Isabelle Bichindaritz
Stefania Montani (Eds.)

LNAI 6176

Case-Based Reasoning Research and Development

18th International Conference on Case-Based Reasoning, ICCBR 2010 Alessandria, Italy, July 2010, Proceedings

Springer

Lecture Notes in Artificial Intelligence 6176

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Isabelle Bichindaritz Stefania Montani (Eds.)

Case-Based Reasoning Research and Development

18th International Conference
on Case-Based Reasoning, ICCBR 2010
Alessandria, Italy, July 19-22, 2010
Proceedings

Springer

Series Editors

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

Volume Editors

Isabelle Bichindaritz
University of Washington, Institute of Technology/Computing and Software Systems
1900 Commerce Street, Tacoma, WA, 98402, USA
E-mail: ibichind@u.washington.edu

Stefania Montani
University of Piemonte Orientale, Department of Computer Science
Viale Michel 11, 15121 Alessandria, Italy
E-mail: stefania.montani@unipmn.it

Library of Congress Control Number: 2010929801

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

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743

ISBN-10 3-642-14273-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-14273-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

The International Conference on Case-Based Reasoning (ICCBR) is the preeminent international meeting on case-based reasoning (CBR). Through 2009, ICCBR (http://www.iccbr.org) had been a biennial conference, held in alternation with its sister conference, the European Conference on Case-Based Reasoning (http://www.eccbr.org), which was located in Europe. At the 2009 ICCBR, the ICCBR Program Committee elected to extend an offer of consolidation with ECCBR. The offer was accepted by the ECCBR 2010 organizers and they have considered it approved by the ECCBR community, as the two conferences share a majority of Program Committee members. ICCBR and ECCBR have been the leading conferences on CBR. From 2010, ICCBR and ECCBR will be merged in a single conference series, called ICCBR. As there had been eight previous ICCBR events and nine previous ECCBR events, the combined series is considered the 18th ICCBR.

ICCBR 2010 (http://www.iccbr.org/iccbr10) was therefore the 18th in this series of international conferences highlighting the most significant contributions to the field of CBR. The conference took place during July 19–22, 2010 in the city of Alessandria, Italy, on the beautiful campus of the University of Piemonte Orientale "A. Avogadro." Previous ICCBR conferences were held in Sesimbra, Portugal (1995), Providence, Rhode Island, USA (1997), Seeon Monastery, Germany (1999), Vancouver, BC, Canada (2001), Trondheim, Norway (2003), Chicago, Illinois, USA (2005), Belfast, Northern Ireland (2007), and Seattle, Washington, USA (2009).

Day 1 of the conference hosted an Applications Track, the second Doctoral Consortium, and the third Computer Cooking Contest. The Applications Track featured fielded applications and CBR systems demos in industrial and scientific settings with an emphasis on discussion and networking between researchers and industrials. The Computer Cooking Contest featured papers selected for their technical quality, originality of the approach, culinary quality, and relevance of the created recipes. It ended with a competition, which showcased intriguing intelligent systems rivaling with humans in the kitchen. The second Doctoral Consortium allowed doctoral students to connect with senior researchers mentors from the CBR community.

Day 2 was dedicated to four workshops and the poster session. The four workshops were dedicated to "Case-Based Reasoning for Computer Games," "Provenance-Aware CBR: Applications to Reasoning, Metareasoning, Maintenance and Explanation," "CBR Startups," and "WebCBR: Reasoning from Experiences on the Web." The poster session allowed for interactive and in-depth discussions of research advances.

Days 3 and 4 comprised scientific paper presentations on theoretical and applied CBR research. The presentations and posters covered a wide range of CBR topics including adaptation, bioinformatics, case mining, case retrieval, computer games, experience on the Web, introspective reasoning, knowledge acquisition, knowledge management, knowledge representation, planning, similarity, temporal reasoning, and textual CBR.

The conference was proud to present three distinguished invited speakers: Riccardo Bellazzi (University of Pavia, Italy) introduced the audience to translational bioinformatics, its challenges and opportunities for CBR and decision support systems; Amedeo Napoli (LORIA, France) explained why and how knowledge discovery can be useful for solving problems with CBR; Ashwin Ram (Georgia Institute of Technology, USA) presented real-time CBR for interactive digital entertainment. We are grateful for their innovative ideas.

This volume includes 18 papers from oral presentations and 17 from posters. These were chosen from a total of 53 submissions. In addition, the volume contains three papers from invited speakers. The accepted papers were chosen based on a thorough and highly selective review process. Each paper was reviewed and discussed by four reviewers and revised according to their comments. Reviewers were encouraged to reach a consensus when they did not agree, which they generally managed to accomplish through lively discussions. The papers in this volume provide a representative snapshot of current CBR research. We have organized the proceedings in three parts: *Invited Talks* (3 short papers), *Theoretical/Methodological Research Papers* (12 papers), and *Applied Research Papers* (13 papers).

Many people participated in making ICCBR possible. First of all, Stefania Montani (University of Piemonte Orientale, Italy) doubled her role as Scientific Chair with that of Conference Chair this year. She also had the initiative to propose ICCBR 2010, thus inviting us all to beautiful Italy. The organization team was very diverse, having Cindy Marling (Ohio University, USA) as coordinator of the Workshop Program; Jerzy Surma (Warsaw School of Economics, Poland) as chair of the Applications Track; Klaus-Dieter Althoff (University of Hildesheim, Germany) as organizer of the Doctoral Consortium. This diverse team together with the authors, the invited speakers, the Program Committee, and additional reviewers are the stars of the CBR community in 2010. They made the conference happen and we want to thank them for their brilliant performances that are recorded in this volume. We gratefully acknowledge the generous support of the sponsors of ICCBR 2010.

Additional help was provided by doctoral students from the University of Piemonte Orientale in Italy. In support of local arrangements, thanks to the Local Arrangements Committee from the University of Piemonte Orientale. The submission and reviewing process was carried out with the use of EasyChair. Finally, we thank Springer for its continuing support in publishing this series of conference proceedings.

May 2010

Isabelle Bichindaritz
Stefania Montani

Conference Organization

Program Chairs

Isabelle Bichindaritz	University of Washington Tacoma, USA
Stefania Montani	University of Piemonte Orientale, Italy

Conference Chair

Stefania Montani	University of Piemonte Orientale, Italy

Application Track Chair

Jerzy Surma	Warsaw School of Economics, Poland

Workshop Coordinator

Cindy Marling	Ohio University, USA

Doctoral Consortium Chair

Klaus-Dieter Althoff	University of Hildesheim, Germany

Cooking Contest Chairs

David Aha	Naval Research Laboratory, USA
Amélie Cordier	University Lyon 1, France

Program Committee

Agnar Aamodt	Norwegian University of Science and Technology, Norway
David Aha	Naval Research Laboratory, USA
Klaus-Dieter Althoff	University of Hildesheim, Germany
Josep-Lluis Arcos	IIIACSIC, Spain
Kevin Ashley	University of Pittsburgh, USA
Paolo Avesani	FBK-IT, Italy
Ralph Bergmann	University of Trier, Germany
Enrico Blanzieri	University of Trento, Italy
Derek Bridge	University College Cork, Ireland

Robin Burke	De Paul University, USA
Hans-Dieter Burkhard	Humboldt University Berlin, Germany
William E. Cheetham	General Electric Research, USA
Susan Craw	Robert Gordon University, UK
Belen Diaz-Agudo	Complutense University of Madrid, Spain
Peter Funk	Mlardalen University, Sweden
Ashok Goel	Georgia Institute of Technology, USA
Mehmet Göker	PriceWaterhouseCoopers, USA
Andrew Golding	Lycos Inc., USA
Pedro González Calero	Complutense University of Madrid, Spain
Christiane Gresse von Wangenheim	Uni. do Vale do Itajai, Brazil
Kalyan Moy Gupta	Knexus Research Corporation, USA
Eyke Hüllermeier	University of Marburg, Germany
Igor Jurisica	Ontario Cancer Institute, Canada
Deepak Khemani	IIT Madras, India
Luc Lamontagne	Université Laval, Canada
David Leake	Indiana University, USA
Jean Lieber	LORIA, France
Ramon López de Mántaras	IIIACSIC, Spain
Michel Manago	kiolis, France
Cindy Marling	Ohio University, USA
Lorraine McGinty	University College Dublin, Ireland
David McSherry	University of Ulster, UK
Alain Mille	University Lyon 1, France
Mirjam Minor	University of Trier, Germany
Héctor Muñoz-Avila	Lehigh University, USA
Petra Perner	Institute of Computer Vision and Applied CS, Germany
Enric Plaza	IIIACSIC, Spain
Luigi Portinale	University of Piemonte Orientale, Italy
Lisa Purvis	Xerox Corporation, NY, USA
Ashwin Ram	Georgia Institute of Technology, USA
Francesco Ricci	ITC-irst, Italy
Michael Richter	University of Calgary, Canada
Thomas Roth-Berghofer	DFKI, Germany
Rainer Schmidt	University of Rostock, Germany
Barry Smyth	University College Dublin, Ireland
Armin Stahl	German Research Center for Artificial Intelligence, Germany
Jerzy Surma	Warsaw School of Economics, Poland

Brigitte Trousse	INRIA Sophia Antipolis, France
Ian Watson	University of Auckland, New Zeland
Rosina Weber	Drexel University, USA
Stefan Wess	Empolis, Germany
David C. Wilson	University of North Carolina at Charlotte, USA
Nirmalie Wiratunga	Robert Gordon University, UK
Qiang Yang	Hong Kong University of Science and Technology, Hong Kong

External Reviewers

Ibrahim Adeyanju, Kerstin Bach, Shahina Begum, Tore Bruland, Yen Bui, Sutanu Chakraborti, Eros Comunello, Amélie Cordier, Sidath Gunawardena, Joseph Kendall-Morwick, Stewart Massie, Regis Newo, Santiago Ontañón, Öztürk Pinar, Jay Powell, Rajendra Prasath, Juan A. Recio-Garcia, Mobyen Uddin Ahmed, Markus Weber, Ning Xiong

Sponsors

ICCBR 2010 was supported by Fondazione Cassa di Risparmio di Torino, Empolis, the Italian Association for Artificial Intelligence, the University of Piemonte Orientale “A. Avogadro” and the City of Alessandria.

Table of Contents

Invited Talks

Theoretical/Methodological Research Papers

Applied Research Papers

Translational Bioinformatics: Challenges and Opportunities for Case-Based Reasoning and Decision Support

Riccardo Bellazzi[1], Cristiana Larizza[1], Matteo Gabetta[1], Giuseppe Milani[1], Angelo Nuzzo[1], Valentina Favalli[2], and Eloisa Arbustini[2]

[1] Dipartimento di Informatica e Sistemistica, Università di Pavia, via Ferrata 1 27100 Pavia, Italy
[2] IRCCS Fondazione Policlinico S. Matteo, Pavia, viale Golgi 19, 27100 Pavia, Italy
{riccardo.bellazzi,cristiana.larizza,matteo.gabetta}@unipv.it,
{giuseppe.milani,angelo.nuzzo}@unipv.it,
{v.favalli,e.arbustini}@smatteo.pv.it

Abstract. Translational bioinformatics is bioinformatics applied to human health. Although, up to now, its main focus has been to support molecular medicine research, translational bioinformatics has now the opportunity to design clinical decision support systems based on the combination of -omics data and internet-based knowledge resources. The paper describes the state-of-art of translational bioinformatics highlighting challenges and opportunities for decision support tools and case-based reasoning. It finally reports the design of a new system for supporting diagnosis in dilated cardiomyopathy. The system is able to combine text mining, literature search and case-based retrieval.

Keywords: Translational bioinformatics, molecular medicine, decision support, dilated cardiomyopathy.

1 Introduction

Translational research focuses on making the results of research applicable to human being and, thus, on *translating* research results into practice. As the ultimate goal of biomedical research is to provide better care to patients, it is easy to understand that translational aspects are of paramount importance [1]. In the last few years, the need of focusing on translational research has become of crucial importance in the light of the enormous amount of results achieved in molecular medicine. In this context, bioinformatics played a crucial role: first, it was the main enabler to analyze the massive amount of data made available by biotechnological tools, including sequencing, genetics, genomics and proteomics; second, it took care to collect and organize the knowledge and "meta-data" which were accumulating during experimental activity. Recently, bioinformatics became so mature to give birth to its "translational" counterpart, called translational bioinformatics [2]. *Translational bioinformatics* is therefore bioinformatics applied to human health [3]. Together with supporting new

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 1–11, 2010.

discoveries, such as new diagnostic tests, new prognostic models or new therapeutic compounds, translational bioinformatics also deals with the full exploitation of "-omics" data to improve the quality and appropriateness of care of the single patient. This latter goal involves the design of genome-enabled electronic health records, and the implementation of novel decision support tools, able to get rid of the potentially vast amount of information available in biomedical data repositories to stratify patients' risk, prioritize diagnostic tests, suggest tailored patient interventions, and choose the most appropriate biomarkers for monitoring the disease progression [4,5]. Although innovative information technology infrastructures are starting to be available in hospitals, the implementation of decision support tools is still under way. In this paper we will describe the current state of art of translational bioinformatics and we will highlight some potential challenges that still need to be considered to design new decision support systems, able to deal with the complexity of modern medicine. The design and first results of a system for automated reasoning in molecular cardiology will be also shown.

