain and our experimental results show that successful imitation is possible in each of those domains.

1 Introduction

Transferring an expert's knowledge to a software agent can be a difficult task. This is especially true if the expert has difficulty modelling its knowledge, possibly if they lack computer programming skills, or they are not fully aware of all details related to how they perform a task. *Learning by observation* attempts to shift the burden of knowledge transfer from the expert to the software agent. The agent learns by watching the expert perform a task and, when faced with the same task, aims to behave in a similar manner.

Ideally, such an agent should operate completely autonomously from the expert and not require the expert to provide any prior knowledge. Existing approaches to learning by observation, however, require some level of background knowledge. This background knowledge can include information about the tasks or goals of the expert [1, 2], which sensory stimuli the expert reasons with [3] or how specific actions influence the environment [4]. While all of this information helps when learning a specific task, bias is added which makes it difficult to learn other unrelated tasks. Ideally we want to be able to imitate unrelated behaviours without adding extra expert knowledge so that the learning system can be deployed in a variety of domains. The primary contribution of our own work [5–7] is a learning by observation system that makes minimal prior assumptions about the expert although we have only experimented in a simulated soccer domain.

The interactions between the expert and its environment can be thought of as a series of environment states and actions by the expert [8]. Both the possible environments states, $\mathcal{S}$, and possible actions, A, are finite sets containing all environment states and actions that may be encountered:

$$\mathcal{S} = \{S', S'', \dots\} \tag{1}$$

$$A = \{A', A'', \dots\} \tag{2}$$

When an agent is observed for a period of time, a *run*, R, of environment states and actions will be observed:

$$R : S_0 \overset{A_0}{\rightarrow} S_1 \overset{A_1}{\rightarrow} S_2 \overset{A_2}{\rightarrow} \dots S_{u-1} \xrightarrow{A_{u-1}} S_u \tag{3}$$

The goal, when learning by observation, is to use one or more runs of an expert to approximate how the expert selects which actions to perform based on the state of the environment:

$$A_i = f(S_i, A_{i-1}, S_{i-1}, A_{i-2}, \dots) \tag{4}$$

The remainder of this paper will detail our domain-independent approach to learning by observing agents and examine what types of agents it is suitable for imitating. Section 2 will provide the model of our agent and case structure. An analysis of several properties of software agents, and their environments, are described in Section 3. Three case studies in the domains of robotic soccer, space combat, and physical robotics will be described in Section 4. These case studies will be used to examine the relation between an agent's properties and the ability of our system to imitate that agent. A summary of related work will be presented in Section 5, followed by conclusions and areas for future work in Section 6.

2 Agent Model

When looking at the ability of a software agent to reason, we can think of the current run as the *problem* that is to be solved and the performed action as the *solution*. This leads us to define a case, C, as a pair containing the current run, R, and the performed action, A.

$$C = \{R, A\} \tag{5}$$

The run is, as shown in Equation 3, a sequence of past environment states and actions. The state of the environment can be decomposed into the sensory stimuli that can be observed. If, in a particular environment, an agent can observe m *types* of stimuli then the environment state can be written as:

$$S = \{V_1, \dots, V_m\} \tag{6}$$

Each type of stimulus, V_i, is represented by a multi-valued attribute that contains all n_i *instances*, o, of that type of stimulus ($V_i = \{o_1, \dots, o_{n_i}\}$). This implies that the number of stimuli of a particular type can vary in different cases. The need to allow for a varying number of instances of each stimulus type is related to the fact that the agent may only have a partial view of its environment at any given time. For example, in a soccer domain the number of teammates an agent can observe in its field of vision may change depending on where the agent is looking. At different times the agent might see no teammates, all of the teammates, or any number in between. Even in domains where an agent has a complete world view it might still be necessary to treat each stimulus as a multi-valued attribute. This is particularly true if objects can be added or removed from the environment, or if the agent is unable to uniquely identify similar objects.

Using such an approach, cases can be generated in an automated manner through passive observation. Using a passive approach, the *expert*, as shown in Figure 1, interacts with the *environment* and may not be aware it is being observed. The *observer* is able to view, and record, both the state of the environment and the actions performed by the expert so that they can later be used to construct cases.

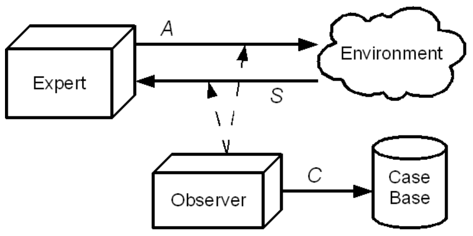


Fig. 1. Passive observation of an expert interacting with the environment

After automatically building a case base by observing an expert, the learning agent will then be able to use that case base when it interacts with the environment. When the agent observes a new environment state, it should ideally perform the same action the expert would have performed had the expert encountered a similar run. This requires using the current run as a query to the case base. The case base is searched using a standard k-nearest neighbour search to find a case with a run that is a minimal distance from the current run. Once the nearest neighbour search has found the case that is most similar to the current state of the environment, the action associated with the retrieved case is used. Our approach does not perform any adaptation since the agent is attempting to behave like the expert and has no idea what goal it is trying to achieve (a more detailed description of our case description and retrieval can be found in [5]).

The data flow of our approach can be thought of as a three step process: acquisition, preprocessing and deployment. As we mentioned previously, the cases are *acquired* in a completely automated manner by observing the interactions between the expert and the environment. While this automatic case acquisition allows for quick and inexpensive case generation it make no guarantees as to what cases will be acquired or what information is contained in those cases. For example, if the expert never performs a certain action or encounters a certain environment state then those actions and states will not be available in the case base. Likewise, since no domain knowledge is provided by a domain expert such knowledge will need to be extracted from the cases. This is addressed through *preprocessing* the raw case base by performing automated tasks like feature selection, removal of redundant cases and ensuring the case base can be searched within real-time limits (a more detailed account of preprocessing can be found in [6]). Once an acceptable case base has been created it can then be *deployed* in order to allow the imitative agent to attempt to imitate the behaviour of the expert.

3 Agent and Environment Properties

Now that we have described how case-based reasoning can be used to imitate the behaviour of agents, we will turn our attention to the agents and their environments. The following lists several properties that will be examined and used to classify agents:
Goals: Each agent will perform a behaviour that attempts to achieve certain goals. The imitation system should ideally be able to imitate agents regardless of the goals they try to achieve and should require no knowledge of what those goals are.
Observability: The agent can have either a complete or partial view of the environment. With a complete view the agent is able to observe the entire environment at once, however this is often not possible due to limitations on the available sensors. Instead, most realistic agents can only view a portion of the environment at once. Partial observability can result in a variable number of objects visible to the agent at any given time which may result in the need to model sensory inputs as multi-valued attributes (it may also be necessary in fully observable environments if objects can be added or removed from the environment).
Noise: The sensory observations received by the agent may be subject to noise. This noise can be small, resulting in values being different from their true values, or large making some sensory information completely unknown to the agent. For example, if an object was located far from an agent the agent might be able to detect what type of object it is observing but not finer details about the object.
State: The agent may not reason exclusively with its current sensory inputs but may maintain an internal state. Since this state information is not directly visible to an observer it makes the imitation task significantly more complicated.
Environment Physicality: The agent could either be in a simulated environment or a physical environment (like controlling a robot).

4 Case Studies

We look to examine several agents with different goals and behaviours. Two simulated domains, soccer and space combat, and one physical domain will be used. A summary of the agents and their properties can be seen in Table 1.

Domain	Agent	Noisy	Fully Observable	State	Physical
Soccer	Krislet	Yes	No	No	No
Soccer	Sprinter	Yes	No	No	No
Combat	Shooter	No	No	No	No
Robot	Avoid	Yes	No	Yes	Yes
Robot	Arm	Yes	No	No	Yes

Table 1. Summary of agents and their properties

4.1 Simulated Robotic Soccer

The first domain we examine is RoboCup [9] simulated robotic soccer (this has been the focus of our previous research [5–7]) . The following types of objects exist in RoboCup: soccer balls, goal nets, boundary lines, boundary flags, teammates, opponents, unknown players[1]. At any given time the agent may see between *0* and *80* total objects, each of which is represented by its continuous-valued distance and direction (relative to the agent). The state of the environment in a simulated RoboCup game can be represented as:

$$S_{SOCCER} = \{ball, goal, line, flag, teammate, opponent, unknown\} \tag{7}$$

Each item in S_{SOCCER} represents a multi-valued attribute containing the objects of that type that are currently visible in the agent's field of vision. The agent can kick, move forward (dash) or turn its body. This makes the set of possible agent actions:

$$A_{SOCCER} = \{kick, dash, turn\} \tag{8}$$

In our experimentation we use two different agents as experts[2]. The first agent we use, Krislet, actively attempts to achieve the goals of soccer. Krislet agents turn until they can see the soccer ball, run toward the ball and then try to kick the ball toward the opponent's goal net. The second agent we use, Sprinter, does not perform typical soccer behaviour. Sprinter agents repeatedly run from one goal net to the other and make no attempt at scoring goals. These agents were selected to show that even in the same domain agents can have very different goals and behaviour, so any background knowledge about the goals of a particular expert may not be transferable to other experts.

For each expert a case base containing *5000* cases was used (this size of a case base was selected so that the case base could be searched within the real-time limits of the agent). A series of *1000* testing cases, or more specifically the

[1] Players can be of an unknown type if the agent can not tell what team they are on due to noise. Although RoboCup is a simulated domain there is a built in noise model to add realism.

[2] While we refer to the agents as experts we do not imply they are complex in nature. We use expert to refer to an agent with some amount of knowledge we wish to attain.

environment states of the cases, were used as input to the imitating agent. The action outputted by the imitating agent, who only considered the most recent sensory input (a run of length 1), was then compared to the known action from the case. The overall performance was measured using the f-measure statistic with values ranging from 0 (low) to 1 (high).

The results are shown in Table 2. In general, the f-measure values are fairly high for both agents (it should be noted that previous work [6] has shown that these results can be significantly improved by automatically preprocessing the case base, although no preprocessing was performed in these experiments). The exception is the results associated with *kick* in Krislet and *turn* in Sprinter, which lower the overall f-measure values. The reasons these values are lower is because the data sets are severely imbalanced, with less than 1% of Krislet cases having the *kick* action and less than 5% of Sprinter cases having the *turn* action (Sprinter never kicks, which is why there is no f-measure value for the kick action). However, even though the f-measure values are low for those actions when the case-based reasoning system is deployed in a game of soccer the actions appear to be performed properly. When watching the CBR agent play soccer, it is difficult to differentiate between that agent and the original expert agent.

	f-measure	f_{dash}	f_{turn}	f_{kick}
Krislet	0.59 (+/- 0.005)	0.71	0.80	0.25
Sprinter	0.69 (+/- 0.005)	0.91	0.48	—

Table 2. Results when learning from RoboCup agents

4.2 Simulated Space Combat

Thus far we have shown the ability of our system to imitate agents, with a variety of different behaviours, in the RoboCup soccer domain but we now look to extend our experiments into another simulated domain. The domain we use is XPilot [10], a simulated space combat game, where an agent controls a space ship and attempts to destroy enemy ships. In this domain the agent is able to detect the location of enemy ships (between *0* and approximately *10* other ships) and can move, turn or shoot:

$$S_{XPILOT} = \{ships\} \tag{9}$$

$$A_{XPILOT} = \{move, turn, shoot\} \tag{10}$$

The agent we imitate, which we will call *Shooter*, continuously turns until it can see at least one enemy ship and then shoots at the nearest ship. The results, when using a case base of *1000* cases and *1500* test cases, are shown in Table 3 (the Shooter agent does not perform the *move* action so that action is not listed in the table). These results are similar to what we got in the RoboCup experiments and is what we would have expected given the similar properties of the agents and the environments. Like with the RoboCup experiments the poorest performance was related to rare actions, in this case the *shoot* action, although this could likely be improved by preprocessing the case base to balance the number of cases related to each action.

	f-measure	f_{turn}	f_{shoot}
Shooter	0.64	0.77	0.50

Table 3. Results when learning from XPilot agent

4.3 Physical Robots

Our experiments up to now have exclusively examined simulated domains, but now we turn our attention to two agents that control physical robots. The first agent, which we will call *Avoid*, controls a small wheeled robot and attempts to avoid collisions with obstacles. The robot has two sensors, sonar and touch, and can perform four movement actions:

$$S_{AVOID} = \{sonar, touch\} \tag{11}$$

$$A_{AVOID} = \{forward, backward, left, right\} \tag{12}$$

If an object is detected as being close to the robot, using the sonar sensor, the robot will turn left or right to avoid a collision. If it collides with an object, as indicated by the touch sensor, it will move backward and then turn. Otherwise it will just move forward. While this behaviour is mostly reactive in nature, the agent does have a simple internal state that determines which direction it should turn. The internal state ensures that the robot toggles between turning left and right.

The second agent, called *Arm*, controls a robotic arm robot. This robot has three sensors and can perform five actions:

$$S_{ARM} = \{sound, touch, colour\} \tag{13}$$

$$A_{ARM} = \{armForward, armReverse, armStop, closeClaw, stopClaw\} \tag{14}$$

Upon detecting a significantly loud sound, on the sound sensor, the arm begins moving forward until the touch sensor signals it has come in contact with an object. If the colour sensor ever determines a red object is within the claw's grasp the claw will be closed around the object and the arm will move in reverse. However, if it ever determines a blue object is within the claws grasp it will not close but instead move the arm in reverse.

There are three primary differences, not including that they control physical robots, between these agents and the ones we have previously examined. First, these agents perform sequences of actions in response to each input instead of a single action. Secondly, since a sequence of actions is performed each action in that sequence is performed for a set amount of time. This requires each action to have an associated parameter related to how long it should be performed. Lastly, the sensory attributes are single-valued, with one attribute per sensor, rather than multi-valued.

For both agents, our system was able to imitate the behaviour very well. For each agent a case base was created by giving the agent *500* randomly created sensory inputs and recording the resulting action sequence. Using a separately generated set of *1000* testing cases each agent was able to perfectly reproduce, with 100% accuracy, the expected behaviour of the agent. This includes selecting the correct action sequence and executing each action for the correct duration[3]. However, for the Avoid agent, this perfect accuracy is only achieve by considering *left* and *right* turns identical. This occurs because two cases can have identical sensory information but different actions since the cases do not contain the internal state of the agent when it selected which actions to perform. This demonstrates that the presence of an internal state, even in a domain that can be imitated with high accuracy, can significantly complicate the imitative process.

5 Related Work

Case-based reasoning has been successfully applied to a variety of games and simulations including robotic soccer [11–13], American football [14], real-time strategy [15] and first-person shooters [16]. One promising use of CBR in games has been to create cases by observing experts. Examples of this include Tetris [17], real-time strategy [1], poker [4], chess [18] and space invaders [19].

The primary difference between these works and our own is that the processes for case generation and case retrieval are optimized for the specific domains and therefore are not suitable for general purpose learning by observation. The work most similar to our own involves using case-based planning in real-time strategy games [2]. They show their approach to be applicable to several different games, however domain knowledge is required for each new game. This knowledge involves having an expert define the goals of the game and the state of the environment associated with each goal.

Learning by observation has also been explored using other machine learning approaches. In robotics, learning from observation has been used in tasks such as controlling a robotic arm [20] and teaching a robotic helicopter aerial manoeuvrers [21]. Both of these approaches use a model for the movements of the robots, with the model parameters learnt through observation. This requires prior knowledge of an appropriate model to use and does not take into account any external stimuli. Grollman and Jenkins [22] have developed a system that allows a robotic soccer player to be simultaneously controlled by a human and

[3] The agent only uses a finite set of discrete action durations. If there was more variability in these durations then we would likely see error in the durations produced by the imitating agent.

an autonomous system. When the human is actively controlling the robot, the autonomous system learns from observing the human. This approach has been successful for learning simple behaviours, like moving or kicking, but has difficulty identifying when transitions between simple behaviours should occur.

Multi-layered perceptrons have been used to control the behaviour of agents in a first-person shooter game [3]. This approach requires a fixed set of input features, so it is not suited for situations where the inputs are multi-valued attributes. Similarly, learning by observation has been used to train virtual agents to move in life-like ways [23]. The movements of the agents are rated based on a fitness function, so each novel behaviour requires a user to define an appropriate fitness function.

6 Conclusions and Future Work

The technique for learning by observation using case-based reasoning described in this paper allows an agent to learn without any knowledge of the goals of the expert being observed. Three domains with different environments and goals have been studied. Software agents, with a variety of properties, were able to be successfully used as experts. The primary contribution of this approach is that a single case-based reasoning system is used in all domains. The only difference is the data contained in the cases, with the types of sensory stimuli and actions varying in the different domains.

While we have shown this approach to be applicable it three different domains, there still exist several limitations that provide areas for future work. The experts that have been observed have been largely reactive in nature, however more complex experts will likely have behaviour that is strongly related to their internal state. Even for the obstacle avoidance robot, which only had a simple internal state, it was not possible to fully account for the agent's state. We will look at how the entire run, instead of just the last environment state, can be used to better imitate stateful behaviour.

References

1. 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
2. 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)
3. 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)
4. Rubin, J., Watson, I.: SARTRE: System overview. A case-based agent for two-player texas hold'em. In: Workshop on CBR for Computer Games at the 8th International Conference on Case-Based Reasoning. (2009)

5. 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
6. 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
7. Floyd, M.W., Esfandiari, B.: An active approach to automatic case generation. In: 8th International Conference on Case-Based Reasoning. (2009) 150–164
8. Wooldridge, M.: An introduction to multiagent systems. John Wiley and Sons (2002)
9. RoboCup: Robocup official site. http://www.robocup.org (2010)
10. XPilot-AI: Xpilot-AI A library for writing XPilot bots. http://www.xpilot-ai.org (2010)
11. Wendler, J., Lenz, M.: CBR for dynamic situation assessment in an agent-oriented setting. In: AAAI Workshop on Case-Based Reasoning Integrations. (1998)
12. Steffens, T.: Adapting similarity measures to agent types in opponent modeling. In: AAMAS Workshop on Modelling Other Agents from Observations. (2004) 125–128
13. Ros, R., Veloso, M., de Mántaras, R.L., Sierra, C., Arcos, J.L.: Retrieving and reusing game plays for robot soccer. In: 8th European Conference on Case-Based Reasoning. (2006) 47–61
14. Aha, D.W., Molineaux, M., Sukthankar, G.: Case-based reasoning in transfer learning. In: 8th International Conference on Case-Based Reasoning. (2009) 29–44
15. 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
16. 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
17. Romdhane, H., Lamontagne, L.: Reinforcement of local pattern cases for playing Tetris. In: 21st International Florida Artificial Intelligence Research Society Conference. (2008) 263–268
18. Flinter, S., Keane, M.T.: On the automatic generation of cases libraries by chunking chess games. In: 1st International Conference on Case-Based Reasoning. (1995) 421–430
19. Fagan, M., Cunningham, P.: Case-based plan recognition in computer games. In: 5th International Conference on Case-Based Reasoning. (2003) 161–170
20. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Fourteenth International Conference on Machine Learning. (1997) 12–20
21. Coates, A., Abbeel, P., Ng, A.Y.: Learning for control from multiple demonstrations. In: 25th International Conference on Machine Learning. (2008) 144–151
22. Grollman, D.H., Jenkins, O.C.: Learning robot soccer skills from demonstration. In: IEEE International Conference on Development and Learning. (2007) 276–281
23. Dinerstein, J., Egbert, P.K., Ventura, D., Goodrich, M.: Demonstration-based behavior programming for embodied virtual agents. Computational Intelligence **24**(4) (2008) 235–256

Provenance-Aware Case-Based Reasoning

Applications to Reasoning, Metareasoning, Maintenance and Explanation

Workshop at the
Eighteenth International Conference on
Case-Based Reasoning

Alessandria, Italy
July, 2010

ICCBR 2010

David Leake, Thomas Roth-Berghofer, Barry Smyth,
and Joseph Kendall-Morwick (Eds.)

Organizing Committee

David Leake (Chair)
Indiana University, USA
leake@cs.indiana.edu

Thomas Roth-Berghofer
German Research Center for Artificial Intelligence DFKI GmbH, Germany
thomas.roth-berghofer@dfki.de

Barry Smyth
University College Dublin, Ireland
barry.smyth@ucd.ie

Joseph Kendall-Morwick
Indiana University, USA
jmorwick@cs.indiana.edu

Program Committee

External Reviewer

Preface

Provenance has emerged as an active research area in scientific computing and the semantic Web. Case-based reasoning (CBR) systems exploit the results of prior reasoning, by storing cases for reuse, but comparatively little attention has been devoted to how CBR can exploit knowledge of the provenance of those cases. Such knowledge may include information about external sources of cases and the context in which they were captured, derivation traces of cases and other meta-data. For example, case provenance information can be used to assess trust in cases, based on their sources, or as a basis for metalearning to determine the effectiveness of the various adaptation strategies, based on their results. This workshop investigates the interplay of case provenance with areas such as trust and reputation, reasoning and metareasoning, and explanation.

Many domains offer opportunities for capture of provenance and meta-data useful for CBR applications. For example, in e-science, provenance information from the execution of scientific workflows provides a knowledge source for generating workflow cases. In e-commerce, information about communities of recommenders may provide useful information for assessing and applying their recommendations. In help-desk domains, information about the context in which faults occur (e.g., available technicians and client locations) may provide important clues for fault diagnosis and recovery.

Likewise, the CBR cycle itself provides numerous opportunities for the capture and use of provenance information, for example to inform reuse and retention. The use of such 'internal' provenance can inform many tasks in the wider CBR process, such as case-base maintenance, explanation, and confidence estimation.

The workshop's broad goals include: (1) clarifying the nature of provenance, trust, and reputation, as they relate to CBR; (2) examining how provenance information may be used at multiple points in the CBR cycle, and (3) advancing the state of the art in relation to how provenance and meta-data should be captured, represented, and exploited in CBR systems. To enable a synthesis of perspectives, the workshop program includes both paper presentations and time for extensive discussion.

The five papers presented here illustrate some of the wide range of opportunities for provenance-aware CBR, including using traces as a knowledge source and container of experience (Cordier & Mille), using provenance to enrich the CBR process (Leake & Kendall-Morwick, McSherry), and using it to support explanation (Kofod-Petersen, Aamodt, & Langseth, Roth-Berghofer & Adrian). We see this work as an exciting first step towards deeper understanding and further exploration of provenance-aware case-based reasoning.

David Leake July 2010
Thomas Roth-Berghofer
Barry Smyth
Joseph Kendall-Morwick

Dynamic Case-Based Reasoning for Contextual reuse of Experience

Amélie Cordier, Bruno Mascret, Alain Mille

Université Lyon 1, LIRIS, UMR5205, F-69622, France
{firstname.lastname@liris.cnrs.fr}

Abstract. This paper reviews Trace-Based Reasoning (TBR), a reasoning paradigm based on interaction traces left by users in digital environments. Because interaction traces record the users' problem-solving experiences in context, TBR facilitates the reuse of such users' experiences. Moreover, interaction traces can be used as a knowledge source to discover other knowledge useful for the reasoning process. This paper describes TBR principles and proposes a common framework for TBR applications. Especially, we focus on the articulation between Trace-Based Reasoning and Trace-Based Systems in the context of user assistance. For this purpose, we describe knowledge and knowledge models involved in reasoning tasks. We discuss the advantages of using traces as knowledge containers for reusing of experience and we show how TBR is related to the research field of provenance. Finally, we report a brief state of the art and a work agenda for TBR. We also show why TBR can take advantage of the current dynamic of the work about traces while globally improving trace-based applications.

1 Introduction

This paper takes up the challenge of developing dynamic and reactive systems that can quickly and incrementally adapt to changes in users' needs and habits. It has been consistently argued that, in many cases, users' needs and habits cannot be fully anticipated *a priori*. The incomplete anticipation of user needs complicates the development of reasoning mechanisms that could make the system self-adaptable. Typically, in reasoning based on previous experiences, such as Case-Based Reasoning, the system's ability to adapt is limited by knowledge models and by the fact that reasoning mechanisms are defined at design stage.

We introduce here the *Trace-Based Reasoning* (TBR) paradigm and argue that this paradigm supports the development of more flexible and scalable systems. While interacting with a system, a user leaves behind *interaction traces* that constitute digital inscriptions of the users' experiences. We consider interaction traces as knowledge containers in which experiences are implicitly stored but are not really organized in advance. Previous experiences, called *episodes*, are only retrieved when a specific need appears. This mechanism ensures flexibility and adaptability of the process, but also raises complex issues (e.g. how to find a previous experience in a complex trace?). In TBR, episodes are always linked to the trace that contains them. Consequently, at any time, it is possible to retrieve contextual elements related to the current episode and to use them to improve the reasoning process. Hence, TBR allows us to handle experiences in a much less constrained way than if they were caught in structured cases.

This paper is organized as follows. In section 2, we quickly describe recent work that seeks to develop more dynamic and scalable applications, in particular within the CBR field. Section 3 presents TBR, starting with a short description of the Trace-Based System concept. Then, we discuss the issues raised by the development of trace-based applications and we show the possible benefits of TBR in the field of user assistance. The paper concludes with a discussion of our research agenda for Trace-Based Reasoning.

2 Towards dynamic and evolving experience-based applications

Recently, several suggestions have been made to push the limits of Case-Based Reasoning. For example, the *Agile CBR* idea described by Susan Craw in [1] makes one step towards a more dynamic approach to CBR. The main idea of Agile CBR is to transform traditional CBR into a "dynamic, knowledge-rich, self-organizing, cooperative problem-solving methodology". Agile CBR draws from agile programming and focuses, amongst other aspects, on individuals and interactions as well as on quick responses to change. Hence, according to Susan Craw, "Agile CBR's opportunism, commitment and flexibility aims to providing the adaptability needed to design processes and workflows, and focusing first on enabling just-in-time information". TBR works the same way. Traces provide us with rich knowledge sources that should be helpful to address the knowledge management issues raised by the Agile CBR challenge. For example, combining several users' experiences to solve one specific problem can be efficiently addressed with a reasoning process based on fusion of individual interaction traces.

The notions of traces and provenance are also closely related. As it is shown in [2], there is a growing interest on the concept of knowledge provenance and on how to use provenance information for various tasks (capturing experience, improving similarity assessment, maintaining case bases, etc.). In this work, the authors show that CBR always records prior problem-solving experiences in the form of cases, but does not record the case provenance. The authors also show that provenance information is often valuable. With traces, we always keep provenance information because we find "cases" in the trace only when we need them. We believe that research on provenance in CBR and in TBR can be usefully combined,

The idea of taking context into account in the reasoning process is also explored. For example, [3] shows that combining Case-Based Reasoning methodology and context awareness is a new and powerful way of modeling and reasoning from contexts. This contribution shows how user behaviors can be learnt in a case-based way, which could be much more efficient with a TBR approach. Illustrating this idea, [4] uses time-stamped activity logs called *paths*. Paths are collected by client-side tracking processes implemented in several tools (such as web browsers and text editors, etc.). These paths are then exploited to gather information used to feed recommender systems.

What we call *episodes* can be seen as historical cases as described by [5] in the Historical Case-Based Reasoning (HCBR) framework. This framework allows the expression of both relative and absolute temporal knowledge, representing case histories in the real world. In HCBR, a case base is formally defined as a collection of (time-independent) cases, together with its corresponding temporal reference. A TBR approach includes all these features and adds several other possibilities, such as context extension, abstraction level navigation, provenance assessment, dynamic elaboration, etc.

Other researches, linked to experience sharing, such as described in [6], can be related to the work described here. In HeyStaks (http://www.heystacks.com), users try to share their search experiences with others. The authors propose a classification approach to increase the relevance of a *stak* (a list of interesting search results) because a stak may have been noised for several reasons (spam, mistakes, out of date records, etc.). The most interesting point here is that even though experience is collected, the system still needs help to determine which real context is used. Instead of using only classifiers, we could combine them with TBR in a revision step based on interesting elements contained in the history of the stak construction. We think that traces can be helpful in such a situation where context is changing and not given directly by the system.

As an example of what could be an assistant using traces and a CBR approach, we can cite [7] who uses Case-Based Reasoning techniques to predict the user's goal(s) from a sequence of his workspace (inter-)actions needed to achieve this goal. A TBR approach generalizes this ability to assist the user for whatever could be found in its interaction traces as previous experiences.

3 Trace-Based Reasoning

A Trace-Based System (TBS) is made of three main components: a Trace-Bases Management System (TBMS), one or more observed applications and one or more trace-based applications. The general architecture of such a system is presented on figure 1 and detailed below.

The core object of this architecture is the trace. A trace reflects the result of the observation of a given situation. In figure 1, the observed situation is that of the use of the application by the user. A trace consists of a set of elements called *obsels* (for observed elements). A modeled trace is a trace to which is associated a model that formally defines the structure and the types of obsels that can be contained in the trace, as well as the relationships between these obsels . The obsels are temporally located within the trace. We distinguish two types of traces: primary traces, and transformed traces. Primary traces are the results of the collect process. By definition, they are intended to be untransformed.

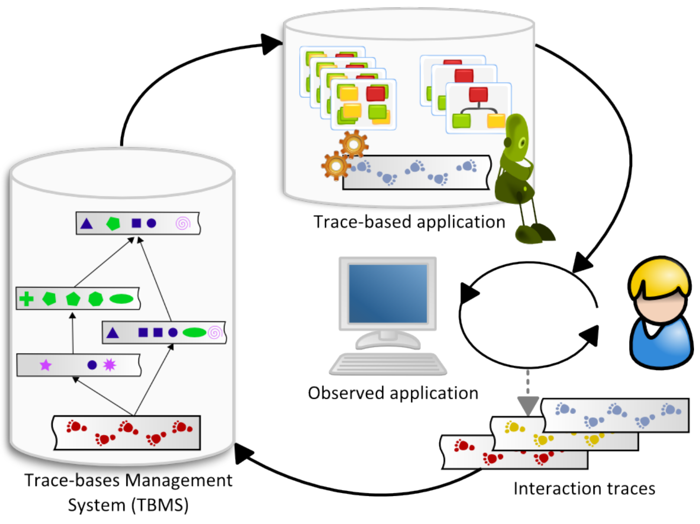


Fig. 1. Trace-Based System: a general architecture. User produces traces stored in the TBMS by interracting with an observed application. The trace-based application uses traces to solve problems and may ask the TBMS to apply transformations. The solution is proposed to the user and this process is also recorded on the trace.

Transformed traces are the results of transformation operations performed on the primary trace or on other existing transformed traces.

The Trace-Based Management System is the component that manages the various trace bases. Trace bases contain primary traces as well as the whole set of transformed traces, and the associated models. A TBMS provides input/output primitive functions to manipulate these traces (read, write, update, transform). Traces are stored in the TBMS during the collect process. This process exploits one or several collect sources that can have various natures. Either the target application is instrumented so that it automatically sends the elements that have to be collected to the TBMS, or an intermediate process builds the primary trace from available observation elements (such as log files of the observed application). Hence, an application, a tool, a set of log files, a trace from another provenance, etc. constitute as many collect sources. In brief, the collect process builds the primary trace that can subsequently be transformed using existing transformation operators.

The notion of trace transformation is of major importance. A transformation is an operation on one or several modeled traces. The result of a transformation is a new modeled trace, linked to the source traces. We can distinguish several types of transformation operators, among them: filtering, merging and reformulating operators. A filtering operator consists in applying a filter on a given trace in order to produce a new trace containing only the obsels matching the rules of the filter. Here is a very simple example. If we consider a trace containing obsels of several colors, we can define a filtering operator that will keep only the "red" obsels. Applying such a filter will transform the initial trace in a transformed trace containing only red obsels. Obviously, transformations can be more complex and may combine several operators. Merging and reformulating operators work the same way but in addition, they can dynamically produce new types of obsels (for example, two obsels in the initial trace may be combined and represented by a third one, dynamically created) in the produced trace.

Transformations can be automatic, semi-automatic or manual. A TBMS guarantees the possibility, at any time, to navigate between transformed traces. Indeed, the TBMS always keeps a link between the transformed trace and the traces from which it comes from. Besides, it records the transformation operators that have been used to perform the transformation. In other words, the TBMS stores a lattice of transformed traces in order to guarantee the possibility to navigate between transformed traces. Transformed traces can be seen as various representations of the same situation, at different abstraction levels (this idea is illustrated in the example at the end of this section). Because traces at different abstraction levels are linked to eachother, it is possible to consider a trace at a specific level of abstraction, and, at some point, to consult another trace representing the same situation at a more specific abstraction level. This possibility to navigate between various modeled traces thus provides an important flexibility.

Another way to understand transformations could be to simply consider them as knowledge: a transformation is performed *because* some knowledge gets the system to activate it. So transformations may be both knowledge and processes. This new point of view raises open issues discussed later on this paper.

The last element of the architecture is the tool that uses traces. This kind of tool can be connected with the TBMS through input/output primitive functions provided by the TBMS. These tools can also connect with the observed application, but this is out of the scope of this paper. These tools have various natures: (interactive) visualization of traces, analysis of usages, user assistance, etc. Tools have to rely on specific knowledge to accomplish certain tasks (described below) with traces. We call this type of knowledge "reasoning knowledge".

Trace-Based Reasoning. While reasoning, the first tasks consist in retrieving previous episodes in the trace to reuse them. For episode retrieving, we introduce the concept of task signature that contains a set of distinctive elements characterizing an episode. With each task signature, we associate a set of similarity measures that allow us to retrieve the most similar episode in the trace, and a set of adaptation rules (see below). Another difficulty adds up to the traditional issue of finding similar episodes: the search has to be performed among the various transformation levels of the trace. Therefore, the search has to take into account the various possible representations of a single problem.

Once retrieved, the episode can be reused in various situations: visualization, interactive visualization, replay, or adaptation. In case of visualization task, we have to exploit the available presentation knowledge in order to display the episode to the user in a nice and understandable way. Interactive visualization relies on the same principle, but additionally allows the user to navigate between the different abstraction levels of the trace. Interactive visualization allows us to handle various viewpoints on the same problem in an efficient way: for the user, it is possible to consider a problem at a high abstraction level and then to go to a lower level (to more specific transformation traces) if more specific details are needed. Replaying episodes consists in retrieving an episode and replaying it the same way. The possibility to replay an episode implies strong constraints on the collect phase. One of these constraints is to make sure that the whole set of observations has been collected in the form of obsels. Last, adapting an episode consists in reusing it and, possibly, transforming it to provide help to the user (assistance, recommendations, automatic task execution, etc.). Within the TBR context, it is necessary to rely on a specific (and possibly generic) structure for adaptation. This structure has to offer a way for the system to identify the elements that can be subject to an adaptation in the episode. We face the same problem than in CBR and, as we have showed it in a previous work [8], it is possible to reuse some of the solutions provided by CBR researches in this context. In TBR, however, an episode adaptation can be envisioned at several levels: content adaptation, interaction modality adaptation, presentation adaptation, navigation between different abstraction levels (i.e. adaptation by navigation between transformed traces). This diversity among the different types of adaptation introduces an additional complexity from the point of view of the knowledge representation necessary for the reasoning.

Figure 1 illustrates one specific issue, e.g. the issue of designing a trace-based assistant. The challenge here is to reuse experiences to provide assistance to the users. The system can reuse the same users' interaction traces or can browse interaction traces to reuse chunks of other users' experiences.

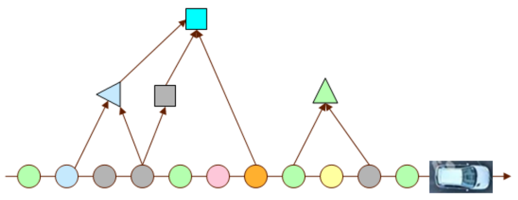


Fig. 2. Several transformations of a single trace. Transformation operators are combined to produce a more abstract trace. The more the trace is abstracted, the more it contains knowledge.

An example with Abstract. To give an example of how traces can be collected, transformed and used at several abstraction levels, we briefly describe the Abstract project [9]. Abstract is a cognitive science project intended to provide means for a better understanding of human activities using a computer recording of these activities. From a computer science point of view, this project aims at discovering knowledge from traces of activity. The Abstract application starts by observing the activity by various means (i.e. various kinds of sensors) and collects a primary trace. The set of data collected is also attached to a moment of the activity identified by a "time-code". The trace can then be manipulated and enriched with more abstract symbols through several transformations. Figure 2 and figure 3 illustrate the way traces are manipulated in Abstract. Figure 2 shows three abstraction levels of the primary trace from a generic point of view. Figure 3 illustrates the same principle, but applied to a specific application domain: driver behavior analysis.

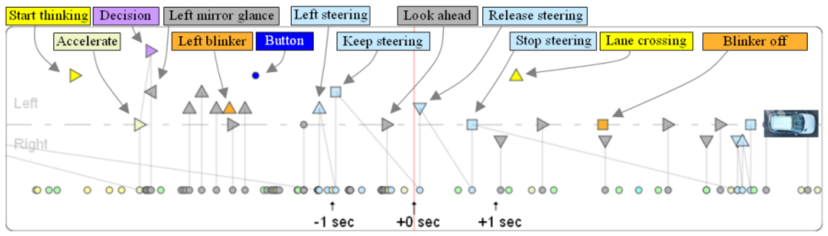


Fig. 3. Abstract: an example of the user interface. The primary trace has been transformed to produce traces "understandable" by users (e.g. analysts).

Summary. In a trace-based application, the Trace-Bases Management System (TBMS) has an operational role: it manages the traces and their transformations. It also provides primitive functions to manipulate traces and allows navigation between the different levels of representation. The trace-based application, meanwhile, relies on presentation and reasoning knowledge. Hence, the application can achieve various goals: visualizing an episode, reusing it, adapting it, etc. A major issue is that of knowledge evolution and of knowledge models. The implementation of mechanisms supporting knowledge evolution is an open problem. If the update of certain knowledge is immediate (by definition of the concept of trace), the update of knowledge models, particularly in the case of modeled traces, is more tedious and may require a process involving an expert. We consider the issue of updating a knowledge model as a major challenge for TBR.

4 Discussion and concluding remarks

Trace-Based Reasoning is a problem solving paradigm in which both the user and the system solve problems and learn new knowledge at the same time. It is through interactions that the user gives additional knowledge to the system and that the system provides the user with solutions to specific problems. In this process, traces constitute a flexible support used at the same time by humans and machines. Traces keep a record of interactions, and thus of the problem solving process. In some extent, this is why we argue that they allow us to record problem solving experiences “in context” and that they constitute a rich knowledge container.

One way of introducing the concept of Trace-Bases Management System is through the analogy with a file system. The file system organizes, stores, and accesses the files, but the users can’t directly exploit the files without any specific applications to read, modify or transform them. It is the same problem with traces: without any specific software to manipulate them, the traces stored in a TBMS usually have little interest for end users. We believe that TBR, because it offers a dynamic, responsive, and scalable approach for the collect and reuse of experience, is very promising. Moreover, the work about traces is currently very active and TBR could advantageously benefit from it.

Among researches on TBR, we can distinguish two categories. The first category includes work concerned with theoretical aspects of traces and their management (modeled trace theory, mining traces to find relevant patterns, interactive visualization, etc.). The second category includes researches that are intended to produce trace-based applications (Trace-Based System for behavior analysis, user assistance, learning purpose, etc).

Within the second category, issues related to transformation are challenging. Indeed, the role of transformations is of utmost importance. A transformation may be seen in several ways: historical, such as a TBR’s process using operators to transform a trace in a higher abstraction level one; knowledge-oriented, which considers transformations as knowledge containers; combined, which mix ideas issued from these two approaches. This last question raises new issues: can all kind of knowledge be represented through transformations, and are transformations a good way to represent knowledge?

Traces and provenance address complementary issues. The common understanding of provenance in CBR is that a system may improve its process if it has knowledge on how cases have been elaborated. As shown previously, traces embed information on provenance and consequently, this provenance information can be used by the TBR approach. Provenance information may be exploited for various purposes. For example, in an experience sharing system, provenance is used for confidence purposes: provenance may help the system to determine what ”kind” of experience should be reused in a given context. Let us consider the copy/paste example (illustrated figure 4. Imagine a blind user trying to use a sighted person’s trace. Most of the sighted people use the mouse to perform a copy/paste with the contextual menu. However, blind people usually prefer to perform this action with the keyboard only. To support experience sharing between blind and sighted people, the first step is to consider the traces at the right abstraction level. In this example, if we consider the low-level obsels (such as ”Ctrl+C” or ”Right Click + Copy”), we cannot provide any help. However, if we consider obsels that belong to a higher abstraction level (such as ”Copy”), we can then support experience sharing between different users. Once the right level of abstraction is found, the second step consists of adapting the way of performing the action (”Copy”) by taking into account the user preferences (e.g. ”only with the keyboard”). This example suggests an interesting observation: while most users use the mouse to perform a Copy/Paste, blind users may prefer less common solutions. Hence, provenance information may help a user to find the best available episodes for him, or at least to give him relevant adaptation knowledge (how to present “paste”: with mouse or with keyboard?). We think this issue should open new paths especially in adaptive systems and assistive technologies.

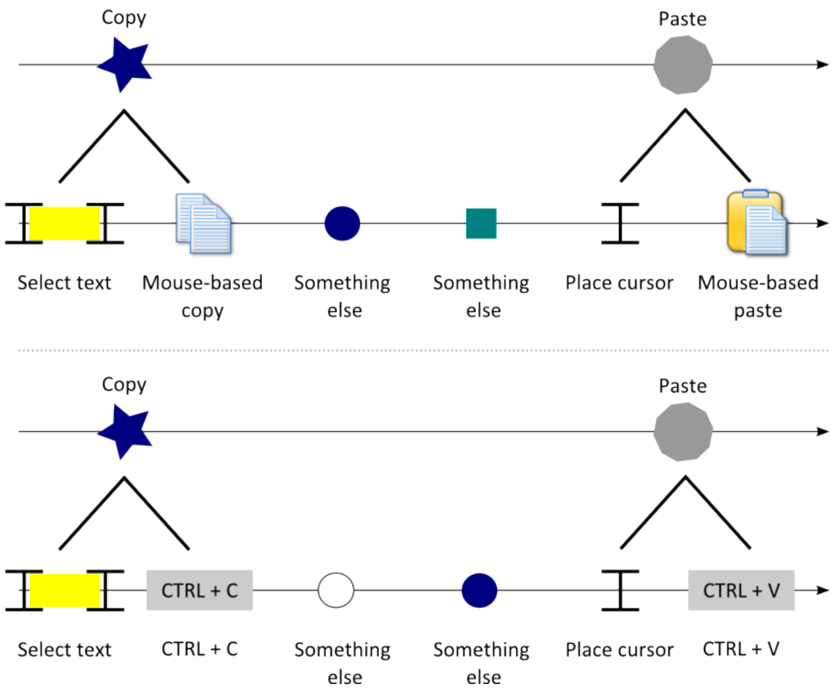


Fig. 4. Copy/paste example. The trace of the mouse-based copy/paste (above) is compared to the trace of the keyboard-based copy/paste (below). At the highest abstraction level, both traces are identical. However they represent two different ways of performing the same task. Note that in both cases, the bottom traces are not primary traces, they are already transformed from other traces (for example ”Right Click + Copy Selection” has been transformed in ”Mouse-Based Copy”).

The issues raised by the implementation of TBR are various and many questions arise: how to gather experience through observing interactions? How to define transformation mechanisms for modeled traces and how to make them evolve? How to adapt retrieved episodes and how to present the results of this adaptation to end users? How to make the system knowledge evolve, and from what sources? Are traces sufficient or should we rely on other sources of knowledge, and if so, how to integrate them in the system? All these questions are research issues that we intend to explore. We will focus our researches on assistance systems which are a challenging experimentation field.

Acknowledgments

Authors wish to thank the workshop organization team for providing this opportunity to discuss the notion of provenance from very different viewpoints, and the reviewers for their constructive and helpful comments.

References

1. Craw, S. Agile case-based reasoning: A grand challenge towards opportunistic reasoning from experiences. In *Proceedings of the IJCAI-09 Workshop on Grand Challenges in Reasoning from Experiences*, pp. 33-39, Pasadena, CA, 2009.
2. Leake, D.B and Kendall-Morwick, J. Four Heads Are Better than One: Combining Suggestions for Case Adaptation. In *Proceedings of ICCBR-09*, pp. 165-179.
3. Zimmermann, A. Context-awareness in user modelling: Requirements analysis for a case-based reasoning application. In *Case-Based Reasoning Research and Development*, pp. 1064–1064, 2003.
4. Chalmers, M. Abstract paths and contextually specific recommendations. In *Proceedings of DELOS/NSF Workshop on Personalisation and Recommender Systems in Digital Libraries*, Dublin, June 2001.
5. Ma, J. and Knight, B. A framework for historical case-based reasoning. In *Case-Based Reasoning Research and Development*, pp. 1067–1067, 2003.
6. Champin, P.A., Briggs, P., Coyle, M. and Smyth, B. Coping with Noisy Search Experiences. In *29th SGAI International Conference on Artificial Intelligence (AI-2009)*. Springer, pages 5-18, Cambridge, December 2009.
7. Schwarz, S. and Roth-Berghofer, T. Towards goal elicitation by user observation. In *Workshop on Knowledge and Experience Management* at GI FGWM, LLWA 2003.
8. Cordier, A., Mascret, B. and Mille, A. Extending Case-Based Reasoning with Traces. In *Grand Challenges for reasoning from experiences*, Workshop at IJCAI'09, Pasadena, CA. 2009.
9. Georgeon, O., Henning, M. J., Bellet, T., and Mille, A. (2007). Creating Cognitive Models from Activity Analysis: A Knowledge Engineering Approach to Car Driver Modeling. *International Conference on Cognitive Modeling*, Ann Arbor, MI: Taylor & Francis, pp. 43–48.

Explanations in Bayesian Networks using Provenance through Case-based Reasoning

Anders Kofod-Petersen, Helge Langseth, and Agnar Aamodt

Department of Computer and Information Science,
Norwegian University of Science and Technology,
7491 Trondheim, Norway
{anderpe|helgel|agnar}@idi.ntnu.no

Abstract. Bayesian Networks are useful for solving a wide range of problems in many domains. Yet, they are exposed to one important challenge when structural and parametrical changes occur. As Bayesian networks lack memory regarding changes over time, there is currently no good way of maintaining a history of changes and their provenance. Thus, any variance in the network's problem solving behaviour will not be explainable to a user. Within the context of systems that integrate case-based reasoning and Bayesian networks, we suggest to add a case-based reasoning functionality that will retain changes and their provenance, as well as approaches to explain any unexpected problem solving behaviour.

1 Introduction

Explanations have been identified as one of the most important properties of intelligent systems (see e.g. [1,2]). One important source of knowledge for generating useful explanations is the information related to changes in used knowledge sources and changes in reasoning processes.

Although provenance-related information has been included in some case-based reasoning (CBR) systems since the early days of the field, and often referred to as book-keeping information, justifications, or meta-information: a systematic treatment of provenance and its role was introduced by Leake and Whitehead [3]. They identified two main roles of provenance in case-based reasoning; to compensate for delayed feedback in the REVISE phase of the case-based reasoning cycle [4], and to improve case base maintenance. The role suggested here is an additional one, namely to produce explanations for unexpected results that stem from changes in parts of the domain model. In a philosophical treatment of provenance and explanation [5] the focus is on "how" and "why" type explanations relate to provenance, and in a treatment of explanation and provenance within a Description Logics framework. An emphasis is on how to track "how-provenance" in a system, referring to how sources of provenance contribute to the observed data.

In the work reported here the domain model could in principle be a case-specific domain model as well as one containing generalised knowledge, but in the present paper we focus solely on changes in the general domain knowledge, and more specifically in the Bayesian network (BN) component of a system.

Bayesian networks [6,7] constitute a modelling framework for uncertain knowledge. With their formal grounding in probability theory and statistical inference, Bayesian networks have become among the most used frameworks for reasoning under uncertainty, and has found applications in areas as diverse as genetics, failure detection, recommender systems, and speech recognition. One important argument for using Bayesian networks is its ability to adapt both its structure [8] and its parameters [9] as new training data is presented to the system; properties which are crucial in domains that are partly unknown or changing.