2 Translational Bioinformatics: State of Art

The large number of initiatives funded by NIH to support biomedical computing witnesses the role of translational bioinformatics. In particular, the NIH National Centers for Biomedical Computing (NCBC) focus on how to deploy molecular medicine results in medical practice. For example, the i2b2 center at Partners HealthCare System, in Boston, is developing an informatics infrastructure to enable clinical researchers to re-use clinical data for discovery research and to facilitate the implementation of personalized medicine [6,7]. Another interesting case is represented by the National Center for Biomedical Ontology at Stanford, which aims at providing biomedical researchers and clinicians with a set of online tools and a Web portal which allow them to access, review, and integrate the different ontological resources currently available. It is expected that these tools will support not only biomedical investigators but also clinicians [8].

It is therefore not surprising that translational molecular medicine and translational bioinformatics are providing every year a variety of astonishing results. In public health, for example, data have been successfully analyzed to monitor the 2009 Influenza A (H1N1) virus [9]. In genetics, the application of novel alignment algorithms for new generation sequencing is leading to the identification of the causes of several diseases [10]. In genomics, several computational methods have been applied to integrate data coming from heterogeneous sources, providing novel instruments for information visualization [11]. In transcriptomics, the automated annotation of the so-called micro-RNA data is helping to elucidate fine regulation of cellular development and stem cell differentiation [12]. In proteomics and metabolomics several research projects are looking for disease biomarkers by resorting to statistics, machine learning and bioinformatics [13].

From the methodological viewpoint, every year, the AMIA Summit on translational bioinformatics reports the most interesting results in the field [14]. Among the different tracks, some areas are of particular interest:

- Mining medical records, which aims at finding relationships between clinical and –omics data [15].
- Applications of text mining, including literature-based discovery, which deals with the automated exploitation of the information contained in the electronic publications available on the Internet [16].
- IT infrastructures for supporting researchers, including knowledge management and workflows of in-silico experiments [5].

The joint availability of data, technological infrastructures and novel methodologies provide a great opportunity to move towards the next step of translational bioinformatics research: the support to clinicians in their day-by-day activity. In the next section we will review some of the recent efforts that we and other researchers put on exploiting automated reasoning modules to support scientific discovery and we will describe the design of a system for genome-enabled clinical decision support.

3 Reasoning, Decision Support and Translational Bioinformatics

3.1 Supporting Translational Research

Decision support methods and technologies are providing support to translational science to improve the data analysis process. As a matter of fact, it is nowadays possible to plan and execute "in-silico" experiments, which are complex sequences of data analysis steps that require to keep track of each intermediate results to allow to reconstruct the discovery process and/or to follow different reasoning strategies. However, the exponential increase of the amount of molecular data and the large number of knowledge sources available in the Internet requires new strategies for effectively planning and executing "in-silico" experiments, too. For this reason, widely used Internet resources, like SRS [17] and NCBI's Entrez [18] have sessions and query management functionality; moreover, the application of workflow management ideas has given rise to a number of innovative software tools to run data analysis sessions [19-21]. Rather interestingly, some systems have been recently implemented to perform fully automated discovery in molecular biology; in particular, the "Robot Scientist" system [22] runs experiments in a fully automated laboratory, interprets results, generates hypothesis and plans new experiments.

Since the mere implementation of linear workflows may be insufficient to support the knowledge discovery process, we have recently proposed a general architecture for the implementation of knowledge-based decision support systems (KB-DSS) in translational medicine. Our approach is based on a general epistemological model of scientific discovery process called Select and Test Model (ST-Model) [23]. The ST-Model was developed in the field of Artificial Intelligence in Medicine as a framework to design and implement expert systems [24]. It represents diagnostic reasoning as an iterative process made of a set of distinguished steps: *abstraction*, which allows to pre-process the available data; *abduction*, which generates a set of hypotheses by applying domain knowledge to the data; *ranking*, which sorts the hypotheses on the basis of their strengths; *deduction*, which derives consequences, e.g. expected findings, from the hypotheses; *eliminative induction*, which eliminates hypotheses that

have conflicting findings/data. In translational bioinformatics, this model can be applied to guide the development of KB-DSS able to integrate high-throughput data and existing knowledge. In particular we have proposed an instance of the ST-Model to support Genome-Wide Association Studies (GWAS), which aim at finding genetic risk factors related to a phenotype/disease of interest. GWAS look for statistically significant differences in the distribution of a set of genetic markers, called Single Nucleotide Polymorphisms (SNPs), between a group of cases, e.g. diseased people, and a group of controls, i.e. healthy subjects. As GWAS exploits very large numbers of SNPs (in the order of hundreds of thousands) in order to achieve a sufficiently good coverage of the entire genome, the analysis of high number of patients and the exploitation of knowledge available on the Internet is mandatory. In our proposal, the knowledge discovery steps of GWAS are explicitly modeled and mapped into computational procedures involving phenotype definition and patients selection (abstraction), statistical analysis and SPNs selection (hypothesis generation and ranking), access to SNP annotation databases to derive SNP-related genes and proteins (deduction), SNP filtering and ranking revision in the light of the available knowledge (eliminative induction).

3.2 Supporting Personalized Medicine

The next challenge for translational bioinformatics is to move from “bench” to “bed”, i.e. not only to support research on human data but also to provide instruments and tools to assist clinical decision-making. Looking at the most recent papers on personalized medicine [25] the frontier is to properly assess patients’ risk and clinical conditions to decide the most appropriate therapy for that particular subject.

The problems that translational bioinformatics are facing with do not differ from the standard decision making framework: they can be broadly classified into risk assessment and diagnosis/therapy planning. Risk assessment typically combines genetic markers with life-style information to predict the probability of a future disease. This assessment may be used to suggest changes in life style and /or to intervene with prevention programs.

Diagnosis and therapy planning are based on a variety of different data, which include, together with clinical findings, genetic and non-genetic molecular markers (i.e. gene expression, proteins, lipids, metabolites). One of the most interesting and distinguished aspects of building decision support systems for translational medicine is that knowledge is accumulating fast. The choice of markers and the evaluation of their potential effects should be therefore both based on formalized knowledge represented in knowledge repositories, such as OMIM, Gene Ontology, KEGG and on new reported results published in Pubmed abstracts and papers.

The current challenge is therefore to properly combine standard technologies for decision support with text and knowledge mining. In the following section we will describe a project on dilated cardiomiopathy that also concerns the development of a knowledge management system, which relies on a set of different technologies to support diagnosis and therapy planning.

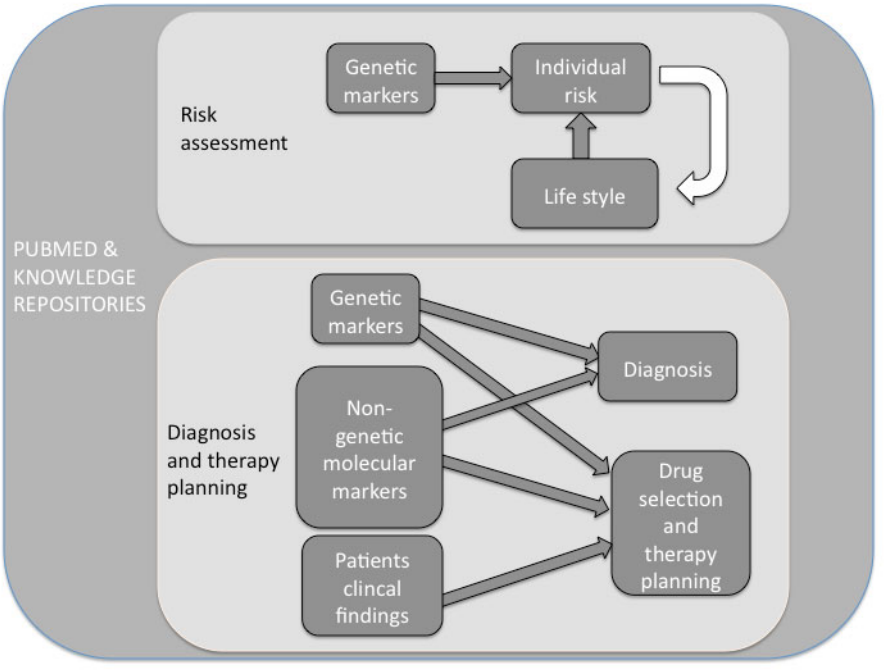


Fig. 1. Areas of decision support for molecular medicine and translational bioinformatics applications

4 Supporting Decisions in Genetic Dilated Cardiomiopathy: The Inheritance Project

4.1 Dilated Cardiomyopathy

Dilated cardiomyopathy (DCM) is a myocardial disorder characterized by the presence of left ventricular dilatation and systolic impairment, in the absence of abnormal loading conditions (e.g. hypertension, valve disease) or coronary artery disease [26]. DCM is the most common cardiomyopathy in the paediatric population [27,28]. The prognosis of DCM is highly variable with five-year survival rates of approximately 20%.

It is known that up to 35% of DCM have genetic causes. Family screening and genetic studies have identified more than 30 disease-causing genes to date [29]. Variation of age of onset, disease severity and prognosis amongst members of families carrying the same causative mutation suggests that modifier genes play a role in disease expression and response to pharmacological therapy [29].

The symptoms and signs of DCM are highly variable and depend partly on the degree of left ventricular dysfunction and on the underlying aetiology. The majority of patients show symptoms of high pulmonary venous pressure and a low cardiac output. Associated cardiac symptoms (such as left ventricular non-compaction or short PR)

and non-cardiac symptoms such as hearing loss, retinitis pigmentosa, myopathy, can be present and are important to identify the causative genes [26].

Currently patients with DCM are treated in accordance with international guidelines for the management of heart failure with little consideration of the possible influence of the underlying aetiology, i.e. of the causative genes on the response to treatment. Recent studies suggest that this might result in sub-optimal or inappropriate therapy in some patients. The influence of genetic factors in determining the response (and timing) of drug therapy is largely unstudied in DCM.

The European Commission has recently funded the Inheritance project, leaded by the IRCCS Policlinico San Matteo Hospital of Pavia, which aims at studying the disease by improving genetic diagnosis and treatment. Within the project, we are working on the definition of a novel decision support system.

4.2 Decision Support

Among the different decision support aspects related to the Inheritance project, we are currently working on the definition of a tool that may guide the clinicians in properly ranking the DCM causative genes, so that their screening can be effectively performed in the clinic. We can therefore consider this task as a diagnostic problem, where we must prioritize around 30 genes for screening on the basis of the patients' symptoms. The large variability of the patients' data and the limited amount of formalized knowledge available requires the design of a decision support tool able to provide instruments for analogical reasoning to clinicians, including case similarity, information retrieval and text mining.

Setting the problem. Each clinical case is usually described by hundreds of features, including anamnesis and family information, life-style, lab tests and exams, ECGs, echo-cardiography data. Among the collected data, some of them are considered as "red flags", i.e. biomarkers that may be related to some gene mutation, as their cause-effect relationships have not yet been fully established. Given the very nature of the problem, we have implemented a decision support strategy that is reported in Figure 2. All Pubmed abstracts are retrieved and included into an abstract corpus. The list of red flags and the list of genes are then searched into the documents together with their synonyms relying on the UMLS meta-thesaurus and on the Gene database. The results can be analyzed in order to find established gene-red flag relationships, as can be found in OMIM, and predicted relationships that can be found by the occurrence of genes and diseases in the abstracts. Let us note that, since UMLS concepts are hierarchically interrelated, the relationships between a gene and a red flag may also occur at different levels of the hierarchy, including their descendants (more specific concepts) or their ancestors (more general ones). The final matching process give rise to an augmented weighted list of red flags related to a gene, where the weight can be calculated on the basis of the frequency of the gene/red flag relationships. Once a single patient case is available, it is possible to compute a matching function with the current patients data and the weighted list of red flag to derive a prioritized list of genes. Moreover, it is also possible to retrieve similar cases with known mutations and therapy and highlight right or wrong previous diagnostic decisions. Rather interestingly, the process evolves over time, and the list of red flags may be varied accordingly to the change in the available knowledge reported in Pubmed and in knowledge repositories.

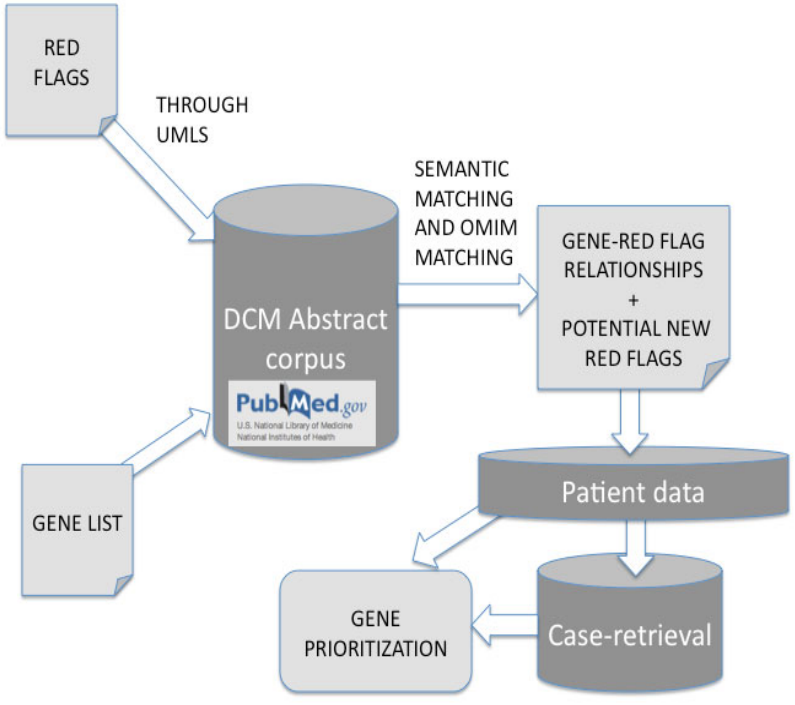


Fig. 2. The current design of the decision support system for gene prioritization, which will be implemented in the Inheritance project. Red flags (symptoms, diseases) and candidate genes are searched in the DCM Pubmed abstract corpus. Thanks to a semantic matching of the genes and red flags the genes are ordered and the list of potential red flag augmented. Matching the information with the patient's data allow to prioritize genes. Case-based retrieval allows extracting similar cases and gene priorities formerly defined in similar cases.

Current implementation. The current stage of implementation of the decision support system is related to the generation of an augmented list of red flags and to the computational mechanisms to prioritize the genes to be screened.

To create the DCM related corpus we developed a search engine based on the Entrez Programming Utilities (EUtils). EUtils are software tools that provide access to Entrez data outside of the regular web query interface, in order to retrieve search results in another environment [30]. In particular, we use the NCBI Web Service implementation, which enables developers to access Entrez Utilities via the Simple Object Access Protocol (SOAP). The module queries PubMed to retrieve scientific papers dealing with the pathology of interest (DCM) in XML format, so that their abstracts can be automatically processed and analyzed by customized NLP tools.

Once the DCM has been obtained, the biological concepts (i.e. genes and red flags mapped on UMLS concepts) contained in the corpus are searched. In order to obtain an exhaustive result, the initial set of UMLS concepts to be queried has been enriched

with the whole set of synonyms extracted from the UMLS thesaurus [31]. In particular, in order to carry on this task, we have implemented a concept extraction system that relies on the framework GATE [32]. The text analysis is handled through eight different steps, scheduled in a pipeline-like architecture. The first five steps are common to many text-mining systems, while the last three modules have been designed specifically for our purposes.

(1) The *Text Tokenizer* operates in two stages: the identification of parts of text separated by blank spaces and the management of the language-dependent exceptions. **(2)** The *Sentence Splitter* separates the sentences within the text. **(3)** The so-called "*ANNIE POS Tagger*" module assigns each previously identified token to the grammatical class (POS) that it belongs to. **(4)** The *Lemmatizer* derives the lemma belonging to every token (i.e. the canonical form). **(5)** The *Noun-Phrase-Chunker* identifies particular syntactical structures, called noun phrases (NP). **(6)** The *Acronym Extractor* identifies within text the typical pattern "*complete name* (*acronym*)". **(7)** The *Gene Extractor* identifies the genes in the analyzed text relying on the NCBI Gene database [34]. **(8)** The *UMLS Concept Extractor* extracts the UMLS concepts belonging to the specific semantic groups related with known red flags. Modules (7) and (8) rely on the information coming form the *Acronym Extractor*, in order to provide more reliable results.

Two output tables are then produced, one for the genes and one for the red flags, containing the most frequent concepts found in the DCM abstract corpus. The importance of these results is twofold: on the one hand they allow to validate the

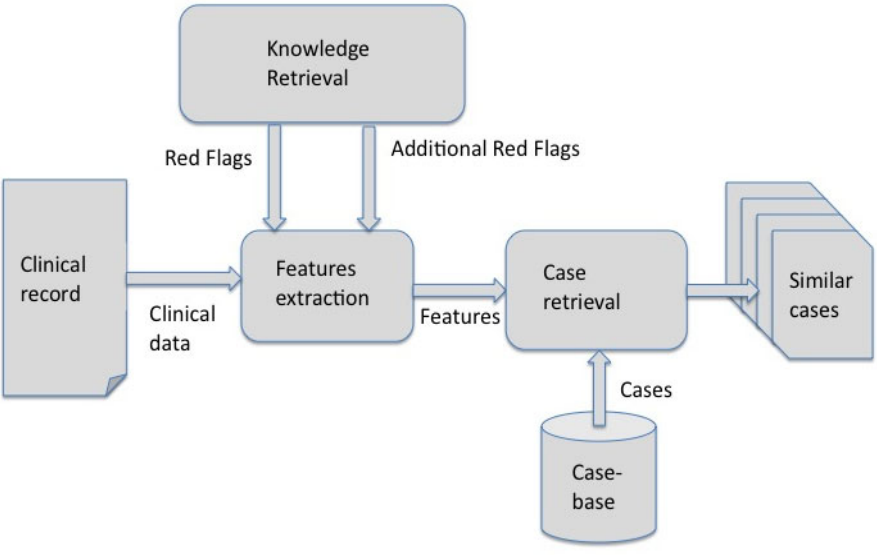


Fig. 3. Once red flags have been augmented, they are extracted from the clinical record and then retrieved in the case base. The knowledge retrieval step allows to (dynamically) change the feature space. Red flags may be weighted with their relevance.

background knowledge available on genes and red flags; on the other hand, they make possible to enrich prior knowledge on the basis of new information coming from the most recent literature, not yet included into the online knowledge resources. This phase could lead to define an augmented list of DCM relevant red flags, which can be searched in the case-base.

We have currently retrieved a corpus of more than 7000 documents by querying Pubmed for DCM in the period 2005-2010. From this corpus we extracted two tables reporting the list of 455 genes and 867 UMLS concepts. As a first task we verified that such lists contain the 27 genes and 20 red flags provided by the physicians as related to DCM. In this way it was possible to confirm the available background knowledge.

Then, we extracted a list of potential new genes or red flags related to DCM. Finally, for each red flag, we produced a list of associated genes ordered by their relevance. The relevance depends on the number of articles that support the red flag-gene association. This information provides a gene prioritization list that can be useful in clinical routine for diagnostic purposes.

As a simple example of this step, let's consider the red flag related to hearing loss problems (Table 1). We extracted the genes reported in the articles that cite any of the UMLS concepts related to this problem. Among the genes extracted, the most strongly associated to the hearing loss group (the one with the highest priority) is EYA4 gene, which also appears in the 27 genes list.

Table 1. Red flags for dilated cardiomiopathy

Atrioventricular Block	Abnormal Phosphate measurement	Fabry Disease	Abnormal Lactic acid measurement
Atrial Fibrillation	Myoglobinuria	Abnormal Creatinine clearance measurement	Proteinuria
Wolff-Parkinson-White Syndrome	Homocystinuria	Leukopenia	Angiokeratoma
Creatine kinase measurement	Hyperhomocysteinemia	Neutropenia	Deafness
Sudden death	Myopathy	Cataract/ Cornea verticillata	Holt-Oram syndrome

Once the revised list of red flags is obtained, similar cases may be found by applying a weighted similarity measures with the cases stored in a case-base of patients. The weights are provided by the relevance of the augmented red-flags to the problem. Case-based retrieval may provide clinicians with a set of past cases where the diagnosis have been already performed, which may further guide the screening and therapy selection process (see Figure 3). The Inheritance project started on January 2010, and the database of clinical cases will be available at the beginning of 2011. We plan to deploy the decision support system by mid 2011.

5 Conclusions

Translational bioinformatics is nowadays facing the challenge to support not only researchers, but also clinical practitioners. To do so, there's the need of merging classical bioinformatics methods, such as knowledge integration and text mining, with decision support tools coming from clinical medicine. Case-based reasoning seem to provide a suitable instrument in this case, as it may easily handle fuzziness and incompleteness of the available knowledge and it may adapt decision support to the constant increase in the body of knowledge. In the next future the paradigm of analogical reasoning is likely to become a crucial enabler to help the transition of bioinformatics from "bench to bed".

Acknowledgments. This work was partially funded by the projects Inheritance, funded by the European Commission and Italbionet, funded by the Italian Ministry of Research.

References

1. Ginsburg, G.S., Willard, H.F.: Genomic and personalized medicine: foundations and applications. Transl. Res. 154(6), 277–287 (2009)
2. Butte, A.J.: Translational bioinformatics applications in genome medicine. Genome Med. 1(6), 64 (2009)
3. Beavis, W.D., Schilkey, F.D., Baxter, S.M.: Translational Bioinformatics: At the Interface of Genomics and Quantitative Genetics. Crop Sci. 47, S-32–S-43 (2007)
4. Hoffman, M.A.: The genome-enabled electronic medical record. J. Biomed. Inform. 40(1), 44–46 (2007)
5. Kawamoto, K., Lobach, D.F., Willard, H.F., Ginsburg, G.S.: A national clinical decision support infrastructure to enable the widespread and consistent practice of genomic and personalized medicine. BMC Med. Inform. Decis. Mak. 9, 17 (2009)
6. Murphy, S.N., Weber, G., Mendis, M., Gainer, V., Chueh, H.C., Churchill, S., Kohane, I.: Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). J. Am. Med. Inform. Assoc. 17(2), 124–130 (2010)
7. Murphy, S., Churchill, S., Bry, L., Chueh, H., Weiss, S., Lazarus, R., Zeng, Q., Dubey, A., Gainer, V., Mendis, M., Glaser, J., Kohane, I.: Instrumenting the health care enterprise for discovery research in the genomic era. Genome Res. 19(9), 1675–1681 (2009)
8. Shah, N.H., Bhatia, N., Jonquet, C., Rubin, D., Chiang, A.P., Musen, M.A.: Comparison of concept recognizers for building the Open Biomedical Annotator. BMC Bioinformatics 10(suppl. 9), S14 (2009)
9. Trifonov, V., Khiabanian, H., Rabadan, R.: Geographic dependence, surveillance, and origins of the 2009 influenza A (H1N1) virus. N. Engl. J. Med. 361(2), 115–119 (2009)
10. Ng, S.B., Buckingham, K.J., Lee, C., Bigham, A.W., Tabor, H.K., Dent, K.M., Huff, C.D., Shannon, P.T., Jabs, E.W., Nickerson, D.A., Shendure, J., Bamshad, M.J.: Exome sequencing identifies the cause of a mendelian disorder. Nat. Genet. 42(1), 30–35 (2010)
11. Huttenhower, C., Haley, E.M., Hibbs, M.A., Dumeaux, V., Barrett, D.R., Coller, H.A., Troyanskaya, O.G.: Exploring the human genome with functional maps. Genome Res. 19(6), 1093–1106 (2009)
12. Song, G., Sharma, A.D., Roll, G.R., Ng, R., Lee, A.Y., Blelloch, R.H., Frandsen, N.M., Willenbring, H.: MicroRNAs control hepatocyte proliferation during liver regeneration. Hepatology 51(5), 1735–1743 (2010)

13. Illig, T., Gieger, C., Zhai, G., Römisch-Margl, W., Wang-Sattler, R., Prehn, C., Altmaier, E., Kastenmüller, G., Kato, B.S., Mewes, H.W., Meitinger, T., de Angelis, M.H., Kronenberg, F., Soranzo, N., Wichmann, H.E., Spector, T.D., Adamski, J., Suhre, K.: A genome-wide perspective of genetic variation in human metabolism. Nat. Genet. 42(2), 137–141 (2010)
14. http://summit2010.amia.org/session-details (last accessed May 4, 2010)
15. Meystre, S.M., Savova, G.K., Kipper-Schuler, K.C., Hurdle, J.F.: Extracting information from textual documents in the electronic health record: a review of recent research. Yearb Med. Inform., 128–144 (2008)
16. Yu, S., Tranchevent, L.C., De Moor, B., Moreau, Y.: Gene prioritization and clustering by multi-view text mining. BMC Bioinformatics, 11–28 (2010)
17. Etzold, T., Ulyanov, A., Argos, P.: SRS: information retrieval system for molecular biology data banks. Meth. Enzymol. 266, 114–128 (1996)
18. Schuler, G.D., Epstein, J.A., Ohkawa, H., Kans, J.A.: Entrez: molecular biology database and retrieval system. Methods Enzymol. 266, 141–162 (1996)
19. Romano, P.: Automation of in-silico data analysis processes through workflow management systems. Brief Bioinform. 9, 57–68 (2008)
20. Demsar, J., Zupan, B., Leban, G., Curk, T.: Orange: from experimental machine learning to interactive data mining. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 537–539. Springer, Heidelberg (2004)
21. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20(17), 3045–3054 (2004)
22. King, R.D., Rowland, J., Oliver, S.G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L.N., Sparkes, A., Whelan, K.E., Clare, A.: The Automation of Science. Science 324(5923), 85–89 (2009)
23. Riva, A., Nuzzo, A., Stefanelli, M., Bellazzi, R.: An automated reasoning framework for translational research. J. Biomed. Inform. (November 18, 2009) (Epub ahead of print)
24. Ramoni, M.F., Stefanelli, M., Magnani, L., Barosi, G.: An epistemological framework for medical knowledge-based systems. IEEE Trans. Syst. Man Cybern. 22(1), 14 (1992)
25. Frueh, F.W.: Back to the future: why randomized controlled trials cannot be the answer to pharmacogenomics and personalized medicine. Pharmacogenomics 10(7), 1077–1081 (2009)
26. Elliott, P., Andersson, B., Arbustini, E., Bilinska, Z., Cecchi, F., Charron, P., Dubourg, O., Kuhl, U., Maisch, B., McKenna, W.J., Monserrat, L., Pankuweit, S., Rapezzi, C., Seferovic, P., Tavazzi, L., Keren, A.: Classification of the cardiomyopathies: a position statement from the European Society of Cardiology Working Group on Myocardial and Pericardial Diseases. Eur. Heart J. 29, 270–276 (2008)
27. Lipshultz, S.E., Sleeper, L.A., Towbin, J.A., Lowe, A.M., Orav, E.J., Cox, G.F., Lurie, P.R., McCoy, K.L., McDonald, M.A., Messere, J.E., Colan, S.D.: The incidence of pediatric cardiomyopathy in two regions of the United States. N. Engl. J. Med. 348, 1647–1655 (2003)
28. Nugent, A.W., Daubeney, P.E., Chondros, P., Carlin, J.B., Cheung, M., Wilkinson, L.C., Davis, A.M., Kahler, S.G., Chow, C.W., Wilkinson, J.L., Weintraub, R.G.: National Australian Childhood Cardiomyopathy Study. The epidemiology of childhood cardiomyopathy in Australia. N. Engl. J. Med. 348, 1639–1646 (2003)
29. Ahamad, F., Seidman, J.G., Seidman, C.E.: The genetic basis of cardioac remodelling. Annu. Rev. Genomics. Hum. Genet. 6, 185–216 (2005)
30. E-utils: Entrez Programming Utilities, http://www.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html
31. Lindberg, D.A., et al.: Methods of information in medicine 32, 281–291 (1993)
32. Cunningham, H., et al.: Proc.of the 40th Anniversary Meeting of ACL, pp. 168–175 (2002)
33. Maglott, D., et al.: Nucleic Acids Res. 33(Database Issue), D54–D58 (2005)

Why and How Knowledge Discovery Can Be Useful for Solving Problems with CBR
(Extended Abstract)

Amedeo Napoli

LORIA, B.P. 70239, Équipe Orpailleur, Bâtiment B, F-54506 Vandœuvre-lès-Nancy
Amedeo.napoli@loria.fr

Abstract. In this talk, we discuss and illustrate links existing between knowledge discovery in databases (KDD), knowledge representation and reasoning (KRR), and case-based reasoning (CBR). KDD techniques especially based on Formal Concept Analysis (FCA) are well formalized and allow the design of concept lattices from binary and complex data. These concept lattices provide a realistic basis for knowledge base organization and ontology engineering. More generally, they can be used for representing knowledge and reasoning in knowledge systems and CBR systems as well.