The nature of Bayesian networks also allows for some explanations to be given regarding the reasoning process (see Lacave and Díez [10] for a review). However, there is one important aspect where Bayesian networks cannot explain changes. That is when the perceived behaviour of the system has changed via learning (via adaptation of the Bayesian network, or – in a hybrid system – by adding cases to the case base). In other words, a Bayesian network might experience changes in structure, e.g. new observable parameters can appear, or changes in parameters, e.g. a change in the probability distribution in the probability function in a node. As Bayesian networks traditionally retains no history of the provenance of such changes made during learning, any variation in results coming from such changes are inexplicable.

In previous work [11], we have been looking into the benefits of combining Bayesian networks and case-based reasoning into a hybrid system, with the goal to utilise the Bayesian networks either as a reasoning engine in its own right, or to help with the case matching in the integrated system. This is an endeavour we have taken up again recently, and although in this paper we will exemplify the reasoning processes using only a Bayesian network, we do still have the total hybrid system in mind.

The work presented here suggests the use of case-based reasoning to address the lack of ability to explain changes in the perceived behaviour of systems including Bayesian components. In brief, we propose to construct cases based on changes in the Bayesian network and use this in combination with a model of the user to construct explanations. Sørmo et al. [12] identified five explanation goals: Explain How the System Reached the Answer (Transparency), Explain Why the Answer is a Good Answer (Justification), Explain Why a Question Asked is Relevant (Relevance), Clarify the Meaning of Concepts (Conceptualization), and Teach the User About the Domain (Learning). Our focus here is on the transparency and justification goals.

The rest of this paper is organised as follows: Section 2 introduces a motivational example for augmenting Bayesian networks with a case-based reasoning provenance part; Section 3 describes how case-based reasoning can be applied and how the cases can be constructed; The paper ends with a summary and an outlook on future work.

2 Motivational Example

To exemplify the need for provenance of changes in a Bayesian network we will use a very simple example. A simple Bayesian network, as depicted in Fig. 1, can be used to tell a user whether to go to the beach or not. Real world application would probably be much more complicated and even make recommendation without being explicitly asked to do so. However, for pedagogical reasons we will stick with simplicity.

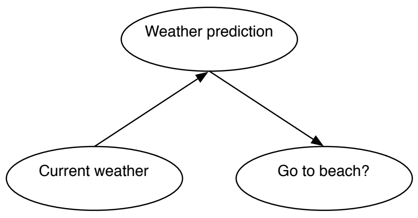


Fig. 1. Bayesian network

The network can observe the current weather and if it is raining or cloudy will tell the user to stay at home, and if it is sunny it will tell the user to go to the beach. The column marked 'Historical recommendation' in Table 1 shows the mapping between weather conditions and expected recommendations. This explicit mapping is naturally not visible to the user. However, we can assume that a user who uses any system on a regular basis will develop some expectation as to how they behave.

Table 1. Condition and response

Condition	**Historical recommendation**	**New recommendation**
Rain	Stay at home	Stay at home
Cloudy	Stay at home	Go to beach / stay at home
Sunny	Go to beach	Go to beach

Let us assume that the system now gets equipped with a barometer, which to some degree can be used to predict the weather. The new Bayesian network, including the barometer is depicted in Fig. 2. Using this new device the network is able to predict that cloudy weather in some situations will improve and become sunny. Thus, the system will sometimes suggest the user to go to the beach even when it is cloudy, for example when the barometer shows a trend towards higher pressure. This new observed behaviour is shown in the column marked 'New recommendation' in Table 1.

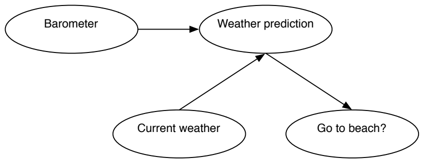


Fig. 2. Modified Bayesian network

The first time the system makes an *unexpected* prediction, i.e. suggests going to the beach even when it is cloudy, the user might like to see an explanation of the apparent inconsistency. A Bayesian network can supply an explanation regarding the relation between the observed variables (the current weather and the barometer reading), the weather forecast, and the query variable. However, as aforementioned no explanation is readily available regarding the fact that there has been a structural change. The types of explanation available from the Bayesian net are clearly not sufficient as they do not include the real provenance of the change in the reasoning process. Since the system behaves in an unexpected manner, the user is likely to be interested in a justification of the change in the expected outcome.

3 Case-based Reasoning as an Explanation Mechanism

The work presented here suggests to augment Bayesian networks with a case-based reasoning component to explain changes in behaviour. This is achieved by retaining the provenance of either procedural or structural changes in the problem-solving network.

When a request for an explanation is received, the case-based reasoning system will retrieve a case containing a relevant part of the previous network that solved the problem (in the example predicting whether or not to go to the beach), and the current network. New cases are constructed whenever a user asks a question to the system. In this example, when a request for a recommendation for visiting the beach is made. If the answer to the question and the input parameters are sufficiently similar to an existing case, no new case is retained. The request for an explanation indicates an unexpected solution or insufficient confidence in the suggested solution, for which a transparency and/or justification explanation is called for. Figure 3 depicts an example case.

In the problem description part of a case, the findings are: a user model, which can be used to influence the explanation goal to be solved and the way to present it; the current network; and the last know accepted network (marked in

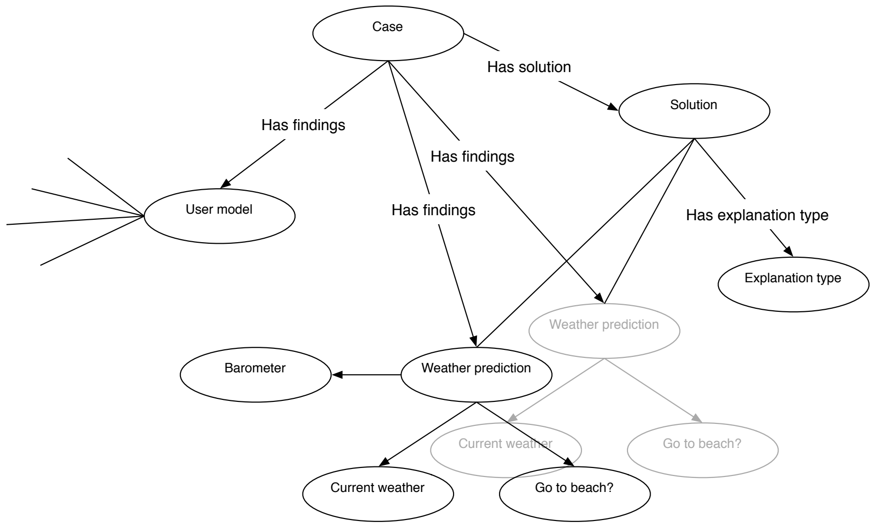


Fig. 3. Case structure

light grey). The solution to the case contains the explanation goal and the two Bayesian networks. The latter can be used as a knowledge container [13] where relevant knowledge to construct the explanation can be acquired.

The user model contains information about the user in question. Typically it would contain some user identifier, and importantly information about the level of skills of the user. This information is important when deciding the type of explanation to provide. As an example, if the user is a highly skilled expert the *transparency* explanation is likely to be the preferred one, where as for novices a *justification* is more likely to be preferred (for a relationship between user competencies and transparency versus justification, see e.g. [14]).

In the case of the user requesting an explanation, as an example in the form of: "The last time it was cloudy, you told me to stay at home. However, now you are telling me to go to the beach. Why?" The current case is matched against the last case recommendation expected by the user. The current case contains the current Bayesian network, which is then matched against the network in the retrieved case. The difference between the two cases from the basis for the explanation.

To continue the example from Section 2 and augment it with the case-based reasoning provenance system, we can now supply an explanation as to why the answer differs from the expected one. Comparing the two networks in question (see Fig. 3) shows that the current network contains a new node, the barometer. This barometer affects the system's ability to predict the weather. Given that it is cloudy and the barometer says that there is a high pressure, it will presumably be sunny shortly. So the explanation offered by the system might be: ”Since the last time that cloudy weather was observed, when I suggested to stay at home, I have received a barometer. The barometer tells me that even though it is cloudy, there is a high pressure. Thus, it is very likely that it will be sunny at the beach.”

4 Summary and Further Work

We are currently investigating suitable approaches to implement the suggestions made in this paper. A suitable implementation in a domain closer to the real world must be carried out in order to test the validity of this Bayesian networks and case-based reasoning hybrid approach.

So far we have suggested how augmenting the Bayesian network with the case-based reasoning system allows the system to supply a *transparency* explanation, to e.g. an expert, and a *justification* explanation, to e.g. a novice.

It is likely that with a suitable structuring of the knowledge model, other types of explanation are possible, either directly from the Bayesian network or from the case-based reasoning provenance system. These types of explanations could include the complete list from [12]. Different types of explanations satisfies different goals [2]. Among these are, perhaps the least explored, namely *learning*. The learning type allows a user to learn about the domain in question. This might be a two-way street, where the user might be able to teach the system why a new solution is worse that an old one. Combined with the other types of explanations –where the user is the producer and the system the consumer– changes to the Bayesian network might be suggested to the system. However, this conversational perspective on explanations is in an early stage, and for far remains unexplored.

Finally, other types of relevant provenance information, such as the context of the problem-solving system and the rationale for changes in the problem-solver's recommendations, could be combined with the suggested approach to further improve users' trust and acceptability of reasoning systems.

References

1. Leake, D.B.: Evaluating Explanations: A Content Theory. Lawrence Erlbaum Associates (1992)
2. Roth-Berghofer, T.R., Cassens, J.: Mapping goals and kinds of explanations to the knowledge containers of case-based reasoning systems. In Muñoz-Avila, H., Ricci, F., eds.: Case Based Reasoning Research and Development – ICCBR 2005. Volume 3630 of LNAI., Chicago, Springer (2005) 451–464
3. Leake, D., Whitehead, M.: Case provenance: The value of remembering case sources. Case-Based Reasoning Research and Development (2007) 194–208
4. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications **7**(1) (1994) 39–59
5. Lockie, R.: Knowledge, Provenance and Psychological Explanation. Philosophy **79**(03) (2004) 421–433

6. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
7. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs. 2nd edn. Springer (2007)
8. Friedman, N., Goldszmidt, M.: Sequential update of bayesian network structure. In: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann (1997) 165–174
9. Spiegelhalter, D.J., Lauritzen, S.L.: Sequential updating of conditional probabilities on directed graphical structures. Networks **20** (1990) 579–605
10. Lacave, C., Díez, F.: A review of explanation methods for bayesian networks. The Knowledge Engineering Review **17**(2) (2002) 107–127
11. Aamodt, A., Langseth, H.: Integrating Bayesian networks into knowledge intensive CBR. In: American Association for Artificial Intelligence, Case-based reasoning integrations; Papers from the AAAI workshop – Technical Report WS-98-15, Madison, WI., AAAI Press (1998) 1–6
12. Sørmo, F., Cassens, J., Aamodt, A.: Explanation in case-based reasoning – perspectives and goals. Artificial Intelligence Review **24**(2) (October 2005) 109–143
13. Richter, M.M.: The Knowledge Contained in Similarity Measures. Invited Talk at the First International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra, Portugal (1995)
14. Kofod-Petersen, A., Cassens, J.: Explanations and context in ambient intelligent systems. In kokinov, B., Richardson, D.C., Roth-Berhofer, T.R., Vieu, L., eds.: Modeling and Using Context, proceedings of the 6th International and Interdisciplinary Conference (CONTEXT 2007). Volume 4635 of Lecture Notes in Artificial Intelligence., Roskilde, Denmark, Springer (August 2007) 303–316

External Provenance, Internal Provenance, and Case-Based Reasoning[⋆]

David Leake and Joseph Kendall-Morwick

School of Informatics and Computing, Bloomington
Indiana University, Lindley Hall 215
150 S. Woodlawn Avenue, Bloomington, IN 47405, U.S.A.
{leake, jmorwick}@cs.indiana.edu

Abstract. The capture, representation, preservation and sharing of provenance information has become a central part of e-Science research, motivated by many benefits of integrating provenance into scientific processing. Initial work on considering provenance in CBR is promising, raising the question of whether provenance should become a core part of CBR as well. This position paper illustrates the potential of provenance-aware CBR by identifying the types of roles provenance can play for CBR systems, and argues for the value of a CBR "provenance revolution" enabled by systematically integrating provenance capture, reuse, and sharing within CBR.

1 Introduction

Provenance traces information about the agents, actions, and inputs responsible for the origin or current state of a product. In information technology, provenance often refers to the derivation trace for a reasoning result. Extensive research in the e-Science community addresses the capture and storage of provenance information, motivated by the recognition that provenance can play many roles in increasing the usefulness of results of *in silico* experiments. For example, Simmhan, Plale and Gannon [1] observe that provenance can serve tasks such as supporting attribution of scientific results, auditing data generation, auditing resource usage, assessing data quality, and replicating results. Recently, CBR researchers have begun to explore the use of provenance within CBR. This position paper identifies key potential uses of provenance within the CBR process to argue for the systematic integration of provenance tracking and use within CBR systems.

Some initial research forays have explored ramifications of provenance for CBR systems [2–4]. Provenance tracing could play important roles in such diverse CBR tasks as case capture, case adaptation, explanation, confidence, case-base

[⋆] This material is based on work supported by the National Science Foundation under Grant No. OCI-0721674. We thank Beth Plale and the Indiana University SDCI group at IU for shaping many of these ideas through their collaboration. We also thank the anonymous reviewers for their helpful comments.

maintenance, and introspective learning to refine case-based reasoning systems. However, there has not yet been an attempt to categorize the many potential roles of provenance in CBR. We see this as a crucial step.

However, we also note that the potential value of provenance collection frameworks goes beyond any specific list of predefined tasks. As has been observed in e-Science, collecting and retaining provenance is important even without immediate benefits. The motivation for retention is that, as for any historical information, routine information gathering can provide a wealth of information for future study, information which might be difficult or impossible to collect after the fact. With the right representations and capture mechanisms, CBR systems will be positioned to take advantage of new provenance-aware enhancements (e.g., learning methods) as they are developed.

To support both current and future steps in provenance-aware CBR, we propose a two-step agenda. The first step is to categorize ways in which provenance may be used within CBR. The second step, building on the first, is to develop a general scheme for representing, manipulating, and examining CBR system provenance, ideally in concert with efforts to develop general tools to support the capture and use of provenance. This position paper focuses primarily on the first step, with an initial categorization of uses for provenance which both motivate the development of a general provenance framework and determine requirements for what it must represent. It also briefly comments on the opportunity to build CBR provenance representations on the existing Open Provenance Model [5]. We intend this position paper as a step towards illuminating the nature of provenance as a framework for broadening the provenance-based capabilities of CBR systems by enabling *cross-task* sharing and exploitation of provenance.

2 Perspective on Provenance and CBR

Explicit consideration of a type of "external provenance"—the provenance of CBR cases received from external sources—dates at least to the HOMER project [6]. HOMER could capture cases from help desk operators as well as confirmed cases, verified by a designated case author, and noted this distinction between the sources of its cases.

The use of "internal provenance"—provenance during internal processing—has deep roots, as well, but has generally been captured implicitly in system processes. For example, early systems which explain in order to ascribe certain types of failures to indexing and learn new indices implicitly consider the provenance resulting in a case retrieval (e.g., [7]). Derivational analogy systems explicitly capture provenance as they record the derivation traces of solutions for replay [8]; replay can also be used to facilitate the management of cases, for example, by replaying the processing used for case adaptation [9, 10]. Metareasoning systems which diagnose problems by reference to a self-model [11–13] can be seen as using the self-model to infer provenance after the fact. More recently, internal provenance has been used to guide case-base maintenance and refinement of case adaptation knowledge [2, 9]. In each of these examples, however, the management of provenance and provenance representations are developed for the requirements of a single task.

This task-specific approach contrasts with the more general approach to provenance taken in e-Science. There a central focus is on developing large-scale provenance capture frameworks which can be made broadly accessible (e.g., [14]). A general-purpose provenance framework could offer practical benefits for CBR, by facilitating the capture and use of provenance. In addition, the availability of such a framework could aid the development of domain-independent tools for manipulating such information (e.g., by providing general explanation capabilities). To enable the right type of information to be captured and to specify its representation requires understanding how it may be used in CBR.

3 Uses of Provenance in CBR

External provenance can play a crucial role in connecting a CBR system to the outside world, by taking into account factors such as trust, uncertainty, and reputation of information sources. Whether information comes from sensors, individual users, social processes, Web knowledge repositories other sources, knowledge of the source is important for reasoning about properties of the information. Likewise, any knowledge transformation process applied to the raw sources (e.g., inference chaining) may affect how the CBR system should use that information.

Internal provenance tracks the lineage of cases—including the processes by which those cases were built—developed over the life-span of an active CBR application. Because aspects such as confidence are propagated as inputs to internal processing, internal provenance helps to connect external provenance from user inputs and parent cases to newer cases (and, in fact, for maximal usefulness must track such information even if the parent cases themselves are deleted). Likewise, it enables external feedback to be connected to relevant cases from a reasoning episode.

Internal provenance can play at least seven major roles in CBR, supporting the tasks of case capture and replay, explanation, assessment, metareasoning, diagnosis and repair, and the interoperability of different learning processes.

1. **Event capture:** Provenance traces capturing a sequence of steps during problem-solving naturally capture a problem-solving case, which can be reused and adapted by transformational analogy [15, 16].
2. **Derivational capture:** If the provenance trace includes representations of the decision-making processes underlying the steps, it can be replayed by derivational analogy.
3. **Explanation:** The event information from (1) can provide a skeleton of steps around which to build causal explanations; the derivational information of (2) provides the basis for motivational explanations and derivational replay.
4. **Result Assessment:** Knowledge of internal provenance can be used to assess confidence in conclusions or lack of trust in that confidence, for example, for the results of case adaptations [9]. Such results can in turn drive maintenance processes [2].

5. **Metareasoning, Diagnosis, and Repair:** Provenance can provide fodder for metareasoning processes to diagnose and repair the functioning of a CBR system. In existing metareasoning systems, the failure sources are sometimes inferred in light of an abstract device model for the CBR system [12, 13]; the routine capture of reasoning provenance would extend the range of usable learning methods.
6. **Interoperability:** Routine provenance capture, coupled with a uniform scheme for representing provenance, could transform individual components of CBR systems from "black boxes" into "glass boxes" (cf. [17]). The routine accessibility of provenance information across components can facilitate the integration of components. For example, a simple CBR retention component might store all new cases, while a more sophisticated one might store only those which would require substantial adaptation effort to generate. In a system which automatically captured provenance information, the amount of derivation effort could be derived from the provenance trace already available to all components. Consequently, the simple retention component could be replaced by the more sophisticated one without requiring any changes to the adaptation component. Thus the additional information channel provided by provenance traces can facilitate the interoperability of components and the substitution of alternative versions of components, potentially facilitating development and testing of CBR systems.
7. **Feedback propagation:** Feedback about any parts of the reasoning process must be correctly ascribed to the responsible parts of processing, through a credit/blame assignment process which depends on knowledge of the derivations of the results receiving comment.

4 Interactive and Multi-Agent Processes

The distinction between internal and external provenance blurs when processes are interactive. In such situations, external inputs may be an integral part of the overall process. With multi-agent processes, or interactive processes with rich sources of input, it may be possible—and worthwhile—to treat the external information source at a finer grained level, considering its own internal provenance. For example, fine-grained provenance for a user's actions could include any relevant and available knowledge about the user's mental states and goals.

5 Specifying a Framework for Provenance in CBR

At first glance, the generation of a general CBR provenance framework is a daunting task. However, recent research in provenance has led to the creation of The Open Provenance Model (OPM)[5], a general purpose framework for representing and communicating provenance. Although this framework is mostly the product of studies of the needs of e-Science workflow users, its scope is intended to be broad, and we believe that it provides a promising foundation for a provenance framework for CBR.

To illustrate OPM, Figure 1 shows an adaptation of a diagram of a provenance sample included in the OPM specification.[1] It represents a portion of the provenance trace for an image generated by a workflow used in the first provenance challenge, a competition testing the capabilities of systems to record and exploit provenance information [18]. Edges in this graph represent relationships between provenance entities, which are represented by various shapes. Ovals in this graph represent "artifacts," specifically data, in this case. The rectangle represents a "process" which both consumes and produces data. Finally, the pointed oval represents an "agent," in this case a user who executed the workflow. Additionally, the process has been annotated with an execution time.

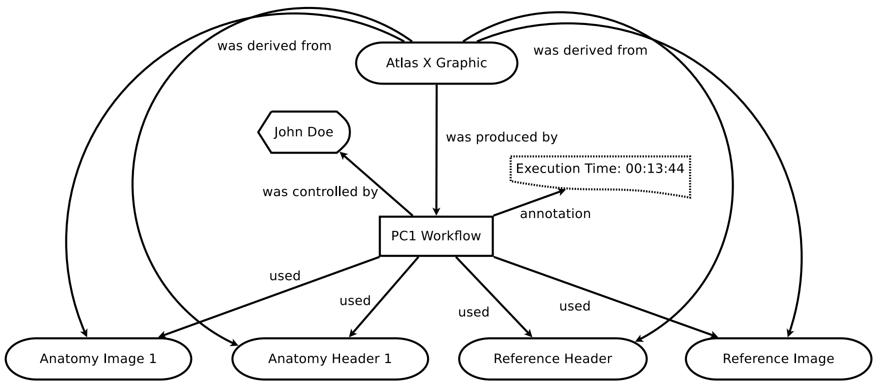


Fig. 1. Example of provenance from the OPM specification, based on [18, p. 8]

Our vision for representing internal provenance of CBR systems involves a structure that, abstractly, matches up well with OPM. Figure 2 illustrates the application of a similar representation scheme to capture parts of the provenance record for an interactive case adaptation episode. As in Figure 1, the representation notes a user who initiated the process. Cases are artifacts and are produced and consumed by adaptation processes. The adaptation process in this instance is annotated with the specific adaptation rule used to produce the new case. We note that this is only an illustrative fragment; we do not claim that it is a complete representation for the needed information. In fact, more elaborated representations could be considered not only for for characterizing the adaptation process, but for all entities in the graph, including relationships.

OPM is abstract, to facilitate interoperability, and does not attempt to specify a formal syntax for expressing provenance. Consequently, leveraging it into a useful CBR provenance framework requires further specification, especially to capture internal provenance. For example, a CBR specific framework must extend abstract entities such as processes and artifacts to more specific ones

[1] This sample can be found on the OPM website: http://openprovenance.org/

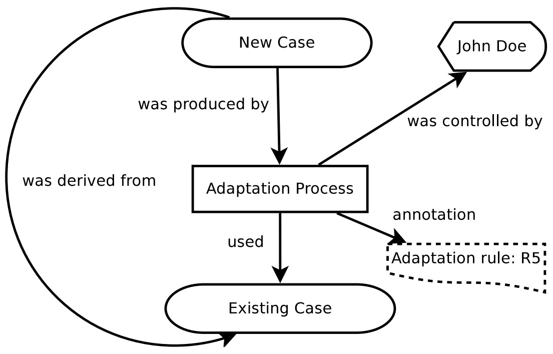


Fig. 2. Example of internal provenance in CBR

such as adaptation processes and cases in order to provide a useful standard. However, this can be built on a number of existing foundational pieces, such as previous work on analytic frameworks and knowledge modeling of CBR [19, 20] and models of the CBR process [12, 13, 21]. Likewise, applying this specification of OPM within a CBR application requires an agreed-upon syntax, e.g., an XML schema for CBR provenance. Once such a schema is available, provenance collection tools designed for other domains could potentially become usable in the CBR context or dedicated CBR tools could be built with that schema as a foundation.

6 Conclusions

Tracking external and internal provenance can aid a wide range of processing and learning tasks in CBR systems. Fully reaping the benefits requires making provenance capture a routine part of CBR and capturing that information in a general form which can be used across different CBR systems and their components. This in turn requires lowering the barriers to entry for provenance-aware CBR systems, by developing a framework for provenance representation. The combination of analysis of types of knowledge and processes in CBR, building on prior studies, and general frameworks such as the Open Provenance Model, provide a path towards this goal.

References

1. Simmhan, Y., Plale, B., Gannon, D.: A survey of data provenance in e-Science. SIGMOD Record **34**(3) (2005) 31–36
2. Leake, D., Whitehead, M.: Case provenance: The value of remembering case sources. In: Case-Based Reasoning Research and Development: Proceedings of the Seventh International Conference on Case-Based Reasoning, ICCBR-07, Berlin, Springer-Verlag (2007) 194–208

3. Leake, D., Dial, S.: Using case provenance to propagate feedback to cases and adaptations. In: Proceedings of the Nineth European Conference on Case-Based Reasoning, Springer (2008) 255–268
4. Briggs, P., Smyth, B.: Provenance, trust, and sharing in peer-to-peer case-based web search. In: Proceedings of the Nineth European Conference on Case-Based Reasoning, Springer (2008) 89–103
5. Moreau, L.: The open provenance model. Technical report, University of Southampton (2008)
6. Göker, M., Roth-Berghofer, T.: Development and utilization of a case-based help-desk support system in a corporate environment. In Althoff, K.D., Bergmann, R., Branting, L.K., eds.: Proceedings of the Third International Conference on Case-Based Reasoning, Springer Verlag (1999) 132–146
7. Hammond, K.: Case-Based Planning: Viewing Planning as a Memory Task. Academic Press, San Diego (1989)
8. Veloso, M., Carbonell, J.: Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. Machine Learning **10**(3) (1993) 249–278
9. Leake, D., Kinley, A., Wilson, D.: Learning to improve case adaptation by introspective reasoning and CBR. In: Proceedings of the First International Conference on Case-Based Reasoning, Berlin, Springer Verlag (1995) 229–240
10. Leake, D., Powell, J.: Mining large-scale knowledge sources for case adaptation knowledge. In Weber, R., Richter, M., eds.: Proceedings of the Seventh International Conference on Case-Based Reasoning, Berlin, Springer Verlag (2007) 209–223
11. Cox, M., Raja, A.: Metareasoning: A manifesto. Technical Memo 2028, BBN (2008)
12. Arcos, J.L., Mulayim, O., Leake, D.: Using introspective reasoning to improve CBR system performance. In: Proceedings of the AAAI 2008 Workshop on Metareasoning: Thinking About Thinking. (2008) 21–28
13. Fox, S., Leake, D.: Modeling case-based planning for repairing reasoning failures. In: Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms, Menlo Park, CA, AAAI Press (March 1995) 31–38
14. Simmhan, Y.L., Plale, B., Gannon, D.: Karma2: Provenance management for data driven workflows. International Journal of Web Services Research, Idea Group Publishing **5** (2008) 1–22
15. Leake, D., Kendall-Morwick, J.: Towards case-based support for e-science workflow generation by mining provenance information. In: Proceedings of the Nineth European Conference on Case-Based Reasoning, Springer (2008) 269–283
16. Leake, D., Kendall-Morwick, J.: Four heads are better than one: Combining suggestions for case adaptation. In: Proceedings of the Eighth International Conference on Case-Based Reasoning, Springer (2009) 165–179
17. Carbonell, J., Etzioni, O., Gil, Y., Joseph, R., Knoblock, C., Minton, S., Veloso, M.: Planning and learning in PRODIGY: Overview of an integrated architecture. In Ram, A., Leake, D., eds.: Goal-Driven Learning. MIT Press/Bradford Books, Cambridge, MA (1995) 297–305
18. Moreau, L., Ludäscher, B., Altintas, I., Barga, R.S., Bowers, S., Callahan, S., Chin, Jr., G., Clifford, B., Cohen, S., Cohen-Boulakia, S., Davidson, S., Deelman, E., Digiampietri, L., Foster, I., Freire, J., Frew, J., Futrelle, J., Gibson, T., Gil, Y., Goble, C., Golbeck, J., Groth, P., Holland, D.A., Jiang, S., Kim, J., Koop, D., Krenek, A., McPhillips, T., Mehta, G., Miles, S., Metzger, D., Munroe, S., Myers, J., Plale, B., Podhorszki, N., Ratnakar, V., Santos, E., Scheidegger, C., Schuchardt,

K., Seltzer, M., Simmhan, Y.L., Silva, C., Slaughter, P., Stephan, E., Stevens, R., Turi, D., Vo, H., Wilde, M., Zhao, J., Zhao, Y.: Special issue: The first provenance challenge. Concurr. Comput. : Pract. Exper. **20**(5) (2008) 409–418

19. Althoff, K.D., Aamodt, A.: Relating case-based problem solving and learning methods to task and domain characteristics: Towards an analytic framework. Artificial Intelligence Communications **9** (1996) 11–6
20. Aamodt, A.: Modeling the knowledge contents of CBR systems. In: Proceedings of the ICCBR 2001 Workshop on Case-Based Reasoning Authoring Support Tools. (2001) 32–37 Naval Research Lab Technical Note AIC-01-003.
21. Birnbaum, L., Collins, G., Brand, M., Freed, M., Krulwich, B., Pryor, L.: A model-based approach to the construction of adaptive case-based planning systems. In Bareiss, R., ed.: Proceedings of the DARPA Case-Based Reasoning Workshop, San Mateo, Morgan Kaufmann (1991) 215–224

Towards a Lazier Approach to Problem Solving in Case-Based Reasoning

David McSherry

School of Computing and Information Engineering
University of Ulster, Coleraine BT52 1SA, Northern Ireland
dmg.mcsherry@ulster.ac.uk

Abstract. Recent research has highlighted the problems caused by delays in the feedback available to a case-based reasoning (CBR) system, and how provenance–guided propagation of feedback to related and/or similar cases can help to alleviate these problems. We argue in this paper that the limited ability of traditional CBR systems to learn from feedback can be attributed to eager commitment to adaptation paths that establish case provenance but which may prove to be sub-optimal in future problem solving. We also present initial work on a "lazier" approach to problem solving in CBR based on demand-driven discovery of adaptation paths that change continuously as the case base grows and in response to feedback received during the system's life cycle.

Keywords: case-based reasoning, adaptation, provenance, feedback.

1 Introduction

In seminal work on the role of provenance in case-based reasoning (CBR), Leake and Whitehead [1] investigate the effects of delays in the feedback available to a CBR system on solution quality. They also demonstrate the benefits of propagating feedback on a given case's solution to related and/or similar cases. In one such algorithm, feedback is propagated to all *descendants* of the reference case (defined as those cases that were generated from the reference case by a series of adaptations). Feedback propagation is guided by provenance information captured by the system as each new case is added to the case base (i.e., the set of cases that contributed – directly or indirectly – to the new case's solution). A related issue noted by Leake and Whitehead [1] is that the solutions provided by a CBR system may depend on the order in which previous problems were presented to and solved by the system.

In this paper, we argue that the limited ability of traditional CBR systems to learn from feedback on cases whose incorrect solutions may have contributed to errors in other cases can be attributed to eager commitment to adaptation paths that establish case provenance but which may prove to be sub-optimal in future problem solving. We also propose that this departure from the lazy approach to problem solving that characterizes CBR [2] is the underlying problem that creates the need for maintenance interventions such as the feedback propagation methods investigated by Leake and his co-workers [1, 3]. It is also a primary cause of the order dependence problem noted by Leake and Whitehead [1].

In Sections 2 and 3, we present initial work on a *lazier* approach to problem solving in CBR that aims to address this issue by avoiding eager commitment to adaptation paths that depend on the order in which problems are presented to the system. Instead, demand-driven discovery of adaptation paths that change continuously during the system's life cycle is combined with a policy of allowing only the solutions of "confirmed" cases (i.e., cases whose solutions are known to be correct) to contribute to problem solving. Our conclusions are presented in Section 4.

2 Lazier CBR

In traditional CBR, a target problem is solved by adapting the solution of the retrieved case that is most similar to the target problem (or simply by reusing the solution from the most similar case without adaptation as in 1-NN) [4-5]. Lazy problem solving is a characteristic of CBR that sets it apart from eager learning algorithms that create abstractions such as decision trees from training data [2]. To simplify our discussion of how CBR might benefit from an even *lazier* approach to problem solving, we do not consider CBR approaches in which two or more retrieved cases may contribute *directly* to the solution of a target problem, for example as in CBR approaches to estimation based on adaptation triples [6].

Adaptable Cases. The set of cases that can be adapted to solve a given problem plays an important role in Lazier CBR. For any problem *P*, we denote by *adaptable-cases*(*P*) the set of cases that can be adapted to solve *P*. Thus for any case *C*, *adaptable-cases*(*problem*(*C*)) is the set of cases that can be adapted to solve *problem*(*C*), the problem described by *C*. In approaches to modeling the competence of a case base, this set of cases is often referred to as a case's *reachability* set [7].

Confirmed/Unconfirmed Cases. We will refer to a case whose solution is known to be correct (e.g., a seed case provided by a domain expert, or a case whose solution is confirmed/corrected by feedback from a reliable source) as a *confirmed* case. All other cases are said to be *unconfirmed* in our approach. In contrast to traditional CBR, only the solutions of confirmed cases can contribute to the solution of a given problem in Lazier CBR. The aim of this policy is to prevent the propagation of possible errors in the solutions of unconfirmed cases. However, there is no loss of coverage in our approach because the coverage relinquished by disallowing the reuse of unconfirmed solutions is reclaimed by recursive adaptation.

Solving a Target Problem in Lazier CBR. The problem-solving process in Lazier CBR is outlined in Fig. 1. A problem is solved using a discovered *adaptation path* from a confirmed case to the target problem. Our demand-driven approach to the discovery of adaptation paths uses breadth-first search [8] to ensure the discovery of an adaptation path of the shortest possible length, if one exists, from a confirmed case to the target problem. However, the initial state in the search process is the target problem and the goal state is any confirmed case. On successful completion of the search process, the discovered *search* path from the target problem to a confirmed case is reversed to provide an *adaptation* path that can be used to solve the target problem.

Algorithm: Problem Solving in Lazier CBR
Input: A target problem *P*
Output: A solution for *P*
Process:

1. Conduct a breadth-first search of the case base in which the initial state is *P*, the goal state is any confirmed case, *children*(*P*) = *adaptable-cases*(*P*), and *children*(*C*) = *adaptable-cases*(*problem*(*C*)) for each case *C*
2. If no search path from *P* to a confirmed case can be found then return failure
3. Let $P \rightarrow C_1 \rightarrow ... \rightarrow C_k$ be the discovered search path from *P* to a confirmed case C_k
4. Reverse the discovered search path $P \rightarrow C_1 \rightarrow ... \rightarrow C_k$ to provide an adaptation path $C_k \rightarrow ... \rightarrow C_1 \rightarrow P$ of the shortest possible length from a confirmed case to *P*
5. Return the solution obtained by using the discovered adaptation path to solve *P*

Fig. 1. Problem solving in Lazier CBR

Retention in Lazier CBR. When a discovered adaptation path has been used to solve a target problem in Lazier CBR, the target problem is stored in the case base, with its solution, as an *unconfirmed* case. The new case's status unchanged until such time as its solution is confirmed (or corrected) by feedback from a reliable source. When this happens, its status changes from unconfirmed to confirmed.

Lazier CBR Example. Fig. 2 illustrates the use of Lazier CBR to solve a target problem (P). Of the 15 cases in the example case base, 3 are seed cases (black) with solutions that are known to be correct, and the rest are unconfirmed cases (white). At this stage of the system's life cycle, no feedback has been received for any case. The cases enclosed by the circle with P at its center are those that can be adapted (in a single step) to solve the target problem. The lines joining cases in the figure show the paths explored in breadth-first search. The confirmed case discovered by breadth-first search is C_8 and the discovered adaptation path is $C_8 \rightarrow C_9 \rightarrow C_{10} \rightarrow$ P. Note that the *adaptation path* reverses the order of the *search path* from the initial state (P) to the goal state (C_8) discovered by breadth-first search. (However, there is no assumption that adaptation is reversible in our approach.) The next stage in the solution of the target problem is to apply the discovered adaptation path. First, the solution for C_8 is adapted to solve the unconfirmed case C_9 (i.e., the unconfirmed solution for C_9 is not reused). The new solution for C_9 is then adapted to solve C_{10}. Finally, the new solution for C_{10} is adapted to solve P. As the target problem has now been solved, it is retained with its solution as an unconfirmed case in the case base. The solutions for the unconfirmed cases C_9 and C_{10} are also revised, but their status (unconfirmed) remains unchanged.

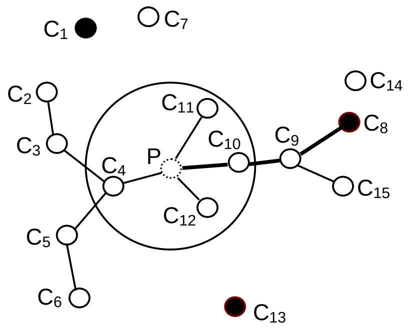


Fig. 2. Solving a target problem (P) in Lazier CBR before any feedback is received. The shortest adaptation path from a confirmed case to the target problem is $C_8 \rightarrow C_9 \rightarrow C_{10} \rightarrow$ P.

Learning From Feedback. Fig. 3 illustrates the use of Lazier CBR to solve the same target problem (P) as in our previous example but in a slightly different context. The case base is the same as in Fig. 2 except that feedback on case C_3 has been received from a reliable source, with the result that its status has changed from *unconfirmed* to *confirmed*. The shortest adaptation path from a confirmed case to the target problem, and therefore the one that will be used to solve the problem in Lazier CBR, is now $C_3 \rightarrow C_4 \rightarrow$ P. Note that the solution provided by Lazier CBR in this example takes account of feedback on a case that would not be considered in a traditional CBR approach to solving the target problem. It is also worth noting that problem-solving efficiency benefits from the feedback on case C_3, not only in terms of search effort but also in terms of adaptation effort.

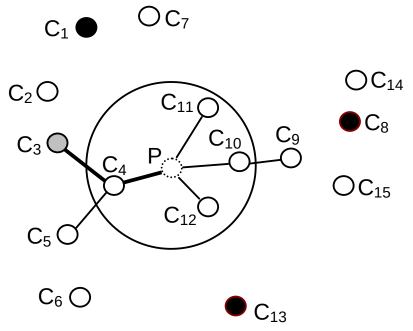


Fig. 3. Solving the same target problem (P) in Lazier CBR after feedback on case C_3 has been received. The shortest adaptation path from a confirmed case to the target problem is now $C_3 \rightarrow C_4 \rightarrow$ P.

Coverage in Lazier CBR. One example of a CBR system in which recursive adaptation is used to increase coverage is CREST [9]. However, in comparison with traditional CBR approaches that do not rely on recursive adaptation, Lazier CBR does not provide any additional coverage. Instead, it uses recursive adaptation to reclaim the coverage relinquished by disallowing the reuse of unconfirmed solutions.

Meta-Data in Lazier CBR. The current status of each case (i.e., confirmed / unconfirmed is used as meta-data to guide the demand-driven discovery of adaptation paths in Lazier CBR. The solutions of unconfirmed cases in Lazier CBR can also be considered as meta-data insofar as they do not contribute to problem solving but can be used for maintenance purposes, for example as discussed in Section 3.

3 Provenance in Lazier CBR

As described in Section 2, Lazier CBR prevents the propagation of possible errors in the solutions of unconfirmed cases by allowing only the solutions of *confirmed* cases to contribute to the solution of a new problem. This means that there is no need for propagation of feedback received by the system to "repair" the case base as proposed by Leake and Whitehead [1], at least not if the goal is to improve the quality of *future* problem solving. There is also no need for case provenance information (i.e., the cases that contributed to each case's solution) to be captured for this purpose. However, one of the issues to be addressed by future work is that a more proactive approach to the use of feedback in Lazier CBR could be beneficial in situations where the effects of a *previous* incorrect or inaccurate solution may be preventable or reversible by providing an improved solution in retrospect. An example in the real estate domain would be a house whose estimated value was too high and has not yet sold for this reason.

While there is no need for feedback propagation based on provenance information to improve future problem solving in Lazier CBR, we expect case provenance to play an important role in enabling the effectiveness of adaptation strategies to be assessed in light of feedback on solution quality. Leake and Dial [3] have already shown how provenance information can be used to guide the maintenance of adaptation rules in traditional CBR approaches, and the fact that Lazier CBR eliminates one source of problem-solving errors (i.e., possible errors in the solutions of unconfirmed cases) may help to simplify the assessment of adaptation rule quality. In this context, another important feature of Lazier CBR that we now discuss is the *provenance shift* that occurs in light of feedback on solution quality.

Provenance Shift. In contrast to traditional CBR, case provenance changes dynamically in Lazier CBR. To illustrate this point, suppose that all the cases in Fig. 3 were already in the case base before feedback on C_3 was received by the system. Also suppose that C_1, C_2, C_3, and C_4 were the first cases to be created in the case base, and that C_4 was created using the adaptation path $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow$ *problem*(C_4) discovered when the problem represented by C_4 was presented to the system. Thus, the cases that originally contributed to the solution of C_4 were C_1, C_2, and C_3 (though the confirmed case C_1 is the only case whose previous solution would be reused in the adaptation process). As the discovery of adaptation paths is demand driven in Lazier CBR, there is no immediate change in the provenance of C_4 (or its solution) when feedback on C_3 is received by the system. In fact, it is only when the target problem P is presented to the system that the feedback on C_3 begins to affect other cases. For example, when the discovered adaptation path $C_3 \rightarrow C_4 \rightarrow P$ is used to solve P, an important side effect is that the solution of C_4 is revised to take account of the shorter adaptation path $C_3 \rightarrow$ *problem*(C_4) from a confirmed case to *problem*(C_4). The provenance of C_4 also changes, as the only case that now contributes to its solution is C_3.

Assessment of Adaptation Strategies. While it is important to consider the provenance shifts that occur in Lazier CBR, there is nothing to prevent the *current* provenance of each case being captured by a Lazier CBR system and used to assess the effectiveness of the adaptation process in light of feedback on solution quality. In this context, an important feature of Lazier CBR is that problem-solving errors can occur only as a result of defects in the system's similarity and/or adaptation knowledge, and not as a result of errors in the solutions of unconfirmed cases. Again with reference to the example case base in Fig. 3, suppose that the system learns from the feedback on C_3 that its previous solution for C_3 was incorrect, and that it also knows that C_3 was created using the adaptation path $C_1 \rightarrow C_2 \rightarrow$ *problem*(C_3). If we also assume that the retrieval process is reliable (cf. [3]), then the error in the previous solution of C_3 can be attributed to one or both of the steps in the adaptation path.

4 Conclusions

The work presented in this paper is motivated by similar concerns to those expressed by Leake and Whitehead [1] in their analysis of the role of case provenance in CBR. In common with provenance-guided approaches to feedback propagation [1, 3], Lazier CBR aims to address the limited ability of CBR systems to learn from feedback on cases whose incorrect solutions may have contributed to errors in the solutions of other cases in the case base. We have proposed an approach to achieving this goal by demand-driven discovery of adaptation paths that change continuously as the case base grows and in response to feedback received during the system's life cycle. Another important feature of Lazier CBR is the use of meta-data about the current status of each case (i.e., confirmed/unconfirmed) to guide adaptation path discovery.

Lazier CBR does not rely on feedback propagation, for example as proposed by Leake and Whitehead [1], to improve the quality of future problem solving. Instead, it aims to prevent the propagation of incorrect solutions that creates the need for such maintenance interventions by allowing only previous case solutions that are known to be correct to contribute to the solution of a new problem. Also in contrast to traditional CBR, the solutions provided by Lazier CBR are independent of the order in which problems are presented to the system. As such, Lazier CBR addresses another important issue identified by Leake and Whitehead [1].

We briefly discussed the potential role of case provenance in enabling adaptation strategies to be assessed in light of feedback on solution quality (cf. [3]) in Lazier CBR. In this context, the fact that Lazier CBR eliminates problem-solving errors that result from errors in the solutions of unconfirmed cases may help to simplify the assessment of adaptation rule quality.

An interesting hypothesis to be investigated in future work is that Lazier CBR may tend to provide more accurate solutions than traditional CBR approaches in situations where solution quality tends to deteriorate as the lengths of adaptation paths increase. However, given the potential costs of demand-driven discovery of adaptation paths based on breadth-first search in large case bases, it will also be important to investigate the trade-offs between solution quality and computational efficiency in the approach.

References

1. Leake, D., Whitehead, M.: Case Provenance: The Value of Remembering Case Sources. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 194–208. Springer, Heidelberg (2007)
2. Aha, D.W.: The Omnipresence of Case-Based Reasoning in Science and Application. Knowledge-Based Systems 11, 261–273 (1998)
3. Leake, D., Dial, S.A.: Using Case Provenance to Propagate Feedback to Cases and Adaptations. In: Althoff, K.-D. *et al*. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 255–268. Springer, Heidelberg (2008)
4. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications 7, 39-59 (1994)
5. López de Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, Reuse, Revision, and Retention in Case-Based Reasoning. Knowledge Engineering Review 20, 215-240 (2005)
6. McSherry, D.: Automating Case Selection in the Construction of a Case Library. Knowedge Based Systems 13, 133-140 (2000)
7. Smyth, B., McKenna, E.: Competence Models and the Maintenance Problem. Computational Intelligence 17, 235-249 (2001)
8. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River (2010).
9. McSherry, D: Demand-Driven Discovery of Adaptation Knowledge. In: 16th International Joint Conference on Artificial Intelligence, pp. 222–227. Morgan Kaufmann, San Francisco (1999)

From Provenance-awareness to Explanation-awareness—When Linked Data Is Used for Case Acquisition from Texts

Thomas Roth-Berghofer[1,2] and Benjamin Adrian[1,2]

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

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

{firstname.lastname}@dfki.de

Abstract. Explanation-awareness in Case-Based Reasoning system development aims at making such systems smarter in interactions with their users. When using Linked Data for case acquisition from text one aspect of being smarter is the provision of evidence for the trustworthiness of the acquired cases. The Trustworthiness of such cases relies not only on the provenance of the text but also on the provenance of the used ontological knowledge. Users can only assess the quality of the case-based reasoner's results, i.e., the cases, if the system provides provenance information and if such a system can justify its results. In fact, explanation capabilities very much rely on provenance information.

Key words: Textual Case-Based Reasoning, Ontology-based Information Extraction, Linked Data, Web of Data, Explanation, Provenance

Explanation-awareness in Case-Based Reasoning system development aims at making such systems much smarter in interactions with their users. It looks at new ways to guide software designers and engineers to a purposeful explanation-aware software system by making their designers and engineers explanation-aware [14]. Systems that intend to exhibit explanation-awareness must be more than simple reactive systems. When the word "awareness" is used in conjunction with the word "explanation" it implies some consciousness about explanation. When a system exhibits explanation awareness, it is capable of reasoning about explanations. When we use the word aware make a strong statement about the capabilities of the entity described. As being knowledgeable is central to being aware, some kind of reasoning capabilities, or intelligence, is implied. Thus, whether we aim at creating computer systems that are explanation-aware—or provenance-aware for that matter—such systems must regard explanations and provenance information from the knowledge level [11].

It has been established that there is a certain explanation quality in the use of cases as explanations [5]. The general argument being that a similar past case will contain additional information, which (together with the target information asked for) provides a context around the target information that should be viewed as a situation-oriented explanation. However, there are also the other knowledge containers [12] related to CBR, i.e., vocabulary, similarity measures, and adaptation knowledge, that can be tapped into for providing additional explanation capabilities [17]. As soon as the vocabulary container is not a closed container anymore, but linked to the Web of Data, provenance issues such as source of information, ownership and authorship arise and need to be dealt with.

Taking the term "provenance-awareness" one of the questions coming to mind probably is: "Who needs to be aware?" Surely designers and users of such software systems, e.g., CBR system developers and knowledge engineers, and the users of the respective CBR system. CBR system developers need to provide means for integrating provenance information and knowledge engineers need to model provenance information or make it available to the CBR system user.

The Linked Open Data project is a Semantic Web effort using the Web (i.e., Web standards such as HTTP, URIs and RDF) to connect related data that was not previously linked together [3]. One of the most visible examples for Linked Open Data is DBpedia, "a community effort to extract structured information from Wikipedia and to make this information available on the Web". Governments and organisations have begun to provide Linked Data in vast amounts.

In [15], Roth-Berghofer et al. describe that case models published as Linked Open Data can be interconnected to information from various data sources. Thus, they can be enriched with additional knowledge from other ontologies. Combining SCOOBIE, an ontology-based information extraction system [1], and *myCBR*, a tool for rapid prototyping of similarity-based retrieval applications [18], allows leveraging available knowledge of the Web of Data for acquiring cases from texts. As standard ontology representation formalism for expressing the case model the Resource Description Framework language (RDF) [19] was chosen, where meaning is expressed by facts encoded in sets of triples [2].

Representing a case model in RDF results in URIs for each attribute and its possible values. If an HTTP request of these URIs delivers RDF triples with additional information about the resource identified with this URI and if several URIs in multiple web domains are interlinked with special types of predicates (e.g., `owl:sameAs`) whether they represent a similar resource, we talk about Linked (Open) Data .

Generating cases from texts using Linked Data from the Web of Data provides new challenges. Textual case-based reasoning (CBR) systems that combine Linked Data from different sources are faced with quality and trust issues. These issues have been recognised and are already addressed by the Semantic Web community (e.g., [10, 9] and by the Web of Data community (e.g., [7]). Both communities provide terminologies for explicitly describing provenance information such as source name, date and author(s) of original information and any updates. Typically, based on the provenance information the quality is assessed. But the vocabularies have lots of weaknesses still and tools are missing.

As soon as CBR system developers and knowledge engineers rely on Linked Data they need to have a vocabulary and tools to express provenance information. They as well as need to provide access to that information for the final recipient of provenance information, the CBR system user who should not only be aware of that kind of information but should demand it.

myCBR provides some general means for extending the vocabulary knowledge container and for accessing it in order to support CBR system users with knowledge about the used terminology [16], namely *concept explanations*. A concept explanation links concepts unknown to a user to already known concepts. It is a comprehensive description in the form of a definition, an example, or a reference to further characterisations, for which any kind of medium such as text, images, audio, or video can be used.

CBR system users can only assess the quality of the case-based reasoner's results if the system provides access to provenance information and if such a system can justify its results. The users might have questions about the history of the case model, which part of the case model was generated by the knowledge engineer and which part comes from other sources. They might ask about why, where, and how provenance information came to pass.

In database research, Buneman et al. [4] distinguished exactly this: why- and where provenance. Why-provenance provides information on the origins that were involved in computing a result, and where-provenance describes the location of the source. Green et al. [6], finally, added how-provenance, which describes how the origins were involved in the computation. These questions are directly feeding into why- and action explanations [13, 17].

Making CBR system developers and knowledge engineers provenance-aware is one step in the direction of making a CBR system provenance-aware. As soon as such CBR systems have provenance information they can act upon it, giving the CBR system user a feeling of how trustworthy a case is or if the data on which the case relies has lost its reliability, as Linked Data today may be different from Linked Data tomorrow [8]. Last but not least, provenance information is also be a good opportunity for triggering maintenance operations.

Acknowledgments

This work was funded by the BMBF project Perspecting (Grant 01IW08002).

References

1. Adrian, B., Hees, J., van Elst, L., Dengel, A.: iDocument: Using ontologies for extracting and annotating information from unstructured text. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009: Advances in Artificial Intelligence. Künstliche Intelligenz (KI-2009), Sept. 15–18, Paderborn, Germany. Lecture Notes in Artificial Intelligence, LNAI, vol. 5803, pp. 249–256. Springer, Heidelberg (2009)
2. Bergmann, R., Schaaf, M.: Structural Case-Based Reasoning and Ontology-Based Knowledge Management: A Perfect Match? Journal of Universal Computer Science

9(7), 608–626 (2003), http://www.jucs.org/jucs_9_7/structural_case_based_reasoning [Last access: 2010-02-26]
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009), http://dblp.uni-trier.de/db/journals/ijswis/ijswis5.html#BizerHB09 [Last access: 2010-02-26]
4. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: ICDT '01: Proceedings of the 8th International Conference on Database Theory. pp. 316–330. Springer, London, UK (2001)
5. Cunningham, P., Doyle, D., Loughrey, J.: An evaluation of the usefulness of case-based explanation. Tech. Rep. TCD-CS-2003-04., Trinity College Dublin (2003)
6. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 31–40. ACM, New York, NY, USA (2007)
7. Hartig, O.: Provenance Information in the Web of Data. In: Proceedings of the Linked Data on the Web LDOW Workshop at WWW (2009)
8. Hellmann, S., Stadler, C., Lehmann, J., Auer, S.: Dbpedia live extraction. In: Debbabi, M., Caragea, D., Yang, X.C. (eds.) Proceedings of the 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE). Lecture Notes in Computer Science, vol. 5871, pp. 1209—1223. Springer (2009)
9. McGuinness, D.L., Ding, L., da Silva, P.P., Chang, C.: Pml 2: A modular explanation interlingua. In: Roth-Berghofer, T.R., Schulz, S., Leake, D.B. (eds.) ExaCt. pp. 49–55. AAAI Press (July 2007), technical Report WS-07-06
10. McGuinness, D.L., Pinheiro da Silva, P.: Infrastructure for web explanations. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) The Semantic Web — ISWC 2003. pp. 113–129 (2003)
11. Newell, A.: The knowledge level. AI Magazine 2(2), 1–20 (1981)
12. Richter, M.M.: The knowledge contained in similarity measures. Invited Talk at the First International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra, Portugal (1995)
13. Roth-Berghofer, T. (ed.): Künstliche Intelligenz: Anwendungen im Semantischen Web (Integriertes Seminar). Technische Universität Kaiserslautern (Juli 2004)
14. Roth-Berghofer, T.: ExaCt manifesto: Explanation-aware computing (July 2009), http://on-explanation.net/content/ExaCt_Manifesto.html
15. Roth-Berghofer, T., Adrian, B., Dengel, A.: Case acquisition from text: Ontology-based information extraction with SCOOBIE for myCBR. In: Montani, S., Bichindaritz, I. (eds.) Advances in Case-Based Reasoning. Springer, Heidelberg (2010)
16. Roth-Berghofer, T.R., Bahls, D.: Explanation capabilities of the open source case-based reasoning tool mycbr. In: Petridis, M., Wiratunga, N. (eds.) Proceedings of the thirteenth UK workshop on Case-Based Reasoning UKCBR 2008. pp. 23–34. University of Greenwich, London, UK (2008)
17. Roth-Berghofer, T.R., Cassens, J.: Mapping goals and kinds of explanations to the knowledge containers of case-based reasoning systems. In: Muñoz-Avila, H., Ricci, F. (eds.) Case-Based Reasoning Research and Development, ICCBR 2005, Chicago, IL, USA, August 2005, Proceedings. pp. 451–464. No. 3620 in Lecture Notes in Artificial Intelligence LNAI, Springer, Heidelberg (2005)
18. Stahl, A., Roth-Berghofer, T.R.: Rapid prototyping of CBR applications with the open source tool myCBR. In: Bergmann, R., Althoff, K.D. (eds.) Advances in Case-Based Reasoning. Springer (2008)
19. W3C: Rdf primer (February 2004), http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ [Last access: 2010-02-26]

WebCBR
Reasoning from Experiences on the Web

Workshop at the
Eighteenth International Conference on
Case-Based Reasoning

Alessandria, Italy
July, 2010

ICCBR 2010

Derek Bridge, Sarah Jane Delany, Enric Plaza,
Barry Smyth and Nirmalie Wiratunga (Eds.)

Co-Chairs

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

Sarah Jane Delany
Dublin Institute of Technology, Ireland
sarahjane.delany@dit.ie

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

Barry Smyth
University College Dublin, Ireland
barry.smyth@ucd.ie

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

Program Committee

Additional Reviewers

Preface

We are pleased to present the proceedings of the Second Workshop in Reasoning from Experiences on the Web (WebCBR), which was held as part of the Eighteenth International Conference on Case-Based Reasoning in Alessandria, Italy, July 2010.

This workshop, which follows a very successful first WebCBR workshop that was held at ICCBR in Seattle in July 2009, promotes CBR as a means to support Web users in at least two ways: firstly by enabling better capture and representation of explicit yet unstructured experiential Web content, and secondly by harnessing Web usage data to improve searching and browsing.

Advances in Web technology have led to vast amounts of user-generated Web content in the form of blogs, emails, reviews and comments. Increasingly, people search and browse other people's experiences on travel, medicine, retail, entertainment, etc. While these records of experience 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 also means significant increases in Web usage and data about this usage. Many users will have similar searching and browsing needs and should ideally benefit from the search and browsing experiences of others. Better capture of usage data and its greater reuse to enhance subsequent searching and browsing is another opportunity for CBR research.

The workshop provided a forum for discussion of these new research directions, provoked by the presentation of six papers, which are collected in these proceedings. The papers include Web 2.0 applications that reuse experiential Web content and usage data to support the authoring of new product reviews, to proactively make recommendations in a conversational setting, and to reuse searches for knowledge that can be used to adapt travel itineraries, for example. Other papers address underlying technological issues such as the similarity and retrieval of textual content, the mining of case vocabulary, and the interoperation of CBR with the emerging Web of Data.

We wish to thank all who contributed to the success of this workshop, especially the authors, the Program Committee, the additional reviewers, the panelists, and Cindy Marling (the ICCBR Workshop Chair).

Derek Bridge July 2010
Sarah Jane Delany
Enric Plaza
Barry Smyth
Nirmalie Wiratunga

Deriving Case Base Vocabulary from Web Community Data

Kerstin Bach, Christian Severin Sauer, and Klaus-Dieter Althoff

University of Hildesheim
Institute of Computer Science - Intelligent Information Systems Lab
Marienburger Platz 22, 31141 Hildesheim, Germany
{bach,althoff}@iis.uni-hildesheim.de
christiansauer@gmail.com
http://www.iis.uni-hildesheim.de

Abstract. This paper presents and approach for knowledge extraction for Case-Based Reasoning systems. The recent development of the WWW, especially the Web 2.0, shows that many successful applications are web based. Moreover, the Web 2.0 offers many experiences and our approach uses those experiences to fill the knowledge containers. We are especially focusing on vocabulary knowledge and are using forum posts to create domain-dependent taxonomies that can be directly used in Case-Based Reasoning systems. This paper introduces the applied knowledge extraction process based on the KDD process and explains its application on a web forum for travelers.

Key words: case-based reasoning, knowledge container, vocabulary extraction

1 Introduction

Since the very beginning and the expansion of the World Wide Web (WWW) the information available increased rapidly. This information overhead lead to new technologies and developments that ended up in the Web 2.0. Moreover, these technologies enhanced the information palette for multimedia data and boosted the amount of user-generated content. Further, the Web 2.0 changed the documents provided in the WWW from rather statical documents on a website towards smaller pieces of user-driven information like forum posts, tweets[1] or ratings for products [19]. The user generated content often contains pieces of user experiences expressed in an explicit or implicit way[15]. However, using these information is still challenging because most of it is still unsystematic and therewith hardly to be efficiently retrieved and reused [3]. In the last years many kinds of web communities that had some kind of structure to the experience available were mostly develop around a certain type of topic and assemble WWW users that share an interest. Nevertheless the amount of web communities that develop based on common interests is still quite large, however, those

[1] So called 140 character messages on twitter.com

communities can be classified into a manageable amount of community types. Those community types can be characterized by their technical realization and the way how knowledge and experiences are presented and shared. Both aspects can be used to differentiate between communities and provide a first type of structure [12]. The information provided in web communities can be described as a combination of experiences. So the information web have to deal with contains common information and information artifacts like web sites or documents that are interwoven with information given during interactions or conversations in web communities.

1.1 The Role of Knowledge Extraction within SEASALT

The work presented in this paper is a part of the further development of the SEASALT (Sharing Experience using an Agent-based System Architecture LayouT) architecture [16], which is especially suited for the acquisition, handling and provision of experiential knowledge as it is provided by communities of practice and represented within Web 2.0 platforms. It is based on the Collaborative Multi-Expert-Systems (CoMES) approach presented by Althoff et. al. [1] which is a continuation of combining established techniques and the application of the product line concept (known from software engineering) creating knowledge lines [10]. The knowledge extraction is a part of the knowledge provision component in SEASALT that is based on software agents representing heterogeneous domains. Each software agent can be realized as an independent Case-Based Reasoning system that is maintained by a *Case Factory* [2]. Each agent retrieves new knowledge from individual knowledge sources like wikis, blogs or web forums in which users provide different kinds of information.

1.2 Discussion of Related Work

According to Richter [17], the knowledge of Case-Based Reasoning systems can be provided in all four knowledge containers that include vocabulary, similarity measures, transformational knowledge and cases. We focus on the problem (task) on how the knowledge containers can be filled using the experiences provided in web communities. For this purpose we have to extract the knowledge before it can be stored.

Further on, we deal with data sources that are mostly free text what usually requires a symbolic representation of keywords. That is the reason why we currently focus on the extraction of taxonomies that can be used for both, enhancing the vocabulary and assigning the similarity. Since we use user experiences to develop our taxonomies, we are able to create taxonomies tailored to represent experiences provided in the according community. However, besides extracting cases from experiences there are many more pieces of knowledge that can be extracted and used by Case-Based Reasoning systems. The work of Smyth et. al. [19], Milne et.al [14], Plaza and Baccigalupo [15] as well as Ihle et. al [9] show the potential of web communities for Case-Based Reasoning systems.

For the generation and the improvement of a Case-based Reasoning system's vocabulary or its domain model different knowledge extraction approaches can be applied. For example, light ontologies can be used for domain modeling and directly extracted from the web community. This can be carried out by creating taxonomies after analyzing social tags, like the extraction of folksonomies [13]. The benefit of using light ontologies and folksonomies is quite clear: the more people are working and contributing in a web community, the more detailed the result gets with almost no increase of the cost.

Another major point while extracting vocabulary knowledge is the fact that the information has to be put into a certain context to be organized in a knowledge model. Mika [13], Chakrabarti [4] and Liu [11] for example present approaches in which they take the role of the user into account. Even if only little information is available: certain properties of the information, the explicit or implicit scope of the web community, or the role of a user can provide information on the context.

2 Knowledge Extraction

Knowledge acquisition is still the bottleneck within the development of Case-Based Reasoning systems. Our approach aims at automatization and for this purpose we propose a schema that extracts knowledge from web community sources and provides it for CBR systems. In the remaining parts of this paper we introduce our approach and apply it in our current real-life application docQuery. docQuery is a SEASALT instantiation that provides travel medicine information based on data extracted from a web community [16].

Currently we work on the extraction of taxonomies that are used to build the vocabulary, to assign similarity values between two symbols as well as provide basic adaptation knowledge. The vocabulary is derived by transferring each term of a taxonomy into the vocabulary repository. Taxonomically ordered symbolic types provide similarity values between two nodes based on their distance in the taxonomy and sibling nodes in taxonomies represent basic adaption possibilities.

Besides the creation of taxonomies, also the improvement of those taxonomies, the adjustments of term weights for the similarity assignment as well as the possibility to resolve disambiguation also has to be focused on. The aforementioned approaches take the web usage or the links pointing at a certain resource of the extracted terms into consideration. Techniques from Information Retrieval and the analysis of social networks are necessary to put the extracted information in the domain context. The goal of our approach is structuring the available information and organizing them within the knowledge model, so the experiences can used by the Case-Based Reasoning system.

2.1 Process

Our approach of extracting knowledge from web communities and applying this knowledge within a Case-Based Reasoning system can be compared to the knowledge discovery in databases (KDD) presented by Fayyard [7].

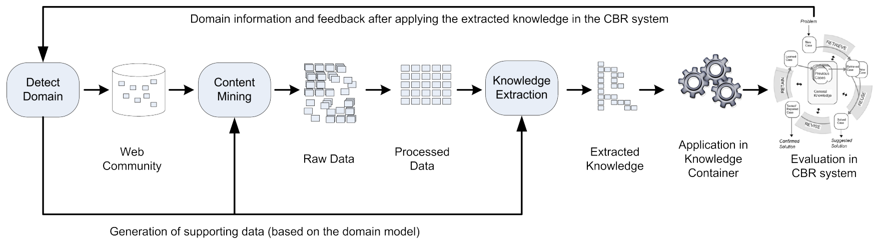

Domain information and feedback after applying the extracted knowledge in the CBR system
Detect Domain
Web Community
Content Mining
Raw Data
Processed Data
Knowledge Extraction
Extracted Knowledge
Application in Knowledge Container
Evaluation in CBR system
Generation of supporting data (based on the domain model)

Fig. 1. Knowledge Extraction Process Model for CBR Systems

The knowledge extraction process presented here does not only apply for CBR systems, of course it can be employed in all kinds of knowledge based systems. In comparison to KDD, we do not create views on data, instead we are extracting knowledge, reorganize it and provide it in knowledge containers. Therefore we do not concentrate only on one particular data type by applying and refining information extraction techniques on it. Instead, our source data is different, because we get our source data delivered from web mining technologies, most of them are web (content) crawlers. The extraction of knowledge and further on its provision in the Case-Based Reasoning system's knowledge containers can be described as an active transformation of knowledge, because we create and/or extend knowledge models, i.e. taxonomies. Figure 1 gives an overview of the Knowledge Extraction processes based on the KDD process. The following nine steps describe successively what kinds of tasks have to be executed in each step to create knowledge from community data:

1. **Domain Detection:** This first step describes the identification of the domain properties and results in the assignment of what kind of information can be extracted and in which knowledge container it should be integrated. Further auxiliary data like word lists or rules that are required for the knowledge extraction are defined and, if possible, created.
2. **Web Community Selection:** In this step an appropriate web community has to be identified. This covers not only technical issues, it also requires the permission to use the knowledge provided within a web community. The technical requirements include the availability of a certain amount of accessible data. This usually starts with structure mining techniques before continuing with the next step.
3. **Content Mining:** This describes the connection of the knowledge extraction process to the source data. Usually web crawlers access the content and transform it so the data can be provided. Also intelligent web forums, as they are described by Feng [8] and further developed within the SEASALT architecture [16] where intelligent agents pass on relevant data, can serve as source web community.
4. **Processing Raw Data:** Noise, stop words and duplicates are removed and the retrieved data from step 3 is transformed to be processed in the next step. This step ensures that the extraction methods receive "clean" data.

5. **Processed Data:** The result of the refining process applied to the raw data.
6. **Knowledge Extraction:** Based on experience an extraction method (like a Named-Entity-Recognition (NER)-Tagger) has to be selected. This step relies on the methods available and the experiences made in past extraction processes as well as it depends on the kind of knowledge that has to be extracted. Each extraction method requires different types of additional input data like vocabulary lists (stop words), rules or ontologies (WordNet) to carry out the information extraction tasks. In this steps this input is provided (if not already done in the domain model step (step 1)) as well as the information extraction application has to be configured (represented by the lower arrow in Figure 1. Within this step this data has to be provided to finish up the final preparation of the knowledge extraction.
7. **Extracted Knowledge:** The knowledge extraction is carried out by executing the extraction processes using the previously described auxiliary material. After this step the extracted knowledge is ready for further use.
8. **Application in Knowledge Container:** The obtained knowledge is transferred to its according knowledge container so it can be applied in the Case-Based Reasoning process. In case new knowledge models have been built, they have to be integrated in the knowledge representation and made available to the processes they are supporting.
9. **Evaluation:** The knowledge obtained is evaluated using the Case-Based Reasoning system(s): this describes evaluation processes of each of the four kinds of knowledge within its application domain. Information on the quality of knowledge can be obtained to further develop the extraction processes by applying experiences made while executing the knowledge extraction process.

The Knowledge Extraction steps are carried out interactively and the degree of automation highly depends on the web community and the target data. The following sections will introduce an interactive knowledge extraction application that uses a discussion forum to build taxonomies.

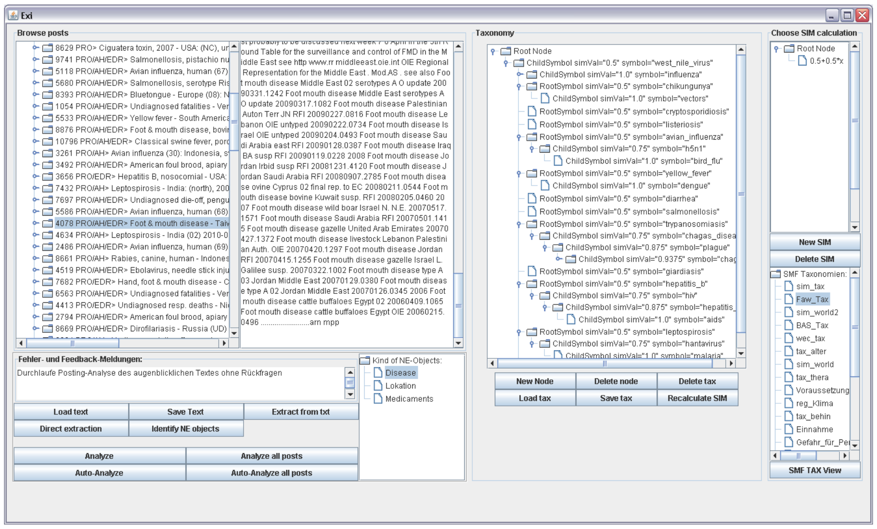
3 Interactive Knowledge Extraction from Community Data

The knowledge extraction process for Case-Based Reasoning systems we have introduced will now be applied in a real life application domain travel medicine. We use a web forum in which experts, ex-pats and travelers discuss various topics regarding traveling to South East Asia. First, we will introduce the knowledge extraction workbench, the graphical user interface presenting the knowledge extraction results to the knowledge engineers, followed by an explanation of our experimental results after applying the knowledge extraction process on the aforementioned forum.

3.1 Knowledge Extraction Workbench

Figure 2 shows the Knowledge Extraction Workbench that we have developed for extracting taxonomies from web community data. This user interface supports the knowledge extraction process steps 4 – 7.

One the left hand side the knowledge engineer can browse forum posts and select the analysis method. The right hand side displays the extracted taxonomies and allows the knowledge engineer to load and save different taxonomies as well as manipulate similarity measures. Further, the knowledge engineer can edit the taxonomy and recalculate the similarity assignments for any node within the taxonomy. A more detailed explanation of the workbench can be found in [18].



```
8629 PRO> Ciguatera toxin, 2007 - USA: (NC), ur
9741 PRO/AH/EDR> Salmonellosis, pistachio nu
5118 PRO/AH/EDR> Avian influenza, human (67)
5680 PRO/AH/EDR> Salmonellosis, serotype Ris
8393 PRO/AH/EDR> Bluetongue - Europe (08): N
1054 PRO/AH/EDR> Undiagnosed fatalities - Ver
5533 PRO/AH/EDR> Yellow fever - South America
8876 PRO/AH/EDR> Foot & mouth disease, bovir
10796 PRO/AH/EDR> Classical swine fever, porc
3261 PRO/AH> Avian influenza (30): Indonesia, s
3492 PRO/AH/EDR> American foul brood, apiary
3656 PRO/EDR> Hepatitis B, nosocomial - USA:
7432 PRO/AH> Leptospirosis - India: (north), 200
7697 PRO/AH/EDR> Undiagnosed die-off, pengu
5586 PRO/AH/EDR> Avian influenza, human (68)
4078 PRO/AH/EDR> Foot & mouth disease - Taiv
4634 PRO/AH> Leptospirosis - India (02) 2010-0
2486 PRO/AH/EDR> Avian influenza, human (69)
8661 PRO/AH> Rabies, canine, human - Indones
4519 PRO/AH/EDR> Ebolavirus, needle stick inju
7682 PRO/EDR> Hand, foot & mouth disease - C
6563 PRO/AH/EDR> Undiagnosed fatalities - Ver
4413 PRO/EDR> Undiagnosed resp. deaths - Ni
2794 PRO/AH/EDR> American foul brood, apiary
8669 PRO/AH/EDR> Dirofilariasis - Russia (UD)
```

```
ound Table for the surveillance and control of FMD in the M
iddle East see http www.rr middleeast.oie.int OIE Regional
 Representation for the Middle East . Mod.AS . see also Foo
t mouth disease Middle East 02 serotypes A O update 200
90331.1242 Foot mouth disease Middle East serotypes A
O update 20090317.1082 Foot mouth disease Palestinian
 Auton Terr JN RFI 20090227.0816 Foot mouth disease Le
banon OIE untyped 20090222.0734 Foot mouth disease Is
rael OIE untyped 20090204.0493 Foot mouth disease Sau
di Arabia east RFI 20090128.0387 Foot mouth disease Iraq
 BA susp RFI 20090119.0228 2008 Foot mouth disease Jo
rdan Irbid susp RFI 20081231.4120 Foot mouth disease J
ordan Saudi Arabia RFI 20080907.2785 Foot mouth disea
se ovine Cyprus 02 final rep. to EC 20080211.0544 Foot m
outh disease bovine Kuwait susp. RFI 20080205.0460 20
07 Foot mouth disease wild boar Israel N. N.E. 20070517.
1571 Foot mouth disease Saudi Arabia RFI 20070501.141
5 Foot mouth disease gazelle United Arab Emirates 20070
427.1372 Foot mouth disease livestock Lebanon Palestini
an Auth. OIE 20070420.1297 Foot mouth disease Jordan
RFI 20070415.1255 Foot mouth disease gazelle Israel L.
Galilee susp. 20070322.1002 Foot mouth disease type A
03 Jordan Middle East 20070129.0380 Foot mouth diseas
e type A 02 Jordan Middle East 20070126.0345 2006 Foot
 mouth disease cattle buffaloes Egypt 02 20060409.1065
Foot mouth disease cattle buffaloes Egypt OIE 20060215.
0496 ........................arn mpp
```

```
Durchlaufe Posting-Analyse des augenblicklichen Textes ohne Rückfragen
```

```
Kind of NE-Objects:
  Disease
  Lokation
  Medicaments
```

```
Root Node
  ChildSymbol simVal="0.5" symbol="west_nile_virus"
    ChildSymbol simVal="1.0" symbol="influenza"
    RootSymbol simVal="0.5" symbol="chikungunya"
      ChildSymbol simVal="1.0" symbol="vectors"
    RootSymbol simVal="0.5" symbol="cryptosporidiosis"
    RootSymbol simVal="0.5" symbol="listeriosis"
    RootSymbol simVal="0.5" symbol="avian_influenza"
      ChildSymbol simVal="0.75" symbol="h5n1"
        ChildSymbol simVal="1.0" symbol="bird_flu"
    RootSymbol simVal="0.5" symbol="yellow_fever"
      ChildSymbol simVal="1.0" symbol="dengue"
    RootSymbol simVal="0.5" symbol="diarrhea"
    RootSymbol simVal="0.5" symbol="salmonellosis"
    RootSymbol simVal="0.5" symbol="trypanosomiasis"
      ChildSymbol simVal="0.75" symbol="chagas_disea
        ChildSymbol simVal="0.875" symbol="plague"
          ChildSymbol simVal="0.9375" symbol="chag
    RootSymbol simVal="0.5" symbol="giardiasis"
    RootSymbol simVal="0.5" symbol="hepatitis_b"
      ChildSymbol simVal="0.75" symbol="hiv"
        ChildSymbol simVal="0.875" symbol="hepatitis_
          ChildSymbol simVal="1.0" symbol="aids"
    RootSymbol simVal="0.5" symbol="leptospirosis"
      ChildSymbol simVal="0.75" symbol="hantavirus"
        ChildSymbol simVal="1.0" symbol="malaria"
```

```
Root Node
  0.5+0.5*x
```

```
SMF Taxonomien:
  sim_tax
  Faw_Tax
  sim_world2
  BAS_Tax
  wec_tax
  tax_alter
  sim_world
  tax_thera
  Voraussetzung
  reg_Klima
  tax_behin
  Einnahme
  Gefahr_für_Pe
```

Fig. 2. Screenshot of the Knowledge Extraction Workbench

The domain detection and provision of the web community data has been done in advance as well as a web forum equipped with intelligent agents was created. The agents are monitoring the forum and pass on posts that belong to a certain topic, currently we deal with the topics location, diseases and medicaments. In advance, we have ensured that our experts agree with the fact that we further process their posts. The intelligent web forum is built upon a data base so we can access the free text via an odbc connection.

Currently, the workbench supports building taxonomies that can be directly imported in the Protégé[2] plugin of MyCBR [20]. Further on, the workbench is connected to GATE, a text engineering tool [6], that we use for information

[2] `http://protege.stanford.edu/`

extraction. The according GATE component is called ANNIE (A Nearly-New Information Extraction System) that we configured using lists of keywords and rules. The goals of the Knowledge Extraction Workbench is creating taxonomies using forum posts. The forum posts can be analyzed in two ways - either post by post - or the whole thread in the forum. In both cases we do a quantitative analysis of the data and based on the number of occurrences of terms we create a parent node for those terms that occur most frequently and add child nodes that occur second most frequently. We are assuming that according to Church and Hanks [5], two terms are more similar the more often they occur next to each other. For example a forum post contains four times the term *children's disease* and three times the term *measles*. Our approach takes children's disease as parent node and assigns measles as its child node.

3.2 Experimental Results

The experiments are carried out on a forum containing posts of travelers, expats and experts who are discussing various issues on travel medicine aspects, especially prevention and travel preparation issues. On the other hand, we have a web forum that contains posts discussing all kinds of travel related topics. In total our experimental data contains about 6500 forum posts and we are extracting knowledge for a medicament, location and disease case base. During the development of the docQuery application, our travel medicine experts provided key word lists for medicaments and diseases that we use in combination with a set of rules for setting up the information extraction environment of ANNIE.

Since the Knowledge Extraction Workbench offers two types of analysis, we tested the performance of those. The first type analyzes the discussion thread in the way that the whole thread is treated as an isolated text. In this process, the term chains are created covering the whole thread. A term chain is a sorted list, descending according to the number of occurrences, of all extracted terms.

In comparison, the second analysis type treads each post as an isolated text and executes the aforementioned process, thus building term chains only covering the isolated post.

Figure 3 depicts two resulting taxonomies with he upper one after the full thread analysis and the lower one post-by-post analysis. The full thread analysis always created deeper taxonomies, because the term chains are longer if built over the whole text of a thread.

4 Discussion

The work presented in this paper focused on web communities and how to systematically use the experiences provided by users of this communities in Case-Based Reasoning systems. We are positive that Case-Base Reasoning can handle and benefit from the information given during discussions, because it is a flexible methodology for integrating different kinds of knowledge in a knowledge based

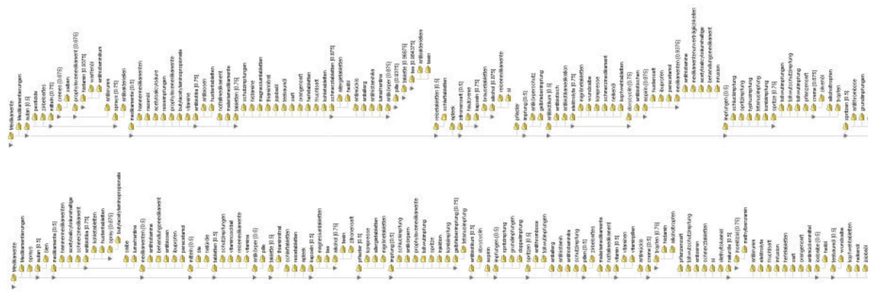


Fig. 3. Taxonomy Depth Comparison

system. We have discovered three different scenarios in which the knowledge extraction process we introduced can be applied: First, the knowledge extraction process can be applied once during the development of a Case-Based Reasoning system supporting the system's design. Second, it can be used for initially filling the knowledge containers and third it can be completely integrated in the maintenance process of Case-Based Reasoning systems. Therefore the extraction has to be scheduled regularly and processes have to be defined that allow a further development of the knowledge structure, comprehending updating, evaluating the knowledge and assuring its quality. Within docQuery, we use the last scenario because web communities are the main type of knowledge sources. Using the third scenario of the extraction process allows us to react to changes in these communities and allows us to discover and integrate new terms quickly.

Web Communities already exist on many topics, however, using those for processing user generated content systematically requires the approval of its users. We have used a web forum that we have built for the docQuery project and that has been designed for extracting knowledge from user discussions. Disadvantages of this approach are the requirement of a sufficient amount of users with expertise and the tendency of web based discussions to cover a broad range of topics thus often failing to concentrate on a given topic.

Usually the discussions are centered around a manageable amount of topics. On the other hand, working with data from existing communities requires their permission. We have made the experience that when communicating why and how their data is processed, they usually agree. However, the recent development of the Web 2.0 focuses more and more on the collaboration and participation of users which both provide further support for automatic experience and knowledge processing.

Applying Case-Based Reasoning within the Web 2.0 can also lead towards demand-oriented and web-based systems that integrate their functionalities in daily working routines of users and therewith provide knowledge gained from their contributions and behavior. Therefore, the technologies for knowledge extraction have to be further developed and semantic web technologies, multi-agent-technologies as well as soft-computing technologies can improve todays systems. Also, a deeper integration of users is possible: it might be feasible to allow the community member actively participate in knowledge modeling.

According to the case factory approach [2], we can also imagine using Case-Based Reasoning systems for supporting or even executing the information extraction. Therefore we have to first fill a case base covering information on what type of extraction method can be applied on which source data to retrieve a certain type of knowledge. In this context, Case-Based Reasoning might be able to keep up with very knowledge intensive IE methods that require additional information like thesauri, gazetteer or annotated training data, because it can handle incomplete information and the similarity-based search might be able to cope with today's problems of achieving an exact analysis and mining of text segments.

5 Summary and Outlook

This paper picks up the task of realizing knowledge extraction for Case-Based Reasoning systems from web communities. First we discussed how experiences occur on the web and in which way they can be further used. Afterwards we presented a selection of related approaches before we have introduced a knowledge extraction process that has later on be applied and evaluated in the travel medicine domain. The paper sums up with a discussion of the results and the further potential of the approach we presented.

The knowledge extraction process is a first realization based on well known and applied methods that have been applied in a new way. After establishing a knowledge extraction process we will now concentrate on applying and developing new approaches that improve the quality of the retrieved data or integrate different sources obtained from Web 2.0 developments. For example folksonomies as a result of social tagging analysis can serve as case base vocabulary.

Another aspect of further research is the application of the knowledge extraction process on topics that are more open like free time activities or more general travel experiences. Currently we are focusing on key word centered data and the transition towards the extraction of more various data with more flexible case representations. Further on, the knowledge extraction workbench is still a prototype and has to be improved aiming at a better usability for knowledge engineers.

References

1. 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 knowlegde-product-lines with case factories and distributed learning systems. In: Baumeister, J., Seipel, D. (eds.) Workshop Proc. on the 3rd Workshop on Knowledge Engineering and Software Engineering (KESE 2007). Osnabrück (Sep 2007)

2. Althoff, K.D., Hanft, A., Schaaf, M.: Case factory – maintaining experience to learn. In: Göker, M., Roth-Berghofer, T. (eds.) Proc. 8th European Conf. on Case-Based Reasoning (ECCBR'06), Ölüdeniz/Fethiye, Turkey. Springer, Berlin (2006)
3. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications, LNCS, vol. 2432. Springer (2002)
4. Chakrabarti, S., Dom, B.E., Gibson, D., Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.: Mining the link structure of the world wide web. IEEE Computer 32, 60–67 (1999)
5. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Computational Linguistics 16(1), 22–29 (1990)
6. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: Gate: A framework and graphical development environment for robust nlp tools and applications. In: Proc. of the 40th Anniv.Meeting of the Assoc. for Comp. Linguistics (ACL'02) (2002)
7. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: The kdd process for extracting useful knowledge from volumes of data. Commun. ACM 39(11), 27–34 (1996)
8. Feng, D., Shaw, E., Kim, J., Hovy, E.: An intelligent discussion-bot for answering student queries in threaded discussions. In: IUI '06: Proc. of the 11th Intl Conf. on Intelligent user interfaces. pp. 171–177. ACM Press, New York, NY, USA (2006)
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. pp. 269–278 (July 2009)
10. van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering. Springer, Berlin, Heidelberg, Paris (2007)
11. Liu, B.: Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2008)
12. Mika, P.: Flink: Semantic web technology for the extraction and analysis of social networks. Journal of Web Semantics 3, 211–223 (2005)
13. Mika, P.: Social Networks and the Semantic Web, vol. 5. Springer Science+Business Media, LLC, Boston, MA (2007)
14. Milne, P., Wiratunga, N., Lothian, R., Song, D.: Reuse of search experience for resource transformation. In: Delany, S.J. (ed.) ICCBR 2009 Workshop Proc., Workshop Reasoning from Experiences on the Web. pp. 45–54 (July 2009)
15. Plaza, E., Baccigalupo, C.: Principle and praxis in the experience web: A case study in social music. In: Delany, S.J. (ed.) ICCBR 2009 Workshop Proc., Workshop Reasoning from Experiences on the Web. pp. 55–63 (July 2009)
16. Reichle, M., Bach, K., Althoff, K.D.: The seasalt architecture and its realization within the docquery project. In: Mertsching, B. (ed.) Proc. of the 32nd Conference on Artificial Intelligence (KI-2009). pp. 556–563. LNCS (Sep 2009)
17. 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)
18. Sauer, C.S.: Analyse von Webcommunities und Extraktion von Wissen aus Communitydaten fr Case-Based Reasoning Systeme. Master's thesis, Institute of Computer Science, University of Hildesheim (2010)
19. Smyth, B., Champin, P.A., Briggs, P., Coyle, M.: The case-based experience web. In: Delany, S.J. (ed.) ICCBR 2009 Workshop Proc., Workshop Reasoning from Experiences on the Web. pp. 74–82 (July 2009)
20. Stahl, A., Roth-Berghofer, T.R.: Rapid prototyping of cbr applications with the open source tool mycbr. In: ECCBR '08: Proc. of the 9th European conference on Advances in Case-Based Reasoning. pp. 615–629. Springer, Heidelberg (2008)

The GhostWriter-2.0 System: Creating a Virtuous Circle in Web 2.0 Product Reviewing

Paul Healy and Derek Bridge

Department of Computer Science,
University College Cork,
Ireland
pjh1@student.cs.ucc.ie,d.bridge@cs.ucc.ie

Abstract. In this paper, we present GhostWriter-2.0, a Case-Based Reasoning system that supports a user who is writing a product review. GhostWriter-2.0's case base is populated with relevant and helpful reviews from Amazon. It extracts phrases from these reviews and offers these as suggestions to the user. In a trial, users made considerable use of the suggested phrases.

1 Introduction

A lot of user-generated content on the Web takes the form of records of *personal experiences* [5, 7, 6]. Epitomizing these records of personal experience are the reviews and ratings that users contribute on everything from movies to books to music to hotels to consumer goods to professional services and to on-line content, whether user-generated or otherwise. Interestingly, Web sites that solicit user reviews often solicit meta-experiences too: users can review or rate other reviews or reviewers. For example, Amazon users vote on reviews, so the site can report review helpfulness; Epinions users additionally vote on reviewers, so the site can report reviewer trustworthiness.[1] Meta-experience like this provides a partial remedy for the problem of how to ignore noisy reviews.

Once shared on the Web, one user's experiences can be reused by another user to support her in some task that she is performing. Case-Based Reasoning (CBR), often defined as reasoning from experiences [4], offers one way to support such a user. Examples of using CBR in this way include the Poolcasting system, which uses CBR to select music to play in social situations [6]; the work of [1], which uses CBR to recommend data visualizations; and CommunityCook, which extracts recipe adaptation knowledge from Web-based recipe corpora [2].

Our position is that there are in fact at least two ways in which records of experience on the Web can be reused:

– to support the user in her real-world task, e.g. booking a hotel, selecting music to play, installing software, and visualizing data; and
– to support the user when she authors new content, e.g. writing a new review.

[1] www.amazon.com, epinions.com

In our own work, we have been developing the GhostWriter family of systems, which uses CBR in the second of these two ways. In GhostWriter-1.0, we showed how CBR could help the user to write more comprehensive descriptions in Web-based exchange systems [8]. GhostWriter-2.0, described for the first time here, is our system which uses CBR to help the user to write reviews.

We developed GhostWriter-2.0 in response to the WebCBR Challenge associated with this workshop.[2] The Workshop Organizers challenged CBR researchers to apply CBR ideas to the Experience Web, the only constraint being that the researchers should "showcase their research in the form of a demonstration system using Amazon's API". The GhostWriter-2.0 case base is populated with existing relevant and high quality Amazon product reviews; and the system uses the case content to assist the user to author a new review.

GhostWriter-2.0 has the potential to create a virtuous circle: if its assistance results in the user writing and submitting a better review than she otherwise would, then a new higher quality record of experience becomes available both to users making purchasing decisions but also to GhostWriter-2.0 next time it assists someone to write a review of a similar item.

Section 2 presents GhostWriter-2.0 from the user point of view; section 3 describes how GhostWriter-2.0 converts Amazon reviews into cases; section 4 explains how GhostWriter-2.0 populates its case base, and how it uses the cases to make suggestions; section 5 reports the results of a real user trial.

2 An End-User View of GhostWriter-2.0

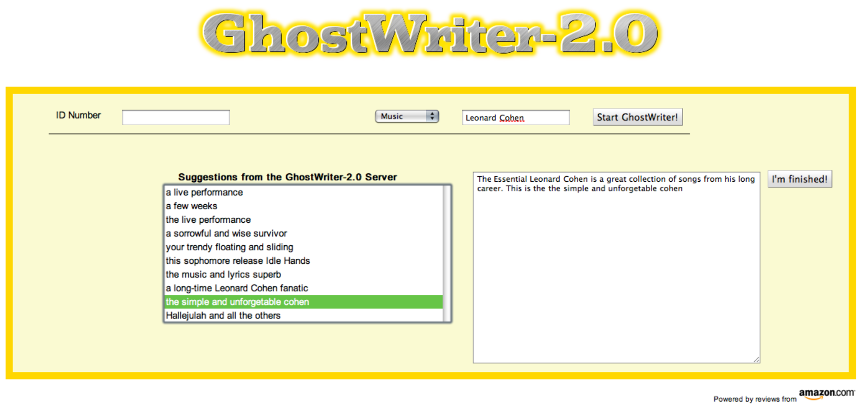
A user who wishes to review an Amazon product with support from GhostWriter-2.0 accesses GhostWriter-2.0 through her browser. We give a screenshot in Figure 1. The user starts at the top of the screen by using the drop-down select list to choose the category of product that she will review; in the example, the user has chosen Music. In the adjacent text-field, she enters keywords to describe the product. This might be the author of a book, the artist for a music CD, or the manufacturer of a digital camera, for example; in the screenshot, the user has entered "Leonard Cohen".[3]

The user next presses the *Start GhostWriter!* button. The GhostWriter-2.0 server takes the user's category and keywords and uses them to query the Amazon information servers. The search returns products and their reviews. These are not shown to the user. Instead, we use the best of these reviews to populate the case base (see Section 4.1).

The user may now type her review into the right-hand text-area. After approximately every 60 characters that the user types, GhostWriter-2.0 will make a set of suggestions, which the browser displays in the selection form to the left of the text-area. GhostWriter-2.0 selects these suggestions from the case base (see Section 4.2).

[2] `www.comp.dit.ie/aigroup/?page\_id=549`

[3] The text-field in the top-left of the screenshot allows us to identify users when we are running user trials (Section 5) and is otherwise not used.



Music

Leonard Cohen

Suggestions from the GhostWriter-2.0 Server

a live performance
a few weeks
the live performance
a sorrowful and wise survivor
your trendy floating and sliding
this sophomore release Idle Hands
the music and lyrics superb
a long-time Leonard Cohen fanatic
the simple and unforgetable cohen
Hallejulah and all the others

The Essential Leonard Cohen is a great collection of songs from his long career. This is the the simple and unforgetable cohen

Fig. 1. A screenshot from GhostWriter-2.0

A user, glancing at the suggestions, may take one of three actions:

– She might decide to include a suggestion exactly ‘as is’ in her review. To do this, she double-clicks on the suggestion.
– She might decide that none of the suggestions can be included verbatim, but one or more of them may inspire her to write on a related topic.
– She may find none of the suggestions helpful. In this case, she continues to type her review.

When she has finished her review, she presses the *I'm finished!* button to submit her review to the GhostWriter-2.0 server, where statistics about the review are logged. Unfortunately, the Amazon API appears not to allow the GhostWriter-2.0 server to directly upload the review to Amazon.

3 Cases in GhostWriter-2.0

Fundamentally, cases in GhostWriter-2.0 are Amazon product reviews. We convert an Amazon product review r into a three-part GhostWriter-2.0 case $c = \langle W, S, h\rangle$ as follows. The first part, W, which can be thought of as the problem description part of the case, is the set of words that occur in the text of review r. Information about which product is being reviewed is not needed in the problem description part of the case because this contextual focus is already provided by the case base as a whole (see Section 4.1). The second part, S, which can be thought of as the problem solution part of the case, is a set of suggestions, i.e. a set of phrases that GhostWriter-2.0 might show to the author of a new review. We explain how we extract these from r in the paragraphs below. The third part of a case, h, is a non-negative integer. Recall that Amazon users can vote to signal that they found a review to be helpful. h is r's helpfulness rating. It can be thought of as the outcome part of the case [4]. It gives an indication of c's quality and, indirectly, an indication of the quality of the suggestions in S.

We have yet to explain how we extract S from r. In GhostWriter-1.0, suggestions were feature-value pairs [8]. There, we were working in the domain of Web-based exchange systems, such as classified ads systems, and so each case described an item that someone wanted to dispose of. We used regular expressions to extract the value of features such as Price (e.g. "€25 or nearest offer"), Condition (e.g. "in excellent condition"), and so on. This was adequately effective because classified ads tend to be written in a domain-specific sub-language [3] that is quite restrictive and uses a lot of recurring 'stock' phrases and phrasing.

Product reviews are written in a much less restrictive way than classified ads. Reviews of novels, for example, rarely mention the price but often give a synopsis of the plot, make comparisons with other novels by the same author or similar authors, describe the passions aroused when reading the book, and offer opinions about the quality of the writing. The author of a new review might welcome suggestions that span all of these. Information Extraction techniques, whether based on regular expressions or something more powerful, could extract simple features such as the price, if present, or the technical attributes of digital electronics goods (e.g. their battery life). Sentiment Analysis techniques could extract the polarity of the opinions expressed, with the risk that it would mine suggestions that would bias new reviews towards the majority opinions of past reviews in the case base. But, even if we were to use both of these techniques together, we would extract only a subset of the potentially useful suggestions.

Hence, we decided to take an approach that was less specific than either Information Extraction or Sentiment Analysis. We took the view that most of the descriptive content of a review lies in its noun phrases. They can cover: technical attributes of consumer goods; characters, places and events in books and films; authors and artists; and some of the opinion.

Noun phrases are easy to extract from text, using just quite shallow natural language processing techniques. We use OpenNLP's maximum-entropy-based 'chunking'.[4] We retain only those phrases that it labels as noun phrases. In fact, we decided to retain only noun phrases that were longer than two words (because shorter noun phrases tend to lack enough descriptive content to make useful suggestions) but shorter than six words (because longer noun phrases tend to make overly specific suggestions). Note that there is an additional scoring process that will determine which of a case's suggestions are actually shown to the user (see Section 4.2).

In summary then, in GhostWriter-2.0 an Amazon review r becomes a case c and comprises: the set of words in r; the set of r's noun phrases that contain three to five words; and r's helpfulness rating.

4 How Ghostwriter-2.0 Works

The GhostWriter-2.0 server sits between the user's browser and the Amazon servers. There are two phases to its processing: populating the case base, and making suggestions.

[4] `opennlp.sourceforge.net`

4.1 Populating the case base

As explained in Section 2, a user who wishes to write a review with support from GhostWriter-2.0 begins by entering data about the product she wishes to review: she enters its category and some keywords into her browser and submits them to the GhostWriter-2.0 server. The GhostWriter-2.0 server forwards this data to the Amazon servers. The operation it requests is an Amazon `ItemSearch` for the given category and keywords, which returns up to 4000 products in pages of up to ten products each. We additionally request that, for each product, the `ItemSearch` returns reviews, ordered by decreasing helpfulness. It returns up to five reviews per product, and it these we use to populate the case base.

We aim to populate the case base with 250 cases. As a way of trying to avoid duplicated products and reviews, we check customer ids and only take the first review for each customer. If the `ItemSearch` fails to provide us with 250 cases, we take each product in turn and use an Amazon `ItemLookup` to obtain a further 10 reviews, again checking customer ids before inserting them into the case base.