Keywords: knowledge discovery in databases, Formal Concept Analysis, knowledge representation and reasoning.

1 Introduction

In this talk, we will discuss and illustrate links existing between knowledge discovery in databases (KDD), case-based reasoning (CBR), and knowledge representation and reasoning (KRR). KDD techniques especially based on Formal Concept Analysis (FCA) are well formalized and allow the design of concept lattices (from binary and complex data). These concept lattices provide a realistic basis for knowledge base organization, ontology engineering, and hierarchical reasoning. They can be used as a backbone for an ontology by transforming formal concepts into concepts representing knowledge [5,4]. From the point of view of CBR, concept lattices may be used for a series of tasks such as:

- case retrieval: assessing similarity between cases with similarity paths, and case or information retrieval [19,20],
- case adaptation: traversing the lattice structure for building adaptation paths between cases [8,9],
- case learning: organizing and updating the case base and the underlying concept lattice.

Moreover, FCA is related to data mining techniques, such a as itemset search and association rule extraction. The use of such techniques can improve the scalability of FCA w.r.t. the volume of data to be analyzed [28,29].

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 12–19, 2010.

By contrast, we will also investigate some aspects of CBR that can be reused in KDD and ontology learning, such as searching and managing ontology design patterns.

2 Knowledge Discovery in Databases (KDD)

Knowledge discovery in databases (KDD) consists in processing a large volume of data in order to extract useful and reusable knowledge units from these data. An expert of the data domain, the analyst, is in charge of guiding the extraction process, on the base of his/her objectives and domain knowledge. The extraction process is based on data mining methods returning information units from the data. The analyst selects and interprets a subset of the units for building "models" that may be further interpreted as knowledge units with a certain plausibility. The KDD process is performed with a KDD system based on components such as domain ontologies, data mining modules (either symbolic or numerical), and interfaces for interactions with the system, e.g. editing and visualization.

The KDD process can be considered along three main steps: data preparation, data mining, and interpretation of the extracted units. At each step, domain knowledge, possibly represented within ontologies, can play a substantial role for improving the KDD process [18]. Moreover, data mining methods can be either numeric or symbolic. In this talk, we will mainly focus on the second type and especially itemset search, association rule extraction, and Formal Concept Analysis (and extensions) [21].

The search for frequent itemsets consists in extracting from binary tables itemsets occurring with a support that must be greater than a given threshold [22,3,29,31]. Given a set of objects and a set of properties, an item corresponds to an attribute or a property of an object, and an itemset (a pattern) to a set of items. The support of an itemset corresponds to the proportion of objects owning the itemset, with respect to the whole population of objects. An itemset is frequent if its support is greater than a given frequency threshold $\sigma_{\mathtt{S}}$: a proportion at least equal to $\sigma_{\mathtt{S}}$ of objects own all items included in the itemset. The search for frequent itemsets is based on monotony constraints (base of the Apriori algorithm [1]). The search of frequent itemsets begins with the search of frequent itemsets of minimal length (or length 1). Then, the frequent itemsets are recorded and combined together to form the candidate itemsets of greater length. The non-frequent itemsets are discarded and all their super-itemsets. The candidate itemsets are tested and the process continues in the same way, until no more candidates can be formed.

From frequent itemsets it is possible to generate association rules of the form $\mathtt{A} \longrightarrow \mathtt{B}$ relating an itemset $\mathtt{A}$ with an itemset $\mathtt{B}$, that can be interpreted as follows: the objects owning $\mathtt{A}$ also own $\mathtt{B}$ with a support and a confidence [1,23]. More precisely, an association rule $\mathtt{A} \longrightarrow \mathtt{B}$ has a support defined as the support of the itemset $\mathtt{A} \cup \mathtt{B}$ and a confidence defined as the quotient $\mathtt{support}(\mathtt{A} \cup \mathtt{B})/\mathtt{support}(\mathtt{A})$ (that can be interpreted as a conditional probability). Then, a rule is valid if its confidence is greater than a confidence threshold $\sigma_{\mathtt{C}}$, and its support is greater

than the frequency threshold for itemsets σ_s (a valid rule can only be extracted from a frequent itemset).

The numbers of extracted itemsets and rules may be very large, and thus there is a need for pruning the sets of extracted itemsets and rules for ensuring a subsequent interpretation of the extracted units. This is especially true when the interpretation has to be done –and this is usually the case– by the analyst who is in charge of interpreting the results of the KDD process [6].

Actually, the search for itemsets and association rules are related to concept lattices: they correspond to a breadth-first search in the concept lattice associated with the formal context under study.

3 Formal Concept Analysis and Derived Formalisms

3.1 The Basic Framework of FCA

The framework of FCA is fully detailed in [12]. FCA starts with a formal context (G, M, I) where G denotes a set of objects, M a set of attributes, or items, and $I \subseteq G \times M$ a binary relation between G and M. The statement $(g, m) \in I$ is interpreted as "the object g has attribute m". Two operators $(\cdot)'$ define a Galois connection between the powersets $(2^G, \subseteq)$ and $(2^M, \subseteq)$, with $A \subseteq G$ and $B \subseteq M$:
$A' = \{m \in M \mid \forall g \in A : gIm\}$ and $B' = \{g \in G \mid \forall m \in B : gIm\}$.

For $A \subseteq G$, $B \subseteq M$, a pair (A, B), such that $A' = B$ and $B' = A$, is called a formal concept. In (A, B), the set A is called the extent and the set B the intent of the concept (A, B). Concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2$ ($\Leftrightarrow B_2 \subseteq B_1$). With respect to this partial order, the set of all formal concepts forms a complete lattice called the concept lattice of (G, M, I). As already mentioned above, natural links exist between between concept lattices, itemsets, and association rules [3,31,28].

When one consider non binary contexts, e.g. numerical or interval data, conceptual scaling is often used for binarizing data and for obtaining a binary formal context [12]. Then, a numerical dataset is described by a many-valued context. (G, M, W, I) is a many-valued context where G is a set of objects, M a set of numerical attributes, W a set of values (e.g. numbers), and I a ternary relation defined on the Cartesian product $G \times M \times W$. The fact $(g, m, w) \in I$ or simply $m(g) = w$ means that the object g takes the value w for the attribute m.

Then, classical algorithms can be applied for designing concept lattices from scaled contexts [16]. However, adapted algorithms for designing a concept lattice may be directly applied on more complex data such as numbers, intervals, or graphs [17,15,14,13].

3.2 Pattern Structures

Instead of applying discretization leading to space and time computational hardness, one may directly work on original data. A pattern structure is defined as a generalization of a formal context describing complex data [11,15].

In classical FCA, object descriptions are sets of attributes, which are partially ordered by set inclusion, w.r.t. set intersection: let $P, Q \subseteq M$ two attributes sets, then $P \subseteq Q \Leftrightarrow P \cap Q = P$, and $(M, \subseteq)$, also written $(M, \cap)$, is a partially ordered set of object descriptions. Set intersection $\cap$ behaves as a meet operator and is idempotent, commutative, and associative. A Galois connection can then be defined between the powerset of objects $(2^G, \subseteq)$ and a meet-semi-lattice of descriptions denoted by $(D, \sqcap)$ (standing for $(M, \cap)$). This idea is used to define pattern structures in the framework of FCA as follows.

Formally, let G be a set of objects, let $(D, \sqcap)$ be a meet-semi-lattice of potential object descriptions and let $\delta : G \longrightarrow D$ be a mapping associating each object with its description. Then $(G, (D, \sqcap), \delta)$ is a pattern structure. Elements of D are patterns and are ordered by a subsumption relation $\sqsubseteq$: $\forall c, d \in D,\ c \sqsubseteq d \Longleftrightarrow c \sqcap d = c$. A pattern structure $(G, (D, \sqcap), \delta)$ gives rise to two derivation operators $(\cdot)^{\square}$:

$$A^{\square} = \prod_{g \in A} \delta(g), \quad for\ A \subseteq G \quad and \quad d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\}, \quad for\ d \in (D, \sqcap).$$

These operators form a Galois connection between $(2^G, \subseteq)$ and $(D, \sqcap)$. Pattern concepts of $(G, (D, \sqcap), \delta)$ are pairs of the form (A, d), $A \subseteq G$, $d \in (D, \sqcap)$, such that $A^{\square} = d$ and $A = d^{\square}$. For a pattern concept (A, d), d is a pattern intent and is the common description of all objects in A, the pattern extent. When partially ordered by $(A_1, d_1) \leq (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2 \ \ (\Leftrightarrow d_2 \sqsubseteq d_1)$, the set of all concepts forms a complete lattice called pattern concept lattice. More importantly, the operator $(.)^{\square\square}$ is a closure operator and pattern intents are closed patterns. Existing FCA algorithms (detailed in [16]) can be used with slight modifications to compute pattern structures, in order to extract and classify concepts. Details can be found in [11,14,15].

Below, we analyze object descriptions as interval in numerical data. Pattern structures allows to directly extract concepts from data whose object descriptions are partially ordered. Considering a numerical dataset with objects in G and attributes in M, a meet operator $\sqcap$ on interval patterns can be defined as follows. Given two interval patterns $c = \langle [a_i, b_i] \rangle_{i \in \{1,...,|M|\}}$, and $d = \langle [e_i, f_i] \rangle_{i \in \{1,...,|M|\}}$, then: $c \sqcap d = \langle [minimum(a_i, e_i), maximum(b_i, f_i)] \rangle_{i \in \{1,...,|M|\}}$ meaning that a convexification of intervals on each vector dimension is operated. The meet operator induces the following subsumption relation $\sqsubseteq$ on interval patterns: $\langle [a_i, b_i] \rangle \sqsubseteq \langle [c_i, d_i] \rangle \Leftrightarrow [a_i, b_i] \supseteq [c_i, d_i],\ \forall i \in \{1, ..., |M|\}$ where larger intervals are subsumed by smaller intervals.

A numerical dataset with objects G and attributes M can be represented by an interval pattern structure. Let G be a set of objects, $(D, \sqcap)$ a meet-semi-lattice of interval patterns ($|M|$-dimensional interval vectors), and δ a mapping associating to any object $g \in G$ an interval pattern $\delta(g) \in (D, \sqcap)$. The triple $(G, (D, \sqcap), \delta)$ is an interval pattern structure (see examples and details in [15,13]).

Pattern structures are very useful for building concept lattices where the extents of concepts are composed of "similar objects" with respect to a similarity measure associated to the subsumption relation $\sqsubseteq$ in $(D, \sqcap)$ [13].

3.3 Relational Concept Analysis

Relational datasets are composed of a binary tables ($\texttt{objects} \times \texttt{attributes}$) and inter-object relations ($\texttt{objects} \times \texttt{objects}$). Formally, these binary tables introduce a set of objects $\texttt{G}_\texttt{i}$ described by a set of attributes $\texttt{M}_\texttt{i}$, and, as well, a set of relations $\texttt{r}_\texttt{k} \subseteq \texttt{G}_\texttt{i} \times \texttt{G}_\texttt{j}$. Relational datasets arise in a wide range of situations, e.g. Semantic Web applications [26], relational learning and data mining [10], refactoring of UML class models and model-driven development [27].

Relational Concept Analysis (RCA) extends FCA to the processing of relational datasets in a way allowing inter-objects links to be materialized and incorporated into formal concept intents. Links are thus scaled to become relational attributes connecting first objects to concepts and then concepts to concepts as role restrictions do in Description Logics (DL) [2]. The new attributes are complex properties reflecting the relational aspects of a formal concept. They nevertheless abide to the same classical concept formation mechanisms from FCA which means that the relational concept intents can be produced by standard FCA methods. Due to the strong analogy between role restrictions and relational attributes in RCA, formal concepts can be readily translated into a DL-based formalism [24], e.g. for ontology engineering purposes as in [5,4,25].