Note that this means that the case base is populated afresh each time a user starts a new review. This ensures that its contents are relevant to the current product being reviewed, and that the case base takes into account the very latest data (products, reviews and helpfulness ratings) on the Amazon site. The downside is that there is an appreciable delay (up to 2 minutes) in inserting reviews into the case base. This is caused by the time it takes to launch OpenNLP and to use it to extract noun phrases from the 250 reviews. A production version of Ghostwriter-2.0 would need to be better integrated with Amazon to bring this time down.

4.2 Making suggestions

After GhostWriter-2.0 has populated its case base, the user starts typing her review into her browser. As explained in Section 2, after approximately every 60 characters that the user types, the browser uses AJAX to request a set of suggestions from the GhostWriter-2.0 server. The browser supplies the server with the current contents of the user's review. The GhostWriter-2.0 server retrieves $k_1 = 50$ cases from the case base. From the suggestions contained in these cases, it selects $k_2 = 10$ suggestions, which it returns to the user. We explain both the retrieval and the selection in more detail below.

Retrieval. Let the current contents of the user's review be called the *new product review* and designated *npr*.[5] This is the set of words that the user has typed. GhostWriter-2.0 retrieves from the case base the k_1 cases that are most similar to the *npr*. We measure similarity using the Jaccard similarity coefficient:

$$sim(npr, c = \langle W, S, h \rangle) = \frac{|npr \cap W|}{|npr \cup W|}$$

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

Suggestion selection. At this point, GhostWriter-2.0 has retrieved k_1 cases C from the case base, and in each case $c = \langle W, S, h \rangle$ there is a set of suggestions S; each suggestion is a noun phrase. GhostWriter-2.0 must select the k_2 suggestions that it will send back to the browser for display. When considering a unique candidate suggestion, $s \in \bigcup_{\langle W,S,h \rangle \in C} S$, several criteria seem relevant, discussed over the next few paragraphs.

Consider the number of retrieved cases in which a suggestion s appears:

$$\mathit{freq}(s) = |\{\langle W, S, h \rangle \in C : s \in S\}|$$

To some extent, the more frequent s is, the more its content is something that different reviewers want to talk about, and this makes it a better suggestion. However, frequency favours short suggestions as these are the ones that are more likely to recur. The downside of this is that short, recurring noun phrases are more likely to be vague or generic.

Suppose instead we consider the length in words of the suggestions, $\mathit{length}(s)$. To some extent, the longer s is, the more descriptive it is, and this makes it a better suggestion. But length alone may favour overly specific suggestions.

We found that the product of frequency and length offered a good balance between the two criteria:

$$\mathit{score}(s) = \mathit{freq}(s) \times \mathit{length}(s)$$

The short, three-word phrase “a fantastic album” needs to appear in two separate cases ($2 \times 3 = 6$) if it is to have a score that betters that of the five-word phrase “an excellent, lively rock classic”, assuming this appears in only one case ($1 \times 5 = 5$).

Many suggestions share the same score. To break ties, we sum and compare the helpfulness of the reviews that contain the two suggestions. Formally, if $\mathit{score}(s) = \mathit{score}(s')$, then s will be ranked higher than s' if

$$\sum_{h_s \in \{h : \langle W,S,h \rangle \in C \wedge s \in S\}} h_s \quad > \sum_{h_{s'} \in \{h' : \langle W',S',h' \rangle \in C \wedge s' \in S'\}} h_{s'}$$

(If s and s' have the same total helpfulness, then tie-breaking is arbitrary.)

In addition to this scoring, we also want to apply other criteria.

We do not want to make a suggestion s if it is already a noun phrase in the user's *npr*. This is quite simple-minded. In the future, it may be worth considering ways of measuring the similarity of semantic content: we would discard s if the *npr*'s content was similar enough to cover the content of s.

We also do not want to make a suggestion s if it is one that we have made several times already. This criterion is not one that we built into an early version of GhostWriter-2.0. But in a small-scale pilot study that we conducted prior to the user trial reported in Section 5, we found that users preferred versions of the system that more often made fresh suggestions over versions that allowed suggestions to linger. We subsequently decided to limit the number of times a suggestion could be made. Specifically, if s has already been suggested $\theta = 4$ times, then it cannot be suggested again, allowing another suggestion to take its place. This increases the number of different suggestions that get made, but the suggestion turnover is not so great as to be distracting to the user.

5 Experimental Evaluation

Here, we report the results of a user trial. For comparison purposes, we developed a version of GhostWriter-2.0 that made less intelligent use of the cases. It populates its case base in the same way as GhostWriter-2.0 (Section 4.1). So we know it has a set of relevant and helpful cases. But instead of retrieving the $k_1 = 50$ cases that are most similar to the *npr*, it retrieves $k_1 = 50$ cases at random from the case base. Like GhostWriter-2.0, it will not make suggestions that are already contained in the *npr*, nor will it make a suggestion more than $\theta = 4$ times. But otherwise, the $k_2 = 10$ suggestions that it makes are drawn at random from the retrieved cases: it does not use the *score* function, nor the helpfulness ratings. In this section we will refer to these two systems as GW-2.0 (for GhostWriter-2.0) and GW-R (for the more random version).

The two systems were evaluated by twenty users. Each user reviewed two products with which they were familiar, either two music CDs or two books. As they wrote their reviews, they received suggestions from one of the GhostWriter systems. Ten users had help from GW-2.0 for their first review, and help from GW-R for their second review. The other ten users had help from GW-R first, and then GW-2.0. They were unaware of the difference between the systems.

During the experiment, GW-2.0 and GW-R recorded: the review the user typed; the time taken to write the review; and the suggestions that were explicitly incorporated, i.e. when the user double-clicked on a suggestion. We also administered a questionnaire to each participant.

At one level, reviews written with support from GW-2.0 and with support from GW-R were similar. Their average length was a little over 150 words in both cases; users created their reviews at an average of 13–14 words per minute in both cases; and the average number of descriptive noun phrases in the reviews (which we took to be noun phrases of more than two words) was also 13–14 in both cases. But what is interesting is the breakdown of those noun phrases.

Figure 2 shows, for each user (designated A to T), how many suggestions they directly used (by double-clicking on them). In total, 116 GW-2.0 suggestions were directly incorporated, an average of 5.8 per user, compared with only 83 GW-R suggestions, an average of 4.15. Fourteen of the twenty users used more GW-2.0 suggestions than GW-R ones; one used the same number of suggestions from both; and one used none from either system. We take this as an indication that GW-2.0's greater reuse of Web-based experience data (case similarity, frequency across reviews, and helpfulness) promotes more useful suggestions.

Figure 2 gives no indication of how many other descriptive noun phrases there are in user reviews. Figure 3 shows this in the case of GW-2.0. It enables us to see that, while users vary, directly incorporated suggestions account on

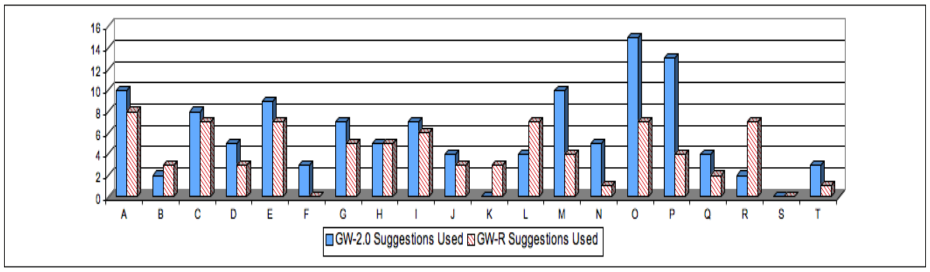


Fig. 2. The number of suggestions that users directly incorporated into reviews

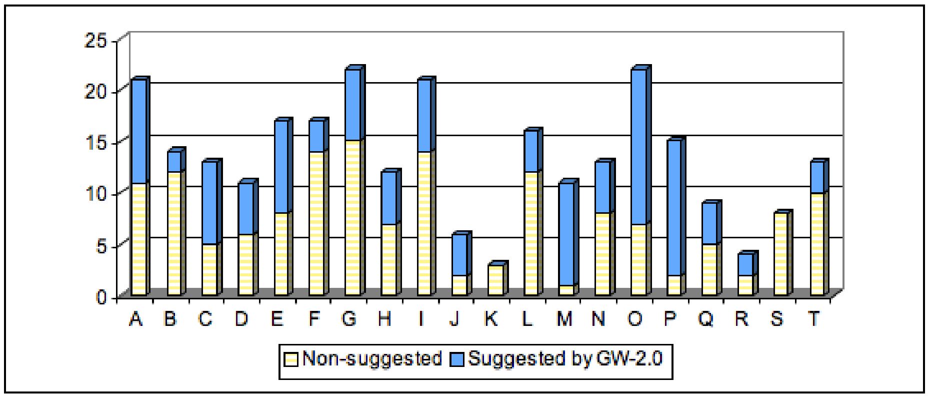


Fig. 3. Noun phrases in reviews written with GW-2.0

average for about 43% of descriptive noun phrases (116 out of the 268 noun phrases of more than two words). We do not have space to show the graph for GW-R but on average only about 30% of descriptive noun phrases were ones GW-R suggested (83 out of 275 noun phrases of more than two words).

Figures 4, 5 and 6 summarize results from the questionnaire. Nineteen of the twenty people agreed or strongly agreed that GW-2.0 helped them to write a comprehensive review; for GW-R, sixteen people agreed or strongly agreed (Figure 4). Eighteen people agreed or strongly agreed that GW-2.0's suggestions were helpful, more than half of them strongly agreeing; for GW-R, sixteen people agreed or strongly agreed, only a minority strongly agreeing (Figure 5). We also asked participants to estimate for each review that they wrote how many times they were inspired by a suggestion but did not actually double-click on it and so did not incorporate it directly. This was obviously very subjective and not likely to be reliable. But it at least gives an impression of the extent to which the numbers reported in Figures 2 and 3 understate the helpfulness of the systems. According to the responses to this (Figure 6) GW-2.0 and GW-R were fairly evenly-matched in their ability to inspire, and inspired their users one or more times in all but three reviews.

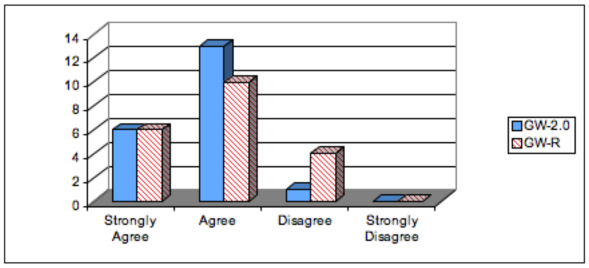


Fig. 4. "The system helped me to write a comprehensive review"

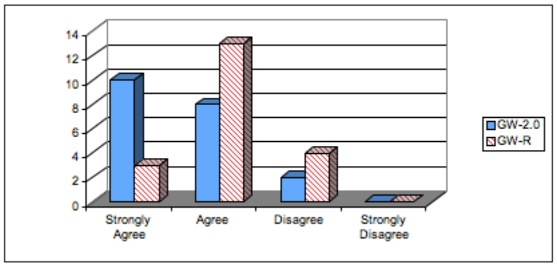


Fig. 5. "I found the suggestions helpful"

We also asked participants for their view of the quality of the reviews they found in on-line stores: nineteen out of twenty thought they were Good or Fair; no-one thought they were Excellent; no-one thought they were Bad; but one person thought they were Terrible. Finally, asked whether they would write more comprehensive reviews if a GhostWriter-style system were available on a store's Web site, everyone agreed or strongly agreed.

6 Conclusions and Future Work

We have presented GhostWriter-2.0, which reuses the experience captured in Amazon product reviews and their meta-review data to make suggestions to a user who is writing a new product review. Our user trial has very promising results, showing a high level of use of the suggestions, both directly and indirectly. In the trial, GhostWriter-2.0 was more helpful to users than a less sophisticated version with an element of randomness to its selections. The differences, however, were small. This may be because, irrespective of which system was being used, users in the trial relied on the suggestions to a degree greater than they would in reality, where they would be writing reviews for endogenous reasons, rather than at our behest.

There are many lines of future research. We would like to measure and, if necessary, explicitly enhance the diversity of the suggestions that GhostWriter-2.0 makes, for example. We would also like to see GhostWriter-2.0 more closely

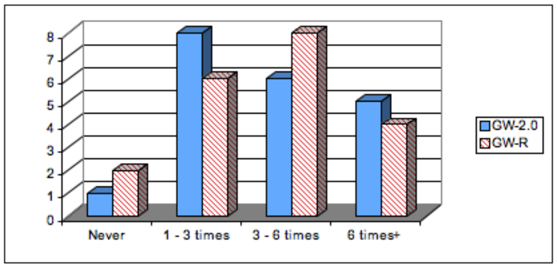


Fig. 6. "The number of times I was inspired by suggestions but I didn't click on them"

integrated with the on-line store that it supports, as this can both reduce the time it takes to populate the case base and make the system less cumbersome to use (e.g. by automatically taking the category and keywords from the page the user is visiting). Finally, we are keen to extend the GhostWriter family into new domains where authors can benefit from the kind of support that GhostWriter systems can offer.

References

1. Jill Freyne and Barry Smyth. Many Cases Make Light Work for Visualization in Many Eyes. In D. Bridge, E. Plaza, and N. Wiratunga, editors, *Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR)*, pages 25–44, 2009.
2. Norman Ihle, Alexandre Hant, and Klaus-Dieter Althoff. Extraction of Adaptation Knowledge from Internet Communities. In D. Bridge, E. Plaza, and N. Wiratunga, editors, *Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR)*, pages 35–44, 2009.
3. Richard Kittredge and John Lehrberger. *Sublanguage: studies of language in restricted semantic domains*. de Gruyter, 1982.
4. Janet L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.
5. 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.
6. Enric Plaza and Claudio Baccigalupo. Principle and Praxis in the Experience Web: A Case Study in Social Music. In D. Bridge, E. Plaza, and N. Wiratunga, editors, *Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR)*, pages 55–63, 2009.
7. Barry Smyth and Pierre-Antoine Champin. The Experience Web: A Case-Based Reasoning Perspective. In S. Craw, D. W. Aha, S. Singh, and B. Smyth, editors, *Procs. of the Workshop on Grand Challenges for Reasoning From Experiences (at the 21st IJCAI)*, pages 53–61, 2009.
8. Aidan Waugh and Derek Bridge. An Evaluation of the GhostWriter System for Case-Based Content Suggestions. In L. Coyle, J. Dunnion, and J. Freyne, editors, *Procs. of the 20th Irish Conference on Artificial Intelligence and Cognitive Science*, pages 264–273, 2009.

Enabling Case-Based Reasoning on the Web of Data

Benjamin Heitmann and Conor Hayes

`firstname.lastname@deri.org`
Digital Enterprise Research Institute
NUI Galway
Ireland

Abstract. While Case-based reasoning (CBR) has successfully been deployed on the Web, its data models are typically inconsistent with existing information infrastructure and standards. In this paper, we examine how CBR can operate on the emerging Web of Data, with mutual benefits. The expense of knowledge engineering and curating a case base can be reduced by using Linked Data from the Web of Data. While Linked Data provides experiential data from many different domains, it also contains inconsistencies, missing data and noise which provide challenges for logic-based reasoning. CBR is well suited to provide alternative and robust reasoning approaches. We introduce (i) a lightweight CBR vocabulary which is suited for the open ecosystem of the emerging Web of Data, and provide (ii) a detailed example of a case base using data from multiple sources. We propose that for the first time the Web of Data provides data and a real context for open CBR systems.

1 Introduction

Case-Based Reasoning (CBR) has made great inroads on the Web as a means of automated technical support and product recommendation. However, it still is open to the criticism made by Kitano and Shimazu that its focus on domain specific problems has led to closed and inflexible data models, typically isolated from other information infrastructures and standards [1]. With the subsequent growth of the Web, there have been several attempts by the CBR community to standardise the different parts of CBR's knowledge containers to enable interoperability and wider dissemination of case-based technology [2–6]. Several iterations of the XML-based Case-Based Markup Language (CBML) [2, 5, 7, 6] were developed between 1998 and 2004, though practical adoption of the language was low. Concurrent work on distributed and multi-casebase reasoning, while assuming case-base interoperability, did not specify how that might be achieved (see [8] for an overview). Lack of adoption may have been partly due to the cumbersome manner in which the structure and the semantics of the case data were described in XML. We suggest that a greater impediment was the lack of real data and real contexts in which distributed case-based reasoning would have been required.

In approximately the same time period, rule and logic-based reasoning has gone through a renaissance in close association with the **Semantic Web** research initiative. However, the particular strengths of case-based reasoning - its suitability for weak theory domains, its relative robustness to noise and its ability to incorporate techniques from machine and statistical learning make it an ideal alternative reasoning paradigm for the Semantic Web, an observation that has been made by Tim Berners-Lee [9]. Furthermore, the constraints for wider adoption of case-based reasoning standards have significantly been loosened through several features of the rapidly emerging **Web of Data**, a highly practical offshoot of Semantic Web research. Firstly, the awkward requirement to define case semantics has been delegated to the hundreds of integrated Linked Data vocabularies already deployed. Secondly, there is an enormous amount of data in structured, semantically annotated format already on the Web. What is missing is a way of creating flexible case views of this data so that reasoning with cases can emerge as standard reasoning paradigm on the Web of Data.

In this paper, we describe how we can create case views of **Linked Data**, which would enable the authoring of and querying of distributed case data on the Web. Linked Data refers to a set of principles for publishing and connecting structured data on the Web, thus forming a Web of Data [10]. While existing CBR systems have been developed under the assumption of a closed ecosystem, the Web of Data represents an open ecosystem, where each data source can evolve and change independently [11]. During the course of this paper, we will explain how Linked Data can be used to provide structured data for the different knowledge containers of a CBR system.

The main contributions of this paper are: (i) the introduction of a lightweight CBR vocabulary which is suited for the open ecosystem of the emerging Web of Data, and (ii) a detailed example of a case base using data from multiple sources.

The rest of the paper is structured as follows: In section 2 we discuss related work and explain our approach for providing a CBR view on Linked Data. Section 3 explains the necessary background about the Linked Data principles and the standards used for the Web of Data, as well as Linked Data sources for experience mining. Section 4 explains the benefits of using Linked Data as sources for the different knowledge containers of a CBR system. Then in section 5 we introduce our lightweight CBR vocabulary, and describe an example case base from the music domain with data from multiple sources in section 6. Finally, section 8 discusses our contributions and concludes the paper.

2 Related work

Despite the conventional focus on reasoning from a single case base, the idea that case knowledge can be distributed in several places has been recurrent in CBR research from its earliest days. Kolodner [12], Barletta and Mark [13] and Redmond [14] all proposed approaches in which case snippets were linked to form a full case episode. Redmond [14] argued that a case-based reasoner cannot know in advance which parts of different cases may be useful during a problem-solving episode. Therefore, storing cases as a linked network of snippets allowed for more flexible retrieval possibilities than their storage as monolithic cases.

While the approach we propose focuses on developing case views of the emerging Web of Data, we draw inspiration from the ideas in this early work. Kitano and Shimazu's proposal [1] for a less domain-fixated role for CBR led to several research initiatives focusing on how the then new XML standards could be used to develop standard case representation formats [2–4]. Several instantiations of Case-Based Mark-up Language (**CBML**) were subsequently developed with the aim of facilitating case base interoperability and wider dissemination and integration of case-based systems on the Web [5, 7, 6]. With the emergence of XML schemas, Hayes and Cunningham proposed a version of CBML that wrapped existing XML domain data to produce domain-compliant case views [5]. The work in this paper shares this perspective. Subsequently, Coyle et al. produced versions of CBML capable of representing hierarchical and object-oriented case data [7] as well as similarity knowledge [6]. Unfortunately, despite concurrent research on distributed/multi-case based reasoning, none of the CBML research gained much traction. This may have been partly due to the cumbersome manner in which XML encoded the structure and the semantics of the case data. Another impediment was the lack of real data and real contexts in which distributed case-based reasoning would be required.

In this paper we suggest that the emergence of the Web of Data, not only provides data, but a context and real requirement for reasoning with cases. As we discuss in the next section, the Web of Data is described using the Resource Description Format (RDF). There has been some work on an RDF-based Case Markup Language, **CaseML** [15]. However, it suffers from the same problems as the earlier versions of CBML in that it shoehorns existing data into a rigid case format, thus hampering interoperability with existing vocabularies and schemas. Furthermore, it subscribes to a view of a closed RDF-based system, which can integrate data from specific controlled sources, but which does not expect to operate on open data from the Linked Data cloud. In contrast, we propose a flexible, lightweight case vocabulary for creating case views on the Web of Data, an open data environment where new vocabularies are regularly added.

The Web Ontology Language (**OWL**) has been proposed for representing cases in an interoperable way. [16] discusses the general applicability of OWL for case representation in biology and medicine. [17] propose using C-OWL with contexts to model the perspectives of different viewpoints. These approaches do not address the cost of engineering the knowledge required for the case base and the OWL reasoning. Our approach is orthogonal, in that it focuses on authoring and curating the knowledge required to bootstrap a CBR system from the Web of Data.

3 Background: Linked Data and the Web of Data

The term **Linked Data** refers to a set of best practices for publishing and connecting structured data on the web [10]. Taken together, all Linked Data constitutes the Web of Data. While the World Wide Web provides the means for creating a web of human readable documents, the Web of Data aims to create a web of structured, machine-readable data. Much of this data can be used for the purpose of mining experiential data from the Web. By making data available which connects the experiences of users from different domains and communities, Linked Data has the potential to provide data which never before has been available for the purpose of experience mining.

The **Web of Data** utilises technologies from the Semantic Web technology stack: the Resource Description Framework (**RDF**) provides a graph based data model and the basic, domain independent formal semantics for the data model [18]; the **SPARQL** Query Language allows querying RDF data with graph patterns, and provides basic means for transforming RDF data between different schemata. In addition, technologies from the World Wide Web provide the fundamental

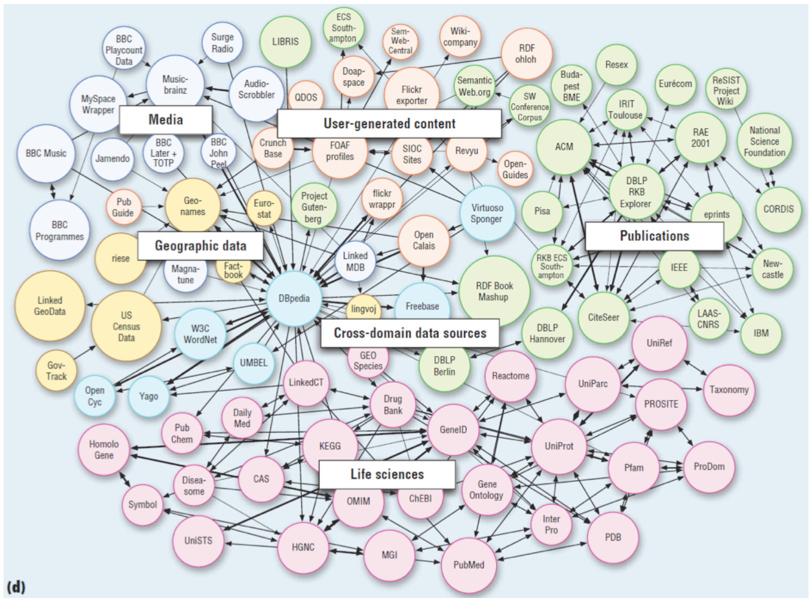


Fig. 1. Overview of Linking Open Data sources as of July 2009 with source types shown

infrastructure: Uniform Resource Identifiers (**URIs**) are used to provide globally unique identifiers for the data, and the HyperText Transfer Protocol (**HTTP**) is used for accessing and transporting the data.

In order to build a single Web of Data, all data providers have to follow the same guidelines for publishing their data and connecting it to other data sources. These guidelines are provided by the **"Linked Data principles"** [10], which specify how to use the different standards of the Web of Data together:

1. Use URIs as names for things (and e.g. persons, places).
2. Use HTTP URIs so that people can look up and access those names via HTTP.
3. When someone looks up a URI, provide useful information, using the standards RDF and SPARQL.
4. Include links to other URIs, so that data about more things can be discovered.

The Linked Data principles have been adopted by an increasing number of data providers, especially from the Linking Open Data community project[1] (see Figure 1), which makes free and public data available as Linked Data. As of November 2009 this includes 13.1 billion RDF triples which are interlinked by 142 million RDF links.

The nucleus of the Linked Data cloud is formed by **DBpedia**, which extracts RDF from wikipedia topic pages, and thus provides authoritative URIs and RDF data about topics from any domain. Figure 2 shows how DBPedia makes use of different vocabularies to describe a resource.

[1] `http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/`

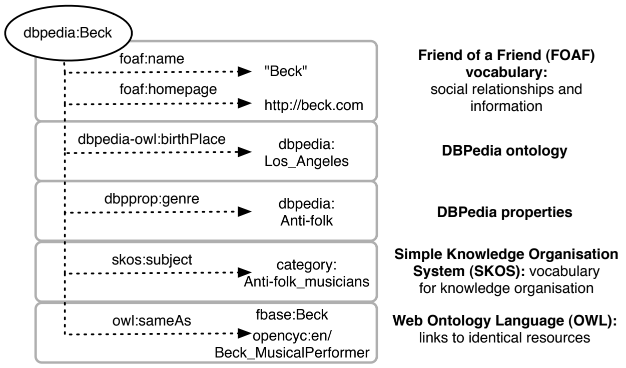

dbpedia:Beck
foaf:name
"Beck"
foaf:homepage
http://beck.com
dbpedia-owl:birthPlace
dbpedia: Los_Angeles
dbpprop:genre
dbpedia: Anti-folk
skos:subject
category: Anti-folk_musicians
owl:sameAs
fbase:Beck
opencyc:en/ Beck_MusicalPerformer
Friend of a Friend (FOAF) vocabulary: social relationships and information
DBPedia ontology
DBPedia properties
Simple Knowledge Organisation System (SKOS): vocabulary for knowledge organisation
Web Ontology Language (OWL): links to identical resources

Fig. 2. Example of a DBPedia resource with descriptions of the different vocabularies used for the properties

The following types of Linked Data sources provide data which could be used for mining experiences:

Major **search engines** such as Google and Yahoo have started to index RDF meta-data which is embedded in web pages. This can be directly used to expose structured data about user experiences and reviews to the search engines. Yelp and Amazon for instance are providing this data, so that e.g. the average rating for an item can be already displayed as part of the search results. **Social Web sites** such as LiveJournal make user generated content from their users available. This data is modelled after the principle of object centred sociality [19], connecting users indirectly into communities via objects of a social focus, such as musical artists.

E-commerce sites use the Good Relations vocabulary to describe their products and their features and prices, payment options, as well as store locations and opening hours. BestBuy has released such data about all of their stores in the US. **Broadcasters and news publishers** provide data about media content and usage. The DBTune project makes data about MySpace users and their connections to musical artists available. Other sources include the BBC's catalogue of broadcasts on TV and Radio, as well as play-count data for different artists played across all of the BBC stations.

4 A case-based view on the Web of Data

While CBR suffers less from the knowledge elicitation problem, it is still not immune from it. In this section we examine how Linked Data can help bootstrap the development of decentralised case-based systems. Customisation of each CBR system for a specific domain is possible via the selection of sources and instances from these sources and through the assignment of data to the four 'containers' which hold the knowledge used in a CBR system (as defined by Richter [20]):

1. **Vocabulary knowledge** consists of the semantic components that can be manipulated by a reasoning system.

2. **Case knowledge** consists of the ‘problem’ episodes or instances represented as cases that can be used to solve similar problems in the future.
3. **Similarity knowledge** represents the similarity measures which are used to match cases in a particular domain.
4. **Adaptation knowledge** is knowledge used to adapt the solution of the matching case for the target problem.

Capturing the knowledge for these containers represents a challenge in every CBR system because of the knowledge engineering costs required in their creation and curation. Vocabulary knowledge may need to be extracted and ordered, case structures may need to be designed and features selected, similarity measures may need to be designed and tested and adaptation strategies modelled. We propose that Linked Data can be used as a knowledge source to ease the creation of these containers in many CBR systems.

Pre-existing vocabulary knowledge is contained in the vocabularies and ontologies already deployed on the Web of Data. Some vocabularies just include class definitions and relations between classes, such as the Semantically-Interlinked Online Communities (SIOC) vocabulary[2]. Then there are topical hierarchies, such as the taxonomy of DBpedia categories. Finally there general knowledge ontologies expressed in OWL such as OpenCyc[3].

Case knowledge is provided by the different data sources which are part of the Linking Open Data cloud, and which have been discussed in the background section. We will show detailed examples of using LD for case knowledge in section 6.

Adaptation knowledge still needs to be engineered for the specific domain of the CBR system. [17] gives an example of how OWL can be used to model adaptation logic. While similarity knowledge is currently not available as linked data, Coyle et al. [6] show how this knowledge can be modelled in XML, and this approach can be adapted for RDF and Linked Data.

5 CBR Vocabulary

In order to use Linked Data for case based reasoning, we are proposing a new, light-weight CBR vocabulary to provide a CBR view on Linked Data. Figure 3 shows the full vocabulary on the left hand side. While CBML [21] and CaseML [15] have attempted to define all of the objects and properties which are broadly associated with CBR, we will just focus on the few main classes and properties which are unique to CBR: cases, solutions and case bases.

This very simple vocabulary allows us to create cases by linking to different objects, properties and concepts from different data sources and vocabularies from the Web of Data. This gives us the ability to reuse case snippets from a massive and increasing data store which includes many different communities and domains. This allows us to build CBR systems on top of all this newly available data.

– We define the classes `cbr:Case` and `cbr:Solution` in order to allow a differentiation between a query and potential results or recommendations. Note that a resource can be an instance of both classes at the same time.

[2] http://rdfs.org/sioc/spec/
[3] http://sw.opencyc.org/

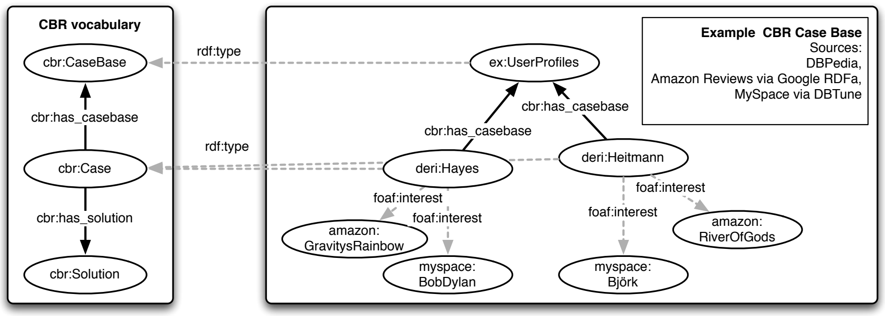


Fig. 3. The CBR vocabulary and a case base with experiential data for user profiling. Properties from the CBR vocabulary are displayed with solid lines, all other properties are displayed in dashed lines.

- Cases are connected to solutions via the property `cbr:has_solution` which can have cases as subject and solutions as object (`cbr:has_solution` has `rdfs:domain cbr:Case` and `rdfs:range cbr:Solution`). Note that usage of the `cbr:has_solution` property is optional, as not every CBR setting requires a solution to be designated in advance.
- Cases are grouped together via the `cbr:CaseBase` class, which can be used to represent the different types of knowledge available to a CBR system.
- Cases are associated to case bases via the property `cbr:has_casebase`, which has `rdfs:domain cbr:Case` and `rdfs:range cbr:CaseBase`.

Modelling decisions: As the Web of Data is an open ecosystem, applications should not make assumptions about the vocabulary used by data sources and they should take frequent changes of sources into account [11]. This has influenced several modelling decisions for our CBR vocabulary: (a) **intentional simplicity** characterises our vocabulary and lowers the barriers for reusing and integrating external data. This leads us to model problems as cases, in order to simplify reuse of external domain data. (b) **reuse of existing properties** and data types, as we leave the definition of the properties which represent the actual features of the data to externally defined vocabularies and ontologies, as appropriate to the domain of a source. (c) **simplified inheritance** and inferencing, as the formal RDF semantics follow the open world assumption, which is only superficially related to object oriented programming. Resources are instances of multiple classes as defined by the properties which the use. Consider as an example, that by using the `cbr:has_solution` property, its subject automatically becomes a `cbr:Case` and its object becomes a `cbr:Solution`.

6 Example: a case base with experiential data for user profiling

In this example we show how to create an example case base by using our CBR view on the Web of Data. This involves three steps: (1) discovering and aggregating relevant data, (2) conversion of external data to a common format, and (3) authoring of the case base. The scenario for the example case base is to aggregate data for user profiles and to use the profiles for personalised recommendations.

First step: discovering and aggregating relevant data. In order to discover relevant data, different approaches can be used [22]: SPARQL queries to specific data sources, Linked Data search engines such as `Sindice.com` or a custom built crawler. The experiential data for our example comes from Amazon.com reviews and MySpace user profiles. We access the Amazon data via the `Sindice.com` search service, the MySpace data is accessed via the SPARQL endpoint provided by the DBTune.org project.

Second step: conversion of external data. After discovering the data, it needs to be converted to RDF as a common format. Amazon exposes its data about reviews as RDFa in its HTML pages. This data can automatically be converted to RDF, see [22] for details. Listing 1.1 shows the data with one review of the book "River of Gods", obtained because the user Benjamin Heitmann wrote the review. Listing 1.2 shows data from MySpace about the musician "Bob Dylan", obtained because user Conor Hayes lists him as a favourite artist. `DBTune.org` provides a SPARQL endpoint which directly returns RDF data.

Listing 1.1. A book review from Amazon

```
amazon:RiverOfGodsReview1 review:itemreviewed "Book: River of Gods, Author: Ian McDonald" .
amazon:RiverOfGodsReview1 review:summary "AIs construct a black hole to escape the AI police" .
amazon:RiverOfGodsReview1 review:rating "5" .
```

Listing 1.2. Data about a musical artist from MySpace

```
myspace:BobDylan rdf:type http://purl.org/ontology/mo/MusicArtist .
myspace:BobDylan foaf:name "Bob Dylan" .
myspace:BobDylan myspace:genreTag http://purl.org/ontology/myspace#Folk%20Rock .
```

Third step: authoring of the case base. After aggregating and converting external data, the entities which should be part of the case base can be selected. This can be either done manually or automatically through the domain logic of the application.

Figure 3 shows the final graph of the example case base. Only the solid lines represent properties from the CBR vocabulary. These properties are created through associating the entities representing the users `deri:Hayes` and `deri:Heitmann` to the case base during the authoring phase. Listing 1.3 contains all the RDF that is required to define the example case base.

Listing 1.3. RDF triples for the example case base

```
deri:Hayes cbr:has_casebase ex:UserProfiles .
deri:Hayes foaf:interest myspace:BobDylan .
deri:Heitmann cbr:has_casebase ex:UserProfiles .
deri:Heitmann foaf:interest amazon:RiverOfGodsReview1 .
```

7 Discussion

In previous work [23], we have shown how Linked Data can be used to ease the bottleneck in knowledge acquisition for recommender systems based on collaborative filtering. In a certain sense, we are now proposing to use Linked Data for Case Based Reasoning out of the same reasons: We are proposing new ways to lower the entry barriers to implementing and deploying CBR systems, by using Linked Data as a short-cut for creating the case base, and for lowering the expense of the required knowledge engineering.

In our example, we show how a user profile can be developed by selecting objects and associated properties and relations from structured data sources currently existing on the Web. On the DBpedia data source alone, rich data exists for at least 3.4 million things, out of which 1.5 million are classified in a consistent Ontology, including 312,000 persons, 413,000 places, 94,000 music albums, 49,000 films, 15,000 video games, 140,000 organisations, 146,000 species and 4,600 diseases. The DBpedia data has meta data and abstracts for these 3.4 million entities in up to 92 different languages; It has 1,460,000 links to images and 5,543,000 links to external web pages; 4,887,000 external links into other RDF datasets, 565,000 Wikipedia categories, and 75,000 YAGO categories. [4]

Such data suggest the possibility of collecting rich experience trails of users and then using these to find similar profiles or to recommend objects of different types. While this is probably the easiest example we can envisage, there are rich possibilities for reasoning on case views of this data and for integrating case-based reasoning as a standard reasoning paradigm for Semantic Web data.

Limitations: In this paper we focus on case and vocabulary knowledge, as it is readily available for many different domains on the Web of Data today. However, we do not address approaches for engineering of adaptation and similarity knowledge, though we believe that here too knowledge costs can be reduced by examining the pre-existing domain concepts and relations on the Web of Data. We also do not discuss the different reasoning approaches which can be used as part of the CBR process. Instead we focus on acquiring the data which is required as the basis for the reasoning process in the first place. Finally, while the Web of Data provides many different sources of experiential data, this also can incur an implementation overhead as obtained data might be noisy or inconsistent.

8 Conclusion

In this paper we have suggested that there is an open opportunity for reasoning with cases of experiential data from the emerging Web of Data. Previous attempts to standardise case representation have failed because of a lack of real data and context for distributed case-based reasoning. The Web of Data now presents such a context. We provide an overview of the Linked Data principles and describe how Linked Data can facilitate the authoring of experiential case data. We present a realistic example of how case-based user profiles can be authored from data currently available from multiple sources. Such cases have the advantage of having rich features, allowing for many different types of matching and retrieval methods. While our example is simple, it demonstrates the rich possibilities for reasoning on case views of this data and for integrating case-based reasoning more extensively within Semantic Web applications.

Acknowledgements: The work presented in this paper has been funded by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Líon-2).

[4] These statistics are taken from `http://en.wikipedia.org/wiki/DBpedia`

References

1. Kitano, H., Shimazu, H.: The Experience Sharing Architecture: A Case Study in Corporate-Wide Case-Based Software Quality Control. In, Case-Based Reasoning: Experiences, Lessons, & Future Directions. Leake, DB (Ed.) pp. 235-268 (1996)
2. Hayes, C., Cunningham, P., Doyle, M.: Distributed CBR using XML. In: Proceedings of the KI-98 Workshop on Intelligent Systems and Electronic Commerce, Citeseer (1998)
3. Shimazu, H.: A textual Case-Based Reasoning system using XML on the World-Wide Web. Advances in case-based reasoning (1998) 274–285
4. Watson, I., Gardingen, D.: A distributed case-based reasoning application for engineering sales support. In: International Joint Conference on Artificial Intelligence. Volume 16., Citeseer (1999) 600–605
5. Hayes, C., Cunningham, P.: Shaping a CBR view with XML. (Case-Based Reasoning Research and Development) 722–722
6. Coyle, L., Doyle, D., Cunningham, P.: Representing Similarity for CBR in XML. (Advances in Case-Based Reasoning) 155–164
7. Coyle, L., Hayes, C., Cunningham, P.: Representing cases for cbr in xml. Expert Update **6**(2) (2003) 7–13
8. Plaza, E., McGinty, L.: Distributed case-based reasoning. The Knowledge engineering review **20**(03) (2006) 261–265
9. Berners-Lee, T., Hall, W., Hendler, J., O'Hara, K., Shadbolt, N., Weitzner, D.: A framework for Web Science. Foundations and Trends in Web Science **1**(1) (2006) 76–77
10. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data–the story so far. Journal on Semantic Web and Information Systems (2009)
11. Oren, E., Heitmann, B., Decker, S.: ActiveRDF: embedding Semantic Web data into object-oriented languages. Journal of Web Semantics (2008)
12. Kolodner, J.: Retrieving events from a case memory: A parallel implementation. In: Proceedings of a Workshop on Case-Based Reasoning. Volume 233. (1988) 249
13. Barletta, R., Mark, W.: Breaking cases into pieces. In: Proceedings of Case-Based Reasoning Workshop. (1988) 12–17
14. Redmond, M.: Distributed cases for case-based reasoning: Facilitating use of multiple cases. In: Proceedings of AAAI. Volume 90. (1990) 304–309
15. Chen, H., Wu, Z.: On Case-based Knowledge Sharing in Semantic Web. In: International Conference on Tools with Artificial Intelligence. (2003) 200–207
16. Bichindaritz, I.: Mémoire: Case Based Reasoning Meets the Semantic Web in Biology and Medicine. In: European Case Based Reasoning Conference. (2004)
17. d'Aquin, M., Lieber, J., Napoli, A.: Decentralized Case-Based reasoning for the Semantic Web. International Semantic Web Conference (2005) 142–155
18. Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., Horrocks, I.: The semantic web: The roles of XML and RDF. IEEE Internet computing **4**(5) (2000) 63–73
19. Bojars, U., Passant, A., Cyganiak, R., Breslin, J.: Weaving SIOC into the Web of Linked Data. In: Linked Data on the Web Workshop. (2008)
20. Richter, M.: The knowledge contained in similarity measures. Invited Talk at ICCBR **95** (1995)
21. Hayes, C., Cunningham, P.: Shaping a CBR View with XML. In: International Conference on Case-Based Reasoning. (1999) 468–481
22. Heitmann, B., Kinsella, S., Hayes, C., Decker, S.: Implementing semantic web applications: reference architecture and challenges. In: Workshop on Semantic Web Enabled Software Engineering. (2009)
23. Heitmann, B., Hayes, C.: Using linked data to build open, collaborative recommender systems. In: AAAI Spring Symposium: Linked Data Meets Artificial Intelligence'. (2010)

On Retaining Web Search Cases

David Leake and Jay Powell

Computer Science Department, Indiana University
Bloomington, IN 47405, U.S.A.
{leake,jhpowell}@cs.indiana.edu

Abstract. Web mining enables CBR systems to draw on rich knowledge sources to supplement their internal knowledge, aiding performance of challenging tasks such as knowledge-rich case adaptation. However, the very richness of Web sources presents a potential problem for system efficiency, because of the amount of information the search process may need to navigate or filter before finding desired results. The potential cost of Web search from scratch suggests applying a CBR solution, by capturing Web mining cases to expedite future searches. However, this in turn raises questions such as how to control the growth of the search case base. This paper explores tradeoffs in retaining Web search cases, in the context of Web mining to support case adaptation, as applied in the WebAdapt system. The paper tests tradeoffs for relying entirely on internal or external knowledge and proposes a selective retention strategy aimed at retaining only "hard" searches addressing new search constraints.

1 Introduction

In case-based reasoning, Web CBR has become an active area pursuing many facets of CBR and the Web (see [2, 1, 5] for some examples), and interest in CBR to guide the search for information dates to the early days of the field [3]. One of the opportunities of the Web is the availability of rich Web knowledge sources, which has prompted optimism in the artificial intelligence community for a new generation of knowledge-rich AI systems [6]. In particular, exploiting Web knowledge may help to alleviate the knowledge acquisition bottleneck for the knowledge supporting CBR processes such as case adaptation. Some promising results have already been achieved in this area [4, 8, 9, 11, 7]. However, important tradeoffs must be addressed for initial research prototypes to scale up to handle very large data sources. A key concern is the costs associated with searching the Web from scratch.

A natural approach to controlling the cost of large-scale Web search is to apply case-based reasoning: to capture relevant search cases for future use, bypassing the need to repeat searches in similar situations. However, the efficiency gained by this approach has two potential tradeoffs. The first is timeliness of results. In rapidly-changing domains, the information in Web sources may be undergoing updates at various frequencies—from up-to-the-minute updates for news sites to much longer time scales for general references. Thus the decision of whether to store search results for future use depends partially on reasoning about whether the possible loss of recency counterbalances the benefits for processing speed. This decision depends on factors specific to the task domain and individual knowledge sources.

The second major tradeoff is space: Capturing search cases imports knowledge from an external source to an internal one, bringing with it issues in controlling case-base growth. Case-base growth issues have been extensively studied in the case-base maintenance literature, which has developed strategies for compacting case bases and guiding retention of cases (see [10] for an overview). In this paper we investigate a retention policy for Web search, in which the constraints guiding a search (categories to which a substitution must belong) are used as an abstract characterization of the search's knowledge goals, in order to retrieve a subset of search cases providing unique coverage of a set of constraints.

The remainder of the paper reviews the WebAdapt system, assesses the efficiency of Web search from scratch for this system, presents its case retention policy, and presents experiments testing the strategy's performance compared to baseline approaches.

2 WebAdapt's Adaptation Process

This section presents a summary of the WebAdapt framework, condensed and adapted from [8]. WebAdapt acquires the knowledge necessary to perform substitution adaptations by mining external sources such as Wikipedia and OpenCyc. Informally, WebAdapt's task is to choose the best source to mine for a given problem and generate a plan capable of searching the chosen source for adaptation-relevant knowledge. To test the generality of the methods, the system has been applied to three domains: travel itinerary planning, menu planning, and software selection.

WebAdapt's adaptation process is shown in Figure 1 and will be described using an example. Consider the scenario in which a user is viewing a travel itinerary and requests an alternative for the Louvre, due to reports of long lines. When the user makes the request for a substitution, WebAdapt generates the goal *Find ranked candidate substitutions for "Louvre,"* which describes the knowledge required to satisfy the user's request (step 1 in Figure 1). WebAdapt then selects a Web source to search for candidate substitutions for *Louvre* (step 2). It does so by retrieving a set of cases from its dispatch case base indicating which sources have previously presented candidate substitutions for *Louvre.* These sources are then ranked based on their source profiles which contain basic statistics describing each source. For illustration, we will assume WebAdapt selects the Geonames GIS database.

WebAdapt then attempts to retrieve a relevant plan from its search plan case base (step 3). If no plan is found, one is generated from scratch using a partial-order planner. The retrieved plan will contain low-level knowledge goals describing the intermediate knowledge necessary for execution such as: (1) *find a Geonames entry for "Louvre,"* (2) *hypothesize constraints which must be satis-*

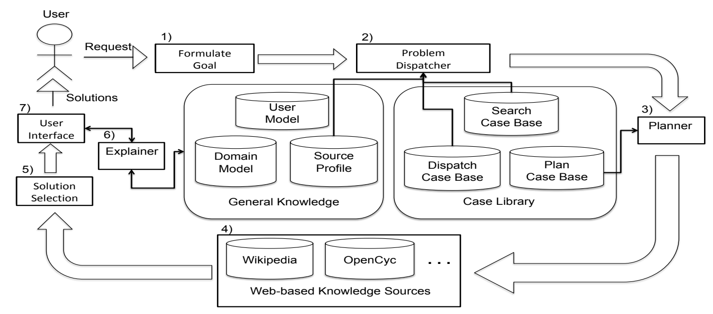


Fig. 1. WebAdapt's adaptation process and knowledge sources, from [8].

fied by a substitution for "Louvre," (3) *find unranked candidate substitutions for "Louvre."* Operators satisfying each sub-goal are used to search the Geonames abstraction hierarchy and return a set of candidate substitutions. During the mining process (step 4), the portion of the Geonames abstraction hierarchy that is traversed is stored in WebAdapt's domain model. Candidate substitutions are then retrieved from Geonames (step 5) and presented to the user for selection.

Knowledge Planning Operators: WebAdapt's operators describe search and transformation operations for interacting with Web sources and modifying existing cases. They specify the knowledge necessary for execution as preconditions and the knowledge acquired during execution as post-conditions, defined in a vocabulary of roles filled during the planning process. WebAdapt's role-fillers are frames describing structured knowledge retrieved from sources, such as URL addresses or database records. Examples of its operators can be found in [8].

WebAdapt's Internal Knowledge: WebAdapt relies on several sources of knowledge to guide its adaptation process.

Search Case Base: The search case base contains solutions to problems from several domains (e.g., travel itineraries and recipes) and provenance knowledge indicating how solutions were derived.

Source Profiles: WebAdapt stores statistical information characterizing each source's performance and content. This information is updated each time a source is accessed and is used to guide source selection. WebAdapt tracks each source's average access times and uptimes, as well as the percentage of queries a source has satisfied. Each source's diversity is estimated using a method similar to that descried in [12].

Cases for Dispatching and Adaptation: WebAdapt also relies on a case-base describing the results of prior adaptation episodes to influence source selection. After an adaptation episode has completed, WebAdapt creates a new dispatch case containing: (1) the adapted item, (2) the knowledge sources searched, (3) any hypothesized constraints discovered, and (4) a Boolean value indicating

Table 1. Query method, hierarchy nodes, and leaves for each source

Source	Query method	Nodes	Leaves
Wikipedia	Google[1]	Categories	Entries
OpenCyc	Cyc Query Engine	Collections	Individuals
Geonames GIS database	SQL Query	Feature class	Locations
USDA SR20 database	SQL Query	Food group	Foods
Widget downloads	Google[2]	Widget category	Widgets

whether any candidate substitutions were generated. Dispatching cases indicate which sources are capable of satisfying particular problems.

Domain Model: The domain model stores a snapshot of the portions of the external sources WebAdapt has searched. Nodes and leaves discovered during the mining process are added to the domain model along with the links between constraints and candidates that have been traversed. WebAdapt's domain model can be used as an alternative to searching external knowledge sources.

WebAdapt's External Knowledge Sources: The external sources leveraged by WebAdapt are defined by explicit abstraction hierarchies, which WebAdapt traverses to find viable substitutions. Nodes in the abstraction hierarchy are viewed as constraints while leaves are candidate substitutions. Table 1 shows examples of constraints and substitutions from each knowledge source, as well as the methods used to process each source. WebAdapt exploits a source by first discovering the item to adapt in a source's hierarchy then expanding reachable nodes to discover candidate substitutions. WebAdapt can utilize any source with a hierarchical knowledge organization, leading to a system capable of leveraging knowledge from multiple domains. To leverage a source, a small set of basic operators for traversing the hierarchy must be defined and provided to the system.

3 Web Search Case Retention

As is illustrated by experiments in Section 6, Web searches can be expensive—in some cases, taking several minutes—but unrestricted search case retention causes rapid growth of the search case base. Consequently, WebAdapt applies a selective search case retention process which is initiated whenever a new adaptation has been completed (i.e., when the user has selected a candidate substitution). At that point, the retention process is applied to all searches conducted to find that substitution.

Triggering retention: Whether to retain a case is based on comparison of the search characteristics for the current search to data WebAdapt collects summarizing its performance during prior reasoning episodes. These data include: (1) the average duration and standard deviation of duration for problem solving

[1] e.g., *site:en.wikipedia.org "Louvre Paris France"*
[2] e.g., *site:www.apple.com/downloads/dashboard "iTunes Widget"*

episodes, and (2) the average number of failures generated per episode and the standard deviation. After a reasoning episode has completed, WebAdapt considers the duration, number of failures generated, and the reliability of the sources used (drawn from WebAdapt's source profiles).

WebAdapt uses two types of criteria for case retention. The first type considers characteristics of the current search. In general, we expect retention to be most worthwhile for longer searches, searches which were difficult to conduct, and searches for which good results were obtained from a generally unreliable source (in which situation, storing the case is valuable to avoid using a different source in the future and missing re-generating the result). Specifically, a case is a candidate for retention when any of the following are satisfied: (1) the duration of the most recent episode is at least one standard deviation above the average duration, (2) the number of failures generated is at least one standard deviation above average, or (3) the source reliability is below some threshold. We note that each of factors (1), (2), and (3) could be weighted in other ways, based on the specific task. The second type considers how the case affects coverage of the case base of search cases, as described in the following section.

Execution: After case retention has been triggered, WebAdapt adds the new case to its search solution case base. Provenance information is stored relating the adapted case (the parent) to the solution case (the child). The adapted item and solution are stored in the provenance trace to indicate how the child relates to the parent. A variety of other information is also captured in the provenance trace, including: (1) why a case was captured, (2) the context in which a case was captured, and (3) how a case was captured, as explained in the following.

Component (1) *why a case was captured*, summarizes the conditions that triggered case retention and includes the duration of the problem solving episode, the number of failures generated, and the reliability of the mined source.

Component (2), *the case capture context*, describes the set of constraints describing the selected substitution in the child case. Consider the scenario where a user replaces *chicken* in a recipe with the vegan meat substitute *seitan*. The provenance trace for this case would be annotated with the constraints *Vegan cuisine* and *Meat substitutes* if the case were derived using Wikipedia.

Component (3), *how a case was derived*, includes the knowledge goals that lead to the child case and the set of sources mined. This knowledge can be used to replay the derivation of the child case as described in Section 5.

4 Controlling Case Base Size by Storing Only Cases with New Constraints

Case retention could simply store a solution case for each specific Web search. However, we can further control case base size by only storing a case if it concerns new constraints. To explain this distinction between cases to discard or retain, we note that WebAdapt's search space can be described by two components, the *candidate substitution* space and the *constraint* space. The candidate substitution space is the set of all substitutions that must be considered, and the constraint space is the set of all constraints (including sub-constraints) that are mined during a search. Some of these search spaces may be large. For example, when mining Wikipedia, *Seine River* is defined by sixty-one constraints (including sub-constraints) and 2,759 candidate substitutions, while *tofu* is defined by eighty-seven constraints and 2,766 candidate substitutions.

Adding cases covering the candidate substitution space of an item can lead to a very large case base. However, a large set of candidate substitutions can be recreated efficiently by expanding a small set of constraints in WebAdapt's domain model. From this observation, WebAdapt can limit the size of its case-base by adding child cases described by unique constraints. Later, when using a child case to retrieve candidate substitutions from the case base, candidates described by a child case's context are re-ordered to appear near the front of the list of candidate substitutions, helping to focus search.

For example, when retrieving a child case suggesting the solution *seitan* for a parent containing *chicken*, the constraints describing the context of the child case are *Vegan cuisine* and *Meat substitutes*. When future candidate substitutions are requested for *chicken*, such options may be preferred during search. WebAdapt also does not need to add new cases to the case base with solutions described by either *Vegan cuisine* or *Meat substitutes* because the child case can be used to re-derive similar items. However, if the user selected the substitution *ground beef*, that would not be covered by the stored constraints, and consequently the retention policy would store a new case.

5 Reusing captured cases

Once search cases have been captured, the question arises of how they should be reused. There are three general approaches: (1) always rely on external knowledge, (2) always rely on local knowledge (when it is available), and (3) selectively choose between local and external knowledge.

There are tradeoffs associated with each approach. When a system uses approach (1) it will always have access to the most up-to-date knowledge. However, this approach is prone to performance bottlenecks, such as unreliable network connections or large search spaces that are time consuming to mine. The second approach is more efficient, though the system developers may be burdened with the task of identifying and updating obsolete local knowledge. The third approach offers a balance between the other two.

WebAdapt adopts a selective approach, as follows. After a user requests a set of candidate substitutions, the system formulates a set of knowledge goals and chooses a source to mine. After a source has been chosen WebAdapt checks its search solution case base for all cases containing items similar to the one being adapted. WebAdapt searches for similar items by first checking its domain model to determine the constraints describing the item being adapted. It then examines each case and uses the domain model to determine the constraints for each case feature. Case features sharing a certain percentage of constraints (e.g., 70%) with the item being adapted are retrieved.

WebAdapt then uses its provenance knowledge to discover child cases that (1) suggest substitutions for the item to adapt, and (2) were derived using similar knowledge goals. For example, if the user requests a substitution for the Eiffel Tower, it retrieves all cases containing items similar to *Eiffel Tower*, then checks the provenance traces of this set of cases for children containing substitutions for *Eiffel Tower*. If no appropriate child cases are found WebAdapt continues to use the suggested external knowledge source.

If appropriate child cases are found, then WebAdapt decides whether to use internal knowledge or external Web mining based on the expected cost and failure rate for external mining. The system uses knowledge described in each child case's provenance trace to determine if the system should rely on local knowledge. The system takes the average access time and average failure rates described in each provenance trace. If the average duration or failure rate is one standard deviation greater than the system's current average then local knowledge is used. If the reliability of the sources used in the provenance traces are less than some threshold then local knowledge can be used as well.

The set of provenance traces typically contain similar knowledge goals. WebAdapt can then retrieve a plan satisfying its current knowledge goals from its plan case base. Because WebAdapt's domain model contains a snapshot of the abstraction hierarchies of each knowledge source, the retrieved plan can be adapted to mine WebAdapt's domain model rather than the chosen knowledge source. WebAdapt does not simply use the set of derived cases to suggest substitutions as this set is often small (i.e., only a few cases). To present a wider range of recommendations, WebAdapt draws on its domain model to present similar candidate substitutions. It also draws on the constraints describing the context of each derived case. These constraints can be used to re-order candidate substitutions to more closely match derived solutions (i.e., when suggesting substitutions for *chicken*, prefer vegetarian options).

6 Experiments

To evaluate WebAdapt's case retention policy, we considered two questions:

1. *Performance Impact:* How do alternative strategies affect WebAdapt's reasoning times?
2. *Case Base Growth:* How do alternative strategies affect the growth of WebAdapt's case base?

For each question, an experiment was run using a combination of problems from three domains: travel planning, menu planning, and software recommendation. For each experiment 49 items were adapted 5 times each, for a total of 245 adaptations. Adaptations were repeated 5 times to average out inconsistencies in Web access times.

Experimental Design for Question 1: The experiment for Question 1 compared three alternative strategies: (1) always use external knowledge, (2) always

Table 2. Performance improvements. All times measured in seconds.

Method	Median	Average	Longest	Reliance on external sources
External knowledge only	12	59	514	100%
Internal knowledge only	1	6	44	0%
Selective strategy	5	17	73	85%

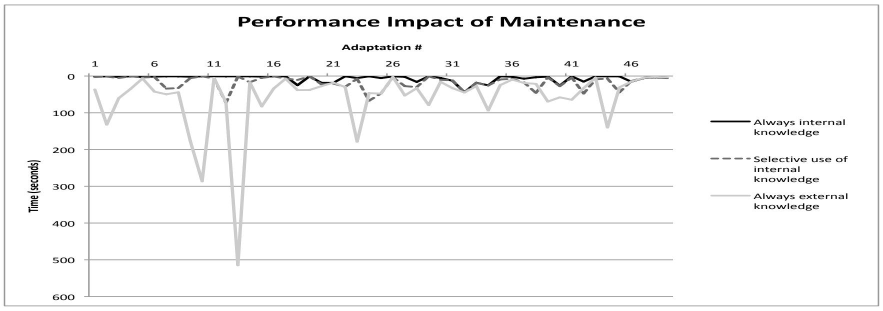


Fig. 2. Average mining time per problem for all configurations

use internal knowledge, and (3) selectively choose when to use internal knowledge. For (2), WebAdapt was provided with a pre-trained memory and adaptation case base, analogous to the results built up in (1). For (3), WebAdapt built an adaptation case base from scratch as it solved problems, and referred to stored adaptations only when the predicted cost of mining external sources exceeded a threshold.

Results for Question 1: Table 2 shows, for each strategy, the average search/-mining times, duration of the longest problem solving episode, and percentage of problems dispatched to Web-based resources. When only using external knowledge, the average mining time was approximately one minute, while the longest was nearly eight minutes. The average time decreases significantly when only internal knowledge is used. The longest problem solving time when using internal knowledge comes from a problem described by a large number of constraints, for which each constraint describes a large number of candidate substitutions. The selective strategy (3) uses external knowledge 85% of the time. However, because it predicts which external searches are likely to be expensive, only using internal knowledge the remaining 15% of the time still results in a drastic speedup over only using external knowledge.

Figure 2 shows the average time per adaptation and maximum time to find the knowledge needed for an adaptation. When only external knowledge is used several problems take several minutes to solve. The internal method is faster for all problems but one. For this problem, the selective method consistently chose an external source with a single constraint, while the internal method consistently mined multiple constraints describing a large number of items.

Experimental Design for Question 2: The second experiment compared the growth rate of WebAdapt's case base for two approaches to case retention: (1) always retain unique cases, and (2) retain a case if it increases the case base's coverage.

The first method adds a case if that case has unique features, attempting to cover the entire candidate substitution space of a problem. The second method only adds a case if that case added a unique constraint to the original case's provenance trace. This method attempts to cover the entire constraint space for a problem.

Results for Question 2: Figure 3 shows the growth rates of the case base for each method. When adding cases with unique features, the case base growth rate is nearly linear. Each knowledge source is capable of presenting dozens or hundreds of candidate substitutions for each adaptation, potentially leading to extremely large case bases. The growth of the second method is nearly linear for the first fifty-five adaptations, after which it starts to level off as constraints are re-encountered. The case base continues to grow slowly because some knowledge sources, such as OpenCyc and Wikipedia, are described by constraints which form a compact vocabulary. The constraint space method generated a total of 397 cases compared to 625 for the candidate substitution method. Both methods had the same coverage.

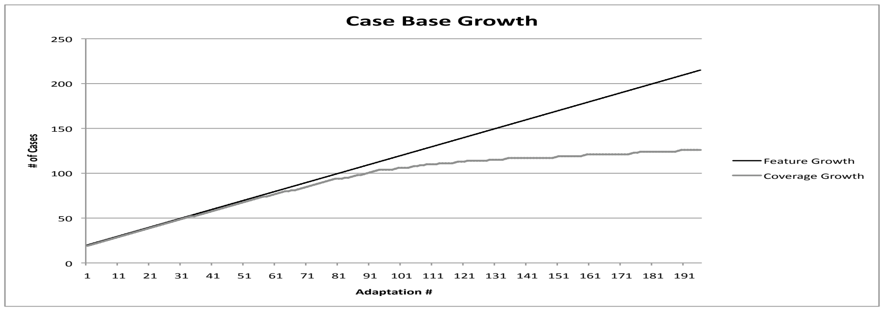


Fig. 3. Case base growth

Discussion When only using external knowledge, WebAdapt's mining times are generally reasonable, though a number of problems took three or more minutes to solve. Using only internal knowledge provides the best speed, but static internal knowledge risks becoming outdated if external sources change, and may sacrifice the ability to find specific solutions available in external sources. By learning to use internal knowledge only when the expected external mining time is long, WebAdapt can markedly increase its overall speed while still using external knowledge for 85% of its adaptation requests.

Each knowledge source's candidate substitution space is much larger than its constraint space. Adding cases based on their coverage of the constraint space leads to much smaller case bases. The constraints can then be used to reach candidate substitutions similar to relevant prior solutions.

7 Conclusions

When mining Web sources to support CBR processes, there is a tradeoff between the benefits of just-in-time Web mining and applying CBR to capture search cases and speed up search. This paper has explored those tradeoffs and proposed a balanced retention/reuse approach which controls growth of the search case base and increases efficiency while retaining the ability to reason largely from up-to-date external information.

References

1. Balfe, E., Smyth, B.: A comparative analysis of query similarity metrics for community-based web search. In: ICCBR'05. pp. 63–77 (2005)
2. Bridge, D., Plaza, E., Wiratunga, N. (eds.): Reasoning from Experiences on the Web. ICCBR, Seattle, WA (July 2009), ICCBR 2009 WORKSHOP
3. Daniels, J.J., Rissland, E.L.: A case-based approach to intelligent information retrieval. In: SIGIR'95. pp. 238–245 (1995)
4. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case base mining for adaptation knowledge acquisition. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07). pp. 750–755. Morgan Kaufmann, San Mateo (2007)
5. Freyne, J., Smyth, B.: Further experiments in case-based collaborative web search. In: ECCBR'06. pp. 256–270 (2006)
6. Hendler, J.: Knowledge is power: A view from the semantic web. AI Magazine 26(4), 76–84 (2005)
7. Ihle, N., Hanft, A., Althoff, K.D.: Extraction of adaptation knowledge from internet communities. In: WebCBR: Reasoning from Experiences on the Web (ICCBR 2009 Workshop) (2009)
8. Leake, D., Powell, J.: Knowledge planning and learned personalization for Web-based case adaptation. In: Althoff, K., Bergmann, R., Minor, M., Hanft, A. (eds.) Advances in Case Based Reasoning: 9th European Conference. pp. 284–298. Springer Verlag, Berlin (2008)
9. Leake, D., Powell, J.: A general introspective reasoning approach to web search for case adaptation. In: Bichindaritz, I., Montani, S. (eds.) Proceedings of the Ninth International Conference on Case-Based Reasoning. Springer Verlag, Berlin (2010), in press
10. Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision, and retention in CBR. Knowledge Engineering Review 20(3) (2005)
11. Plaza, E., Baccigalupo, C.: Principle and praxis in the experience web: A case study in social music. In: WebCBR: Reasoning from Experiences on the Web (ICCBR 2009 Workshop) (2009)
12. Smyth, B., McClave, P.: Similarity vs. diversity. In: Case-Based Reasoning Research and Development: Proceedings of the Fourth International Conference on Case-Based Reasoning, ICCBR-01. pp. 347–361. Springer-Verlag, Berlin (2001)

Similarity Assessment through blocking and affordance assignment in Textual CBR

R.Rajendra Prasath[⋆] and Pinar Öztürk

Department of Computer and Information Science (IDI)
Norwegian University of Science and Technology (NTNU),
Sem Sælands Vei 7-9, NO - 7491, Trondheim, Norway,
{rajendra,pinar}@idi.ntnu.no,
http://www.idi.ntnu.no/~rajendra;
http://www.idi.ntnu.no/people/pinar;

Abstract. It has been conceived that children learn new objects through their affordances, that is, the actions that can be taken on them. We suggest that web pages also have affordances defined in terms of the users' information need they meet. An assumption of the proposed approach is that different parts of a text may not be equally important / relevant to a given query. Judgment on the relevance of a web document requires, therefore, a thorough look into its parts, rather than treating it as a monolithic content. We propose a method to extract and assign affordances to texts and then use these affordances to retrieve the corresponding web pages. The overall approach presented in the paper relies on case-based representations that bridge the queries to the affordances of web documents. We tested our method on the tourism domain and the results are promising.

1 Introduction

World Wide Web (WWW) is a massively distributed and decentralized medium for information and services, and also one of the most egalitarian discoveries of mankind in modern times. However, the use of the web technology to its maximum possible extent requires development of flexible and effective searching approaches. To this end, we propose an approach in which the web can be searched through case representations that capture plausible connections between users' queries and affordances of web documents.

This work presents an approach to document retrieval in the tourism domain, yet the underlying research objective is to develop a method for explication and capture of the *affordance* of documents that are available on the WWW. Gibson[3] introduced the term *affordance* to refer to the opportunities for action provided by a certain *object*. We suggest that web documents should similarly have affordances that refer to use-purposes of documents. The meaning of a content to the query of a user lies in its affordance. The question transforms then into how the affordance of documents are conveyed in textual format and can be extracted and represented in a reusable way.

[⋆] This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

A central idea of the presented method is that a document may contain information that matches different queries, each of which corresponds to an affordance. For example, a web document about *New Delhi* may provide information about the non-vegetarian restaurants, the bazaars, as well as the transportation within the city while the main focus may be on shopping. This would mean that most of the content revolves around bazaars, shopping malls, and the shopping norms (i.e., whether/where to bargain and when/where not). The main strategy of the approach is to divide the Web pages into text segments, determine affordance of each and use these to determine the information needs the documents afford. This may be considered as a special type of *tagging* technique, in the classical natural language processing terminology.

The use of Case Based Reasoning (CBR) in web context has attracted researchers for a while. For example, Limthan et al. [6] applied CBR in order to compose a complex web service from heterogeneous web services residing in different parts of the web, and [2] in order to search and select web services. Ha [4] investigated how the web usage data can be used to discover navigation patterns which, in turn, can be used to predict the user behavior. In this work, a web document has a corresponding case representation capturing information that bridges queries to documents through affordances. Query experiences are used to adjust the affordances of documents. This provides search-useful and re-usable information for future web searches. The key rationale of the proposed approach is based on two assumptions: *(i)* a web document embraces a number of text blocks each of which can be connected to one or several affordances, *(ii)* the web can be searched CBR-wise and relevance of a document can be judged on the basis of its affordance alignment with the current query case. Figure. 1 illustrates the proposed approach in comparison with the web search.

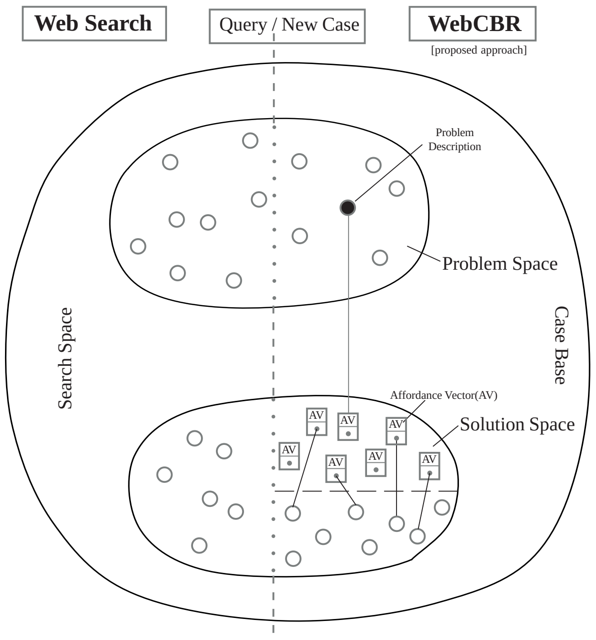


Fig. 1. Blocking and affordance assignment approach in WebCBR

The paper is organized as follows: Next section presents the idea behind the web-affordance notion. Section 3 explains the proposed case-based approach and the case population. Section 4 presents the experimental results and discussions. Finally, section 5 wraps up with conclusions and future directions.

2 The affordance-guided querying approach

A main rationale behind the proposed approach is the hypothesis that a document may serve more than one information need, that is, it may have multiple affordances. This because different parts (here we consider them as *blocks*) of a document may have slightly different focuses. We defined a list of topics in advance (manually at the moment) each of which fulfills an information need of the user. In tourism domain, a user may need information related to accommodation at a particular place, easy and economical transport options, places to roam around, time to travel from one place to another place, food / types of restaurants, historical / important places to visit, shopping at famous places, so on and so forth. Currently we have a list of 18 topics for the tourism domain. Each web page may afford to one or several of such information needs.

Affordance of a text segment/block is represented as a vector of which each element specifies the extent the text affords a certain information need in the affordance list for the task domain. The affordance of a whole document is determined on the basis of the affordance vectors (AV) of its constituent blocks. The size of an affordance vector, consequently, is m in the example (ie., tourism) domain (here m = 18).

The querying approach proposed in this paper relies on retrieving a number of documents relevant to the query using information retrieval (IR) techniques, and then employing an affordance-based ranking on this set. In the rest of the paper, we use the following notation:

$$AV_{\text{text}} = \{A_1, A_2, ..., A_m\} \tag{1}$$

where *AV* is a vector while A_i is a scalar representing the affordance with respect to the i^{th} element in the list of *m* affordances.

3 Case-Based, Affordance-Guided Web

The objective of this work is, given a query, to retrieve the web documents that meet the user's information need in the best possible way, using a refined assessment of the document guided by affordances. The underlying assumption is that the web is informed about the affordance of each document, in a case base. There is a case for each document of which the problem description part consists of a term-set that represents the document in a concise way, and the identification of the document it represents. The term-set is, in a sense, a dimension-reduced version of the web page. The solution part informs about the affordance of the document indirectly, through affordances of its constituent blocks. Hence a case is represented as follows:

C_i= $\{ProbDescription, Solution\}$

where *ProbDescription* consist of a term-set representation of the web page while *Solution* has two components:

$Solution$= $\{AV, I\}$

AV is the affordance vector of size *m* (see Equation 1) and represents the affordances of the web page. AV, together with *I* (which is the identification of the corresponding document), constitutes the *solution* part of a case. The next section describes how the case base is populated.

3.1 Population of the Case Base

Initially, case base contains no cases and cases are constructed incrementally in the off-line mode. To build the case base, we apply *link-to-text* ratio which is defined as the ratio between the size of the text tagged with hyperlinks and the text without hyperlinks.

Problem description: A web page is first segmented into blocks and a textual description of each block b_i is extracted (after removing the markups and stop words) using *link-to-text* ratio. If *link-to-text* ratio is more then, all hyperlinked text will be skipped and *link-to-text* ratio is scaled in the normalized interval [0-1]. Then from each extracted block text, top k discriminative terms are selected (after stop word removal) and added to *ProbDescription* in the problem description of the case. This process is described in the 'problem part' of Algorithm 1.

Solution: The extracted text, say *text*$_i$, of the block b_i is processed to identify its affordance with the help of the resource terms (topics and # terms considered in each topic, are presented in the table 1). For each topic, the matching terms are identified, the affordance with respect to this topic is computed and the affordance vector of the block is updated (see **ComputeBlockAffordance** in the Algorithm 2) accordingly. The AV of the document, in turn, is computed by **ComputeDocAffordance**, as described in the 'solution part' of the Algorithm 1. A case is generated by coupling the problem description and the solution parts and is added to the case base.

Algorithm 1 Population of Case Base

```
Input:  List of topic: $L_t$ (from table 1) ;
        A web document: $d$;
Procedure:
I: identification of $d$;
Problem Part:
 1: Initialize probDesc to NULL;
 2: for each block text data do
 3:     remove stop words and punctuation;
 4:     filter out top $k$ words from the block text and add it to probDesc;
 5: end for
Solution Part:
 1: $AV_d = \mathbf{ComputeDocumentAffordance}(d)$
 2: Soln = $< AV_d, I >$
Case Base:
 1: Add the new case having <probDesc, Soln> to the case base;
Output: The Case Base;
```

3.2 The Querying Process

Two main processes underlying the querying method are described in Algorithm 3. This part is similar to information retrieval for the given query, but retrieval is performed on a specific amount of extracted text from each block and the assigned affordances. During the retrieval, the problem description part of the cases are matched with the user's query and top k cases are retrieved. These are then ranked using the solution of the case, where the AV of the query and the AV of cases are compared. AV of the query is computed on the basis of the terms (also called as 'resource terms') in each topic. After each such a retrieval episode, the AV of the top *k* number of cases are revised and modified in such a way that its currently experienced relevance to the query is properly reflected in the case representation.

Algorithm 2 Procedures to compute Document and Block Affordance Vectors

Procedure: ComputeDocAffordance(d)

Input: List of topics - L_t; A web document - d;

Procedure:

```
1: Segment $d$ into blocks $b_i$ Initialize $AV_d$:= NULL;
2: for each block $b_i$ in $d$ do
3:     Compute $AV_{b_i} := \mathbf{ComputeBlockAffordance}(b_i)$
4:     Update $AV_d := AV_d + AV_{b_i}$
5: end for
6: return $AV_d$
```

Output: The AV of the given document b_i

Procedure: ComputeBlockAffordance(b_i)

Input: List of topics - L_t (as in table 1); AV_{b_i} - affordance vector;

Procedure:

```
1: for each $topic_j \in L_t$ do
2:     Compute the number of matching terms in $b_i$ and terms in $topic_j \in L_t$
3:     Update this score for the corresponding affordance in $AV_{b_i}$.
4: end for
5: return $AV_{b_i}$
```

Output: The affordance vector AV_{b_i} of the block b_i

4 Experimental Results

4.1 Web Corpus

The effectiveness of the proposed method is analyzed through experimental results on a corpus containing the web pages mostly related to the tourist places in India. The tourism web pages were collected by applying the crawling process according to a set of policies that filter the supplementary files. We have omitted the web pages having only hyperlinks, images, advertisements and graphical layouts (like the index page of the

Algorithm 3 Similarity Assessment through blocking and affordance assignment

```
Input:  A query having $n$ terms: $q = \{q_{t_1}, q_{t_2}, \cdots, q_{t_n}\}$
        Case base having $m$ cases: $\{c_1, c_2, \cdots, c_m\}$
        $L_t$ - List of topics;
        $AV_q$ - the query affordance vector
        $AV_c$ - the affordance vector of a case.
Procedure:
 1: Retrieve top $k$ cases using
        $$sim(q, c_j) = \sum_{\substack{t_i \in q; \\ t_j \in c_j; \\ \text{matching terms}}} sim(t_i, t_j)$$
    where $q$ is the query; $c_j$ is the case and $t_j$ are the matching terms in the problem part of $c_j$
 2: for each retrieved case $c_k$ do
 3:     compute the query affordance vector $AV_q$ with respect to the topic list $L_t$;
 4:     get $AV_c$ from the solution of $c_k$;
 5:     compute $sim(AV_q, AV_c)$ using cosine metric; (see equation 5)
 6: end for
 7: return the ranked list of top $k$ cases with respect to $sim(AV_q, AV_c)$
Output: The ranked list of cases sorted by their similarity scores
```

most of the sites). Additionally we skipped the pages containing redirect options, less significant textual description, only copyright information, etc. The remaining pages are collected to form a raw web corpus. Then preprocessing tasks were performed to generate the case base having problem description and solution parts through content and structure mining with focused information extraction.

4.2 Preprocessing

We have applied focused content filtering which performs the structural mining on each collected web page. This structural mining, based on table OR paragraph OR div tags, decomposes the given web page into blocks. Then for each block, we applied the *link-to-text* ratio to distinguish content noise and content text description. We perform duplicate sentences elimination both at the phrase level and at the whole sentence level in order to avoid repeating sentences in the solution parts of each block text. The extracted text content, if they are represented in hex code, are converted into unicode. So multilingual content using hex representation can also be processed (except for the pages using certain proprietary fonts / encodings). We have retained the headings and paragraph markers with selected top k terms for the problem description. But due to link to text ratio, some of the headings might have been removed. Specific patterns are hardly seen for eliminating the unlinked noise from such pages.

Among the total number of 112,522 web documents [1315 seed URLs were crawled to the depth 3], 14,033 web pages, containing both tourism and non tourism pages but having sufficient textual data (after filtering the web pages having spam contents like unwanted, restricted contents, adult contents, etc), were selected for our experiments. We have manually crafted the list of 18 affordances related to tourism domain including the affordance *miscellaneous*.

Affordances	# Terms	Affordances	# Terms
Accommodation	59	Retreats	59
Attractions	59	Shopping	59
Beaches	59	Spirituality	59
Deserts	59	Sports	66
HealthCare	60	ThemeParks	59
Heritage	59	TourPackages	59
HillStations	59	Transport	61
Landscapes	59	Wildlife	59
Nature	59	Miscellaneous	Rest

Table 1. List of predefined affordances with number of terms in each affordance

4.3 Queries / New Cases

We have considered 25 tourism queries in English language used in Cross Lingual Information Access (CLIA) Project[1] - a large project on cross-lingual information access systems for Indian languages, that is being funded by the Government of India, and being executed by a consortium of several academic institutions and industrial partners[7]. Each query is presented in three forms: title - the actual query, desc - the expanded query and narr - the narration of the query. At present, we have attempted with title, desc parts. Here we considered each query (title / desc) as a new case.

4.4 Evaluation Methodology

In the experiments, we compared the effectiveness of the retrieval using Lucene[2] and the proposed approach. In Lucene, the similarity scoring function[5] is derieved from its conceptual formula as follows:

$$\begin{aligned} sim(q,d) := & \, coord(q,d) \cdot queryNorm(q) \\ & \cdot \sum_{t \in q} (tf(t \in d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d)) \end{aligned} \quad (2)$$

where $tf(t \in d)$ is the term frequency - the number of time t occurs in d; $idf(t)$ is the inverse document frequency of the term; $coord(q,d)$ is the score factor based on how many of the query terms are found in the specified document; $queryNorm(q)$ is the normalizing factor used to make scores between queries comparable (does not affect the document ranking); $t.getBoost()$ is the field boost and $norm(t,d)$ encapsulates a few (indexing time) boost and length factors. Here *norm* values is encoded during index time and decoded during search time. Thus encoding/decoding comes with the precision loss - that means $decode(encode(x)) = x$ is not guaranteed. Lucene allows

[1] http://www.clia.iitb.ac.in/clia-beta-ext/
[2] http://lucene.apache.org/java/docs/

the users to customize its scoring formula by changing the boost factors for calculating the score of similarity between the query q and the document d. Lucene[5] sorts the retrieved results based on either their *relevance* or *index order* for the given query. Here we sort the retrieved results based on their relevance to the given query.

In the proposed approach, we weight the AV of the case in the similar way to [1,8]:

Weighted affordance of a case(W_{c_i}):

$$\frac{w_{c_i}}{\sqrt{\sum_{i=1}^{m} w_{c_i}^2}} \tag{3}$$

where w_{c_i} is the weight of the affordance i in the AV of the case.

Weighted affordance of a Query(new case)(W_{q_i}):

$$\frac{w_{q_i}}{\sqrt{\sum_{i=1}^{m} w_{q_i}^2}} \tag{4}$$

where w_{q_i} is the weight of the affordance in the AV of the query.

Similarity between the query (new case) and the case is computed by:

$$sim(q_i, c_i) := sim(AV_{q_i}, AV_{c_i}) := \sum_{\text{matching features}} W_{q_i} \times W_{c_i} \tag{5}$$

During the estimation of case affordance vectors, the values of the elements in the vector increase with the number of matching terms between the solution part and the new case. In such situations, we could apply affordance vector length normalization. To length normalize the elements of the affordance vector AV_c, for the case c having m affordances: $\{A_{t_1}, A_{t_2}, \cdots, A_{t_m}\}$, to the unit vector, we do the following: $AV_c := AV_c/|AV_c|$ where denominator denotes the Euclidean length of the vector of the affordance t_i in c. In the mean time, we will take care of the effect of normalization factor in decreasing the chances of retrieval of the document.

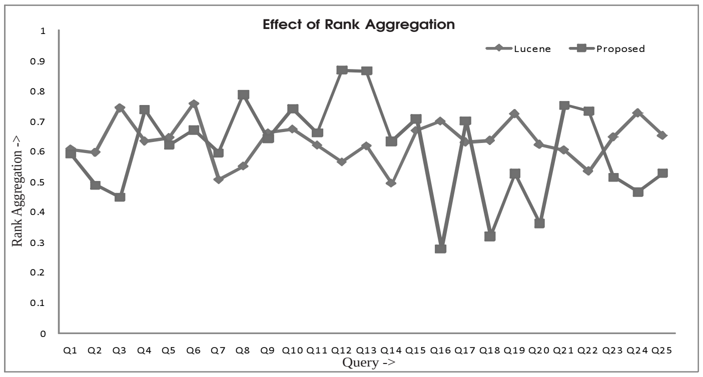


Fig. 2. Effect of rank aggregation of lucene retrieval vs the proposed approach

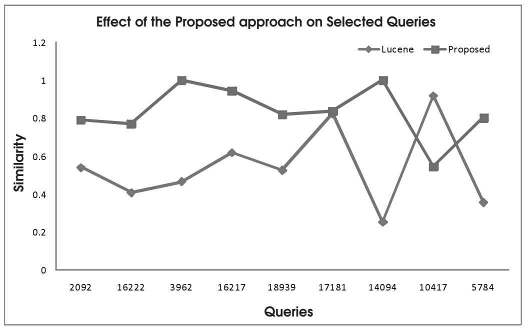


Fig. 3. Effect of the proposed approach on a few selected queries

We perform *rank aggregation*: Given a new case, retrieve top candidate cases using the problem description. Then similarity estimates of the affordance vector (solution part) of each of the candidate cases with the affordance vector of the query are computed. Finally the rank of all top k cases are aggregated with respect to their actual similarity scores and the results are compared. The figure 2 shows the effects of rank aggregation for the 25 tourism queries [title / desc parts are considered here]. Lucene retrieval score is influenced by index time boot factors and applies tf - idf tradeoff with overall content of the textual description. In the proposed approach, the point of focus is the affordance assignment with respect to the block text based on its maximum affordance. The queries Q12, Q13 and Q14 are proper names representing the places and each extracted block text, that speaks about these places specifically, contributes to the overall affordance. Similarly Q21 and Q22 focus on the specific event / hotel in the particular place. This gives combined affordance score with with the proper names. Hence this leads to a better performance for most of the queries (particularly for Q12, Q13, Q21 and Q21).

Next we considered a few sample queries whose similarities are effectively computed through blocking and affordance assignment(fig. 3). For example, in Q1 (Query: sunderbans national park), the document with ID - 2092, ranked 14 in Lucene retrieval and the proposed method has brought it to rank 2. Here affordances related to *wildlife*, *heritage* and *attractions* are captured where as the affordances related to *nature* is hardly captured. This is due to the fact that park under the affordance nature contributed less to the overall affordance than to wildlife. In another example, for the query Elephant Safari in Kaziranga, the document, with ID: 10417, having the dominating term of “safari”, has been brought to the top in lucene where as its affordance value score very less. At the same time, for the query Goddess Meenakshi Temple, the proposed approach captured the document, with ID: 5784, whose blocks describe different topics related to Meenakshi temple in Madurai, Tamil Nadu, India.

Even though the performance of the proposed system is promising, the effect of the noise and the accuracy of filtering approach along with the list of resource terms play a vital role in the effective retrieval of cases. The effect of spam pages will reduce the chances of retrieving the relevant document through boosting their scores by projecting the related themes. Owing to paucity of resources, we have limited our spam filtering to filter pages containing adult content along with tourism related textual content. This effectively reflects in the retrieved results with the proposed approach. This is our preliminary attempt with manually crafted term list for each identified (predefined) affordance related to the tourism domain. Developing an automated process for the affordance identification irrespective of the domain may be attempted in the future.

5 Conclusion

We presented an approach for achieving an effective case retrieval through the similarity assessment based on blocking and identifying affordances of web documents. Affordance provides a visual clue to the case identification. Traditional methods dealing with textual content try to apply similarity metrics collectively for the heterogeneous blocks of text together presented in the same web content. This reduces the chances of the user expected contents (solutions). The proposed approach solves this issue by applying page segmentation through blocks, identifying valid text from these blocks, scoring each of the blocks with certain affordance scores and then applying similarity metrics to achieve the effective case retrieval. The actual performance of the proposed approach can be seen with the similarity scores computed using query affordance vector and case affordance vectors. The preliminary results show that the proposed approach would be promising for identifying the specific block text as the relevant solution.

Acknowledgement

We gratefully acknowledge: Prof. Sudeshna Sarkar (IIT Kharagpur) for initiating this work; the efforts of Mr. Parin and Mr. Gyan (IIT, Kharagpur) in getting the list of seed urls for crawling and Mr. David Kabath (NTNU) in preparing the list of (tourism related) keywords. Also authors thank the anonymous reviewers for their insights which improved the quality of the paper

References

1. C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3. In *TREC*, pages 0–, 1994.
2. O. G. F. Diaz, R. S. Salgado, I. S. Moreno, and G. Rodrguez-Ortiz. Searching and selecting web services using case based reasoning. In *ICCSA (4)*, Lecture Notes in Computer Science, pages 50–57. Springer, 2006.
3. J. J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, *In Perceiving, Acting, and Knowing*, 1977.
4. S. Ha. A personalized counseling system using case-based reasoning with neural symbolic feature weighting (cansy). *Appl. Intell.*, 29(3):279–288, 2008.
5. E. Hatcher and O. Gospodnetic. *Lucene in Action (In Action series)*. Manning Publications Co., Greenwich, CT, USA, 2004.
6. B. Limthanmaphon and Y. Zhang. Web service composition with case-based reasoning. In *ADC '03: Proceedings of the 14th Australasian database conference*, pages 201–208, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.
7. P. Majumder, M. Mitra, S. Parui, and P. Bhattacharyya. Initiative for indian language ir evaluation. In *The First International Workshop on Evaluating Information Access*, pages 14–16, Tokyo, Japan, 2007. EVIA.
8. A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Inf. Process. Manage.*, 32(5):619–633, 1996.

Conversational Framework for Web Search and Recommendations

Saurav Sahay and Ashwin Ram

ssahay@cc.gatech.edu, ashwin@cc.gatech.edu
College of Computing
Georgia Institute of Technology
Atlanta, GA

Abstract. In this paper, we describe a Conversational Interaction framework as an innovative and natural approach to facilitate easier information access by combining web search and recommendations. This framework includes an intelligent information agent (Cobot) in the conversation that provides contextually relevant social and web search recommendations. This setup leverages the information discovery process by integrating web information retrieval along with proactive connections to relevant users who can participate in real time conversations. We describe the conversational framework and report some preliminary experiments in the system.

1 Introduction

The medium of online conversation allows for sharing ideas, asking questions or discussing issues and solutions interactively along with others. It is an age-old communications practice that helps cultivate creativity, exploratory ideas, perspectives and experiences to take better decisions individually or collectively in the process. Several problems persist with using existing search tools as a means of learning, investigating or exploring about some complex and open-ended information topic. Collaborative social search involves different ways for active involvement in search related activities such as use of social network for search, use of expertise networks, involving social data mining or crowdsourcing to improve the search process.

Social psychologists have experimentally validated that the act of social discussions have facilitated cognitive performance[16]. People in social groups can provide solutions (answers to questions), pointers to databases or other people [1][3], validation of ideas[2], can serve as memory aids[5] and help with problem reformulation.

The goal, we envision, is to move search from being a solitary activity to being a more participatory activity for the user using natural dialogue conversations mixing social search with traditional web search techniques. The search agents perform multiple tasks of finding relevant information and connecting the users together; participants provide feedback to the system during the conversations that allows the agent to provide better recommendation temporally in the conversation. This framework is different from classical IR or Question Answering (QA). The focus of classic IR systems is on retrieving relevant documents from a large document collection in response to a query.

While QA deals with more complex understanding of natural language queries, it does not involve a back and forth interaction to continuously monitor, adapt and explore in continuum about some information or questions. This Conversational approach helps users search, explore and ask questions in natural language, leaving the task of user intent comprehension on the system, while the conversational search agents bring together people and different artifacts like documents, facts and opinions together in the conversation to provide a knowledge-rich participatory atmosphere. Cobot uses technology for operationalizing a user's intent into computational form, dispatching to multiple, heterogeneous services, gathering and integrating results, finding people in the community who best match the ability to respond to user's request and presenting them to the user as a set of solutions to their request. This conversational framework process involves a series of dialogue interactions, agent recommendations and feedback activities.

2 Framework

Figure 1 gives a high level architecture of the Conversational Interaction framework. The framework is built around constructs to support memory update and access, categorization and learning in the system. The framework allows for the ability to start conversations, get connected to people and get relevant information for the information need in context.

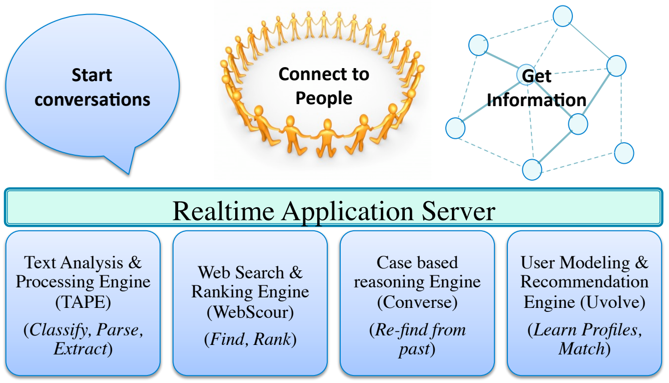

Start conversations
Connect to People
Get Information
Realtime Application Server
Text Analysis & Processing Engine (TAPE) (Classify, Parse, Extract)
Web Search & Ranking Engine (WebScour) (Find, Rank)
Case based reasoning Engine (Converse) (Re-find from past)
User Modeling & Recommendation Engine (Uvolve) (Learn Profiles, Match)

Fig. 1. System Architecture

While developing the Conversational Interaction framework, we are adhering to some requirements and guidelines which are as follows:

- Cobot is an Information Agent with Memory, Categorization and Learning modules to remember, understand and improve recommendations over time for the user.
- Different conversation facets (topic, message, asker, presence, time of asking) should have different metrics for comparison to provide for search criteria beyond query-relevance

– Ability to reformulate relevant queries from conversational sentences and paragraphs
– Ability to understand the progression of conversation context to determine suitable interference points.
– Critique based feedback in search results (eg. ability to like different facets) to support personalization of results
– Support for quick access to past conversations (Ability to re-find information)

Some differences between searching conversations and traditional web search can be attributed to factors like chronological ordering of conversations, lots of coreferences and informal nature of the language. Traditional text ranking algorithms like BM25[9] might not work due to factors like short length of these conversations.

Text Analysis and Processing Engine(TAPE) processes conversations, pushing it through the various steps of analysis, processing and storage within the system. The current system is being designed and developed for health domain and engages in it the use of medical ontologies coupled with natural language processing components. TAPE (Figure 2) produces and maintains the knowledge representation by processing information from agent's working memory of conversation, user models and knowledge-bases. The agent's task is to use the sub-modules for extracting meaningful queries from conversations, classifying messages into relevant categories, and calling the right combination of algorithms for retrieving candidate recommendations.

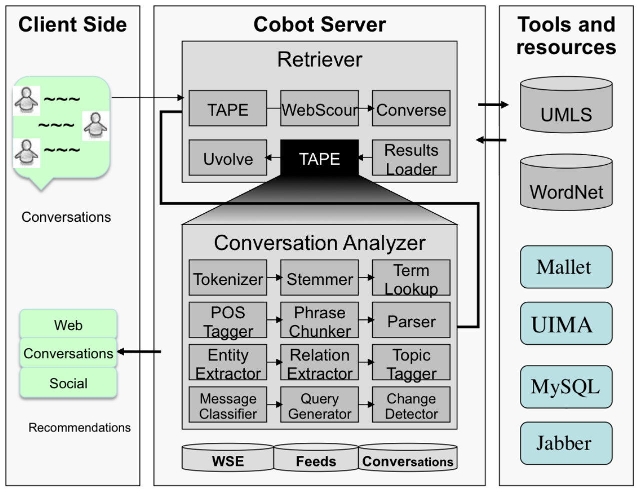


Fig. 2. TAPE Engine

2.1 Memory

Language and interaction (percepts) creates usable memories, useful for making decisions about what actions to take and what information to retain. Cobot framework (we interchangeably use the terms Cobot and the Conversational Interaction framework) leverages these interactions to maintain users' episodic and long term semantic models, agent's per conversation working memory of topics, users and messages (Figure 3). The agent analyzes these memory structures to bring in external recommendations into the system by matching with the contextual information need(Categorization). The social feedback on the recommendations are registered in the indices for the algorithms to generate their contextual relevance. Paper [13] also describes the architecture of Cobot System in more detail.

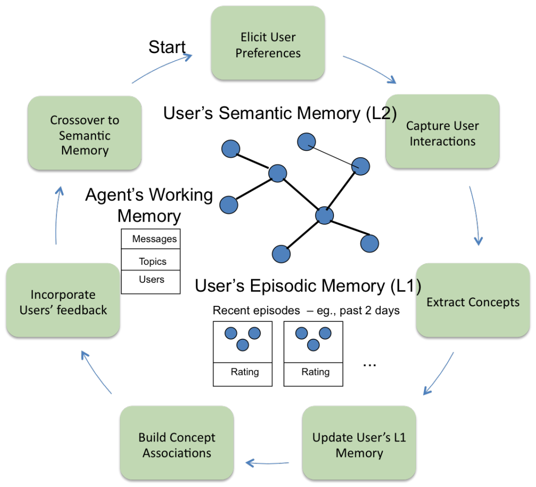

Start
Elicit User Preferences
Capture User Interactions
Extract Concepts
Update User's L1 Memory
Build Concept Associations
Incorporate Users' feedback
Crossover to Semantic Memory
User's Semantic Memory (L2)
Agent's Working Memory
Messages
Topics
Users
User's Episodic Memory (L1)
Recent episodes – eg., past 2 days
Rating
Rating
...

Fig. 3. Memory Structures

The purpose of Episodic Memory is to capture the user's short-term interactions and interests. Based on user's frequency of interactions and diversity in topics, this memory empirically varies in the range of a few days for different users. The Semantic Memory captures the user's long-term profile. These are the topics that interest the user in general and for a prolonged time. These interests change less frequently and represent general criteria of recommendation to the user. Many times, users might be interested in some temporary information need. Such information need not be incorporated in the long term user memory. The episodic memory captures such short-term interests. The episodic memory forms a sort of staging area and the concepts from this memory are selectively and periodically moved to the semantic memory in a crossover process.

The nodes of the semantic memory are concepts extracted from user's interactions. The concepts are connected with associations which develop when concepts co-occur frequently. Over a period of time when the user participates in more interactions, new concepts are added to the semantic memory. Episodic Memory is represented as a Case based Reasoning like kNN system. Short term interaction episodes containing frequent concepts from conversations with interaction feedback are stored. We also call this episodic store as our Level 1 (L1) memory. This memory is searched first during the recommendation stage to prune the search space to a smaller size. Semantic Memories for this smaller search space (Level 2 or L2 Memory) are searched next to refine the ordering of recommendations and find the best matches.