RCA was introduced and detailed in [24]. The data structure is described by a relational context family, composed of a set of contexts $\{\mathcal{K}_\texttt{i}\}$ and a set of binary relations $\{\texttt{r}_\texttt{k}\}$. A relation $\texttt{r}_\texttt{k} \subseteq \texttt{G}_\texttt{j} \times \texttt{G}_\ell$ connects two object sets, a domain $\texttt{G}_\texttt{j}$ ($\texttt{dom}(\texttt{r}_\texttt{k}) = \texttt{G}_\texttt{j}$) and a range $\texttt{G}_\ell$ ($\texttt{ran}(\texttt{r}_\texttt{k}) = \texttt{G}_\ell$). RCA is based on a "relational scaling" mechanism that transforms a relation $\texttt{r}_\texttt{k}$ into a set of relational attributes that are added to the context describing the object set $\texttt{dom}(\texttt{r}_\texttt{k})$. To that end, relational scaling adapts the DL semantics of role restrictions.

For each relation $\texttt{r}_\texttt{k} \subseteq \texttt{O}_\texttt{j} \times \texttt{O}_\ell$, there is an initial lattice for each object set, i.e. $\mathcal{L}_\texttt{j}$ for $\texttt{O}_\texttt{j}$ and $\mathcal{L}_\ell$ for $\texttt{O}_\ell$. For a relation $\texttt{r}_\texttt{k} \subseteq \texttt{O}_\texttt{j} \times \texttt{O}_\ell$, a relational attribute, is associated to an object $\texttt{o} \in \texttt{O}_\texttt{j}$ whenever $\texttt{r}_\texttt{k}(\texttt{o})$ satisfies a given constraint, where $\texttt{r}_\texttt{k}(\texttt{o})$ denotes the set of objects in $\texttt{O}_\ell$ in relation with $\texttt{o}$ through $\texttt{r}_\texttt{k}$. The relational attribute is denoted by $\forall \texttt{r}_\texttt{k}.\texttt{C}$ (universal scaling) when $\texttt{r}_\texttt{k}(\texttt{o}) \subseteq \texttt{extent}(\texttt{C})$ with $\texttt{r}_\texttt{k}(\texttt{o})$ possibly empty. The relational attribute is denoted by $\exists \texttt{r}_\texttt{k}.\texttt{C}$ (existential scaling) when $\texttt{r}_\texttt{k}(\texttt{o}) \cap \texttt{extent}(\texttt{c}) \neq \emptyset$. Other relational scaling operators exist in RCA and follow the classical role restriction semantics in DL.

Actually, RCA is a powerful mechanism for managing relations in the framework of FCA. In CBR, it could be used for example for associating elements of problem statements with elements of problem solutions, an association that was not possible in [9].

4 Elements for Discussion

Usually, considering knowledge systems, and CBR systems as well, knowledge units may have two major different origins: explicit knowledge (and cases) can be given by domain experts and implicit knowledge can be extracted from databases of different kinds, e.g. domain data or textual documents. Moreover, a KDD system, as any other knowledge system, improves its performance when it is

guided by domain knowledge [18]. Hereafter, some requirements for KDD systems, adapted from [6,30], are listed for discussion:

- A KDD system is a knowledge system: it should present to the user the underlying domain in an appropriate fashion and rely on domain knowledge (e.g. an ontology).
- Extending the system knowledge: domain representation should be extensible by addition of new concepts or classes resulting from mining or querying processes. Concepts and their instances must be reusable in queries. The question of extracting cases from data, which have to be made precise, remains open [7].
- Alternative classification and mining tools: it should be possible to define alternative classifications of data, e.g. alternative concept lattices. A set of different classification and mining tools should be available, possibly combining numerical and symbolic methods.
- Support to analysts: analysts should be supported by adequate visualization tools and in the interpretation of extracted units as well, in particular by domain knowledge.
- Monitoring and documenting the system evolution: tools managing versions can be used for monitoring changes in classes or concepts over time. The system should document the different steps of the knowledge discovery process.
- KDD is a flexible process and its results should reflect the plural nature of knowledge, i.e. extracting procedural or declarative knowledge units, and, as well, meta-knowledge units.
- KDD provides knowledge units for extending ontologies, and, reciprocally, knowledge systems and CBR systems can be used to guide and improve KDD.

Finally, the relations between knowledge representation, reasoning, and knowledge discovery with FCA, are explained as follows in [30]. Formal concepts and concept lattices provide a mathematization of real-world concept hierarchies. This yields a mathematical support to human reasoning, especially using the graphical representation of concept lattices. Then, conceptual knowledge discovery, considered as pattern discovery plus knowledge creation, can be guided by the design of concept lattices and a subsequent representation of the formal concepts within a knowledge representation formalism such as description logics. The process can be repeated until a satisfactory knowledge base is obtained.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast Discovery of Association Rules. In: Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI Press/MIT Press (1996)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)

3. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining frequent patterns with counting inference. SIGKDD Exploration Newsletter 2(2), 66–75 (2000)
4. Bendaoud, R., Napoli, A., Toussaint, Y.: Formal concept analysis: A unified framework for building and refining ontologies. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 156–171. Springer, Heidelberg (2008)
5. Bendaoud, R., Toussaint, Y., Napoli, A.: Pactole: A methodology and a system for semi-automatically enriching an ontology from a collection of texts. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 203–216. Springer, Heidelberg (2008)
6. Brachman, R., Anand, T.: The Process of Knowledge Discovery in Databases. In: Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 37–57. AAAI Press / MIT Press (1996)
7. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case base mining for adaptation knowledge acquisition. In: Veloso, M. (ed.) IJCAI 2007, pp. 750–755. Morgan Kaufmann, San Francisco (2007)
8. Diaz-Agudo, B., Gonzales-Calero, P.: Classification based retrieval using formal concept analysis. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 173–188. Springer, Heidelberg (2001)
9. Diaz-Agudo, B., Gonzales-Calero, P.: Formal Concept Analysis as a support technique for CBR. Knowledge-Based Systems 14, 163–171 (2001)
10. Dzeroski, S., Lavrac, N. (eds.): Relational Data Mining. Springer, Berlin (2001)
11. Ganter, B., Kuznetsov, S.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 129–142. Springer, Heidelberg (2001)
12. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Berlin (1999)
13. Kaytoue, M., Assaghir, Z., Messai, N., Napoli, A.: Two Complementary Classication Methods for Designing a Concept Lattice from Interval Data. In: Link, S., Prade, H. (eds.) FoIKS 2010. LNCS, vol. 5956, pp. 345–362. Springer, Heidelberg (2010)
14. Kaytoue-Uberall, M., Duplessis, S., Kuznetsov, S., Napoli, A.: Two FCA-Based Methods for Mining Gene Expression Data. In: Ferré, S., Rudolf, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 251–266. Springer, Heidelberg (2009)
15. Kuznetsov, S.: Pattern structures for analyzing complex data. In: Sakai, H., Chakraborty, M.K., Hassanien, A.E., Ślęzak, D., Zhu, W. (eds.) RSFDGrC 2009. LNCS, vol. 5908, pp. 33–44. Springer, Heidelberg (2009)
16. Kuznetsov, S., Obiedkov, S.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental & Theoretical Artificial Intelligence 14(2/3), 189–216 (2002)
17. Kuznetsov, S., Samokhin, M.: Learning closed sets of labeled graphs for chemical applications. In: Kramer, S., Pfahringer, B. (eds.) ILP 2005. LNCS (LNAI), vol. 3625, pp. 190–208. Springer, Heidelberg (2005)
18. Lieber, J., Napoli, A., Szathmary, L., Toussaint, Y.: First Elements on Knowledge Discovery guided by Domain Knowledge (KDDK). In: Yahia, S.B., Nguifo, E.M., Belohlavek, R. (eds.) CLA 2006. LNCS (LNAI), vol. 4923, pp. 22–41. Springer, Heidelberg (2008)
19. Messai, N., Devignes, M.D., Napoli, A., Smaïl-Tabbone, M.: Many-valued concept lattices for conceptual clustering and information retrieval. In: Ghallab, M., Spyropoulos, C., Fakotakis, N., Avouris, N. (eds.) 18th European Conference on Artificial Intelligence (ECAI 2008), Patras, Greece, pp. 127–131. IOS Press, Amsterdam (2008)

20. Messai, N., Devignes, M.D., Napoli, A., Smaïl-Tabbone, M.: Using domain knowledge to guide lattice-based complex data exploration. In: 19th European Conference on Artificial Intelligence (ECAI 2010), Lisbon, Portugal. IOS Press, Amsterdam (to appear, 2010)
21. Napoli, A.: A smooth introduction to symbolic methods for knowledge discovery. In: Cohen, H., Lefebvre, C. (eds.) Handbook of Categorization in Cognitive Science, pp. 913–933. Elsevier, Amsterdam (2005)
22. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1998)
23. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Pruning closed itemset lattices for association rules. International Journal of Information Systems 24(1), 25–46 (1999)
24. Rouane-Hacene, M., Huchard, M., Napoli, A., Valtchev, P.: A proposal for combining formal concept analysis and description logics for mining relational data. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 51–65. Springer, Heidelberg (2007)
25. Rouane-Hacene, M., Napoli, A., Valtchev, P., Toussaint, Y., Bendaoud, R.: Ontology Learning from Text using Relational Concept Analysis. In: Kropf, P., Benyoucef, M., Mili, H. (eds.) International Conference on eTechnologies (MCETECH 2008), pp. 154–163. IEEE Computer Society, Los Alamitos (2008)
26. Staab, S., Studer, R. (eds.): Handbook on Ontologies, 2nd edn. Springer, Berlin (2009)
27. Stahl, T., Voelter, M., Czarnecki, K.: Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, Chichester (2006)
28. Szathmary, L., Valtchev, P., Napoli, A., Godin, R.: Constructing iceberg lattices from frequent closures using generators. In: Boulicaut, J.-F., Berthold, M.R., Horváth, T. (eds.) DS 2008. LNCS (LNAI), vol. 5255, pp. 136–147. Springer, Heidelberg (2008)
29. Szathmary, L., Valtchev, P., Napoli, A., Godin, R.: Efficient Vertical Mining of Frequent Closures and Generators. In: Adams, N., Boulicaut, J.F., Robardet, C., Siebes, A. (eds.) IDA 2009. LNCS, vol. 5772, pp. 393–404. Springer, Heidelberg (2009)
30. Wille, R.: Why can concept lattices support knowledge discovery in databases? Journal of Experimental & Theoretical Artificial Intelligence 14(2/3), 81–92 (2002)
31. Zaki, M.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Transactions on Knowledge and Data Engineering 17(4), 462–478 (2005)

Real-Time Case-Based Reasoning for Interactive Digital Entertainment

Ashwin Ram

Cognitive Computing Lab
School of Interactive Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0760, USA
ashwin@cc.gatech.edu
www.cc.gatech.edu/~ashwin

Abstract. User-generated content is everywhere: photos, videos, news, blogs, art, music, and every other type of digital media on the Social Web. Games are no exception. From strategy games to immersive virtual worlds, game players are increasingly engaged in creating and sharing nearly all aspects of the gaming experience: maps, quests, artifacts, avatars, clothing, even games themselves. Yet, there is one aspect of computer games that is not created and shared by game players: the AI. Building sophisticated personalities, behaviors, and strategies requires expertise in both AI and programming, and remains outside the purview of the end user.

To understand why authoring Game AI is hard, we need to understand how it works. AI can take digital entertainment beyond scripted interactions into the arena of truly interactive systems that are responsive, adaptive, and intelligent. I will discuss examples of AI techniques for character-level AI (in embedded NPCs, for example) and game-level AI (in the drama manager, for example). These types of AI enhance the player experience in different ways. The techniques are complicated and are usually implemented by expert game designers.

I propose an alternative approach to designing Game AI: Real-Time CBR. This approach extends CBR to real-time systems that operate asynchronously during game play, planning, adapting, and learning in an online manner. Originally developed for robotic control, Real-Time CBR can be used for interactive games ranging from multiplayer strategy games to interactive believable avatars in virtual worlds.

As with any CBR technique, Real-Time CBR integrates problem solving with learning. This property can be used to address the authoring problem. I will show the first Web 2.0 application that allows average users to create AIs and challenge their friends to play them—without programming. I conclude with some thoughts about the role of CBR in AI-based Interactive Digital Entertainment.

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, p. 20, 2010.

Applying Machine Translation Evaluation Techniques to Textual CBR

Ibrahim Adeyanju, Nirmalie Wiratunga, Robert Lothian, and Susan Craw

School of Computing
Robert Gordon University, Aberdeen, Scotland, UK
{iaa,nw,rml,smc}@comp.rgu.ac.uk

Abstract. The need for automated text evaluation is common to several AI disciplines. In this work, we explore the use of Machine Translation (MT) evaluation metrics for Textual Case Based Reasoning (TCBR). MT and TCBR typically propose textual solutions and both rely on human reference texts for evaluation purposes. Current TCBR evaluation metrics such as precision and recall employ a single human reference but these metrics are misleading when semantically similar texts are expressed with different sets of keywords. MT metrics overcome this challenge with the use of multiple human references. Here, we explore the use of multiple references as opposed to a single reference applied to incident reports from the medical domain. These references are created introspectively from the original dataset using the CBR similarity assumption. Results indicate that TCBR systems evaluated with these new metrics are closer to human judgements. The generated text in TCBR is typically similar in length to the reference since it is a revised form of an actual solution to a similar problem, unlike MT where generated texts can sometimes be significantly shorter. We therefore discovered that some parameters in the MT evaluation measures are not useful for TCBR due to the intrinsic difference in the text generation process.