2.2 Categorization

The next important step in the development of an information agent is to enable it with constructs to identify important signals from the conversations, classify them in the right schemas and group them together to further aid in generating good recommendations. In order to test some of Cobot's algorithms, we crawled WebMD forum that consists of posts and responses on different health topics. The crawler extracted all posts dating back up to one year or 20 pages of posts for each subforum. The data so extracted includes forums, subforums, conversations, users and their ratings. We extracted more that 64000 conversations from WebMD forums.

Here's an annotated sample post and one of its responses that are typical of the dataset. Bold face maps medical concepts and extracted relationships (highlighted in bold italicized). The method for extracting terms and relationships is described in detail in this paper[12].

Post (**AskQuestion** category): *Has anyone experienced* **cystic acne** *appearing once you started taking* **Adderall***? I have found that when I take my* **daily dose***, by the end of the day a cystic-like* **pimple** *has appeared on my face. If I skip a* **daily dose** *or two of* **medication***, I don't have any real* **acne** *issues. I am 42 years old and have had* **acne** *before taking* **Adderall***. But I have never had these large painful bumps. Can anyone help me???*

Response (**SuggestSolution** category): *I don't think it's the* **medication***. I've had* **cystic acne** *for a long time - including years before I started taking* **ADD Meds***. It's linked to two things. My time of the month and* **STRESS***.* **AD/HD stimulants** ***can increase*** **stress***. Instead of an* **antidepressant***, like some people have, get a* **beta blocker***. You don't get* **sleepy***. I also don't think it's* **depression** *that people get with the* **meds***, it's the* ***anxiety*** **which can cause** ***depression*** *like* **symptoms***.*

Conversational interactions are classified into one of the following categories in Cobot to strategize for query reformulation stage and to help make the decision if the agent should insert some type of recommendation into the conversation:

– *ASK QUESTION*: Asking a question, e.g. somebody posts a problem. This is usually, but not always, the first post of a thread.
– *DITTO*: Repeating a question, e.g. "Yes, I also have the same (or a very similar) problem".
– *ASK CLARIFICATION*: Asking for more details about the problem, e.g. "Can you please provide more details?"
– *FURTHER DETAILS*: The person who is facing a problem provides more detailed information about it, possibly after somebody asks for more details.

- *SUGGEST SOLUTION*: Suggesting a solution
- *EXPRESSIVE* (Thanks for suggestion/solution, complaints about suggestion/solution, reject/accept solution)
- *OTHER* (Not fitting the above categories)

Choice of Features

The choice of features to predict the type of message labels is extremely important to get good results for this problem. In most text classification problems, a simple 'bag of words' approach is taken to populate the vector space of features. These features are statistically extracted using techniques like 'term frequency - inverse document frequency' (TFIDF) or z-score method. These statistical features make the space of possible feature set extremely large thus requiring huge training data to come up with good decision boundaries for classification of data into the right categories. In contrast, we have used a mix of syntactic and semantic features for our data exploiting medical ontologies like UMLS (Unified Medical Language System) and WordNet. We extracted the following features for the Message Classification problem:

- Position of the message thread
- Length of message
- Number of responses of the user for that forum
- Emotive Features (vector of words, testing for binary presence)
- Question words (vector of words, testing for binary presence)
- Previously responded in the forum or not
- Number of previous responses
- response time windows
- words in the thread (high information gain 5950 words vector from the corpus)

In order to develop a message classifier that could categorize the messages into one of the above categories, we manually tagged 412 different conversation threads with different message categories. We used this labeled data from different WebMD forums to evaluate the classifiers using 80% of data for training and the rest for testing the models.

We used three standard algorithms to compare the accuracies of message classification system using rich feature extraction to aid in classification. In the first two approaches involving Bayes' Classification and Support Vector Machines, this problem is a standard multi class text classification problem. Third approach using CRFs formulate the problem as a Sequence Labeling problem. Conditional Random Fields (CRFs)[6] are discriminative conditional probability distribution models that allow to take advantage of the sequential nature of conversations better. From the experiments, we see that CRF was able to pick up the right categories from the messages and was able to do better (Table 1) that the other standard methods (but was slower than other classifiers).

2.3 Recommendations

Cobot provides three types of recommendations. It recommends and notifies relevant people who may be interested in joining conversations. It provides topic specific web recommendations and it also suggests past similar conversations from the system.

Table 1. Message Classification

	Accuracy	Time(sec)
Bayes' Classifier	53.5	0.1
SVM Classification	56.1	1.8
Linear Chain CRF	67.9	9.8

People Recommendation: While designing a recommender system, it is important to take into account the domain implications and fine-tune the algorithms accordingly. To provide social recommendations with a high degree of conversion rate, the system needs to identify people who can provide answers to asked questions, share similar health experiences and provide topic specific opinions and advice. Our system is built around health information domain therefore users are generally not concerned with building their social ties, instead, the goal is to serve the user's contextual information need. One important aspect in this domain is reputation of the recommended users, since there is no prior information and relationship of these users with the person who starts a conversation. We are building the reputation system by allowing users with the ability to rate conversations. The system takes into account factors for weighting the users differently (based on types (asker, responder, viewer), length of conversation, topics, etc.)

Our system currently tries to find a recently active user first who participated in similar conversations. Different conversational facets are matched with episodic memories and a spreading activation search on the semantic net is performed for recommending the best 3-4 users for the conversation. The activation is spread to the neighboring nodes proportional to the weight of each connecting association in the semantic net. There are several parameters in the system that can be learnt based on activity of users. Parameters for episodic memory window size, semantic memory learning and unlearning rates, concept co-occurences and feedback strengths for associations are initially set heuristically and can be fine-tuned to suit individual users.

Knowledge Recommendation: For web search and conversation recommendations, we reformulate queries from the conversation snippets based on occurrence of concepts and relationships and types of messages. For a given target query Q_t, past conversations are ranked so that the results which are most likely related to the learned preferences of the community are promoted[14][8][7]. This kind of personalization is based on the reuse of previous search episodes: the promotions for Q_t are those results that have been previously selected by community members for queries that are similar to Q_t.

Cases are represented as tuples made up of the query component (a set of query terms, Q_i used during some previous search session) along with web recommendations and past conversations with their community hit counts. Our formulation is based on similar work reported in Paper [14]. Each case is a summary of the communitys search experience relative to a given query.

Each new target problem (corresponding to a new query Q_t) is used to identify a set of similar cases in the case base by using a term-overlap similarity metric to select the n most similar search cases for Q_t .

These search cases contain a range of different result pages and their selection frequencies. Bearing in mind that some results may recur in multiple cases, the next step is to rank order these results according to their relevance for Q_t. Each result R_j can be scored by its relevance with respect to its corresponding search case, C_i by computing the proportion of times that R_j was selected for this cases query Q_i.

During the development of retrieval stage of the CBR system for Cobot, it was often observed that number of results retrieved were very large since the retrieval stage entailed a meta-search which queried many search engines which returned large number of results. We wanted to show only the top 2 to 3 results /conversations from the retrieved case base. Consequently sorting and ranking results according to their relevance to the ongoing conversation was necessary.

Relevance of a result with respect to the current target query Q_t) is calculated by computing the weighted sum of the individual case relevance scores, weighting each by the similarity between Q_t and each Q_i. In this way, results which come from retrieved cases (C_1, ..., C_n) whose query is very similar to the target query are given more weight than those who come from less similar queries. The relevance of a Result R_j to a target query Q_t and the case library comprising of cases from C_1 to C_n cases is expressed as:

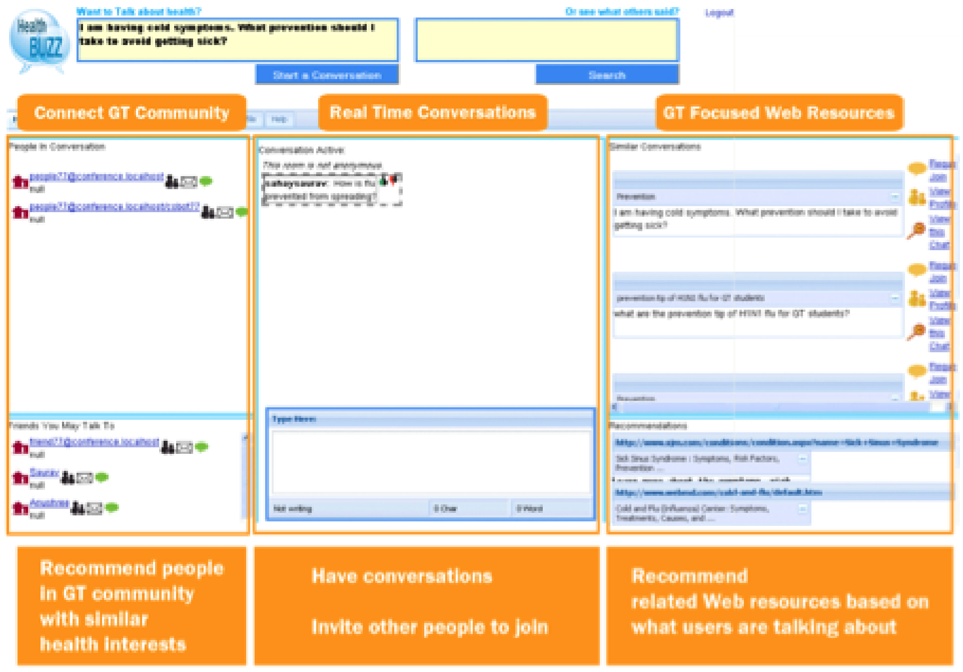
$$WRel(R_j, Q_t, C_1...C_n) = \frac{\sum_i Relevance(R_j, C_i) * Similarity(Q_t, C_i)}{\sum_i Exists(R_j, C_i) * Similarity(Q_t, C_i)}$$

Similarity between the query and case is computed by finding the similarity between the query and case queries. We are using Jaccard Similarity as the similarity metric in our design. In this way, for given user, with query Q_t we produce a ranked list of results R_j that come from the communitys case base and that, as such, reflects the past selection patterns of this community. If the case base doesnt retrieve cases or the similarity confidence of the retrieved results is less than a user specified threshold t then, Q_t is used by the meta-search module to retrieve a set of web search results.

The top 3 results from the ranked results obtained either from the case base or the meta search engines are shown to the user. In this way, results that have been previously preferred by community members are either promoted or marked as relevant to provide community members with more immediate access to results that are likely to be relevant to their particular needs. This framework promotes community preferred results and conversations to the user.

3 Discussion

From a brief usability study of the system prototype (Figure 4)[11], we learnt that socially powered search feature and the ability to collaboratively search together and discuss issues with real people instead of solitary search engine is very useful. Websites like Vark.com[4] are doing social search for generic question answering effectively using IM based messaging bots and other channels.



I am having cold symptoms. What prevention should I take to avoid getting sick?

This room is not anonymous

sahaysaurav: How is flu prevented from spreading?

Prevention

I am having cold symptoms. What prevention should I take to avoid getting sick?

prevention tip of H1N1 flu for GT students

what are the prevention tip of H1N1 flu for GT students?

Fig. 4. System Prototype

There are many technical challenges in community based information and recommendation systems. Cobot is being developed around the principle of *'Suit the user, make it easy, make it good'*. Cobot's approach and solution to next generation of socially enabled search is uniquely driven by new trends on the web, requiring new technologies for an integrated socio-semantic search experience. Instead of relying on search engines that inundate the user with a multitude of information, Cobot models the information finding task as an interactive and collaborative recommendation process within a social community. The user describes his need in natural language to a trusted community which is modeled via text conversations familiar to most users. Our agent based conversational framework for web search and recommendations uses a 'wisdom of crowds' approach to compensate for the limitations of traditional search engines and uses the experience of real users by proactively bringing them to participate in the conversations.

4 Acknowledgement

We acknowledge and thank our past project members, Alejandro Dominguez for writing the WebMD forum crawler, Bharat Ravisekar for working on a Personalized Feed Recommender based on Semantic Nets and Hrishikesh Pathak for implementing the case based reasoning module in Cobot. We also thank the contributions of Anushree Venkatesh and Stephanie Ahn as members of the Cobot project.

References

1. R. Cross, R. E. Rice, and A. Parker. Information seeking in social context: structural influences and receipt of information benefits. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 31(4):438–448, 2001.
2. B. M. Evans and E. H. Chi. Towards a model of understanding social search. In *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 485–494, New York, NY, USA, 2008. ACM.
3. E. A. Fox, D. Hix, L. T. Nowell, D. J. Brueni, D. Rao, W. C. Wake, and L. S. Heath. Users, user interfaces, and objects: Envision, a digital library. *J. Am. Soc. Inf. Sci.*, 44(8):480–491, 1993.
4. D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *WWW*, 2010.
5. I. Karasavvidis. Distributed Cognition and Educational Practice. *Journal of Interactive Learning Research*, pages 11–29, 2002.
6. J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
7. K. McCarthy, L. McGinty, B. Smyth, and M. Salamó. The needs of the many: A case-based group recommender system. *Advances in Case-Based Reasoning*, 4106:196–210, 2006.
8. M. J. Pazzani and D. Billsus. Content-based recommendation systems. pages 325–341, 2007.
9. S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.
10. B. Rogoff. *Apprenticeship in thinking: Cognitive development in social context*. Oxford University Press New York, 1990.
11. S. Sahay, S. Ahn, S.-C. Lu, B. Sherwell, A. Venkatesh, and A. Ram. Healthbuzz: Contextual social search and conversations. In *The Third Annual Workshop on Search in Social Media (SSM 2010)*, February 2010.
12. S. Sahay, S. Mukherjea, E. Agichtein, E. V. Garcia, S. B. Navathe, and A. Ram. Discovering semantic biomedical relations utilizing the web. *ACM Trans. Knowl. Discov. Data*, 2(1):1–15, 2008.
13. S. Sahay, A. Venkatesh, and A. Ram. Cobot: Real time multi user conversational search and recommendations. In *Recommender Systems and the Social Web*, volume 532. CEUR Workshop Proceedings, 2009.
14. B. Smyth, P. Briggs, M. Coyle, and M. P. O'Mahony. A case-based perspective on social web search. In *Proceedings of the 8th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, pages 494–508, Berlin, Heidelberg, 2009. Springer-Verlag.
15. B. Wilson and H. Meij. Constructivist learning environments: Case studies in instructional design. *IEEE Transactions on Professional Communication*, pages 0361–1434, 1997.
16. O. Ybarra, E. Burnstein, P. Winkielman, M. C. Keller, M. Manis, E. Chan, and J. Rodriguez. Mental Exercising Through Simple Socializing: Social Interaction Promotes General Cognitive Functioning. *Pers Soc Psychol Bull*, 34(2):248–259, 2008.

CBR Startups

Workshop at the
Eighteenth International Conference on
Case-Based Reasoning

Alessandria, Italy
July, 2010

Ashwin Ram and Saurav Sahay

Co-Chairs

Ashwin Ram
Georgia Institute of Technology, USA
Founder: Enkia, OpenStudy
ashwin@cc.gatech.edu

Saurav Sahay
Georgia Institute of Technology, USA
Founder: Cobot Health
ssahay@cc.gatech.edu

Invited Speakers

Barry Smyth
University College Dublin, Ireland
Founder: HeyStaks

Peter Funk
Mälardalen University, Sweden
Founder: FunkAI

Frode Sørmo
Norwegian University of Science and Technology, Norway
Founder: Verdande

Workshop Overview

Over the past twenty-five years, Case-Based Reasoning has matured into a full-fledged discipline within AI, with an international community, strong research momentum, and many commercial successes. However, despite its many advantages as a technology, CBR is not well known in the entrepreneurial world. In part, this is due to few startups being created by CBR researchers, who are the best people to initiate commercialization of their ideas.

This workshop focuses on the merits and challenges of creating a technology startup out of cutting-edge research in academia or research labs. We will discuss technological issues, such as the application areas best suited for CBR approaches and scalability of CBR technologies. We will also discuss practical issues, such as the tension between academic goals (e.g., publishing papers) and commercialization goals (e.g., building applications), and the different types of expertise required to create a vision (researchers), market a product (marketers), and build a company (entrepreneurs).

In this workshop, we will hear from CBR researchers who have created CBR startups, and use their experiences to discuss different ways to commercialize CBR technologies. Some have chosen a hands-on approach, taking on a management role (CEO) or a technology role (CTO). Others have partnered with experienced business people, who have taken their ideas forward. We will also provide a forum to help participants with their startup ideas. This workshop will comprise short talks, a panel discussion with open audience questions on the merits and challenges of doing a startup, alternative ways of commercializing CBR technologies, and advice to people interested in doing this.

We will conclude with a hands-on session with 3 minute pitches by participants. Think of it as throwing down the gauntlet - a friendly competition where you pitch your CBR startup idea. We will identify most innovative use of CBR technology, best business idea, and idea most likely to succeed. Our intention is to provide advice and mentoring by community members who have been-there-done-that, using these ideas as case studies for all of us to learn from.

Ashwin Ram July 2010
Saurav Sahay

Computer Cooking Contest

Workshop at the
Eighteenth International Conference on
Case-Based Reasoning

Alessandria, Italy
July, 2010

Amélie Cordier and David W. Aha (Eds.)

Co-Chairs

Amélie Cordier
LIRIS, University of Lyon 1, France
amelie.cordier@liris.cnrs.fr

David W. Aha
Naval Research Laboratory, USA
david.aha@nrl.navy.mil

Program Committee

Preface

The Third Computer Cooking Contest (CCC) was held as part of the Eighteenth International Conference on Case-Based Reasoning (CBR), which took place in Alessandria, Italy during 19-22 July 2010. The purpose of this annual event is to provide a challenging, intuitive, and fun application focus for AI research that is open to all indviduals and teams. Our goal is to encourage the creation of novel development or integrations of AI techniques to solve problems related to cooking, where no restrictions were placed on the types of techniques that could be used. To accomplish this, we held an open call for software systems that can solve one of a set of cooking challenges, a workshop at which the entrants' solutions were presented, and a live competition in which a chef and the program committee members could judge the capabilities of the submitted systems.

This year's CCC included four challenges. First, the Main Challenge involved answering queries that required the selection and possible modification of a recipe for a single dish. Second, the task of the Adaptation Challenge was to adapt a known recipe in response to additional constraints (e.g., some of the recipe's necessary ingredients were unavailable). Third, the Open Challenge was just that: an invitation for researchers to use their imagination in the context of the cooking domain. For example, this could involve representing and reasoning with numeric quantities, formulating adaptation as a planning task, or discovering pertinent cooking information automatically from the Web. Finally, the Student Challenge was designed to permit students to focus on specific tasks of their interest (e.g., natural language understanding) so that they would not need to compete head-on with larger teams that have been participating repeatedly in this contest. Each team is permitted to enter any challenge, although only student teams can enter the Student Challenge.

Each team was required to submit (1) the URL of the user interface for their running system and (2) a paper (max 10 pages) describing their corresponding research, development, and evaluation activities. We provided them with a resource - a recipe database that contains, for each recipe, its title, list of ingredients, and cooking instructions in text format.

The CCC workshop included technical presentations by the contest entrants, whose papers can be found in this volume. The purpose of this workshop, which was open to all conference attendees, was to facilitate discussion on the approaches used to solve the contests' challenges so that all of the attendees can study these solutions for future reference (or to improve their cooking skills!).

This year there were five terrific entrants. They all address the Main Challenge and the Open Challenge. Three teams are also involved in the Adaptation Challenge and we have only one student team! Together, these entrants have extended the state-of-the-art for the application of AI techniques for automated cooking. For example, contributions have included reasoning with quantitative ingredient information, representations (e.g., workflows) and reasoning methods for adapting cooking instructions, and the use of semantic web technologies, among others.

The 2010 CCC succeeded in encouraging innovation, and we hope that next year's contest will continue this tradition. For example, the 2011 CCC may benefit from a focus on more extensive evaluations of the submitted systems, which could potentially be accomplished through the on-line collection of feedback from a pre-selected pool of volunteer judges or through the extensive use of a cooking simulator for "testing" generated solutions to cooking challenges.

Many thanks to our outstanding program committee, the ICCBR-10 Co-Chairs (Isabelle Bichindarize and Stefania Montani), the ICCBR-10 Workshop Chair (Cindy Marling), and our sponsors (Empolis, Germany and LIRIS, France).

We look forward to next year's contest!

David W. Aha July 2010
Amélie Cordier

JADAWeb: A CBR System for Cooking Recipes

Miguel Ballesteros, Raúl Martín and Belén Díaz-Agudo

Department of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, Spain.
miballes@fdi.ucm.es, rmb2000@gmail.com, belend@sip.ucm.es

Abstract. JaDaWeb has been developed to participate in the 3rd Computer Cooking Contest 2010. The system has been developed over the system JaDaCook 2.0 that participated in CCC-09. As the main novelties JaDaWeb includes a Web based natural language interface with parsing of the input data, a fuzzy similarity function and Wordnet reasoning to identify and include new ingredients in the ontology. In this paper we present a brief review of the technical characteristics of the system, and some experimental results comparing JADAWeb with Cookiis, the winner system of the CCC-09. The online system is accesible via http://supergaia.fdi.ucm.es:8810/CCCWeb

1 Introduction

In this paper we describe JADAWeb, a CBR system based on JADACook[1, 2] that suggests recipes using as input a set of ingredients to include and to avoid, and some optional dietary and cooking restrictions. JADAWeb retrieves a recipe from the system case base and includes the capability of adapting it by substituting its ingredients by other similar ingredients that appear in the user query. The system uses a fuzzy similarity function to determine if two ingredients can be swapped in the recipe. JADAWeb has been implemented using the jCOLIBRI framework [3] and its facilities to design CBR systems.

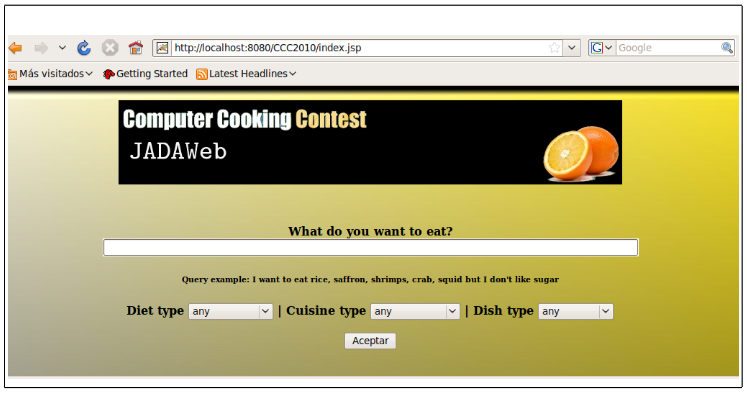
JADAWeb has a Web interface built using JSP [4] technology where the input of data is a single character string on natural language in English. We have included an algorithm to detect the negation in a sentence, so the system is able to infer which ingredients the user wants to include and to avoid. The algorithm is based on multilingual dependency parsing[5]. JADAWeb also includes a number of substantial improvements in the decision of which nouns identified in the query are ingredients and which are not. JADACook only checks if the noun appears in the ingredient ontology. However, JADAWeb also searches the noun in Wordnet[6] to determine if the noun is an ingredient or not. If an ingredient is identified in the query and it was not defined in the system ontology, JADAWeb suggests the area of the ontology is better to classify the new ingredient.

After retrieval, JADAWeb suggestions are ordered by similarity and also according with the characteristics of the season. Hotter meals, like a soup, will be presented before in winter, and fresh meals, like salad, will be presented before if the query is in summer. The information that JADAWeb uses to make this decision is the presence, or not, of some important words that are present in each recipe, like ‘hot’, ‘oven’, or ‘fresh’.

This paper is organized as follows: Section 2 describes the web-based graphical user interface and the parsing of the input data. In Section 3 we define how the acquisition of knowledge works. Section 4 describes the case-based reasoning process, focusing on the similarity function. Section 5 describes a set of examples tested in the system. We have compared JADAWEb results with the CookIIS system[7] results. Finally, Section 6 shows the conclusions and suggests some lines of future work.

2 Web-Based Graphical User Interface

The web interface allows the user writing a natural language sentence with the recipe requirements (see Figure 1). The following is an example of a query where the user explains what (s)he wants: ”I want to eat some fruits, like apple, but I don’t like kiwi”. The system connects to the dependency text parser to obtain the information contained in this query. It extracts the list of ingredients to include and to avoid in the recipe.



Query example: I want to eat rice, saffron, shrimps, crab, squid but I don't like sugar

Diet type any | Cuisine type any | Dish type any

Fig. 1. JADAWeb Web-Based Graphical User Interface

The system infers the lists of ingredients from the query text and shows the result of the parser. The user also can express if he/she wants to choose some dietary practices, or type of cuisine using a form type of interface.

2.1 Textual Parsing of the Input

In this version of JADAWeb the textual input query must be a complete and correct sentence in English. The user query might include the ingredients to include and to avoid in the recipe. We have implemented an algorithm to detect the negation on sentences by using dependency parsing[5]. Dependency parsing is an important area of research related with artificial intelligence, computational linguistics and machine learning. We have used a dependency parser called Minipar[8] that is limited to English, but it would be possible to use another parsers like Maltparser [9] which allows multilingual dependency parsing. Minipar has an API implemented in C language. Given a plain text sentence, Minipar returns a dependency tree (see Figure 4). Using a wrapper[1] for JAVA we have studied the information given by the parser. Given the tree with the dependency analysis we have implemented an algorithm that detects the attachment of each noun and those who share the same branch of a negation are marked as candidates for unwanted ingredients. All other nouns are marked as wanted ingredients.

The evaluation results show that MINIPAR is able to cover about 79 % of the dependency relationships in the SUSANNE corpus[10] with about 89 % precision[8] for English parsing, so we have a small quantity of errors, but it does not mean that our algorithm committed 10% of errors because is possible that the errors make by Minipar are not important for our purposes.

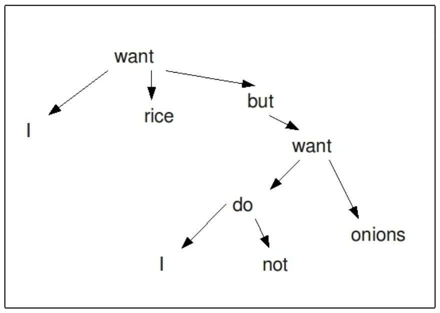


Fig. 2. Dependency tree

Figure 2 shows the information that the Minipar parser returns for the following sentence: 'I want rice but I don't want onions'. Using this information our algorithm builds a structured query that is compared to the recipes in the case base. Although we have found a few labeling errors from the dependency parser, the results obtained by the algorithm are quite reliable in general.

3 Knowledge acquisition

JADAWeb reasons using two main knowledge sources: the recipe base provided by the contest organization, and the ontology that has been built incrementally and collaboratively from the first versions of JADACook. The ontology is conceptualized and formalized in the OWL language[2].

[1] http://nlp.shef.ac.uk/result/software.html
[2] http://www.w3.org/2004/OWL

As a novelty, JADAWeb uses Wordnet[6] to let the user include ingredients that are not previously integrated in the ontology. The textual parsing of the input query searches in Wordnet to decide if a not identified noun in the query is an ingredient or not.

3.1 Ontology

JADAWeb reasons with an ontology that formalizes the cooking domain knowledge. The ontology organizes ingredients in categories. In the first level there are ingredients of animal origin grouped in some subsets: meat, fish, milk, cheese and eggs. There are also, ingredients of plant origin grouped in subsets like cereals, fruits, and vegetables. And finally, there are other specific families of ingredients like sweets, drinks, or basis ingredients like oil or salt.

The ontology in its current state is very extensive and complete, it has 255 classes, 202 of them are ingredients. Although we made little changes to make easier the search and the equality between concept names because Wordnet has different terms than our previous ontology. For example, we changed 'AnimalOrigin' and 'PlantOrigin' for 'animal' and 'plant'.

3.2 Ingredient treatment using Wordnet

Despite the richness of the ontology and the wide range of definitions in the cooking domain, we have implemented a system to detect new ingredients in the input by searching the nouns in the lexical database Wordnet. Wordnet is a lexical database in English, developed at Princeton University under the direction of George A. Miller. This lexical database allows searching a concept to get the definition, synonyms and antonyms, among other things.

For every noun in the query that is also in Wordnet, is a candidate to be an ingredient, the system checks if it is already included the ontology. If the ingredient is not in the ontology, we try to automatically include and classify it in the ontology. Using WordNet the application checks whether the definition of that noun is feasible to belong to any of the categories present in the ontology, if it is, JADAWeb adds it to the system memory, if the system cannot select a unique branch, it asks the user to select the concept to allocate the ingredient.

3.3 Algorithm definition

JADAWeb includes the capability of classifying a new ingredient in the ontology using the information from Wordnet.

Our algorithm first searches an ingredient in the ontology using its name and also using all the synonyms that Wordnet gives for this concept in the same synset. If any of the synonyms belongs to the ontology, the search is done. If not, the system looks for the best node to attach the ingredient.

The parser returns two lists of ingredients. A first list with the ingredients that the user wants to use. And a second list with the ingredients that the user wants to avoid. The ontology update algorithm proceeds as follows: it takes each element of those lists and checks if the item is already included in the ontology. If the item or any of those synonyms are registered then the algorithm is done; if the ingredient does not appear then the system searches the best node to put the new ingrediente, but if the best node is not found, the user guides the system to include the ingredient in the ontology. A branch is defined as the part of the tree whose items have a predecessor or successor relationship to each other. A child is the successor of his parent, grandparents ... and the parent predecessor of children, grandchildren (see Figure 3). To decide the best branch to introduce an ingredient in the ontology, the system uses the word definition in Wordnet. As only nouns are feasible to be ingredients, the system discards the other parts of speech.

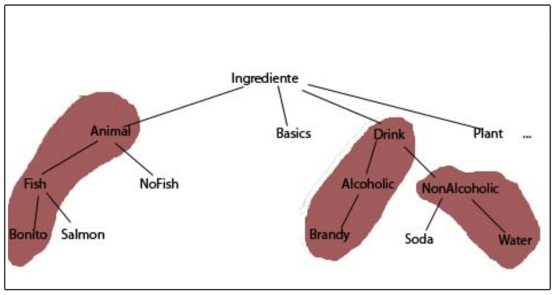


Fig. 3. Fragment of the ontology using by JadaWeb, using two examples of branches within the tree.

All the nouns in the text of the candidate ingredient definition are searched in the ontology.

If every noun in the definition appears in the same branch, then a candidate predecessor to allocate the new ingredient would be the lowest item of the branch that is not a leaf. If it is a leaf, the new ingredient will be attached to the predecessor of the leaf.

If the definition includes words that are in different branches, each branch is checked. The system will select the branch which has more nouns of the definitions of the new ingredient. If two or more branches have the same number of ingredients, the item would have a number of potential predecessor (remember that the system is searching for the best predecessor to hang the new ingredient). This is the only case in which both predecessors are returned to the user. The remaining cases, we obtain a unique solution of the problem, although not necessarily the best.

In this way, the system puts the ingredient as a child of a node according its definition and synonyms. Example: We have tested the algorithm with Shark (the parent the system recomends is fish), saffron (the parents the system recomends ar flower and citrus), yeast (cruciferous) and others, and we obtain a very satisfactory result in an 85 % of the new ingredients included.

3.4 Cases

In this new version of the system we have used the case base that is provided by the organizers of the CCC-10, which is the same than for the CCC-09. The recipe cases are organized in an XML file that is processed using SAX and DOM to extract the information to load it in the system memory.

3.5 Non wanted ingredients treatment

Previous versions of JADACook only remove the ingredients that the user does not want. JADAWeb includes a new algorithm for removing the ingredients that shares the same branch in the tree with the non wanted ingredients. This option is good when the user is not very specific. However, if the user is very specific, or (s)he does not want a special ingredient, our algorithm could remove other ingredients that the user might want. For example, if the user specifies (s)he does not want *sardine*, our system will remove the term *sardine* and other kind of similar fishes like *anchovies* or *herring*.

4 Case-Based reasoning process

4.1 JadaCook2

JADAWeb inherits the similarity function of JADACook2[2], and adds some functionality to it. The original similarity function of JADACook2 works as follows: first, the system takes the ingredients for the wish list of the user, and it searches for recipes that contain these ingredients. If a recipe contains an ingredient it increases by 1.0 the similarity value. The similarity value is normalized by the number of ingredients in the query.

During adaptation JadaCook2 does not reject an ingredient when it is not in the recipe. When it happens, the system searches for ingredients that share the same branch of the original ingredient in the ontology, if it is feasible the algorithm increase by 0.8 the similarity value of this recipe. JadaCook2 searches in all recipes contained by the case base. Finally, the system gets a list of recipes with the information of how similar is each one in according to the ingredients introduced by the user.

The user can also specify in the query the type of meal (s)he wants, the type of diet, the type of cuisine and if (s)he wants some dietary practices. The system adds some ingredients if the user wants a specific type of meal, i.e. if the user selects a *chinese* meal the system will add some ingredients like rice, bamboo, carrot ,ginger, chicken, duck, pig, pork or soy sauce.

4.2 JADAWeb

JADAWeb includes new features. The first one is the ability to look for ingredients to adapt not only among the siblings, but also going up in the hierarchy. The similarity function has the same criteria as the degree of consanguinity of two people. Two siblings have a similarity value of 0.6, because both are descendent of the same parents. Therefore, a rise of level, subtract 0.4 of similarity. For two cousins, subtract 0.8: 0.4 because of the rise to the parent and other 0.4 because of the climb to the grandparent. Going down in the level, not penalize. Thus, another example: the children of a sibling of an ingredient have the same similarity as the sibling (0.6), the similarity of a cousin's children are the same as the cousin and also the same as any of the grandfather's descendants (which do not fall for the same branch of the one that is being compared) which is 0.2 in this case. In this way, you can use other similar ingredients. The number of levels can be parameterized and the default value is 2, i.e, the system only searches among siblings and their descendants.

Table 1. Some examples of the fuzzy table.

Ingredient Origin	**Ingredient targez**	**Value**
Rice	Macaroon	0.4
Stock	Water	0.9
Stock	Bouillon cubes	0.9
Milk	Cream	0.6
Milk	Water	0.6

Another novelty in JadaWeb is a table of similarity between fuzzy ingredients. It is a table with specific values (fuzzy) that gives meaning to say that rice is replaceable by macaroni with a value of 0.4 to add to the similarity function. All these pairs of ingredients are defined in an XML file which indicates what ingredients are similar and the associated fuzzy value. The fuzzy table is not only used to replace pairs of ingredients. The system also can decide replacing one ingredient by a group of similar ingredients: stock instead of water and bouillon cubes with a similarity of 0.9, milk instead of water and cream in a similarity value of 0.6.

4.3 Measure evaluation of the similarity function.

We have proposed the following experiment as a measure for evaluating the similarity function of the system.

We made a survey with a set of people(12), each member tested the system with 5 JADAWeb queries. Each user has introduced his/her a query in natural language in English and we gave them the choice of which level they want. The tests were carried out with the three levels of adaptation of ingredients as shown in Section 4.2. After the CBR process the user evaluates the system results telling us if the output of the system is satisfactory for him/her, both the recipes and the order. Table 2 shows the results of the survey.

Table 2. Results obtained from the measure of evaluation

Similarity level	**Satisfied user**	**Substitutions**
1	90.0%	5.0%
2	70.0%	10.0%
3	40.0%	30.0%

If the similarity level increases then the number of substitutions also increases. With a similarity level that allows only adaptation between ingredients that share the same branch (similarity level = 1), the satisfaction with the given recipe is very high. However, the result is worse when we use higher levels. This is because people do not expect those adapted elements. In various dishes (e.g., replacing the lemon with bread), the result is quite concerned, but we could see that a higher-level change (similarity level ≥ 2) makes the user obtain another recipe recommended that he or she could not expect it. With a deeperd ontology, higher level adjustments would score much more satisfactory results. The result of 70 % for level 2 is good, but the 40 % of level 3 is very low.

5 Results and examples

In this section we show a list of examples of the JADAWeb system. For each query we describe the recipe proposed by the system in the first place. However, for each query the system returns five suggestions ordered by the similarity value.

First Experiment In this experiment we have introduced the list of ingredients for a typical Spanish recipe: paella. We have not specified any type of diet, cuisine or dish and there are no dietary practices.

Q1: "I want to eat rice, saffron, shrimps, chicken, crab, squid but I don't like sugar".
Answer: the first answer is Paella (Paella in a Pot) that contains many of the ingredients contained in the query.

Second Experiment In this experiment, we write a query to obtain any type of pizza with ham, tomato and cheese, and the pizza must not contain onion. We do not specify any type of diet or type of cuisine, but we introduce a type of dish (pizza). Although the term "pizza" is written in the main query in natural language, the system will recognize it as a type of dish.

Q2: "I like pizza with ham, cheese and tomato but I don't want onions".

Answer: The system returns some types of pizza ("Pizza Quiche", "Idiot Bread Pizza", "Breakfast Pizza", "Fruit Pizza" and "Pineapple Cream Cheese

Pizza") which contains ham and cheese or cheese and tomato. We also obtain some pizzas with ingredients adapted changing tomato with pepper and cheese with egg, penalizing in 0.4 each one.

Third Experiment In this experiment, we tested the system with a fruit salad. We do not specify any type of diet or type of cuisine but we specify that the type of dish is a salad.

Q2: "I like to eat some fruit, like orange, lemon and apple but I don't want kiwi".
Answer: The system returns five salads: first a salad that contains a variety of fruits: "Portable Salad Dressing" in addition to other salads as "Creamy Waldorf Salad".

Comparing JADAWeb with Cookiis As a measure of the behaviour of our system, we have compared JADAWeb results with last year winner: CookIIS[7]. We show the similarities and differences between the two systems by comparing the retrieved sets of recipes of both systems.

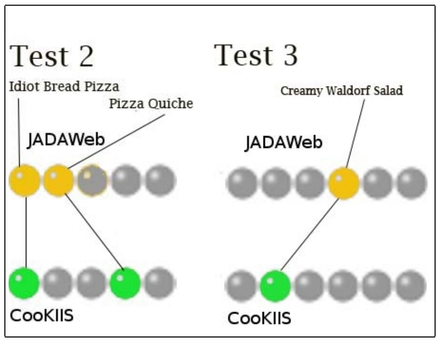


Fig. 4. Comparation with test 2.

In the first query there is no coincidence between the retrieved recipes. JADAWeb retrieves *Paella* in the first place that is a highlight answer because we used the paella ingredients to make this query. Cookiis obtains *Pad Thai* in the first place. In the other queries we have some coincident results between the first 5 suggested recipes. In query 2 the two systems retrieve two common recipes, although in a different order. *Idiot Bread Pizza* is the first option in both. Pizza Quiche is the second in JADAWeb, but the fourth in Cookiis. In query 3 we only have one coincident result: Creamy Waldorf Salad.

6 Conclusions

JADAWeb system improves and extends the JADACook2 [2] system. Its web-based interface gives a broader access to the system and Web accessibility. Accomplishing the requirements of AA or AAA W3C makes the system useful for any person although the person has an inability, i.e blindness.

By using a textual input, we enable a more natural treatment improvement in the interaction with the final user. We can suggest some future work by using a system that allows voice recognition that could be connected with our dependency parser.

JADAWeb extends the possibilities by incorporating a fuzzy table which allows similarity between two distant elements or types in the ontology, and the inclusion of various levels in the adaptation algorithm.

It should be noted the great contribution obtained by using WordNet, it contributes allowing more synonyms for some terms and it extends largely the knowledge that it has, avoiding the limitation of the use of a finite ontology of ingredients.

References

1. Herrera, P.J., Iglesias, P., Romero, D., Rubio, I., Díaz-Agudo, B.: Jadacook: Java application developed and cooked over ontological knowledge. In Schaaf, M., ed.: ECCBR Workshops. (2008) 209–218
2. Herrera, P.J., Iglesias, P., Sánchez, A.M.G., Díaz-Agudo, B.: Jadacook 2: Cooking over ontological knowledge. In: ICCBR Workshops. Computer Cooking Contest. (2009)
3. Recio-García, J.A., D'iaz-Agudo, B., González-Calero, P.A.: Boosting the performance of cbr applications with jcolibri. In: ICTAI, IEEE Computer Society (2009) 276–283
4. Fields, D.K., Kolb, M.A., Bayern, S.: Web Development with Java Server Pages. Manning Publications Co., Greenwich, CT, USA (2001)
5. Buchholz, S., Marsi, E.: Conll-x shared task on multilingual dependency parsing. In: Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL–X). (2006) 149–164
6. Fellbaum, C., ed.: WordNet: an electronic lexical database. MIT Press (1998)
7. Hanft, A., Ihle, N., Bach, K., Newo, R.: Cookiis - competing in the first computer cooking contest. KI **23** (2009) 30–33
8. Lin, D.: Dependency-based evaluation of minipar. In: Proc. Workshop on the Evaluation of Parsing Systems. Granada (1998)
9. Nivre, J., Hall, J., Nilsson, J.: Maltparser: A data-driven parser-generator for dependency parsing. In: In Proc. of LREC-2006. (2006) 2216–2219
10. Sampson, G.: (Briefly noted english for the computer: The susanne corpus and analytic scheme)

TAAABLE 3: Adaptation of ingredient quantities and of textual preparations

Alexandre Blansché, Julien Cojan, Valmi Dufour-Lussier, Jean Lieber, Pascal Molli, Emmanuel Nauer, Hala Skaf-Molli, and Yannick Toussaint

LORIA UMR 7503 CNRS, INRIA, Nancy Universities BP 239
54506 Vandœuvre-lès-Nancy, France, `First-Name.Last-Name@loria.fr`

Abstract. TAAABLE 3 is a textual case-based cooking system which is a contestant in the third computer cooking contest, inheriting most features of its previous versions (TAAABLE and WIKITAAABLE) and adding new features. The reused features are information retrieval and text mining for building a domain ontology and for annotating the ingredient part of the recipes, a semantic wiki in which the whole knowledge base (recipes, domain knowledge, retrieval knowledge, and adaptation knowledge) is encoded, and an inference engine based on minimal generalisations of the query and adaptation by ingredient substitutions. The two new features are related to adaptation. First, ingredient quantities adaptation is handled using conservative adaptation techniques on numerical constraints, permitting to maximally preserve, for instance, the quantity of sugar in the recipe. Second, adaptation is learned from a set of recipes using text mining techniques. Each recipe is analysed and represented as a tree which identifies ingredients, food components and actions that are performed. Formal concept analysis is used on these data in order to find an appropriate way to insert a new ingredient in the final recipe.

Keywords: textual case-based reasoning, case-based cooking, semantic wiki, conservative adaptation, adaptation of ingredient quantities, natural language processing, formal concept analysis, adaptation of recipe preparations

URL of the system: At `http://taaable.fr`, the reader can find a link to the homepage of the TAAABLE 3 system.

1 Introduction

TAAABLE and WIKITAAABLE have participated in the first and second computer cooking contests [1, 2]. TAAABLE 3 is the 2010's version of TAAABLE. It inherits from features of the two previous versions of the system, as described in Section 2. In particular, it reuses the semantic wiki of WIKITAAABLE and improves it (Section 3). It also has two original features. The first one is related to ingredient quantity adaptation using revision-based adaptation in metric spaces. It is presented in Section 4 and is based on previous work [3]. The second one is related to the textual adaptation of recipe preparations. This feature is based on natural language processing and formal concept analysis. It is presented in Section 5 and detailed in [4].

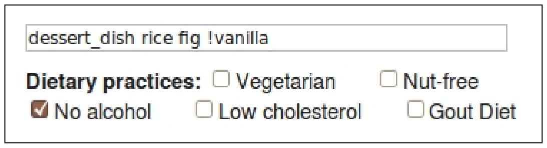
2 Previous work on the TAABLE project

The knowledge-based system TAABLE is composed of a CBR inference engine, a knowledge base stored in a semantic wiki, and some tools for acquiring or editing knowledge. These tools have a great importance for the different versions of the system, but no major improvements have been made on these tools this year, so they are not detailed here.

The knowledge base stored and edited in a semantic wiki is composed of a cooking domain ontology $\mathcal{O}$, the recipe base indexed by concepts of $\mathcal{O}$ (thanks to an annotation process), and some knowledge useful for retrieval and for adaptation. The ontology $\mathcal{O}$ contains about 5000 concepts organised in several hierarchies, including an ingredient hierarchy and a dish type hierarchy. For example, the concept `mango` is under the concept `tropical_fruit`: every mango is a tropical fruit. Some improvements on the wiki concerning ingredient properties and recipe formalisation are detailed in Section 3.

A query Q to the CBR engine is expressed as a conjunction of literals, where a literal is either a concept of $\mathcal{O}$ or the negation of such a concept. For example, let us consider the following request, expressed in natural language: "I want an alcohol-free dessert with rice and figs, but without vanilla." This request is expressed by

$$Q = \neg\texttt{alcohol} \land \texttt{dessert\_dish} \land \texttt{rice} \land \texttt{fig} \land \neg\texttt{vanilla} \qquad (1)$$

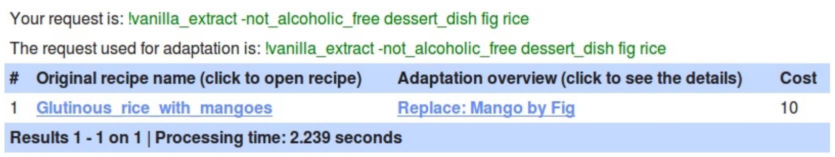
or  in the interface

dessert_dish rice fig !vanilla

Dietary practices: ☐ Vegetarian ☐ Nut-free ✓ No alcohol ☐ Low cholesterol ☐ Gout Diet

Since the recipes are indexed by concepts from $\mathcal{O}$, finding the recipes that exactly match Q consists in finding the recipes R such that (1) for each positive literal `C` of Q, there exists a concept `D`, under `C` in $\mathcal{O}$, such that R is indexed by `D`; (2) for each negative literal $\neg$`C` of Q, there exists no concept `D` under `C` in $\mathcal{O}$, such that R is indexed by `D`. When such an exact match occurs, the matching recipes are returned by the retrieval process and no adaptation is required. Otherwise, the retrieval process aims at finding a minimal generalisation function Γ such that there exists at least one recipe R exactly matching the generalised query $\Gamma(Q)$. Γ is a composition of substitutions `A` $\rightarrow$ `B` where `A` is a direct subconcept of `B` in $\mathcal{O}$. The minimality of the generalisation Γ is computed thanks to a cost function, described in [2]. For example, the recipe R named "Glutinous rice with mangoes" is indexed by the conjunction $\texttt{dessert\_dish} \land \texttt{coconut\_cream} \land \texttt{mango} \land \texttt{salt} \land \texttt{sugar} \land \texttt{rice} \land \texttt{sesame\_seed}$.The retrieval process retrieves this recipe for the query (1) with generalisation $\Gamma = \texttt{fig} \rightarrow \texttt{tropical\_fruit}$.

Thus, the result of retrieval is a set of recipes R matching a generalisation $\Gamma(Q)$ of the initial query (when there is an exact match, Γ is the identity function). Given R, Γ, and Q, the adaptation of R to Q consists simply in substituting some concepts indexing R , by (a) following the match from R to $\Gamma(Q)$, and (b) applying the specialisation function Γ^{-1}. For example, the adaptation of the recipe "Glutinous rice with mangoes" will consist in (a) substituting `mango` with `tropical_fruit` and (b) substituting `tropical_fruit` with `fig`, which is equivalent to the substitution of mangoes with figs (Figure 1). An adaptation may be composed of several substitutions but, in the current version of TAAABLE, each substitution of a single ingredient replaces it with a single ingredient. This is linked with the search space of the function Γ: generalisations of conjuncts (e.g., $\Gamma = a \wedge b \to c$) do not belong to this search space. However, the use of adaptation rules for TAAABLE would lead to substitutions of several ingredients by several ingredients (see [2]), though TAAABLE 3 does not use such rules.



Your request is: !vanilla_extract -not_alcoholic_free dessert_dish fig rice

The request used for adaptation is: !vanilla_extract -not_alcoholic_free dessert_dish fig rice

#	Original recipe name (click to open recipe)	Adaptation overview (click to see the details)	Cost
1	Glutinous_rice_with_mangoes	Replace: Mango by Fig	10

Results 1 - 1 on 1 | Processing time: 2.239 seconds

Fig. 1. Retrieved recipe with the proposed substitution for the query example.

3 New content of the TAAABLE's Semantic Wiki

3.1 Ingredients

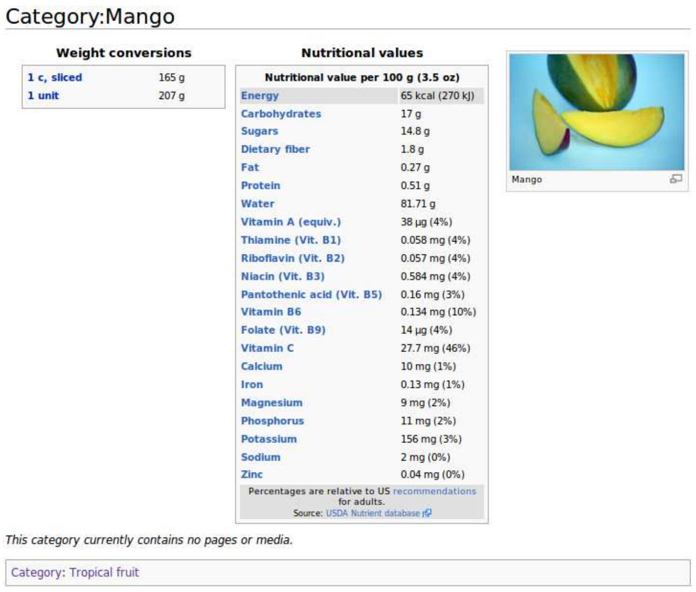
In the previous versions of TAAABLE, ingredients were represented by wiki categories. Each ingredient is encoded as a wiki category in which the ingredient is only described according to its relation with more generic categories. For example, the category `Mango` is described as a sub-category of the category `tropical_fruit`.

In the version of the TAAABLE 3 wiki, ingredient categories have been enriched by additional knowledge extracted from the USDA Nutrient database (`http://www.nal.usda.gov/`). An ingredient is now described by its nutritional values (sugar, fat, protein, vitamins, etc.) and some weight conversion equivalences, two types of knowledge required for the adaptation of ingredient quantities (cf. Section 4). Ingredient nutritional values are represented using semantic wiki properties and weight conversions thanks to semantic wiki multi-value properties. Multi-value properties are needed because a weight conversion is based five element: an amount, a unit, a property, an ingredient, and the equivalent weight in grams, (e.g. one cup of sliced mango is 165g). For linking nutritional values and weight conversions to their corresponding wiki categories, a semi-automatic alignment has been processed. This alignment makes the link between the ingredient label in the USDA Nutrient database and its label in the wiki as a category. As the mapping is incomplete, some of the ingredients in the wiki do not possess such information.

Figure 2 shows the content of the category `Mango`. One can see the knowledge about weight conversion and nutritional values organised into tables. Users might also add a picture of the ingredient.

3.2 Recipes

The description of recipes has also been improved. A natural language analysis process, described in Section 5, is used to add a formal description of the preparation of a recipe.



Category:Mango

Weight conversions

1 c, sliced	165 g
1 unit	207 g

Nutritional values

Nutritional value per 100 g (3.5 oz)	
Energy	65 kcal (270 kJ)
Carbohydrates	17 g
Sugars	14.8 g
Dietary fiber	1.8 g
Fat	0.27 g
Protein	0.51 g
Water	81.71 g
Vitamin A (equiv.)	38 μg (4%)
Thiamine (Vit. B1)	0.058 mg (4%)
Riboflavin (Vit. B2)	0.057 mg (4%)
Niacin (Vit. B3)	0.584 mg (4%)
Pantothenic acid (Vit. B5)	0.16 mg (3%)
Vitamin B6	0.134 mg (10%)
Folate (Vit. B9)	14 μg (4%)
Vitamin C	27.7 mg (46%)
Calcium	10 mg (1%)
Iron	0.13 mg (1%)
Magnesium	9 mg (2%)
Phosphorus	11 mg (2%)
Potassium	156 mg (3%)
Sodium	2 mg (0%)
Zinc	0.04 mg (0%)

Percentages are relative to US recommendations for adults.
Source: USDA Nutrient database

Mango

This category currently contains no pages or media.

Category: Tropical fruit

Fig. 2. Mango category.

All the steps of the preparation are inserted in the wiki as a list of items. A graphical representation of the preparation is also included.

Figure 3 shows the final part of the formal preparation of "Glutinous rice with mangoes" as a graph. This graph is included in the wiki page of the corresponding recipe.

4 Adaptation of ingredient quantities with belief revision

4.1 Integration into TAAABLE

The use of TAAABLE remains as presented in Section 2, in particular the queries must be given in the same format. The retrieval and a first adaptation step that computes the set of substitutions ignore ingredient quantities. With the example query given in Section 2 the retrieved recipe is "Glutinous rice with mangoes", and the first adaptation step recommends to substitute mangoes with figs.

The adaptation on ingredient quantities is triggered when calling the adaptation result page, it takes as inputs:

- The source recipe with its list of ingredients with their amounts and units.
- A set of ingredient substitutions of the form $\texttt{from\_ing}_i \rightarrow \texttt{with\_ing}_i$.
- Numerical data about the ingredients: unit conversions (e.g. the weight of a cup of mango) and nutritional data (e.g. how much energy in calories a gram of mango contains).

The result is shown in Figure 4.

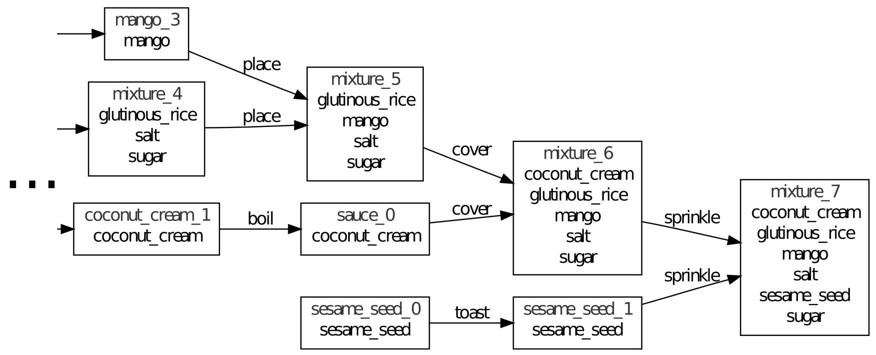
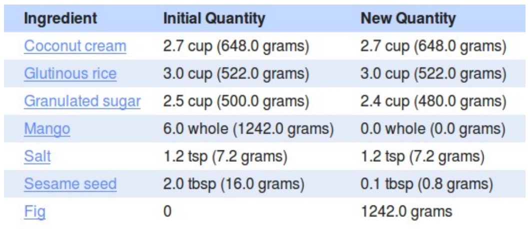


Fig. 3. Excerpt of the formal description of the preparation for "Glutinous rice with mangoes".



Ingredient	Initial Quantity	New Quantity
Coconut cream	2.7 cup (648.0 grams)	2.7 cup (648.0 grams)
Glutinous rice	3.0 cup (522.0 grams)	3.0 cup (522.0 grams)
Granulated sugar	2.5 cup (500.0 grams)	2.4 cup (480.0 grams)
Mango	6.0 whole (1242.0 grams)	0.0 whole (0.0 grams)
Salt	1.2 tsp (7.2 grams)	1.2 tsp (7.2 grams)
Sesame seed	2.0 tbsp (16.0 grams)	0.1 tbsp (0.8 grams)
Fig	0	1242.0 grams

Fig. 4. Ingredient quantity adaptation for the example. The evolution of ingredient quantities are under the influence of several criteria, in particular the preservation of nutritional components, see Section 4.2. This explains, for instance, the reduction of the sugar quantity: figs are slightly sweeter than mangoes. The reduction of the sesame seed quantity can also be explained by the preservation of nutritional components (fibres, calcium, magnesium, ...), though this justification can be discussed (see the distance definition in Section 4.2).

4.2 How does it work

This adaptation step follows the $\dotplus$-adaptation approach [3] applied to an attribute-value formalism. The cases are formalised as constraint satisfaction problems over the attributes; solving a case amounts to specifying the value of all its attributes.

Representation space. Only numerical attributes are considered:

– The ingredient quantities in grams and in another unit if they were given in this other unit. The ingredient values will be stored in variables $[\texttt{ingredient}]_{[\texttt{unit}]}$, for instance as mangoes are given in cups, two variables will be used for mangoes: $\texttt{mango}_{\text{cups}}$ and $\texttt{mango}_{\text{grams}}$. In addition to the ingredients expressly given in the recipe source and in the substitutions, their generalisations are considered as well. For instance, as `fig` is classified as a `tropical_fruit` and `tropical_fruit` is a subclass of `fruit` there will be a dimension for each in the representation space.

– All the nutritional components that appear in the wiki (see fig. 2) represented by the variables $\texttt{lipid}_{\text{grams}}$, $\texttt{energy}_{\text{Kcal}}$, etc.

Domain Knowledge. The domain knowledge is expressed by a set of linear constraints:

– Unit conversion, for instance: $\texttt{mango}_{\text{grams}} = 165\ \texttt{mango}_{\text{cups}}$ (a cup of mango contains 165 grams of mango).
– Ingredient hierarchy, for instance: $\texttt{tropical\_fruit}_{\text{grams}} = \texttt{fig}_{\text{grams}} + \texttt{mango}_{\text{grams}}$ (as in this adaptation session no other tropical fruit are considered).
– Nutritional data calculus, for instance: $\texttt{energy}_{\text{Kcal}} = 0.65\ \texttt{mango}_{\text{grams}} + 0.74\ \texttt{fig}_{\text{gram}} + 0.97\ \texttt{glutinous\_rice}_{\text{grams}} + \dots$

Source case. The solution of a case (i.e. its ingredients quantities) is expressed as well as a set of linear constraints. For instance, in the source recipe “Glutinous rice with mangoes”: $\texttt{rice}_{\text{cups}} = 3$.

Target Case. For each substitution $\texttt{from\_ing} \rightarrow \texttt{with\_ing}$, a constraint $\texttt{from\_ing} = 0$ is added to the target case. This transcription of the substitutions is based on the fact that the substitution has been obtained by a generalisation-specialisation adaptation path (see Section 2): $\texttt{from\_ing} \overset{\text{generalisation}}{\longrightarrow} \texttt{parent} \overset{\text{specialisation}}{\longrightarrow} \texttt{with\_ing}$. The domain knowledge contains then a constraint: $\texttt{parent}_{\text{grams}} = \texttt{from\_ing}_{\text{grams}} + \texttt{with\_ing}_{\text{grams}}$. Stating that $\texttt{from\_ing}_{\text{grams}} = 0$ in the target case while preserving $\texttt{parent}_{\text{grams}}$ will entail the substitution from $\texttt{from\_ing}$ with $\texttt{with\_ing}$.

As the target case is also expressed as a set of linear constraints over ingredient quantities, the request could contain constraints of the form $\texttt{fig}_{\text{gram}} \leq 500$ to state that only 500 grams of figs are available. However this is not yet implemented in the interface.

Distance. The adaptation cost is evaluated according to a distance $\texttt{dist}$ between the source case and the proposed solution for target. This distance is taken as a weighted sum of each value change (i.e. $\texttt{dist}(x, y) = \sum_i w_i |y_i - x_i|$ where i stands for one of the dimensions).

Not only ingredient quantities are taken into account in $\texttt{dist}$ (see the representation space definition), in particular, terms corresponding to nutritional components are included as well. This entails some quantities adjustments that are not explicitly related to the substitution, see the sugar and sesame seeds quantity evolution in Figure 4.

The weights are w_i are set to 1 by default, this choice leads to questionable value changes; (see sesame seeds quantity in Figure 4); a more relevant valuation should be considered in future work.

Computation. $\dot{+}$-adaptation relates the CBR principle of minimal change for adaptation to the theory of belief revision [5]. A revision operator $\dot{+}$ encodes some meaning to the expression “minimal change” in the context of making two formulas consistent.

Here, the ingredient quantities given by the source recipe are revised by the target case constraints. $\dotplus$ is defined from `dist`: the adaptation process returns the quantities that are closest to the source recipe quantities according to the `dist` while remaining consistent with target constraints. The computation follows the reduction of $\dotplus$-adaptation computation to linear programming proposed in [3].

5 Adaptation of the preparation: natural language processing and formal concept analysis

At this stage we have a satisfactory glutinous fig rice recipe as far as picking the right ingredients is concerned. But if one simply substitutes "fig" for "mango" wherever it appears in the preparation text, as a user of the system might be led to believe we suggest, the apprentice cook will be facing an insurmountable problem. The recipe text demands that the mangoes be peeled, sliced lengthwise and pitted, all operations ranging somewhere between awkward and utterly impossible to accomplish when it comes to figs.

We introduce a method using natural language processing techniques and formal concept analysis to benefit from information hidden in the preparation text and improve the adaptation stage. The guiding idea behind this work is that, for each ingredient, there exist preparation "prototypes" describing the different ways in which the ingredient can be used. What we then need to adapt our mango rice recipe into a fig rice recipe is a mango prototype, a variety of fig prototypes, and some similarity metric to pick the fig prototype that is closest to our mango prototype. The algorithms that perform those tasks are defined in more details in [4].

5.1 From Natural Language to Formal Representation

Acquiring meaningful case representations from textual sources remain one of the foremost challenges of textual case-based reasoning[6]. Because a recipe basically consists of taking a set of ingredients and combining them in specific ways until we obtain something edible, it seems natural to represent them as trees as shown in Fig. 3. Each node represents the state of a food component at a given time, and each edge $\langle \alpha, \beta \rangle$ represents an action applied on food component α that yields a new food component β. A set of edges sharing the same head (β) represents a "combining action", i.e., an action applied to many food components at once that yield out one new food component.

This representation is used even as it is being built to resolve some of the difficult situations that occur frequently in recipe texts. In a sentence such as "Peel the mangoes, slice [the mangoes] lengthwise and remove the pits [from the mangoes]", it is easy as humans to understand that mangoes are being sliced and pitted, but some heuristic is needed is order for a computer to realise that. We have attributed an arity to each action in order to be able to tell when an argument is missing. Whenever this happens, it is taken that the missing argument is in fact the last node that was added to the tree. In that way, we are able to deal with anaphora, the phenomenon wherein a different word, or no word at all as it might be, is used to represent an object.

Other types of anaphora appear in our mango rice recipe. The expression "seasonings ingredients" clearly refer to some set of food components, so we use the ingredient hierarchy to find all the nodes of the tree that fit under the "seasonings" category. A phrase such as "cover with sauce" is a lot trickier because there is no obvious clue either in the text or in the ontology which food component this "sauce" may be. We built, from the analysis of thousands of recipes, "target sets" of ingredients that usually appear in the food components being referred to by word such as "sauce" or, say, "batter". Because a quantity of recipes include coconut cream-based sauces (and few contain, say, rice-based sauces) we are able to guess rightly that the food component containing coconut cream really is the one the recipe author had in mind when he wrote "sauce".

Once the tree representation is built, it is trivial to identify the sequence of actions being applied to any ingredient of a given recipe. We define the prototype of an ingredient by the actions applied to this ingredient, up to and including the first combining action. In practice this means we only use the leftmost part of the tree. This has the limitation that we may be missing relevant actions, but we are also getting rid of actions that, while being applicable to certain mixtures containing an ingredient, may not make sense when applied directly to this ingredient. In the mango rice, we are keeping the *chill, peel, slice lengthwise, remove pits,* and *place on top* actions, and getting rid of *cover with, sprinkle with,* and *serve* which are indeed too generic to be interesting. As an added benefit, the leftmost part of the tree is the one we are able to build with the higher reliability at this time.

5.2 Recipe Adaptation Using Formal Concept Analysis

The mango prototype used in the mango rice is, then, characterised by the set of actions { `chill`, `peel`, `slice`, `remove-pit`, `place` }. In the recipe base, there are two recipes with figs. Recipe nr. 1163 sports a { `halve`, `sprinkle-over`, `dot-with`, `cook`, `brown`, `place` } fig prototype, and nr. 53 a more modest { `cut`, `combine` }. In order to simplify the processing and avoid discriminating near-synonyms, we can group classes of actions using the action hiearchy. This has the effect of grouping `peel` with `remove-pit`, `cut` with `slice`, and `cook` with `brown`.

To select the more appropriate fig prototype, a concept lattice from the fig prototypes is built with the mango prototype merged in. This is similar to techniques used in document retrieval by [7] wherein a lattice is built according to keywords found in documents and a query is merged in. The formal context contains the set of fig prototypes as its set of objects and its set of attributes is the set of action classes applicable to figs. It is concatenated (see [8]) with a "query" context consisting of the mango prototype from the retrieved recipe as its only object and the set of action classes applied to this mango as its attributes, yielding the formal context shown in Table 1.

Using formal concept analysis [9], the concept lattice shown in Figure 5 is built from Table 1. All prototypes having actions in common will appear together in the extension of at least one concept. The "lower" this concept is in the lattice, the more attributes the prototypes have in common. A set of candidate fig prototypes is taken from the extension of all concepts "immediately above" the lowest concept with `mango` in its extension. Whenever there is more than one candidate, the one that minimises the distance between its attributes and the mango's attributes is selected. In the example, `fig_53`

	cut	halve	remove	place	pour	dot	cook	chill	combine
fig_1163		×		×	×	×	×		
fig_53	×								×
mango	×		×	×				×	

Table 1. Formal context of figs and a mango.

is selected since, while both `fig_53` and `fig_1163` appear directly above `mango`, replacing `mango` with `fig_1163` would require removing three actions and adding four, whereas replacing it with `fig_53` would require only removing three actions.

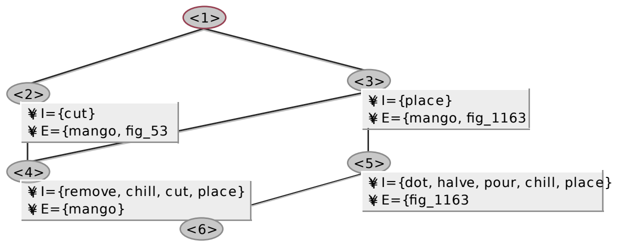


Fig. 5. Concept lattice corresponding to the context of Table 1. Each node (formal concept) is a set of objects O (extension, noted "E") and a set of attributes A (intension, noted "I") such that all objects in O have all attributes in A in common in exclusion to any other. They are ordered according to the inclusion of their extension.

The process is then finalised by replacing the textual parts of the retrieved recipe dealing with mangoes with the parts of recipe nr. 53 dealing with figs:

> [. . .] Blend the sauce ingredients in a pot and heat until it just reaches the boiling point. Let cool. ~~Peel the mangoes, slice lengthwise and remove the pits.~~ Cut figs into wedges. Divide the rice mixture among 6 plates. [. . .]

6 Conclusion

TAABLE 3 is a textual case-based cooking system which is a contestant in the third computer cooking contest, inheriting most features of its previous versions (TAABLE and WIKITAABLE) and adding new features. The reused features are information retrieval and text mining for building a domain ontology and for annotating the ingredient part of the recipes, a semantic wiki in which the whole knowledge base (recipes, domain knowledge, retrieval knowledge, and adaptation knowledge) is encoded, and an inference engine based on minimal generalisations of the query and adaptation by ingredient substitutions. The two new features are related to adaptation. First, ingredient quantities adaptation is handled using conservative adaptation techniques on numerical constraints, permitting to maximally preserve, for instance, the quantity of sugar in the recipe. Second, adaptation is learned from a set of recipes using text mining techniques. Each recipe is analysed and represented as a tree which identifies ingredients, food components and actions that are performed. Formal concept analysis is used on these data in order to find an appropriate way to insert a new ingredient in the final recipe.

The evaluations of the whole system and of its components constitute an important ongoing work. Since there is no known computing test asserting whether a recipe is cookable (i.e., giving a satisfactory dish, from a human average taste viewpoint), these evaluations are subject studies.

Acknowledgements. The participants of the TAABLE project wish to thank the organisers 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. They also wish to thank the reviewers for their work which has impacted the state of the paper and will impact the future work on TAABLE.

References

1. Badra, F., Bendaoud, R., Bentebitel, R., Champin, P.A., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J., Meilender, T., Mille, A., Nauer, E., Napoli, A., Toussaint, Y.: Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In: ECCBR Workshops, Workshop of the First Computer Cooking Contest. (2008) 219–228
2. Badra, F., Cojan, J., Cordier, A., Lieber, J., Meilender, T., Mille, A., Molli, P., Nauer, E., Napoli, A., Skaf-Molli, H., Toussaint, Y.: Knowledge Acquisition and Discovery for the Textual Case-Based Cooking system WikiTaaable. In: Workshops of the 8th International Conference on Case-Based Reasoning (ICCBR-09). (2009)
3. Cojan, J., Lieber, J.: Belief Merging-based Case Combination. In David C. Wilson, Lorraine McGinty, eds.: 8th International Conference on Case-Based Reasoning - ICCBR 2009 Case-Based Reasoning Research and Development. Volume 5650 of Lecture Notes in Computer Science., Seattle United States, Springer Berlin (07 2009) 105–119 The original publication is available at www.springerlink.com.
4. Dufour-Lussier, V., Lieber, J., Nauer, E., Toussaint, Y.: Text adaptation using formal concept analysis. In Bichindaritz, I., Montani, S., eds.: Case-Based Reasoning Research and Development. Volume 6176 of Lecture Notes in Artificial Intelligence., Alessandria, Italy, Springer (2010) 96–110
5. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet functions for contraction and revision. Journal of Symbolic Logic **50** (1985) 510–530
6. Asiimwe, S., Craw, S., Wiratunga, N., Taylor, B.: Automatically acquiring structured case representations: The SMART way. In: Applications and Innovations in Intelligent Systems XV, Springer (2007)
7. Carpineto, C., Romamo, G.: Order-Theoretical Ranking. Journal of the American Society for Information Science **51**(7) (2000)
8. Bendaoud, R., Napoli, A., Toussaint, Y.: Formal Concept Analysis: A unified framework for building and refining ontologies. In Aldo Gangemi, Jérôme Euzenat, eds.: 16th International Conference on Knowledge Engineering and Knowledge Management - EKAW 2008. Volume 5268., Acitrezza, Catania Italie, Springer Berlin / Heidelberg (2008) 156–171
9. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Heidelberg (1999)

Adaptation of Cooking Instructions Following the Workflow Paradigm

Mirjam Minor, Ralph Bergmann, Sebastian Görg, Kirstin Walter

Business Information Systems II
University of Trier
54286 Trier, Germany
{minor|bergmann|goer4105|walt4701}@uni-trier.de

Abstract. Former CCC systems have considered mainly the ingredients of cooking recipes. This paper contributes to the open challenge with a novel approach that targets the preparation instructions. Our demo system ***CookingCakeWf*** employs the workflow paradigm to represent and adapt cooking instructions in the form of cooking workflows. Adaptation cases on former modification episodes of cooking workflows are reused for current change requests. A small experimental evaluation with cooking workflows created from pasta recipes of the CCC 2010 recipe base provides first insights into case-based adaptation of cooking workflows.

1 Introduction

The International Computer Cooking Contest (CCC) is going into its third year. Cooking recipes are given in a case base to be retrieved and reused with the assistance of a computer system. The participating teams compete by providing their systems with recipes in order to answer cooking requests. The main task has been nearly the same over the years while additional challenges are changing to address recent developments in Case-based Reasoning. This year, the contest includes an open challenge for the first time. This paper addressed the open challenge by focusing on the cooking instructions and on how to adapt them to cooking requests. For this, the textual cooking instructions are formally represented. We employ the workflow paradigm to represent cooking instructions as cooking workflows. A case-based adaptation method [1] developed for agile workflow technology is extended to adapt cooking workflows. Both the control flow (of cooking steps) and the data flow (of ingredients and their products) are considered. The latter has not yet been addresses by previous work on automated workflow adaptation [1]. The output of our CCC open challenge system **CookingCakeWf** is an adapted workflow but not yet conventional cooking instruction that consists of natural language sentences. The automated generation of text from the workflow representation is a topic of future work as well as the automated transformation of the original textual description of a cooking description into a formal workflow. Additionally, we will participate in the CCC main challenge with our 2009 system **CookingCake**[1] which is described in [2].

The paper is organized as follows: Section 2 introduces a formal representation of the cooking workflows. Section 3 presents a novel case format for adaptation knowledge. Section 4 describes the methods for applying the adaptation knowledge to the workflows. Section 5 demonstrates the feasibility of the approach by means of a first experimental evaluation.

```
<RECIPE>
        <TI>Baked Spaghetti</TI>
        <IN>1 c Chopped onion</IN>
        <IN>1 c Chopped green pepper</IN>
        <IN>1 tb Butter/margarine</IN>
        <IN>1 cn (28 oz.) Tomatoes with liquid; cut up</IN>
        <IN>1 cn (4 oz.) Mushroom stems and 1 piece - drained</IN>
        <IN>1 cn (2-1/4 oz.) Ripe olives, sliced and drained</IN>
        <IN>2 ts Dried oregano</IN>
        <IN>1 lb Ground beef, browned and drained (optional)</IN>
        <IN>12 oz Spaghetti, cooked & drained</IN>
        <IN>2 c (8 oz.) shredded cheddar cheese</IN>
        <IN>1 cn (10-3/4 oz) Condensed cream</IN>
        <IN>Mushroom soup; undiluted</IN>
        <IN>1/4 c Water</IN>
        <IN>1/4 c Parmesan cheese, grated</IN>
        <PR>In a large skillet, saute onion and green pepper in butter until
        tender. Add tomatoes, mushrooms, olives and oregano. Add ground
        beef if desired. Simmer, uncovered, for 10 minutes. Place half of the
        spaghetti in a greased 13-inch x 9-inch x 2-inch baking dish. Top
        with half of the vegetable mixture. Sprinkle with 1 cup of cheddar
        cheese. Repeat layers. Mix the soup and water until smooth; pour
        over casserole. Sprinkle with Parmesan cheese. Bake, uncovered, at
        350 degrees for 30-35 minutes, or until heated throughout. </PR>
</RECIPE>
```

Fig. 1: Sample cooking recipe for Baked Spaghetti from the CCC recipe base.

2 Cooking workflows

Cooking recipes are usually described by a list of ingredients and a cooking instruction. Fig. 1 shows a sample recipe description of a pasta recipe from the CCC recipe base in XML. The title of the recipe (<TI>…</TI>) is followed by a list of ingredients (<IN>…</IN>) and a textual cooking instruction (<PR>…</PR>). Many internet platforms on cooking use a comparable XML representation of the cooking data.

This semi-structured representation has been transformed into a more formal workflow representation for our CCC system by a human expert. Workflows describe processes by means of *tasks* (activities) that are organized within a control flow. The

[1] See also http://proj2.wi2.uni-trier.de/

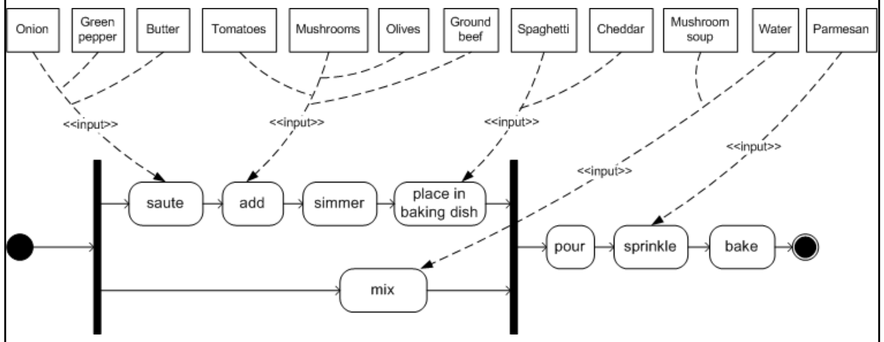
CAKE workflow modeling language (see [3]) consists of several types of *workflow elements* whose instances form block-oriented control flow structures. A workflow element can be a task, a *start symbol*, an *end symbol*, or a *control flow element* like an AND, XOR, loop etc. Control flow elements always define related blocks (AND-block, XOR-block etc.). Blocks cannot be interleaved but they can be nested. In addition to the control flow, a workflow can have a data flow, which is the flow of *data objects* from one workflow element to a successor workflow element. The data flow is specified by means of *data links* that connect the data objects with workflow elements. In a *cooking workflow*, every ingredient is considered a data object while the preparation steps form the workflow tasks within a control flow. Preparation steps and control flow are extracted from the textual cooking instruction as well as the data links that connect the data objects (ingredients) with the tasks (preparation steps). Fig. 2 a) illustrates this by showing the workflow representation for the sample recipe from Fig. 1.

Some challenges concerning the control flow (1), the data objects (2), and the data links (3) had to be solved during the transformation:

(1) **Control flow**: The cooking instructions suffer from both a granularity and a paraphrase problem. *Granularity* means that some authors describe instruction steps in great detail (e.g. peel and chop onions, melt butter in a large skillet, add onions and beef, wait until brown) while others aggregate minor steps into higher-level activities (e.g. brown beef and onions). Although granularity models have been discussed in the workflow literature [4], we have decided to avoid the granularity problem here by leveling the granularity of workflow tasks by hand. However, the granularity problem has to be solved in future work on the automated transformation of recipes into the workflow representation. The second challenge when defining the control flow of cooking tasks is the vocabulary. The *paraphrase problem* occurs here, which concerns the non-uniform wording used in texts from different authors (e.g. sauté vs. brown). Using a task ontology would solve this problem to a large extent (compare the discussion at the end of Section 4).

(2) **Data objects**: The representation of data objects (ingredients) raises some issues concerning the preparation states, amounts, and double (or multiple) occurrences of ingredients. Although these issues have been addressed by former CCC systems, they deserve special attention in the context of workflow adaptation. The *preparation states* of the ingredients (one piece, chopped, sautéed, cooked etc.) could be represented by multiple data objects (whole onions, sliced onions, chopped onions, etc.) or by the same data object with the preparation state represented as a variable property. The *amounts* of the ingredients can be represented as a data object property as well. *Double (or multiple) occurrences* of ingredients in different roles (e.g. butter to brown onions and flakes of butter for a topping) can be represented either as different objects (e.g. butter and flakes of butter, or butter one and butter two) or as a single object. For simplicity reasons, we have chosen the latter and to abstain from representing any properties (amounts, state of

a) Original workflow:

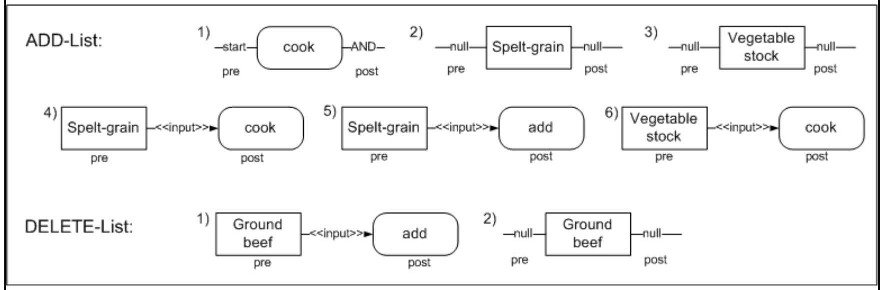


b) Change request:
Replace ground beef by spelt-grain

PROBLEM PART

SOLUTION PART

c) Adaptation steps:



d) Adapted workflow:

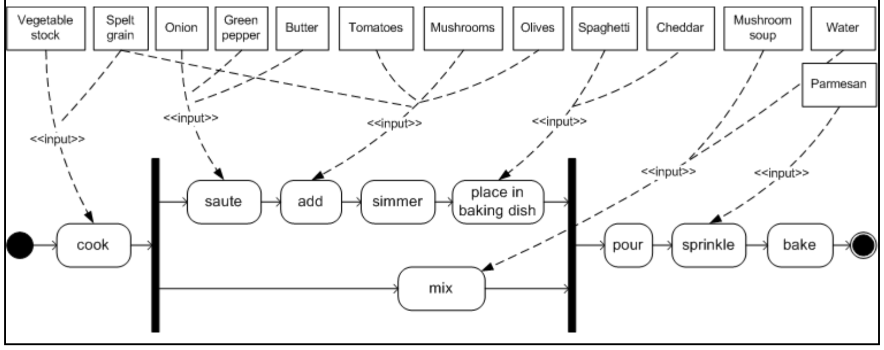


Fig. 2: Adaptation case on the sample cooking recipe from Fig. 1.

preparation) at the moment. In the future, this approach can be improved by an extended representation.

(3) **Data links**: Preparation tasks consume ingredients and produce transformed ingredients or aggregates of ingredients. Data links connect data objects with tasks. In general workflows, the data objects can play the role of *input or output data* for the tasks. A complete model would include an input connection between a data object and every task using it as an input and an output connection for every product resulting from a task. Data objects for aggregated objects (e.g. a document collection made from two different forms and a copy of a certificate, or a sauce made from onions and tomatoes) would be created by the task that aggregates the objects. However, textual cooking instructions tend to be underspecified due to the human principle of economy in language. Our analysis of the CCC recipe base has shown that the data links for the input of ingredients are more frequently described than for the outputs. Transformed ingredients and aggregates occur very seldom in the instructing texts. Hence, we consider inputs only at their first occurrence, except for those ingredients that occur in multiple roles as described above.

3 Adaptation cases for cooking workflows

An adaptation case represents the knowledge from a previous adaptation episode of a workflow. We refer to our previous work [1] for a discussion of related work. Our case representation consists of a problem and a solution part as follows: The *problem part* contains a semantic description of the *change request* (specifies a desire to modify the recipe, e.g. to replace ground beef by spelt-grain (an ancient grain becoming very popular again) to reduce the fat content) and the *original workflow* prior to the adaptation. The *solution part* contains the *adapted workflow* and the description of the *adaptation steps* that have been executed to transform the original workflow into the adapted workflow (added and deleted workflow elements, e.g. delete the ground beef with its data links and add new data objects for spelt-grain and stock plus the preparation steps for preparing them and the required data links). Fig. 2 shows a sample adaptation case by which baked spaghetti have been made according to the original recipe given in Fig. 2 a) but with spelt-grain instead of ground beef. Fig. 2 d) shows the adapted workflow: The spelt-grain has to be cooked in stock before it can be added. Aggregates and preparation states are omitted in the current representation. A complete model would include an aggregated object for "cooked spelt-grain" with an output data link from the task "cook" and an input data link to the task "add".

As in our previous work [1], the adaptation steps are organized within an *add* and a *delete* list. In extension of the previous work, an add or delete step may refer to a workflow element, a data object or a data link. The add and delete lists organize the add and delete steps in the form of chains. A chain encapsulates a set of adaptation steps on connected workflow elements, data objects, and data links. A chain is intended to be either fully applied or not applied at all while reusing the adaptation case. Furthermore, each chain records a pair of *anchors*. A *pre anchor* is the workflow element or data object (in the original workflow) after which the add or delete steps from the chain have been applied. A special ‘null’ element is used as anchor in case an appropriate data or workflow element is not available, for instance if a data object is newly created or deleted. A *post anchor* is the workflow element from the original workflow following the last element of the chain. We assume that data objects do not play the role of a post anchor at the moment. This might change in the future in case output data links will be included. Hence, the pre anchor describes the position after which the adaptation starts and the post anchor describes the first position in the original workflow that is not affected by the adaptation described in the chain. Fig. 2 c) illustrates this by sample add and delete lists. We have decided to model very fine granular chains in order to apply as much of the adaptation steps as possible. Another modeling strategy would be to maximize the chains, for instance to combine the add chains 1, 2, 3, 4, and 6 to one chain with one anchor pair. This would reduce the effort for the anchor matching at the cost of a higher modeling effort to revise the workflow.

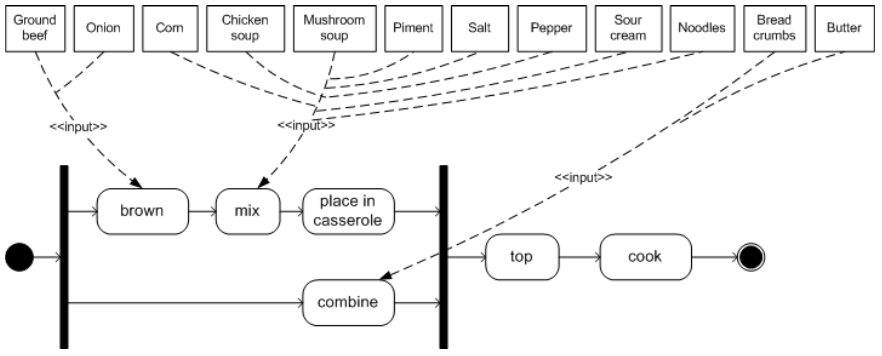
4 Applying adaptation cases to new situations

We now focus on the retrieve and reuse phase of the CBR cycle [5]. The revise and retain phases are not yet considered to be automated. The retrieval method for workflow adaptation cases that may be reused for the adaptation of new cooking workflows is also not within the scope of this paper. We refer to [1] for a discussion of similarity measures that come into consideration for the retrieve phase. Fig. 3 depicts a sample query for which the adaptation case shown in Fig. 2 would be applicable.

The reuse phase has to solve two main tasks: First to determine the locations where the adaptation steps from the retrieved adaptation case should be applied to the target workflow and, second, to execute these adaptation steps. The change locations in the target workflow are determined by mapping the anchors from the retrieved case. The composite anchor mapping method described in our previous work [1] is extended in order to consider the data flow in addition. The mapping method consists of two steps, which we briefly sketch below:

(1) Valid candidate positions for anchors are chosen within the set of workflow elements and data objects of the target workflow. A data object anchor can be mapped to the position of a data object in the target workflow only if the data objects are sufficiently similar (above a validity threshold for data objects). Workflow element anchors can be mapped only to similar workflow element positions analogously (above a validity threshold for workflow elements). Similarity functions for data objects and workflow elements will be discussed below. The valid candidate positions are

a) Original workflow:



b) Change request:

Replace ground beef by spelt-grain

Fig. 3: Sample query to adapt a recipe for a beef noodle casserole.

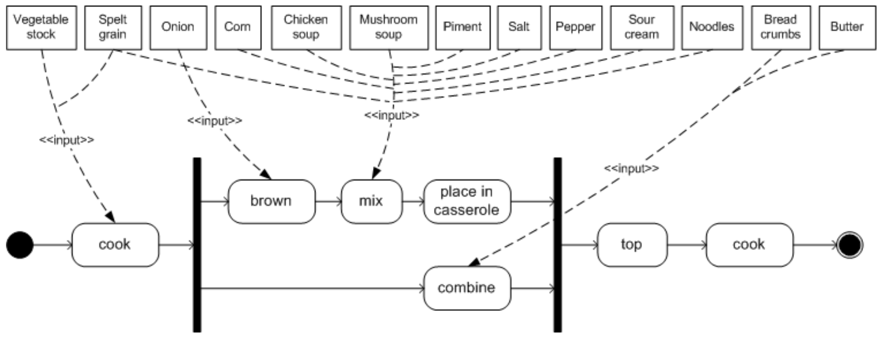


Fig. 4: Adaptation result (optimal solution, constructed by hand).

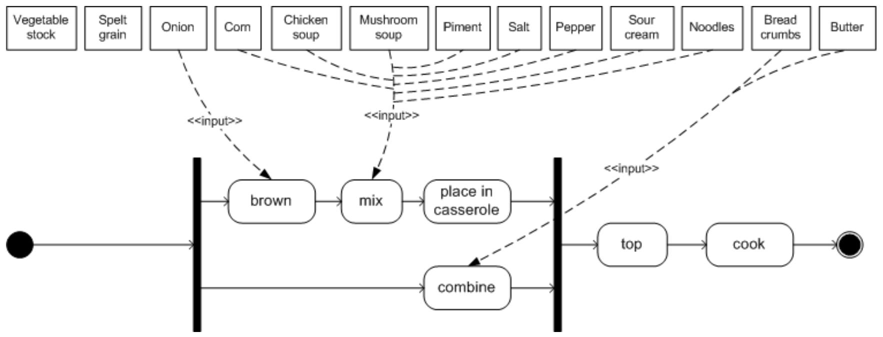


Fig. 5: Adaptation result (actually generated solution, some locations for changes could not be determined).

described by a set of triples [$wfl\_el_{retrieved}$, $wfl\_el_{target\_wfl}$, $sim_{wfl\_el}(wfl\_el_{retrieved}, wfl\_el_{target\_wfl})$] and [$data\_obj_{retrieved}$, $data\_obj_{target\_wfl}$, $sim_{data\_obj}(data\_obj_{retrieved}, data\_obj_{target\_wfl})$] whose similarity values are higher than the specified validity thresholds.

(2) The best matching positions from the set of valid candidate positions are selected to construct the anchor mapping. Pairs of workflow element anchors (pre and post anchor) have to preserve their order in the target workflow according to a precedence relation of workflow elements that is induced by the control flow. Additionally, the mapped positions of pairs of workflow element anchors must be at direct neighbors according to the same precedence relation in case they belong to a chain from the add list while they must not be direct neighbors in case they belong to a chain from the delete list. The mapping algorithm employs a hill climbing search to find mapping positions with an optimal overall anchor similarity.

The application of the adaptation chains is different for add and delete operations. The add operations are executed immediately for all chains of which at least one anchor has been mapped successfully. Chains from the delete list are applied only if a complete mapping of their particular delete steps to workflow and data flow elements can be constructed. The mapping algorithm for delete steps considers pairs of elements (workflow elements, data objects and data links) from the retrieved and the target workflow whose similarity value is above a *delete threshold*. Additionally, the elements to be deleted must be organized in exactly the same structural order (control flow or data flow) as those in the chain. Thus, in order to be applied, the delete operations have to fulfill stronger constraints than the add operations.

The similarity measures for workflow elements (sim_{wfl_el}), data objects (sim_{data_obj}) and data links (sim_{data_links}) that are required during the reuse phase are specified as follows:

$$sim_{wfl\_el}(x,y) = \begin{cases} sim_{task}(x,y) & , \quad x,y \text{ are tasks} \\ sim_{ctrl\_flow\_el}(x,y) & , \quad x,y \text{ are control flow elements} \\ 0 & , \quad else \end{cases}$$

sim_{wfl_el} distinguishes tasks from control flow elements. In case of tasks it aggregates the similarity measure for the sets of input parameters (sim_{inputs}), the similarity measure for the task name (sim_{task_name}), and the similarity of the task description (sim_{task_descr}) by a weighted sum in sim_{task}. The three constituent measures of sim_{task} are specified based on the Levenshtein distance. The Levenshtein distance is purely syntactic and measures the minimum number of edit operations on the character level that is required to transform one string into another. sim_{task_name} and sim_{task_descr} employ the Levenshtein distance directly on the task name and on the textual task description. sim_{inputs} matches the input parameters (data links) based on the Levenshtein distance of the names of their corresponding data objects by means of a hill climbing search. $sim_{control\_flow\_el}$ measures the similarity of the set of tasks included in the block in case of a block-building control flow element like AND-split or AND-join (again, a hill climbing search is applied to map the two sets of tasks to each other). sim_{data_obj} employs the Levensthein distance on the names of the data objects as the data objects do not yet store any properties like amounts or preparation states. If the special data object 'null' is one of the arguments of sim_{data_obj}, the value is set to 0 and to 1 if both arguments are 'null'. sim_{data_links} aggregates the local similarity measures sim_{data_obj} and sim_{wfl_el}.

Figures 4 and 5 illustrate that the chosen similarity measures require further improvement because the automatically generated solution in Fig. 5 could not transfer all adaptation steps from the case described in Fig. 2 c). For instance, the data links from spelt-grain to mix is missing because the syntactic similarity measure was not able to map the anchor at "add" in the retrieved workflow on "mix" in the target workflow. Also, the task names within the AND block of the retrieved workflow {"saute", "add", "simmer", "place in baking dish", "mix"} have not been similar enough to the task names within the AND block of the target workflow {"brown", "mix", "place in casserole", "combine"} so that the post anchor for adding the task "cook" could not be positioned in the target workflow. Semantic similarity measures, for instance based on a task ontology, would probably provide better results.

5 Experimental evaluation and discussion

The system is implemented in a demo version. As a starting point for a first evaluation we use a reduced recipe base containing 39 pasta recipes from in the CCC recipe base. Based on seven recipes taken from this pasta excerpt 30 different change requests were constructed by using our own experiences in cooking. Typical change requests replace one ingredient by another or avoid a certain ingredient. Thereby, an experimental case base of 30 adaptation cases (see Chapter 4 for the case structure) was created by hand. Most of the included adaptation steps concern the data objects and the data links, where some adaptations involve adding or deleting workflow elements (preparation steps). We conducted two experiments described below.

The aim of the first experiment was to evaluate whether the proposed adaptation method is able to correctly reconstruct the adapted workflows from the adaptation steps described in the 30 adaptation cases. This requires finding the proper anchor positions, which should not be affected by the similarity measures chosen for retrieval purposes as only identical matches are required for this self-reconstruction. In line with our hypothesis, all adapted workflows could be reconstructed correctly.

In the second experiment the adaptation cases were applied to new recipes. For this purpose, 14 test queries have been created by arbitrarily selecting 14 different change requests from the adaptation case base of the first experiment and combining them with other cooking workflows. We selected recipes from the pasta recipe base for that the change requests still leads to tasteful new recipes. For instance, a change request on replacing ground beef by spelt-grain applies only to a recipe that contains ground beef as an ingredient and that would still be tasty after this replacement.

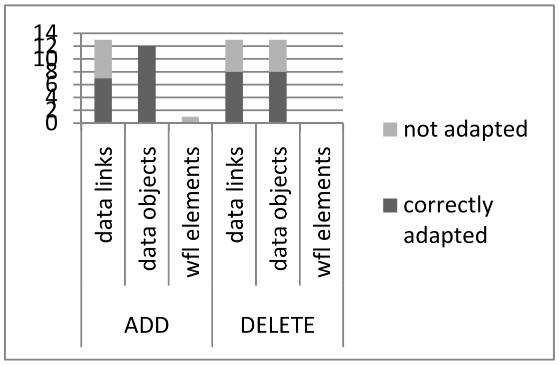


Fig. 6: Results of applying adaptation cases to new recipes in Experiment 2.

As the retrieval is not part of this evaluation the best-fitting adaptation case is chosen for each query by hand. The resulting adapted workflow is compared to a solution obtained by manually transferring the respective adaptation steps to the workflow (see Fig. 6). Altogether, half of the test cases were adapted totally correctly. In the remaining cases, at least the data objects could be added correctly, while the correctness of added data links is only approximately 50%. Also only about 60% of the possible delete operations of data links are applied. This is a clear indication that the currently used syntactic similarity measure is insufficient for our purposes. Further experiments with improved similarity measures using the CookingCake's ingredient ontology [2] as well as a task ontology (for preparation steps) will be necessary. Furthermore, the data flow could be extended by further data objects than ingredients like aggregated ingredients or utensils. However, the first results are quite promising and confirm that our idea of using the workflow paradigm to automatically adapt cooking instructions is feasible in principle.

6 References

1. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards Case-Based Adaptation of Workflows. In: Proc. ICCBR'10, to appear.
2. Fuchs, C., Gimmler, C., Günther, S., Holthof, L., Bergmann, R.: Cooking Cake, In: S. Delany (ed.) ICCBR'09 Workshop Proceedings, pp. 259 – 268, 2009.
3. Minor, M., Tartakovski, A., Schmalen, D., Bergmann, R.: Agile Workflow Technology and Case-Based Change Reuse for Long-Term Processes. International Journal on Intelligent Information Technologies 4(1), pp. 80-98, 2008.
4. Holschke, O., Rake, J., Levina, O.: Granularity as a Cognitive Factor in Effectiveness of Business Process Model Reuse, In: U. Dayal et al. (Eds.), BPM 2009, LNCS 5701, pp. 245 – 260, Springer, 2009.
5. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications, 7(1), pp. 39-59, 1994.