1 Introduction

Textual Case Based Reasoning (TCBR) deals with reusing past experience stored in the form of text such as reports, frequently asked question (faqs) and emails. However, there is the need to evaluate textual solutions proposed by a TCBR system. User evaluation is generally accepted as the best form of text evaluation but it is expensive and the aggregation of results from repeated experiments is likely to be difficult due to subjective user judgements. This is different and far more demanding than automated evaluation where experts provide reference texts only once. Therefore automated evaluation techniques that lead to metrics such as precision and recall (also known as accuracy and coverage) obtained by comparing proposed texts with reference solutions are preferred [6,13,1]. Although there have been reports to show good and reliable results in some domains [13,1], these simple metrics have also been reported to be insufficient to capture grammatical and semantic variations in texts that occur in other domains [2].

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 21–35, 2010.

Machine Translation (MT) on the other hand deals with producing an equivalent text from one language to another. Evaluation of machine translated text must therefore attempt to capture semantic meaning as well as differences in word choice and order (grammatical/ semantic variations). More sophisticated metrics than precision and recall have therefore been developed and used for text evaluation in MT research, since semantic meaning is crucial for successful translations. These metrics such as BLEU [17] and NIST [9] have also been reported to correlate highly with human judgements.

This paper presents the evaluation challenges for TCBR and how MT metrics can be employed to address them. We present the similarities and differences in MT and TCBR evaluation requirements and accordingly propose strategies to adapt MT metrics for TCBR. MT evaluation techniques are adaptable for use in TCBR because the common goal is to quantify the goodness of a piece of text suggested by text generation systems. We experiment with datasets from a health and safety incident reporting domain and compare results from applying MT evaluation with using the simple metrics of precision and recall. Analysis of our results show that MT metrics are generally better in capturing grammatical and semantic variations due to their use of multiple human references.

Other sections in this paper are as follows. Related works are reviewed in Section 2, while the text evaluation challenge and MT evaluation metrics are discussed in Sections 3 and 4 respectively. Experimental setup, evaluation and discussion of our results appear in Section 5, before conclusion in Section 6.

2 Related Work

The need to evaluate natural language texts is common to several research areas in computer science. These areas include (but not limited to) Information Retrieval (IR) [14,3], TCBR [7,19], Natural Language Generation [18,4] and MT [20,11]. Generally, we can group text evaluation techniques into two broad categories: qualitative and quantitative.

Qualitative techniques involve the use of humans (experts and non-experts) to determine the quality of some text produced by a machine. The results from several humans are then aggregated using statistical methods to judge the average quality of such texts. The major disadvantages are that these techniques are very expensive especially when expert knowledge is required and results are not easily reproducible as human judgement is subjective. Nevertheless, qualitative techniques have been used for evaluation across many application domains involving natural language processing and generation (e.g. [18,5]).

On the other hand, quantitative techniques involve the comparison of machine texts to one or more gold standards written by humans (usually experts). Here quality of the method is gauged according to similarity at the syntactic or semantic level. Quantitative techniques are typically less reliable as most of them depend on finding matching string patterns between the machine-produced texts and human gold standard(s). However, such techniques can be automated, are less expensive and are easily reproducible. This also allows for easy comparison across several algorithms that are designed for the same purpose.

Precision and Recall are two basic quantitative metrics [6,13] widely used for text evaluation across several disciplines especially IR and TCBR. The basic idea is to regard a piece of text as a bag of (key)words and to count common words between the machine and human texts. Proportions of these common words to the machine and human texts give a metric of precision and recall respectively. A major drawback is that the sequence of words in a piece of text is ignored and this can adversely affect the grammatical and semantic meaning. In other words, a machine text with high precision and recall might not necessarily be grammatically and/or semantically correct.

The edit distance (also called Levenshtein distance [16]) has also been used for text evaluation (e.g. [4]). This technique takes the sequence of words into account and is calculated in the simplest form as the number of delete, insert and substitute operations required to change the machine text into its human solution equivalent. Typically, different costs are associated with each of these edit operations. Nevertheless, the edit distance can give misleading values as well because the same piece of text can be written in several ways without loss of meaning. In particular, machine texts with a longer length will be unfavourably penalized by this technique.

The link between MT and TCBR has been previously employed to enhance retrieval [12] . MT models are used to predict links between each keyword in the problem to one or more solution keywords in the vocabulary. Such alignments were used to generate a pseudo-solution for a new query using the statistically best solution keywords linked to keywords in the query. The pseudo-solution and original query texts are used to retrieve similar cases rather than the query text alone. This led to improvements in retrieval accuracy. Our focus is different from this; we apply MT evaluation techniques rather than MT models to TCBR.

3 Challenges with Evaluating Textual Solutions

This section provides an overview of a textual reuse approach called Case Retrieval Reuse Net (CR2N) which helps to identify relevant sections in a retrieved solution text. Detailed discussion of the technique can be found in previous work [1,2]. The focus here is to highlight the challenges faced during experimental evaluation on a health and safety incident reporting domain. We present the domain of application and our task of generating textual solutions before discussing the related evaluation challenges.

3.1 Health and Safety Incident Reports

Our corpus consists of health and safety incident reports (H&S dataset) provided by the National Health Service in Grampian. A report consists of a textual description of the incident and the action taken by the health personnel on duty. Each record is also labelled with 1 of 17 care stage codes which identifies a group of records such as accidents that result in personal injuries, incidents during treatment or procedures etc. Our intention is to build a TCBR system

that assists less experienced health personnel to generate reports when resolving/recording incidents by using previous similar experiences. Therefore, the incident description serves as our problem while the solution text is the record of actions taken to resolve the incident for each case in our experiments.

3.2 Textual Solution Generation with CR2N

In previous work, we introduced the CR2N architecture for text reuse [1,2]. Here we discuss how this architecture is used to generate textual solutions and briefly outline the key steps. The CR2N architecture consists of two Case Retrieval Nets CRNs [15]: one to index the problem space and the other referred to as the Case Reuse Net (CReuseNet) for the solution space. Figure 1 illustrates the CR2N approach to annotating a retrieved solution text on a simple case base of six cases. There are five terms from the problem vocabulary (i.e. Problem Information Entities, PIEs) and four terms from the solution vocabulary (i.e. SIEs) respectively. Given a query, the best case (C_2 in figure 1) is retrieved by activating all relevant PIEs to the query which consists of PIE_1, PIE_2, PIE_4. Generally the more activations the more relevant a case is to the query. The activations are shown as solid arrows as opposed to dotted arrows for inactive links between information entities and the cases.

Generation of a solution text begins with the activation of SIEs from the most similar case. An SIE is a textual unit such as a keyword, phrase or sentence. When an SIE activates similar cases to those activated by the query within a specified k-neighbourhood of the retrieved solution, it is considered relevant to the query. Such a relevant solution term (SIE) becomes part of the solution text generated by the CR2N, otherwise it is discarded. The optimal k-value is determined empirically but has been found to be about one-third (or less) of the

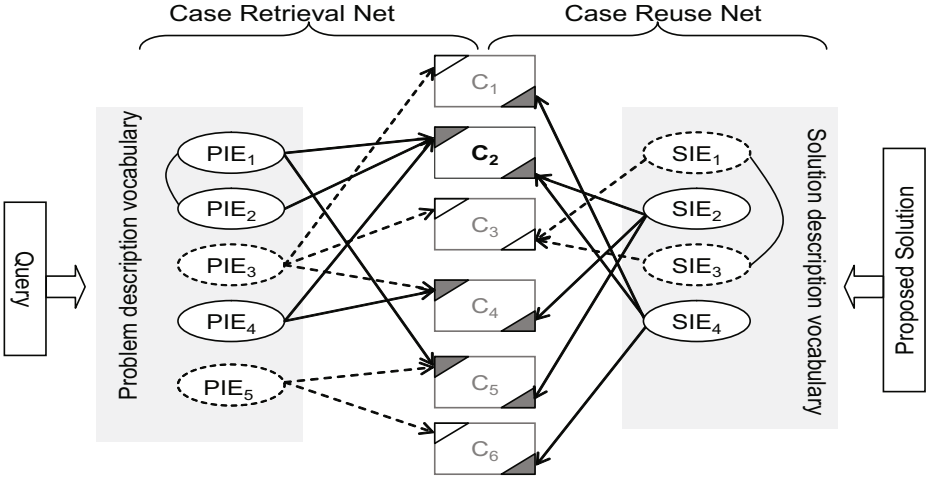


Fig. 1. The Case Retrieval Reuse Net (CR2N) architecture

size of the casebase [2]. Notice that this approach to solution text generation is different from directly proposing the solution from the best match case. Instead CR2N's approach is more akin to solution reuse, whereby the best solution is analyzed and only relevant parts of the solution are utilized to generate the proposed solution for a query. The proposed solution generated by CR2N consists of a list of reusable textual units. A complete solution is obtainable when other relevant textual units absent in the retrieved solution are added and all textual units are then put together to form a contextually coherent piece of text during revision.

3.3 Challenges with Evaluation

Quality of generated solution text from CR2N measured with precision, recall and accuracy metrics is reported in [2] using two domains: H&S incident reporting and weather forecast text generation. Both domains have the problem and solution in textual form. However, they also exhibit different textual characteristics such as vocabulary size, problem and solution vocabulary overlap and the use of synonyms. These characteristics influence the evaluation results; for instance, a large vocabulary size could mean that semantically similar texts will have few keywords in common. We compared our CR2N results with a baseline retrieve-only system and it showed a significantly better performance in both domains. However, we observed that the precision, recall and accuracy scores were comparatively lower (less than 0.5) with the H&S dataset compared to the weather forecast corpus (greater than 0.7).

Further investigation showed that these values were misleading in that proposed solutions judged relevant by a human would be judged otherwise by these basic metrics. This is because our evaluation measures (precision, recall & accuracy) only count matching keywords using their stems, lemma or synonyms.

Table 1. Sample retrievals from the H&S dataset

	Query	**Retrieved Problem (PIEs)**	**Similarity**	**Retrieved Solution (SIEs)**	**Reference Solution**	**Precision**
1	nurse slipt and fell on wet floor	staff member slid on something wet and fell to the floor	0.6124	examined by nursing staff	nurse given first aid	0.333
2	patient fell to the ground as nurse assisted him to bed	patient fell out of bed	0.7071	examined by medical staff	patient was advised to get assistance in and out of bed	0.0
3	needlestick injury sustained	needlestick injury sustained by a member of staff	0.7746	first aid, blood sample taken, visited occupational health	occupational health contacted	0.333

Therefore, they are unable to capture variation in phrases/sentences that have similar meanings but expressed by a completely different set of keywords. Poor accuracy results were also reported when retrieved solutions are more verbose than the reference solution.

Table 1 shows three incident queries as well as the retrieved case, similarity value and retrieval accuracies. With query 1, although the retrieved and reference solutions are similar in meaning, retrieval accuracy is just 0.333. This is because 1 out of 3 keywords ("nurse/nursing") is matched in the retrieved solution and the remaining keywords though semantically similar are lexically different. Query 3 poses a similar challenge while query 2 highlights a slightly different problem. Here, the level of detail/abstraction in the reference solution is different from retrieved solution thereby causing the accuracy to be calculated as 0.0.

Our hypothesis is that the use of multiple references in MT evaluation techniques will better capture the inherent variability in vocabulary as observed in the H&S dataset. The use of multiple references might also be able to reduce the problem associated with different levels of abstraction.

4 MT Evaluation Techniques

Machine Translation (MT) is a research area that deals with techniques to enable automated translation from one language to another. There is therefore a need to evaluate such machine generated translations (usually in textual form) for grammatical and semantic correctness. Initial research in MT used human expert translators for evaluating several aspects of a translated text in terms of adequate coverage, semantic meaning and grammatical correctness [20,11]. However, more recent work [17,9] has reduced the demand for user-driven quality assessments by developing automated text comparison techniques with high correlation to human judgements. As a result, automated MT evaluation techniques are quick, inexpensive, language independent and repeatable.

4.1 BLEU

BLEU [17] (**Bi**-**L**ingual **E**valuation **U**nderstudy) is an automated MT evaluation technique and it was used as an understudy of skilled human judges in translation. The idea is to measure the closeness of a machine text to its human equivalent using weighted average of phrases matched with variable length (n-grams). It enables the use of multiple reference solutions from different experts and this allows for legitimate differences in word choice and order. The BLEU score is a precision-based metric which can use multiple reference solutions and aggregates the precision scores from different word lengths; this concept is known as *modified n-gram precision*. BLEU also ensures that the machine text's length is comparable to the reference solutions' using *brevity penalty*.

Modified n-gram precision matches position independent n-grams; where $n \geq 1$ and grams are typically keywords but can include stand-alone special characters and punctuations. This is similar to precision measure in Information

Retrieval (IR). However, it is modified to ensure that n-grams can be matched across multiple reference solutions. Each n-gram is matched in only one of the reference solutions with the maximum count for such n-gram. The overall n-gram precision is a geometric average of all individual precisions from 1 to n. Using $n = 4$ has been found to give the best correlation to human judgements [17]. In comparison to the criteria used in human evaluation, uni-gram precision (i.e. $n = 1$) measures adequate coverage of a machine text while n-gram precision (when $n > 1$) shows grammatical correctness.

Brevity Penalty (BP) on the other hand ensures that the length of machine text is penalized if it is shorter than all the reference solutions. This is because a text of shorter length might have a very high n-gram precision if most of its keywords occur in any of the reference solutions. Therefore, modified n-gram precision alone fails to enforce proper translation length. BP focuses mainly on penalizing shorter machine texts as unnecessarily long texts will have been penalized by the modified n-gram precision. Although recall has been combined with precision to overcome problems with text lengths in some areas like IR, it cannot be used in BLEU because it employs the use of multiple references and each reference might use different word choices and order. Also, recalling all choices is bad since a good translation will only use one of the possible choices. BP is formulated as a decaying exponential function which gives a value of 1 when machine text's length is greater than or identical to any of the reference solutions length otherwise $BP < 1$. The BLEU metric is calculated as follows.

$$p_n = \frac{\sum_i \left(\begin{array}{c} \#\text{ of } n\text{-grams in segment } i \text{ of machine text} \\ \text{matched in segment } i \text{ of any of the reference solutions} \end{array} \right)}{\sum_i (\#\text{ of } n\text{-grams in segment } i \text{ of machine text})}$$

$$BP = \begin{cases} 1 & \text{if } l_{sys} > l^*_{ref} \\ \exp^{(1-\frac{l^*_{ref}}{l_{sys}})} & \text{if } l_{sys} \leq l^*_{ref} \end{cases}$$

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} \frac{1}{N} \log p_n)$$

where

$p_n = n$-gram precision $\quad BP =$ brevity penalty
$l_{sys} =$ length of machine text $\quad i = 1$ for TCBR
$l^*_{ref} =$ nearest reference solution length to machine text
$N =$ maximum size of n-gram (i.e. $n = 1 \ldots N$)

It is important to note that the entire text is typically regarded as one segment in TCBR (i.e. $i = 1$) when calculating p_n. This is because there is usually no knowledge of aligned segments between proposed and reference texts unlike MT where translations are done segment by segment. Figure 2 shows an example from our H&S dataset with multiple references and is used in the sample BLEU calculation shown in figure 3. Here, we compare the generated solution with

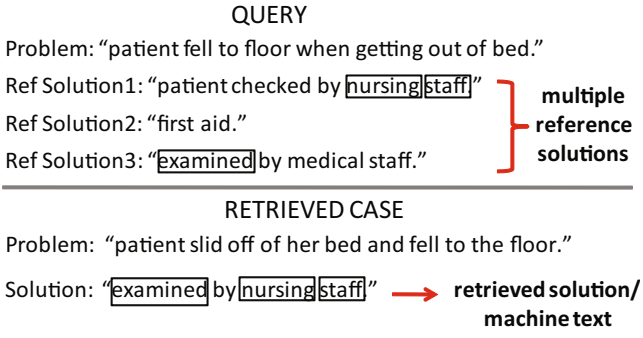


Fig. 2. A test case with multiple reference solutions

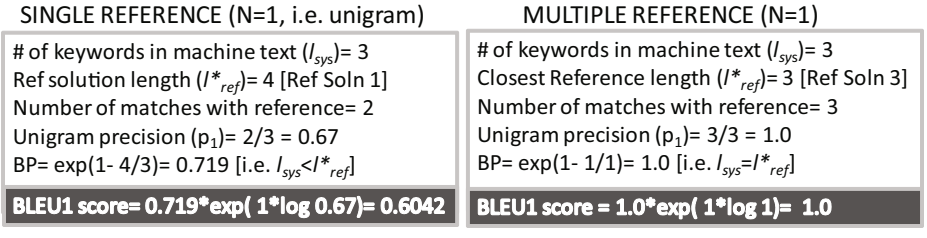


Fig. 3. A sample BLEU calculation with H&S dataset

the three reference solutions. Precision with a single reference solution (say Ref Solution1) matches only keywords "nursing" and "staff" from the machine text. However, keyword "examined" is also matched when multiple reference solutions are in use. A larger BLEU score is therefore obtained with multiple references.

4.2 NIST n-gram Co-occurrence Statistics

NIST n-gram co-occurrence statistics [9] is a more sophisticated MT evaluation technique. It was designed while experimenting with BLEU for stability and ability to reliably predict human quality assessments. NIST builds on the BLEU idea by modifying the weighting scheme for calculating precision. This is done by using information weights rather than frequency of occurrence and an arithmetic average of n-gram weights as opposed to geometric mean of n-gram precisions. Information weights are computed for n-grams such that those that occur less frequently have more weights as they are deemed to be more informative. In addition, brevity penalty was modified to minimize the impact of small variations in the generated text's length as they do not generally affect human judgements.

A significant improvement in stability and reliability was reported with NIST when compared with BLEU from experiments across several copora [9]. In other words, NIST is less sensitive to variation in the level of human expertise. Its correlation to human judgement is also more consistent across corpora from different languages. The NIST formula is given below.

$$BP = \begin{cases} 1 & \text{if } l_{sys} > \bar{l}_{ref} \\ \exp\left\{\beta \log^2 \left[min\left(\frac{l_{sys}}{\bar{l}_{ref}}, 1\right)\right]\right\} & \text{if } l_{sys} \leq \bar{l}_{ref} \end{cases}$$

$$\begin{aligned} \text{info}(n\text{-gram}) &= \text{info}(w_1 \dots w_n) \\ &= \log_2 \left(\frac{\#\text{ of } w_1 \dots w_{n-1} \text{ in reference solutions}}{\#\text{ of } w_1 \dots w_n \text{ in reference solutions}}\right) \end{aligned}$$

$$NIST = BP \cdot \sum_{n=1}^{N} \left\{ \frac{\sum \forall n\text{-gram} \in \text{sys} \, \text{info}(n\text{-gram})}{\#\text{ of } n\text{-grams in machine text}} \right\}$$

where

w= a word in the machine text
info= information weight
N = maximum size of n-gram (i.e. $n = 1 \dots N$)
$\beta = -4.3218$, chosen such that BP=0.5 when $l_{sys}/\bar{l}_{ref}$=2/3
l_{sys} = number of words in machine text (sys)
$\bar{l}_{ref}$ = average number of words in reference solutions

A sample NIST calculation which also uses the example test case from our H&S dataset (see figure 2) is shown in figure 4. NIST penalizes shorter machine text more as shown by the smaller BP score as compare to BLEU's for a single reference. As expected, the NIST values obtained are larger than BLEU's due to the use of information weights. NIST values can also be greater than 1 as opposed to BLEU values which are always between 0 and 1. Larger NIST (or BLEU) scores indicate better machine text's correlation to human judgement.

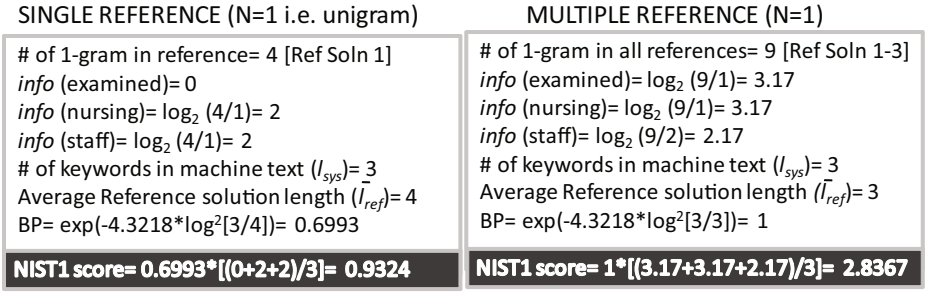


Fig. 4. A sample NIST calculation with H&S dataset

5 Experimental Setup

We evaluate the quality of the text generated by CR2N on the H&S dataset using the MT evaluation metrics, BLEU and NIST, discussed in sections 4.1 and 4.2 respectively after creating a new dataset with multiple references. Our

new dataset is also evaluated using the previous metric of precision and compare results with those obtained from the MT metrics. Multiple references give better evaluation results as they are better able to capture grammatical variations in texts but obtaining multiple references is not trivial. Therefore a novel introspective approach is employed to generate these references for our evaluations.

5.1 Generation of Dataset with Multiple Human References

Our original H&S incidents dataset consist of 362 cases belonging to the same care stage code. Each case has just 1 sentence in both the problem and solution texts since our evaluation metrics work at keyword granularity and alignment of sentences across cases are unknown. A new dataset with multiple reference solutions is needed to test our hypothesis that multiple references capture variability in word choice/order during evaluation. However, such multiple references were absent in the original H&S dataset. The CBR assumption that *similar problems have similar solutions* implies that identical problems should have same solutions. We therefore exploited this similarity assumption to create a new dataset from the original dataset with multiple references which was hitherto absent. This is done in a leave one out experiment design where each case is used as a query to retrieve the nearest neighbours. Solutions from neighbours with a similarity of 1 are then selected to form multiple reference solutions for each case while ignoring identical solutions. Here, a similarity of 1 does not necessarily mean that the problem texts are identical. This is because our similarity metric uses a bag of word representation in which stop words are removed and keywords stemmed. This process led to the extraction of 34 cases generally with 2 to 4 multiple non-duplicated reference solutions. An example of such a test case with three solutions is shown in Table 2. These 34 cases were used as test cases while the remaining 328 cases formed our case base.

Table 2. A sample test case with multiple solutions created from the previous dataset

Problem: "patient fell to floor when getting out of bed."
Solution1: "patient checked by nursing staff."
Solution2: "first aid."'
Solution3: "examined by medical staff."

The problem and solution texts are preprocessed using the GATE library [8] where texts are split into keywords. Stop words are removed and keywords stemmed to cater for morphological variations. During evaluation, synonym keywords are matched using WordNet [10] as well as keywords with the same lemma but different stems (e.g. gave/ given, fallen/ fell etc).

5.2 Evaluation and Discussion

We explore the usefulness of MT metrics, BLEU and NIST, when comparing two text reuse techniques.

1. Baseline retrieve-only system
2. Textual solution generation with CR2N

The average precision is also measured in addition to the two MT metrics using single and multiple reference solutions. The evaluation results for average precision, BLEU and NIST are shown in Tables 3A, B & C respectively. It can be seen across all three tables that the use of multiple reference solutions for text (retrieved or CR2N generated) evaluation always gives better results than using a single reference solution. Close examination of the 34 test cases suggests that these improvements are intuitive and better aligned with human judgement. This is because multiple references reduce the effect of variability in the domain vocabulary on our evaluation metrics thereby giving higher values that correlate better with human judgements. This also aligns with the reason why qualitative text evaluation typically involves the use of multiple human experts to reduce bias to a certain style of writing. We therefore suggest that multiple reference solutions (when available) should be utilized for TCBR evaluation but they can also be learnt introspectively from the casebase as explained in Section 5.1.

The result in Tables 3A & B also show that the precision scores are identical to BLEU when $N = 1$; this means that the length of most retrieved or CR2N generated solution texts were identical to one of the references implying that the brevity penalty has no effect. The brevity penalty is the only thing that differentiates precision from BLEU when $N = 1$. Therefore, the average BLEU score is expected to be less than precision's if it has an effect. This effect is illustrated in figure 3 when a single reference is used; precision is 0.67 while BLEU score is 0.6042 due to a brevity penalty of 0.719. The fact that the brevity penalty has no effect is generally true for TCBR since generated textual solutions are obtained from reference solutions to similar problems unlike MT where generated text can be shorter.

We use $k = 9$ for the CR2N after conducting an empirical study on the neighbourhood size. As shown in the Table 3, average retrieval and CR2N results are generally comparable across all 3 metrics; precision, BLEU, and NIST. Tests of statistical significance also showed no significance between each pair of retrieval/CR2N results ($p = 0.7107 > 0.05$ at 95% confidence). This shows that the CR2N has no considerable improvement over retrieval for the 34 test cases with multiple solutions used in our experiments. This can be explained by the fact that most of the retrieved solution texts (description of the action taken) were sufficient to assist a health personnel to solve the test queries (incident descriptions) when checked manually. Over 80% (28 out of 34) of the retrieved solution texts can also be reused verbatim during documentation of incidents with very little modifications. It is important to emphasize here that CR2N captures this since it is not worse than retrieval's results according to the three metrics. Nevertheless, averages are not able to show certain patterns if the difference in average between two result sets is small but the data is skewed with a comparatively large standard deviation (SD).