On-Demand Recipe Processing based on CBR

Régis Newo, Kerstin Bach, Alexandre Hanft, and Klaus-Dieter Althoff

Intelligent Information Systems Lab
University of Hildesheim
Marienburger Platz 22
31141 Hildesheim, Germany
{lastname}@iis.uni-hildesheim.de

Abstract. The third version of CookIIS, our candidate for the third Computer Cooking Contest (CCC) is focusing on pre-processing and enriching the source data for a better performance of the 4R processes. Our goal for this year is to improve the *retrieval* process by considering the ingredients' weights, strengthen the *reuse* phase by enhancing the knowledge model and further developing the adaptation methods, introducing light *revision* approaches of retrieval results. This paper presents the improvement of CookIIS in preparation for the CCC-2010.

1 Introduction

CookIIS is a Case-Based Reasoning system realized in the cooking domain for providing recipes according to given user preferences. It is a research prototype that deals with a given standardized case base of 1,484 recipes. Each recipe consists of a title, a list of ingredients and the preparation advices. We represent each recipe as one case. In terms of CBR, the list of ingredients serve as problem description and the recipe's title as well as the preparation is the according solution. Nevertheless, usually only a subset of ingredients is given by a user and it is CookIIS's task to find the most appropriate recipe to the given desires.

The CCC takes place the third time and the challenges evolved in the comparison to the two competitions. There are four challenges this year[1]:

The *Main Challenge* focuses on retrieving recipes according to given constrains, like the ingredients, the type of ingredients, the type of dish, the provenance of a dish as well as some dietary practices. A possible question to be answered in this challenge might be "I would like to cook a Mediterranean fish dish as starter, but please avoid lemon and other citrus fruits". Since all competitors work with the same case base, the requests do not have a perfect match in the case base and it is the system's task to modify the recipes in order to match best.

The *Adaptation Challenge* this year focuses on a specific adaptation task that everybody might know from personal experiences. Once you have found a great sounding recipe and start cooking you figure out that you do not have all

[1] http://vm.liris.cnrs.fr/ccc2010/doku.php?id=rules

the required ingredients available. This year, the participating systems should be able to consider this information and help the amateur chef to improvise and resolve problems like "I have retrieved the Banana Butterfinger Cake recipe, but I have no sour cream or vanilla. What should I do?".

A novelty of this years CCC is the *Open Challenge* where each team can show their creativity and moreover present what technology today can do. We decided to take numeric quantities for the retrieval process into account as well as we further developed the creation of menus since this has been a challenge in the last two years and we still have some ideas on this topic.

Another new aspect of this year's CCC is the *Student Challenge* that features student teams to enter the contest and compete with other student teams. This year, our team mainly consists of PhD students and research assistants, so CookIIS will not compete in this challenge. Like the other years, CookIIS is a web-based application and is available and can be tested by visiting the CookIIS web page at the University of Hildesheim's IIS lab[2].

The remaining paper is structured as follows: Section 2 introduces the architecture of CookIIS featuring the underlying knowledge model, retrieval processes, similarity measures and workflows of the system. The following three sections focus on the novelty of this year's system. Section 3 describes the pre-processing that enables the consideration of the ingredients' amounts; section 4 introduces the adaptation processes including the already existing ones and its improvements. Section 5 shows how user feedback can influence to retrieved recipes. The experimental results including the system's responses for a set of training queries is presented in section 6, before the final section sums up the paper and gives an outlook on future work.

2 Architecture of CookIIS

Before we explain the CBR processes implemented in CookIIS, we will first present in this section how the system is built. CookIIS is built using an industrial strength tool called empolis Information Access Suite (e:IAS). The architecture of our system therefore leans on the architecture of e:IAS, which can be seen in Figure 1.

The bottom of the architecture shows the data sources. In CookIIS these are the recipes given by the CCC organizers that we have split up so each recipe is in one XML file and further preprocessed as described in Section 3. e:IAS then uses our defined knowledge model and similarity measures in order to process the user's queries. The retrieval process is defined in the search engine and it can be supported/assisted by rules and a text miner. We will now give a short explanation of some parts of the architecture.

[2] `http://cookiis2010.iis.uni-hildesheim.de/ccc`

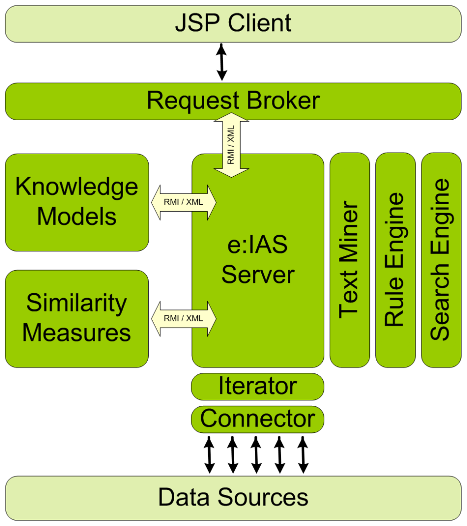


Fig. 1. The architecture of e:IAS used for CookIIS

2.1 Knowledge Model

Our knowledge model contains different elements. The main elements are ingredients (about 1000 in our model) which are organized in the following eleven classes:

Basic Ingredients	Fish	Meat	Vegetable
Supplement	Fruit	Drinks	Milk
Intermediate Ingredients	Oil and Fat	Spice and Herb	

Each ingredient is modeled as a concept with possible synonyms in English and German. The concepts are mostly organized as taxonomies and each class can contain several taxonomies. One can for example organize meat by part (fillet, haunch) or by kind (pork, beef).

Furthermore, our model contains several rules which are used for the computation of meta information like:

- the type of meal,
- the type of cuisine,
- the consideration of some diets (for example low cholesterol).

Further details on the knowledge model as well as the computation of meta information can be found in [1, 2].

2.2 Similarity Measures

Within CookIIS, similarities between recipes (or between a query and a recipe) are computed in two steps. First, a local similarity measure is defined for each class of the knowledge model in order to compare the different types of ingredients. The underlying taxonomies give a strong basis for the computation of the similarities between ingredients within a class. Here, adequate values are given for the generalization and the specialization step so that the similarities can be automatically computed. These values indicate the similarity between a concept and a child or a parent. In addition to the taxonomy based similarity measures, we also used table based similarity measures for some classes, because the values computed with taxonomies do not always reflect the reality in our opinion. The similarity between some pairs of elements is (manually) entered in the corresponding matrix. When several local similarity measures are defined within a class, the highest similarity value is always used.

In the second step, we use a global similarity measure to compute the similarity between cases (i.e. the recipes). It is a weighted sum of the local similarities of the attributes in the cases, following the local-global principle for similarity modeling [3]. The global similarity measure do not only consider the ingredients of the recipes, but also other attributes like the computed meta information mentioned in the previous Section.

2.3 Indexing and Searching

Indexing and Searching recipes in CookIIS is done using workflow-like constructs, called *pipelines* in e:IAS. The pipelines consist of the so-called *pipelets* which represent the different steps (actions) for each operation.

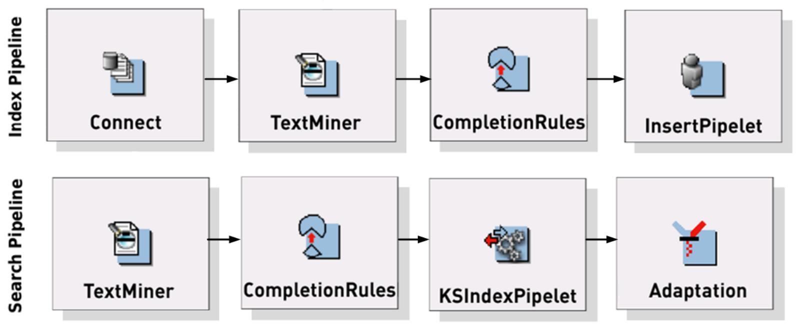


Fig. 2. The index and search pipeline for CookIIS

The Index Pipeline The aim of the index pipeline is to create an index representing the case base (i.e., insert the recipes in the case base). This is done by performing the following actions (as depicted on the left hand side of Figure 2): First a connection to the data sources has to be established before an analysis of the data and the extraction of the concepts by the text miner is carried out. The third pipelet computes of meta information based completion rules and the final pipelet inserts the cases in the retrieval structure.

The Search Pipeline The aim of the search pipeline is to retrieve and adapt the recipes from the case base according to a query. The actions performed are the following (right hand side on Figure 2): First, the query is analyzed and concepts are extracted using the text miner. The following pipelet computes meta information based on the input given by the query. The third pipelet executes the retrieval of the most similar cases from the case base with the aforementioned similarity measure. The final piplet carries out the adaptation of the retrieved cases (this is done in many steps as explained in Section 4).

3 Recipe Pre-Processing

The information given in recipes still has potential to improve the CookIIS system. There are different ways of using the case base obtaining knowledge that can be provided in within the knowledge containers. This point has also been discussed by the Taable team [4] as well as the JaDaCook team [5]. For example, we have applied association rules aiming at adaptation knowledge, but caused by the high variation we only received rules containing very obvious associations, like eggs and sugar usually come with flour. We think that a higher amount of recipes is required for acquiring more special association rules.

The global similarity measure of the last versions of CookIIS only takes the ingredient without the according amount into account. Hence, recipes that contain many ingredients are retrieved more often because the probability that one of the containing ingredients was requested is higher. However, users that are looking for recipes usually name those ingredients that should be characteristic for the whole dish: when looking for a recipe with carrots you might not be satisfied with a casserole that includes half of a carrot. Therefore, we are taking the amounts of the ingredients into account.

3.1 Pre-processing Process

We decided to enrich the information given in the case base including an attribute that marks the major ingredients of a recipe. Since the amounts are given in various measures, we built up a dictionary that normalizes each amount to milligram. Further, we used the normalized amount information for improving the retrieval aiming at retrieving recipes that contain the requested ingredient as a major ingredient. Algorithm 1 shows how the major ingredients of a recipe are determined.

Given the normalized amount of each ingredient, we first determine the median of the amounts occurring in the recipe. The median splits the ingredients into major and minor ingredients of a recipe. In the third step we mark all major ingredients by adding the attribute "major" to the ingredients tag. In the last

Algorithm 1 Marking Major Ingredients

```
1: Order ingredients descending according to its amounts of one recipe
2: Calculate the median over all amounts of one recipe
3: Mark all ingredients with $amount > median$ as major
4: Mark the position of major ingredients according to their amounts
```

step we determine the order (based on the ingredients' amounts) of the major ingredients and assign it to the ingredients. This results in an enhanced ingredient tag in the xml source file, like the following:

<IN amnt='425243' major='5'>1 cn (15oz) tomato sauce**</IN>**

Obviously this information can be used in various ways. However, currently we only use it within the retrieval process. For that purpose, we doubled the weight of major ingredients within the global similarity calculation. In the similarity assignment, we have an attribute that represents a set of major ingredients, so each major ingredient that has been requested gets the double weight. As a result, the retrieved recipes containing the desired ingredients are more emphasized. We are aware that for instance spices and herbs, which have a huge influence on a recipe, but only occur in small amounts, are not considered in this approach. However, in the CookIIS knowledge model and within the global similarity calculation spices and herbs are treated separately.

4 Adaptation

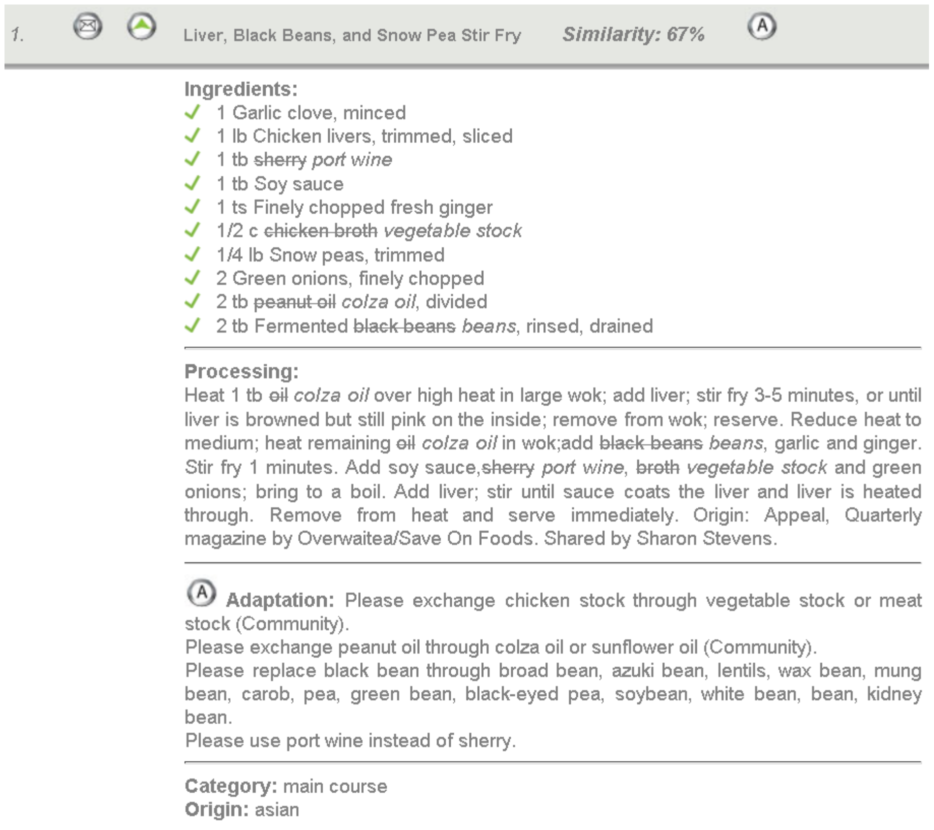
The given recipe base contains 1484 recipes, which are not enough recipes for the whole variety of desired as well as unwanted ingredients. Therefore, an adaptation of the queried recipes to the users needs is necessary.

This section describes three different kind of adaptations we use in CookIIS. Two approaches are known from former versions of CookIIS: the model-based and the community-based adaptation. Furthermore, we implemented an in-place adaptation and an adaptation approach regarding user feedback.

All of our adaptation approaches share some assumptions and have a certain generality in the used methods with the aim to be able to transfer these methods to other domains. First, we denote ingredients that are excluded by a diet or explicitly by the user as *forbidden* ingredients. If a *forbidden* ingredient occurs in a retrieved recipe we consider it as *critical* (for this recipe) because it has to be omitted or replaced. Second, we restrict the replacements to the same class as the replaced ingredient to assure certain a level of applicability.

4.1 Integration of the three Approaches

All three adaptation approaches are performed sequentially and each of them uses the results of the preceding one. First, the community-based adaptation is



1. Liver, Black Beans, and Snow Pea Stir Fry *Similarity: 67%*

Ingredients:
- ✓ 1 Garlic clove, minced
- ✓ 1 lb Chicken livers, trimmed, sliced
- ✓ 1 tb ~~sherry~~ *port wine*
- ✓ 1 tb Soy sauce
- ✓ 1 ts Finely chopped fresh ginger
- ✓ 1/2 c ~~chicken broth~~ *vegetable stock*
- ✓ 1/4 lb Snow peas, trimmed
- ✓ 2 Green onions, finely chopped
- ✓ 2 tb ~~peanut oil~~ *colza oil*, divided
- ✓ 2 tb Fermented ~~black beans~~ *beans*, rinsed, drained

Processing:
Heat 1 tb ~~oil~~ *colza oil* over high heat in large wok; add liver; stir fry 3-5 minutes, or until liver is browned but still pink on the inside; remove from wok; reserve. Reduce heat to medium; heat remaining ~~oil~~ *colza oil* in wok;add ~~black beans~~ *beans*, garlic and ginger. Stir fry 1 minutes. Add soy sauce,~~sherry~~ *port wine*, ~~broth~~ *vegetable stock* and green onions; bring to a boil. Add liver; stir until sauce coats the liver and liver is heated through. Remove from heat and serve immediately. Origin: Appeal, Quarterly magazine by Overwaitea/Save On Foods. Shared by Sharon Stevens.

Adaptation: Please exchange chicken stock through vegetable stock or meat stock (Community).
Please exchange peanut oil through colza oil or sunflower oil (Community).
Please replace black bean through broad bean, azuki bean, lentils, wax bean, mung bean, carob, pea, green bean, black-eyed pea, soybean, white bean, bean, kidney bean.
Please use port wine instead of sherry.

Category: main course
Origin: asian

Fig. 3. Replacement of forbidden concepts (crossed out) with new replacements (italicised) in the ingredient list and preparation: sunflower oil replaces olive oil in the ingredient list and only oil in the preparation because olive oil does not exist there.

executed. For those forbidden ingredients where no community-based replacements can be offered, the standard model-based adaptation is carried out afterwards. Both approaches only add adaptation advice in an extra text attribute of the retrieved recipes but leave all attributes for ingredients and preparation unchanged. In contrast, the third approach changes the ingredient list and preparation text visible according to the advice of both preceding approaches.

The integration of the all approaches is realised with three subsequent pipelets within the the Search Pipeline used for retrieval and reuse. A user-defined pipelet for the community adaptation is inserted into this pipeline which is followed by the a standard AdaptationRulesPipelet implementing the model-based adaptation. The in-place adaptation as third approach is realised as another user-defined pipelet and integrated after the AdaptationRulesPipelet in the same pipeline. All approaches are described in the next sections.

4.2 Model-based and Community-Based Adaptation

For the community-based adaptation we collect lots of adaptation advices from comments of recipes within a large cooking community using the knowledge model of CookIIS. With the aim of this model we classify each ingredient of the comment as OLD (if it exits in the recipe already) or NEW (if it exists only in the comment). Afterwards we interpret the co-occurrence of a OLD and a NEW ingredient belonging to the same ingredient class as an adaptation advice and sum them up over all 70.000 recipes. This results in 2967 adaptation suggestions for 410 different ingredients. If there exist a suggestion for a forbidden ingredient, the two most frequent adaptations are presented. A more detailed description and evaluation of this approach is given in [6].

In contrast to the community-based adaptation, which delivers adaptation suggestions for 40% of the ingredients, the model-based adaptation works for all concepts, based on the similarity between ingredients. In general, it determines similar ingredients to the *forbidden* ingredients and suggest them as replacement. The similarity between two ingredients is based on the knowledge model that are applied along with thresholds to determine nephew and sibling concepts as replacements [2].

4.3 In-place Adaptation

Both aforementioned approaches only add adaptation advices in an extra text attribute, but did not change the original attributes for ingredients and preparation which makes it harder to take the ingredient list as shopping list or work through the preparation step by step. Therefore we add an additional adaptation component which executes these adaptation advices directly on the original recipe. By investigating the preparation instructions we detect the following difficulties that prevent using a simple string matching: ingredients are mentioned in their (irregular) plural form, referred in an abbreviated form or more general term or described by their synonyms. Additionally, the user should recognise in the modified recipe what the changes are, especially if an adaptation suggestion is not good as expected by the user. Therefore, the in-place adaptation has to tackle two problems: Simple string matching for the replacement often does not succeed and the original ingredient should be visible if one of the replacement suggestions is not accepted by the user.

The replaced and the new text are marked up with HTML tags in order to make changes on the original text of the case visible. Hence, the user can notice the original ingredient. To identify concepts that should be replaced in the given text four consecutive possibilities are tested until one is successful:

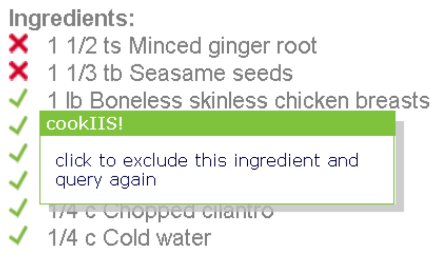
1. find the concept name in the text
   (a) search and replace plural form (considering -es, -ies, -ves and -oes) and
   (b) search and replace singular form,
2. consider common synonyms belonging to certain concepts,
3. if it is a 2-word-concept, replace only the last word, otherwise

4. look for the class name of this concept and replace if exists.

Figure 3 shows an example where ”black bean*s*” (1.b) are replaced by ”beans” (1.a). In the preparation only ”oil” instead of *forbidden* ”peanut oil” (3.) occurred and is replaced by ”colza oil” (from Community). Furthermore, ”broth” in the preparation stands for ”chicken stock” (2.) and is replaced by ”meat extract”.

5 Revise through User Interaction

During a small evaluation we investigated that users often discover that they either dislike or do not have certain ingredients available that are required for preparing the desired recipe. To make this more faster and comfortable the user can now click on an icon before each ingredient to exclude this ingredient instead of adding the text by hand into the exclude text box in the GUI. At start, each ingredient has a green check mark, which is changed into a red cross if the user clicks on it to exclude this ingredient. In a subsequent search request these ingredients where excluded as well. According to the fact that in our current implementation the amount of forbidden ingredients in an recipe did not affect the similarity of a recipe to a query, the same recipes would be retrieved again, but with additional adaptation suggestions. Of course, the exclusion can be undone by clicking again on the icons. The complete functionality is realised with Javascript called by the JSP code of website. Figure 4 shows a scenario where the user has excluded the ingredients minced ginger an Sesame seeds after the first retrieval.



Ingredients:

✗ 1 1/2 ts Minced ginger root
✗ 1 1/3 tb Seasame seeds
✓ 1 lb Boneless skinless chicken breasts
✓
✓
✓
✓ 1/4 c Chopped cilantro
✓ 1/4 c Cold water

cookIIS!

click to exclude this ingredient and query again

Fig. 4. Feedback: The user can easily exclude additional ingredients by clicking on the green icon in front of each ingredient

6 Experimental Results

As stated earlier, many features of CookIIS have been improved (from the previous versions), like for example, the computation of meta information (type of meal, type of cuisine), the adaptation of recipes (model based, community based and in-place). We will concentrate in this Section on two new features of CookIIS, namely the impact of the preprocessing of the data source and the revise step in CookIIS.

We will consider for this purpose the query "I want a dessert with strawberries". CookIIS returns many recipes which are 100% similar to the query because they contain strawberries and were tagged as desserts. These recipes are automatically ranked according to the amount of strawberries contained (in comparison to the remaining ingredients). CookIIS thus ranks the recipe "My Strawberry Pie" (which contains few ingredients) higher than the "Summertime Strawberry Gelatin Salad" which contains much more ingredients (because strawberry is this case less important). Furthermore, if an ingredient is not available, the user can explicitely select the failing or unwanted ingredients as can be seen in Figure 4. Gelatin for example in the "Summertime Strawberry Gelatin Salad" is then replaced through agar-agar.

7 Conclusion

The paper presents new developments of the CookIIS system - mainly the preprocessing for taking ingredients' amounts into account, applying adaptation knowledge to recipes and replacing the modified ingredients within the recipe as well as enabling user interaction for interactive search refining.

However, we have addressed the first three of the four CBR processes and since the case base is standardized and static, the only way for retention is acquiring and updating modification knowledge, similarity and vocabulary knowledge.

Future work will be focused on including the amount information, now available for each recipe, in the adaptation process as well. This also affects the substitution of ingredients or a modification that leaves out certain ingredients instead of looking for a substitute.

References

1. Ihle, N., Newo, R., Hanft, A., Bach, K., Reichle, M.: Cookiis - A Case-Based Recipe Advisor. In Delany, S.J., ed.: Workshop Proceedings of the 8th International Conference on Case-Based Reasoning, Seattle, WA, USA (July 2009) 269–278
2. Hanft, A., Newo, R., Bach, K., Ihle, N., Althoff, K.D.: Cookiis - a successful recipe advisor and menu advisor. In Montani, S., Jain, L., eds.: Successful Case-based Reasoning applications. Springer (2010)
3. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. Volume 2432 of LNAI. Springer-Verlag (2002)
4. Badra, F., Cojan, J., Cordier, A., Lieber, J., Meilender, T., Mille, A., Molli, P., Nauer, E., Napoli, A., Skaf-Molli, H., Toussaint, Y.: Knowledge acquisition and discovery for the textual case-based cooking system WikiTaaable. In Delany, S.J., ed.: ICCBR 2009, Workshop Proc. (2009)
5. Herrera, P.J., Iglesias, P., Sanchez, A.M.G., Diaz-Agudo, B.: Jadacook 2: Cooking over ontological knowledge. In Delany, S.J., ed.: ICCBR 2009, Workshop Proc. (2009)
6. Ihle, N., Hanft, A., Althoff, K.D.: Extraction of adaptation knowledge from internet communities. In Delany, S.J., ed.: ICCBR 2009, Workshop Proc. (2009) 35–44

Approximating Knowledge of Cooking in High-Order Functions, a Case Study of Froglingo

Kevin Xu, Jingsong Zhang, Shelby Gao

Bigravity Business Software LLC
2306 Johnson Circle, Bridgewater, New Jersey, United States
{kevin, jingsong, shelby}@froglingo.com

Abstract. Without accurately representing the knowledge of cooking, a computer would never react as if it were a professional chef. Accumulating the knowledge into a computer presentation, maintaining the accuracy of the presentation, and offering practically valuable advice face many challenges in the current technologies. Froglingo is a computer system that uniformly represents both business data and business logic as high-order functions. It has a set of built-in operators stemming from the relationships among the high-order functions. As a case study in this paper, we use Frolingo to approximate the knowledge of cooking and to approximate the reasoning on the knowledge.

Keywords: High-order Function, Ordering Relation, Query, Similarity, Adaptation, Approximation.

1 Introduction

To be a valuable recipe advisor, a system shall have a data presentation semantically approximating the knowledge of cooking including adaptation processes. In other words, the system allows users to accumulate data that approaches closely to the knowledge of cooking, and thereafter to accurately offer advice reflecting the managed data.

To implement such a system, the authors suggest a data model that generally and therefore uniformly represents knowledge. Only with the generality, we have a chance to unlimitedly accumulate data that approaches the knowledge more closely; and the mission would not be hindered by any pre-mature assumptions of data presentation.

Modeling knowledge in high-order functions offers an opportunity. A collection of high-order functions embed rich relationships among them [6]. In addition, a class of total recursive functions, in which high-order functions are the sole members, is the upper bound of what a computer can do practically. (We attempt to exclude those partial recursive functions which don't terminate on some inputs and therefore are not desired in software practice [8 and 11]).

In this paper, we use Frolingo to approximate the knowledge of cooking. Froglingo has a data model, called the EP (Enterprise-Participant) data model. The EP data model is equivalent to a class of total recursive functions [9]. Therefore it mathematically defines the entire semantic space for software applications. The variables and the sequential terms in Froglingo, beyond the EP data model, allow one to construct business logic, i.e., infinite data [10]. The addition of the new concepts doesn't alter the semantic space defined by the EP data model. Instead, it allows one to stuff the infinite semantic space in finite expressions. As a result, the relationships among high-order functions are preserved and thereafter the built-in operators introduced by the EP data model can be equally applicable to the data expressed in variables [6]. The unified semantic space for both business data and business logic makes the architecture of Froglingo a monolith [7]. Froglingo reaches the greatest possible generality in uniformly representing both business data and business logic including knowledge.

The system, "Chef Froglingo", implemented for the 2010 Computer Cooking Contest (CCC), has the following objectives: 1) A consistent method of representing the knowledge of cooking including adaptation; 2) The ability of accepting a format of the inputs producible by the customers who don't know the knowledge representation of system; 3) A reasonable precision of answering customer queries as if it were a professional chef.

In Section 2, we briefly outline the main concepts of Froglingo. For a detailed introduction of Froglingo syntax, readers may reference the user's guide [10]. In Sections 3, we give a Froglingo presentation of the primary cooking concepts such as ingredient, dish origin, recipe, and menu. At the same time, we discuss our strategy of parsing the XML document file, the Recipe Book from CCC, containing a set of recipes. (A sample XML block for a recipe is attached in Appendix A for reference.) In Section 4, we describe how customer queries are expressed and answered by a Froglingo built-in operator. In Section 5, we describe the implementation of an adaptation task, and therefore demonstrate that the system can approach the accuracy of a professional chef in responses to future events as close as the volume of the data approximating the knowledge of cooking is allowed. In Section 6, we make a few remarks about system architecture, related work and future work.

2 Froglingo

In traditional data models, an entity is either dependent on one and only one other entity, or independent from the rest of the world. The functional dependency in relational data model and the child-parent relationships in hierarchical data model are the typical examples. This restriction, however, doesn't reflect the complexities of the real world that are manageable by using a computer. The EP (Enterprise-Participant) data model suggests that if an entity is dependent on others, it precisely depends on two other entities. Drawing the terminologies from the structure of an organization or a party, one depended entity was called enterprise (such as organization and party), the other called participant (such as employee and party participant), and the dependent entity called participation. An enterprise consists of multiple participations. Determined by its enterprise and its participant, a participation yields a value, and this value in turn is another enterprise.

Specifically, a Froglingo database is a set of assignments, an assignment has an assignee and possibly an assigner, and both assignee and assigner are terms. The concept of term is essential. A term is a constant, an identifier, a variable, or a pair of parenthesized terms. A few sample terms are `3.14`, `$x`, `"A String"`, `mike`, `(mike` `salary`), and `((a b) (c d))`. A sample database is: {`mike salary = 3.14; fac 0 = 1; fac $n = ($n * (fac ($n - 1))); ((a b) (c d)) = something; x y z;`}.

A term is called a sub-term of itself. When a term consists of a pair of two other terms, it is called an application, where the first element is the function, and the second the argument. Given a sub-term of a term, recursively, an inner sub-term of the sub-term is also a sub-term of the given term. For example, `a, c, (c d),` and `((a b) (c d))` are sub-terms of the term `((a b) (c d))`. If the right sub-term of an application is not another application, the parentheses surrounding the application don't have to be written. For example, `((a b) (c d))` is equivalent to `a b (c d)`. In this article, we alternatively use the phrase "plus-term" in the place of the function of an application to avoid any confusion with the word "function" in generic term.

In terms of ordering relations, we call an application *neutrally* depends on a sub-terms, i.e., either its function, its argument, or any inner sub-term, in contrast to the fact that the function and the argument play slightly different roles in determining the application. For example, `a b (c d)` neutrally depends on `c`. Because of assignment, two or more terms can be derived to be equal in a given database. Given an application and its equal peers in a database, we say that the application or any one of its peers is functionally derivable from the function of the application, argumentatively derivable from the argument, and *neutrally* derivable from a sub-term of the application. Given the sample database discussed earlier, for example, both `a b (c d)` and `something` are neutrally derivable from c.

3 Knowledge of Cooking

During the design of the knowledge representation, we first consider of arranging the knowledge of cooking in certain orders that are available among high-order functions. The aim is to leverage Froglingo built-in operators stemming from the orders in support of the queries expressible from customers. Secondly, we consider the challenges in retrieving data from the given Recipe Book, a loosely structured text file. In this section, we introduce a Froglingo presentation of preliminary concepts in the knowledge of cooking. The query and adaptation processes are discussed separately in Sections 4 and 5.

3.1 Ingredient

An ingredient may have an ingredient type, and an ingredient type may have its parent type. Sometimes, an ingredient may belong to multiple types. For example, chicken wing has the type chicken, and chicken has the type meat (or poultry precisely). Chicken also belongs to the type broth for the ingredient chicken broth. Here are a few examples in Froglingo terms:

```
meat beef corned;
meat beef (short loin);
meat chicken;
vegetable artichoke;
vegetable (Chinese artichoke);
soda (baking powder);
broth (meat chicken);
broth vegetable;
```

To collect all the ingredients, we first inventory a list of preliminary ingredients such as meat, vegetable, nut, alcohol, grain, sauce, herb, and spice. To obtain granular ingredients appeared in Recipe Book, e.g., cilantro and all-purpose flour, we collect all the possible names from an ingredient dictionary [3], and classify each of them as one of the preliminary ingredients by parsing the body of the ingredient dictionary.

3.2 Preparation Method and Step

Preparation method is an important piece of information in a recipe. It may determine the type of dish. Taking it into consideration in our data presentation helps the case study more closely modeling the interactions between customers and wait staffs. Like ingredients, it participates in preparation steps. For example: `bake`, `grill`, `steam`, and `stir fry` are preparation methods.

A recipe normally involves multiple preparation methods and therefore a sequence of preparation steps, and each step may have its sub steps. To avoid the difficulty of parsing the multiple steps in the Recipe Book, we identify one preparation method for a recipe. If there are multiple methods appeared between a preparation block (i.e., a pair of <PR> and </PR>), we choose only one by pre-defining weights for all the methods. For example, if both `grill` and `marinate` appear in a recipe preparation block, we will choose `grill` because `grill` is assigned with a heavier weight.

The cooking temperatures and the cooking time periods of preparation methods are the factors considered when the weights are assigned.

3.3 Origin

Many ingredients and dishes originate from certain cultures. For example, curry is generally regarded as an Indian ingredient, wok as an Asian equipment, and a Fajita beef as a Mexican dish. Therefore, a recipe may have multiple origins. To support this feature, we inventory a list of food origins. Here are a few sample terms inventoried in Froglingo terms:

```
Asian Chinese Szechuan;
French Dijon;
Irish Kilkenny;
```

The complete list of origins is to be identified through a world atlas.

When the ingredient dictionary is parsed (as discussed in Section 3.1), and when an XML block between <RECIPE> and </RECIPE> is parsed, we search the key words from the origin list, and tag ingredients and preparation steps if a key word is found.

3.4 Dish Type

The food described by a recipe can be served as an appetizer, a main course, or a dessert. It also can be simultaneously served for two or more dish moments. Instead of explicitly labeling them for the purpose of determining dish moments, we derive dish moments according to the ingredients and cooking methods of recipes. For example, cafe, tea, sugar, chocolate, and refrigerator most likely imply dessert; and soup, salad, fry, and the rest of non-sweet foods can imply appetizer.

Appetizers and desserts are normally further classified to different types, such as cake, cookie, ice cream, and pizza. We observe that recipe titles sometimes embed this information. The Chef Froglingo uses the embedded information to identify dish types.

We also allow customers to express key words appeared in the title of recipes, e.g., Mom, and Carne Asada. It facilitates customers to make requests easier.

3.5 Synonym

Ingredients may have multiple names. We identify them when the ingredient dictionary is processed (discussed in Section 3.1). Here are a few sample pairs of equivalent names in Froglingo assignments:

```
lichee = lichi;
chile = hot pepper;
```

3.6 Recipe

We discussed many concepts in the earlier sub-sections from which our simplified version of the concept recipe is assembled. To Froglingo, however, all of them are indiscriminately stored as data, i.e., high-order functions. Here we give the database for a few sample recipes.

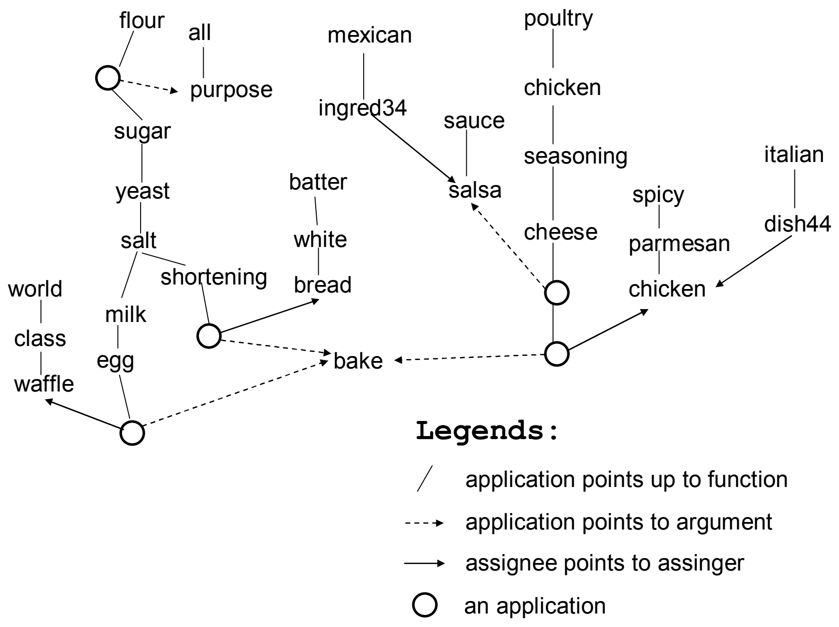
```
(flour (all purpose)) sugar yeast salt milk egg bake
     = world class waffle;
(flour (all purpose)) sugar yeast salt shortening bake
     = batter white bread;
(meat chicken) cheese seasoning (sauce salsa) bake
     = spicy parmesan chicken;
mexican ingred34 = sauce salsa;
Italian dish44 = spicy parmesan chicken;
```

To better understand the knowledge representation that is arranged in orders among high-order functions, we provide a graphical view of the database.

To support the queries from customers, we keep the text message from "Recipe Book" for each recipe. Here is an example:

```
spicy parmesan chicken title = "Spicy Parmesan Chicken";
spicy parmesan chicken body  = "...";
```
(The texts in the XML block <PR>)

Here the terms `title` and `body` are the key words indicating that the text strings of the recipe title and body follow as assigners.



3.7 Menu

A term in Froglingo can be used to express a menu too. Here is a sample expression for a dinner:
`dinner (Potato Salad 2) (Spicy Parmesan Chicken) (Sorry Cake);`
However, enumerating all the possible dinner choices is not preferred. We choose to have variables for the set of dinners producible from dishes. The following sample expression is equivalent to a set of dinners including the one described earlier:

```
dinner $s:[$s {=+ salad or $s {=+ soup]
     $m: [$m {=+ chicken or $m {=+ marinate]
     $d: [$m {=+ cake];
```

4 Query

As readers might have realized, the Froglingo expressions for recipes, as the examples demonstrated in Section 3, embed all the relevant information including ingredients, preparation methods, and cultures. In addition, this information is arranged in certain ordering relations, i.e., the dependent relations and pre-ordering relations. Given an entity, e.g., all-purpose flour, salt, bake, and Chinese, one can find all the dishes that include the entity by navigating the ordering relations. When a customer gives a list of entities, the same process is repeated for one entity. At the end, the remaining recipes are those satisfying the initial request. This process is done by the built-in operator `(=`, a pre-ordering relation in Froglingo (called neutrally derivable in this article). Here is a sample query expression retrieving the recipes that have all-purpose flour as an ingredient and that use the preparation method bake:

```
select $dish title, $dish body where
      $dish (= flour (all purpose) and
      $dish (= bake and
      $dish title != null;
```

In the expression, we use the clause `$dish title != null` to only retrieve those terms representing recipes because not all the terms in the database have correspondences of recipes.

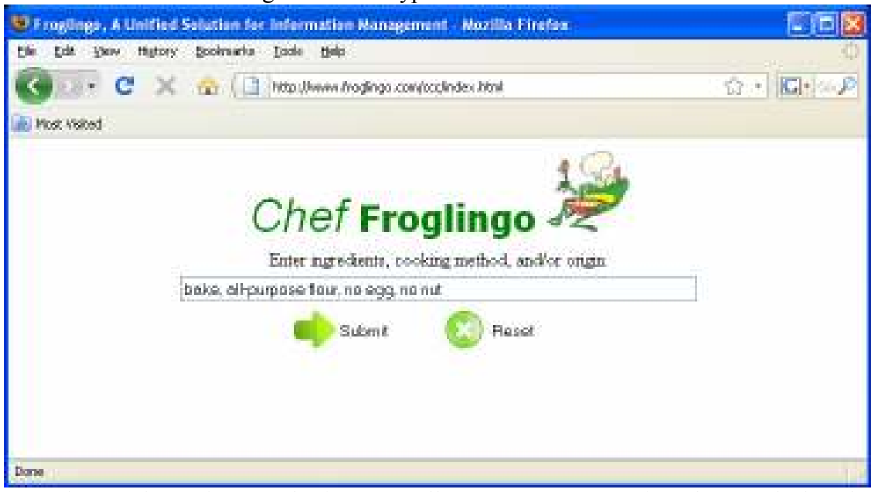
It is clear that the recipes "World Class Waffles" and "Batter White Bread" will be retrieved for this query expression.

To support negation, we simply use the negation operator available in Froglingo `not`. When the boolean expression `not $dish (= egg` is added into the `select` operation above, the query will only return the recipe "Batter White Bread".

The dietary practices of vegetarian, nut-free, and no alcohol are translated into the following expressions:

```
not $dish (= meat;
not $dish (= nut;
not $dish (= alcohol;
```

where all the meats have the ingredient type `meat`, all the nuts have the type `nut`, and all the alcohol beverages have the type `alcohol`.


Froglingo, A Unified Solution for Information Management - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.froglingo.com/ccc/index.html
Most Visited
Chef Froglingo
Submit
Reset
Done

Enter ingredients, cooking method, and/or origin

bake, all-purpose flour, no egg, no nut

What we have done so far is eventually to support a user interface such that the customers who don't know the system representation of recipes can easily express their requests, and the system can precisely process the requests. The resulting system allows users to enter a list of phrases or words delimited by a comma, and each phrase or word can be preceded with the word "no" for negation. See a sample screenshot of the interface above.

In the sample query string in the screenshot, the system doesn't break the phrase "all-purpose flour" into individual words, but translate it to the Froglingo term `flour (all purpose)`. In other words, customers are required to be precise on the phrases representing ingredients, preparation methods (such as stir fry), and recipe titles if they want to be specific. If a customer is not sure about the exact spelling of a phrase, he or she should enter individual words divided by commas. Individual words lead to more recipes in return than phrases do.

The same interface is equally applicable to dinners that have variables in their expressions. When customers enter a string: "dinner, no soup, chicken", for example, the system will retrieve all the dinner choices, each of which will include a salad as the starter, a main dish with chicken, and an ice cream as the dessert.

5 Adaptation

If we view the knowledge of cooking discussed above as experience, or precisely historical events, the concept of adaptation in the field of Case-Based Reasoning refers to a systematic ability of proactively planning future events that may not repeat the past experience [5]. Adaptation may mean various systematic abilities. The one required in the 2010 CCC requests a system to response reasonably when a customer wants a specific recipe and certain ingredients for the recipe are not available. The reasonability may mean different things from time to time and from person to person. When there is not a grill available, for example, a recipe which requires fresh salmon and a grill may be desired to be adapted to a recipe with a broiler some times, and to a recipe of a Sushi at another occasion.

For the 2010 CCC, what we choose as a reasonable response is to adapt some existing and similar recipes to replace those wanted but not available. Further, we rule that two applications in a Froglingo database are similar if they share the same function, i.e., the plus-term. This rule applies to all the cooking related entities including ingredients, preparation methods, preparation steps, and origins. Given the sample ingredients in Section 3.1, for example, `meat beef corned` and `meat beef (short loin)` are similar because they share the same plus-term `meat beef`. Similarly, the sample preparation steps `(flour (all purpose)) sugar yeast salt milk` in Section 6 is similar to `(flour (all purpose)) sugar yeast salt shortening`.

Given a recipe and a set of excluded cooking entities, i.e., a set of entities that are required by the recipe but not available, we attempt to identify alternative recipe(s), as a solution for the adaptation task of the 2010 CCC, by taking the following steps:
1). Identify the set of non-excluded entities required by the recipe, and retrieve those recipes that are neutrally derivable from the non-excluded entities.
2). If there is no recipe returning from Step 1, we relax the constraints by taking a non-excluded entity as an excluded entity, and rerun Step 1.
3). Taking the returning recipes from Step 1 as the initial input, we select those neutrally derivable from the plus-terms of the excluded entities, but filter out those

recipes that are neutrally derivable from the excluded entities. The outcome would be those similar recipes, but not the recipes requiring the excluded entities.

The interface introduced in Section 4 is also used for the adaptation task. For example, the request strings “World Class Waffle, no egg” is for the query: “I want World Class Waffles, but I don’t have egg. Please show me the similar recipes”. This query would return a set of recipes including Batter White Bread, but not World Class Waffle.

When taking a member out of the set of non-excluded entities in Step 2, we choose the one that is less important to the recipe. We rule again that the preparation method of a recipe is the least important, and an ingredient toward the end of the ingredient list in a recipe is less important than a one closer to the beginning.

The relaxation of query constraints in Step 2 is the initial step of the approximation in searching for similar recipes, and the additional constraints based on similar entities in Step 3 is the second step approaching closer to the most similar recipes.

6 Remarks

Reasoning on the knowledge of cooking has been identified as a typical domain in the field of case-based reasoning in [2]. Under the benchmark of CCC, many systems were implemented for this domain. The tools used for the reasoning in the implementation were based on traditional technologies such as the propositional logic for WikiTAAABLE [1], and Java for CookIIS [4].

Aimed to be a true recipe advisor, Chef Froglingo represents knowledge of cooking in high-order functions, and uses a neutral derivative relation, i.e., (=, to approximate the adaptation processes of substituting a desired but not available dish with the most similar ones. The real cooking knowledge is much more complicated than what we have discussed in this article. But this approach can reach our expectation if we accumulate as much knowledge of cooking, and develop as many adaptation tasks as our business needs. The knowledge accumulation process is at multiple dimensions, such as the dimension of nutrition categories, e.g., protein and fat levels, the dimension of additional ingredient attributes, e.g., volume and texture, and the dimension of more complicated preparation steps, e.g., sub steps and cooking equipment.

Chef Froglingo, available at http://www.froglingo.com/ccc/index.html, is implemented in Froglingo, a single tool for business data, business logic, and web server. Therefore, we also easily offer a web page allowing users to update recipes.

References

1 F. Badra, J. Cojan, A. Cordier, J. Lieber, T. Meilender, A. Millle, P. Molli, E. Nauer, A. Napoli, H. Skaf-Molli, and Y. Toussaint. “Knowledge Acquisition and Discovery for the Textual Case-Based Cooking System WIKIAAABLE”. 8th International Conference on Case-Based Reasoning – ICCBR 2009, Workshop Proceedings 2009, Page 249 - 258.

2 K. J. Hammond. "CHEF: A model of case-based planning." AAAI Proceedings of the 5th National Conference on Artificial Intelligence, page 267-271. Morgan Kaufmann, 1986.
3 S. T. Herbst, "Food Lover's Companion, The (Barron's Cooking Guide) 2nd Edition", available at: http://www.epicurious.com/tools/fooddictionary.
4 N. Ihle, R. Newo, A. Hanft, K. Bach, M. Reichle. "CookIIS – A Case-Based Recipe Advisor". 8th International Conference on Case-Based Reasoning – ICCBR 2009, Workshop Proceedings 2009.
5 D. Leake. "Case-Based Reasoning: Experience, Lessons, and Future Directions". Menlo Park: AAAI Press/MIT Press, 1996.
6 K. H. Xu, J. Zhang, S. Gao. "High-Order Functions and their Ordering Relations". The Fifth International Conference on Digital Information Management, 2010.
7 K. H. Xu, J. Zhang, S. Gao. "Froglingo, A Monolithic Alternative to DBMS, Programming Language, Web Server, and File System". The Fifth International Conference on Evaluation of Novel Approaches to Software Engineering, 2010.
8 K. H. Xu, J. Zhang, S. Gao. "An Assessment on the Easiness of Computer Languages". To appear in the Journal of Information Technology Review, 2010.
9 K. H. Xu, J. Zhang, S. Gao, R. R. McKeown. "Let a Data Model be a Class of Total Recursive Functions". The 2010 International Conference on Theoretical and Mathematical Foundations of Computer Science (TMFCS-10), 2010.
10 K. H. Xu, J. Zhang. "A User's Guide to Froglingo, An Alternative to DBMS, Programming Language, Web Server, and File System, Release 1.0, January 2010". Available at the website: http://www.froglingo.com/ FrogUserGuide10.doc.
11 K. H. Xu, J. Zhang, S. Gao. "Assessing Easiness with Froglingo". The Second International Conference on the Application of Digital Information and Web Technologies, 2009, page 847 - 849.

Appendix A: A Sample XML Block from Recipe Book

```
<RECIPE>
<TI>Spicy Parmesan Chicken</TI>
<IN>2  Whole chicken breasts, split and fat removed -or-</IN>
<IN>1  Broiler chicken, cut up and skin removed</IN>
<IN>1/4 c Grated Parmesan cheese</IN>
<IN>1 ts Italian seasoning</IN>
<IN>1/2 c Mozzarella cheese, (2 ounces)</IN>
<IN>3/4 c Salsa</IN>
<PR>Preheat oven to 375 degrees. Place chicken, bone-side down, in a 9x13-inch
baking pan. Combine Parmesan cheese and Italian seasoning and sprinkle over
chicken. Bake for 15 minutes if using two breasts, 45 minutes for a broiler chicken.
Sprinkle with mozzarella cheese and bake 1 more minute. Serve with salsa.
</PR>
</RECIPE>
```