Further investigation revealed that the standard deviation of the individual 34 results were large as compared to the average; for instance, $SD = 0.46$ against

Table 3. Evaluation of textual solution generation quality in H&S incident reporting (A)Average precision (B) Average BLEU scores (C) Average NIST scores

(A)

Average Precision	$N = 1$		$N = 2$	
	Single Ref	**Multiple Ref**	**Single Ref**	**Multiple Ref**
Retrieval	0.28595	0.52124	0.14706	0.35294
CR2N ($k = 9$)	0.29160	0.53072	0.14706	0.35294

(B)

Average BLEU	$N = 1$		$N = 2$	
	Single Ref	**Multiple Ref**	**Single Ref**	**Multiple Ref**
Retrieval	0.28595	0.52124	0.15161	0.40167
CR2N ($k = 9$)	0.29160	0.53072	0.15161	0.40167

(C)

Average NIST	$N = 1$		$N = 2$	
	Single Ref	**Multiple Ref**	**Single Ref**	**Multiple Ref**
Retrieval	0.43575	1.31441	0.43575	1.34382
CR2N ($k = 9$)	0.44511	1.34139	0.44511	1.37081

Table 4. Clusters of precision results indicating where CR2N improves significantly over retrieval for the 34 cases with multiple reference solutions

Precision ($N = 1$, multiple Ref)	**Number of cases**	**Average Retrieval**	**Average CR2N**
$Score = 0$	6	0	0
$0 < Score < 1$	12	0.1435	0.1704
$Score = 1$	16	1	1

average precision = 0.52 for the retrieval results with multiple references. The same phenomenon applies to the results in Table 3 where N=1 for the three evaluation metrics and N=2 for NIST. The SD for results from the use of single references was generally greater than their averages. We discovered that the results where CR2N slightly improves over retrieval formed three natural clusters: $score$=0, $0 < score < 1$ and $score$= 1 as shown in Table 4 and Figure 5. The 6 cases with zero retrieval scores (cluster 1 in Figure 5) cannot be improved since it means that none of the retrieved keywords matches the query's reference solutions. The CR2N aptly identifies this by discarding all of these keywords during it text generation process. However, this cannot be captured by the precision measure as well as the MT metrics since they do not take true negatives into account. CR2N also uses all keywords in its generated text for the 16 cases where retrieval precision is one (cluster 3). Importantly, it is able to identify when all keywords in the retrieved solution text should be included in its generated text solution. The CR2N generated text outperforms retrieval for the 12 middle cases with retrieval scores between 0 and 1 (cluster 2) and this is significant at 95%

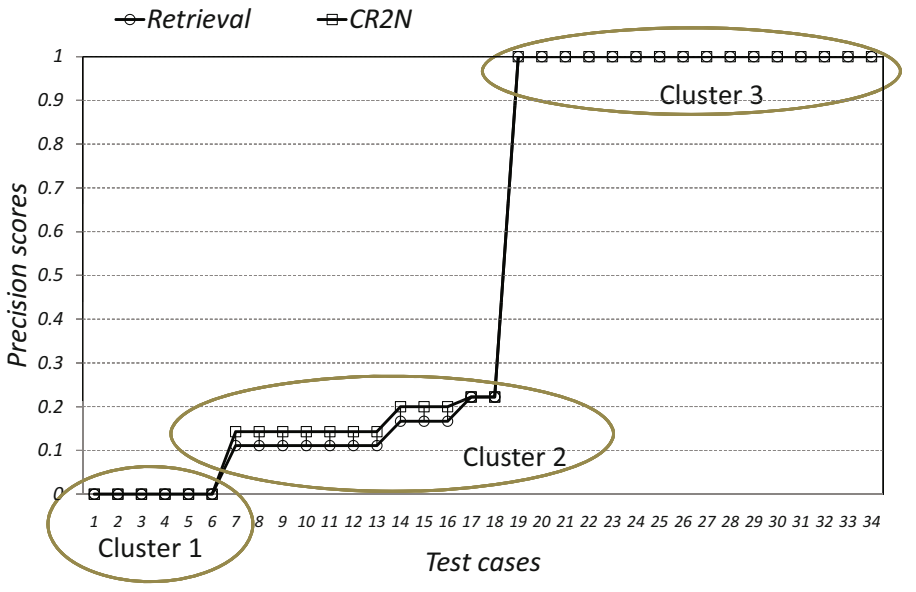


Fig. 5. Graph of precision results for the 34 test cases with multiple references

($p = 0.045 < 0.05$). A similar trend is observed for the BLEU and NIST results. Here, precision and the MT metrics are therefore only able to show improvements in retrieval when the retrieval scores are greater than zero.

6 Conclusion

The use of MT evaluation metrics to evaluate quality of generated textual solutions for TCBR is the main contribution of this paper. Two MT metrics, BLEU and NIST are adapted for TCBR evaluation with multiple reference solutions. We also propose a novel introspective method to generate multiple references when they do not naturally occur in a domain. Multiple references reduce the effect of different writing styles or variations in word choice on text evaluation. They therefore give more reliable and accurate results that correlate better with human judgements. Experimental results on a health and safety incidents dataset gave better results that were closer to human judgements with multiple reference solutions as opposed to the use of single references. We intend to carry out an extensive user evaluation to quantify the correlation of these MT metrics with human judgements for this dataset.

We also discovered that parameters like brevity penalty are not very important for TCBR because the generated texts are usually not significantly different from the reference solutions in length. We intend to verify this further by applying the MT metrics to other TCBR domains where multiple references are available or can be created introspectively.

Acknowledgements. This research work is funded by the Northern Research Partnership (NRP) and the UK-India Education and Research Initiative (UKIERI).

References

1. Adeyanju, I., Wiratunga, N., Lothian, R., Sripada, S., Craw, S.: Solution reuse for textual cases. In: Petridis, M. (ed.) Proceeding of 13th UK Workshop on Case-Based Reasoning, pp. 54–62. CMS Press, University of Greenwich (2008)
2. Adeyanju, I., Wiratunga, N., Lothian, R., Sripada, S., Lamontagne, L.: Case retrieval reuse net (CR2N): Architecture for reuse of textual solutions. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 14–28. Springer, Heidelberg (2009)
3. Baeza-Yates, R., Ribeiro-Neto, B., Bertino, E., Brown, E., Catania, B., Faloutsos, C., Ferrari, E., Fox, E., Hearst, M., Navarro, G., Rasmussen, E., Sornil, O., Ziviani, N.: Modern Information Retrieval. Addison-Wesley, Reading (1999)
4. Belz, A.: Statistical generation: Three methods compared and evaluated. In: Proceedings of the 10th European Workshop on Natural Language Generation (ENLG 2005), pp. 15–23 (2005)
5. Belz, A., Reiter, E.: Comparing automatic and human evaluation in NLG. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006), pp. 313–320 (2006)
6. Brüninghaus, S., Ashley, K.D.: Evaluation of textual CBR approaches. In: Proceedings of the AAAI 1998 Workshop on Textual Case-Based Reasoning, pp. 30–34. AAAI Press, Menlo Park (1998)
7. Brüninghaus, S., Ashley, K.D.: Reasoning with textual cases. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 137–151. Springer, Heidelberg (2005)
8. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A framework and graphical development environment for robust NLP tools and applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics, ACL 2002 (2002)
9. Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurence statistics. In: Proceedings of the 2nd International Conference on Human Language Technology, pp. 138–145. Morgan Kaufmann, San Francisco (2002)
10. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998), http://wordnet.princeton.edu
11. Hovy, E.H.: Toward finely differentiated evaluation metrics for machine translation. In: Proceedings of the Eagles Workshop on Standards and Evaluation (1999)
12. Lamontagne, L., Langlais, P., Lapalme, G.: Using statistical models for the retrieval of fully-textual cases. In: Rusell, I., Haller, S. (eds.) Proceedings of FLAIRS 2003, pp. 124–128. AAAI Press, Menlo Park (2003)
13. Lamontagne, L., Lapalme, G.: Textual reuse for email response. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 234–246. Springer, Heidelberg (2004)
14. Lenz, M.: Textual CBR and information retrieval - a comparison. In: Gierl, L., Lenz, M. (eds.) Proceedings of the Sixth German Workshop on Case-Based Reasoning (1998)
15. Lenz, M., Burkhard, H.D.: Case retrieval nets: Basic ideas and extensions. In: Görz, G., Hölldobler, S. (eds.) KI 1996. LNCS, vol. 1137, pp. 227–239. Springer, Heidelberg (1996)

16. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics- Doklady, Cynernetics and Control theory 10(8), 707–710 (1966)
17. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: A method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 311–318 (2002)
18. Sripada, S., Reiter, E., Hawizy, L.: Evaluation of an NLG system using post-edit data: Lessons learnt. In: Proceedings of European Natural Language Generation Workshop, pp. 133–139 (2005)
19. Weber, R.O., Ashley, K.D., Bruninghaus, S.: Textual case-based reasoning. Knowledge Engineering Review 20(3), 255–260 (2006)
20. White, J., Connell, T.: The ARPA MT evaluation methodologies: evolution, lessons and future approaches. In: Proceedings of the First Conference of the Association for Machine Translation in the Americas, pp. 193–205 (1994)

Intelligent Data Interpretation and Case Base Exploration through Temporal Abstractions

Alessio Bottrighi, Giorgio Leonardi, Stefania Montani, Luigi Portinale, and Paolo Terenziani

Dipartimento di Informatica, Università del Piemonte Orientale, Alessandria, Italy

Abstract. Interpreting time series of measurements and exploring a repository of cases with time series data looking for similarities, are non-trivial, but very important tasks.

Classical methodological solutions proposed to deal with (some of) these goals, typically based on mathematical techniques, are characterized by strong limitations, such as unclear or incorrect retrieval results and reduced interactivity and flexibility.

In this paper, we describe a novel case base exploration and retrieval architecture, which supports time series summarization and interpretation by means of Temporal Abstractions, and in which multi-level abstraction mechanisms and proper indexing techniques are provided, in order to grant *expressiveness* in issuing queries, as well as *efficiency* and *flexibility* in answering queries themselves.

Relying on a set of concrete examples, taken from the haemodialysis domain, we illustrate the system facilities, and we demonstrate the advantages of relying on this methodology, with respect to more classical mathematical ones.

1 Introduction

Analysing and interpreting long and complex **time series** of measurements is a key requirement in several real world applications, in which the pattern of feature changes is critical for decision making. This consideration strongly applies to the medical domain [16], especially in those situations in which a continuous monitoring of the patient's health indicators is required - such as in intensive care units [26] or in chronic diseases [3,23]. As a matter of fact, in these applications several process features are naturally collected in the form of time series, either automatically generated and stored by the monitoring instruments, or obtained by listing single values extracted from temporally consecutive visits/situations (as e.g. the series of glycated haemoglobin values, measured on a diabetic patient once every two months).

Interpreting time series features on screen or on paper can be tedious and prone to errors. In a medical setting, physicians may be asked to recognize *small or rare irregularities* in the series itself, or to identify *partial similarities* with previous situations: for instance, it could be important to highlight the past occurence of a similar trend behaviour in time in a given feature (referring to

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 36–50, 2010.

the same patient, or to a different one), independently of its actual values. Such identification may be extremely relevant for patient care, but may also require a significant amount of expertise in the specific field [25]. Similar consideration can be applied to any other application field, having time series interpretation as a core business (e.g. financial analysis, wheather forecasting, etc...). In all these domains, an automated decision support and data interpretation strategy is therefore strongly desirable.

Case-based Reasoning (CBR) [1] has recently being recognized as a valuable decision support methodology in time dependent applications (see e.g. [22], and section 4), provided that proper case representation and retrieval strategies are implemented. In particular, most of the approaches proposed in the literature to this end are based on the common premise of dimensionality reduction; this allows one to reduce memory storage and to simplify time series representation, still capturing the most important characteristics of the time series itself. Dimensionality is typically reduced by means of a *mathematical transform*, able to preserve the distance between two time series (or to underestimate it). Widely used transforms are the Discrete Fourier Transform (DFT) [2], the Discrete Wavelet Transform (DWT) [7] and the Piecewise Constant Approximation (PCA)[15]. Once a series is transformed, retrieval then works in the resulting time series space.

However, mathematical transforms have several limitations. First, they can be computationally complex, and often require additional pre-processing steps, such as mean averaging and zero padding (see e.g. [25] for details). Additionally, they work well with series with relatively simple dynamics, but they can fail to characterize more complex patterns [8]. Moreover, in many cases, they operate in a black-box fashion with respect to the end users, who just input the query and collect the retrieved cases, but usually do not see (and might not understand the meaning of) the transformed time series themselves; furthermore, users are usually unable, in such approaches, to interact with the system, during the retrieval process. Finally, mathematical transforms force the user to issue a very precise query, in which both trend and value information must be provided; indeed, retrieval is usually based on standard Euclidean distance among the transformed time series points. This requirement may be hard for an end user, as it is actually the case for a physician. Moreover, such methods may incorrectly answer a query in which e.g. only trend similarity is looked for, or get lost in time series details (e.g. small oscillations), thus not retrieving cases with a basically similar behaviour.

Our research group has recently proposed [21] to exploit a different technique for dimensionality reduction and flexible retrieval, namely **Temporal Abstractions** (TA) [30,4,27,20]. TA is an Artificial Intelligence (AI) methodology able to solve a *data interpretation task* [30], the goal of which is to derive high level concepts from time stamped data. Operatively, the principle of *basic* TA methods is to move from a *point-based* to an *interval-based* representation of the data [4], where the input points (*events*) are the elements of the time series, and the output intervals (*episodes*) aggregate adjacent events sharing a common behavior, persistent over time. Basic abstractions can be further subdivided into *state*

TA and *trend* TA. *State* TA are used to extract episodes associated with *qualitative levels* of the monitored feature, e.g. low, normal, high values; *trend* TA are exploited to detect specific *patterns*, such as increase, decrease or stationary behaviour, from the time series.

Through TA, huge amounts of temporal information can then be effectively mapped to a compact representation, that not only *summarizes* the original longitudinal data, but also highlights meaningful behaviours in the data themselves, which can be *interpreted* by end users as well (in an easier way if compared to mathematical methods outputs). TA-based dimensionality reduction appears to be well suited for several application domains, and in particular for medical ones (see also section 4).

In this paper, we describe the functionalities of an intelligent case base exploration and retrieval system, which implements the techniques previously described in [21], to support data interpretation through TA.

More precisely, our system allows for *multi-level abstractions*, according to two *dimensions*, namely a taxonomy of (trend or state) symbols, and a variety of time granularities. This functionality provides **(1)** great **expressiveness** in issuing queries, which can be defined at any level of detail in both dimensions. Moreover, we provide end users with **(2)** a significant degree of **flexibility** in the retrieval process, since they are allowed to interact with the system, and progressively reduce/enlarge the retrieval set, depending on their current needs. Finally, we grant for **(3)** computational **efficiency**, since query answering takes advantage of *multi-dimensional orthogonal index structures* for focusing and early pruning.

In the next sections, by means of a set of concrete examples taken from the haemodialysis domain, we illustrate such facilities, and we demonstrate the advantages of relying on this methodology, with respect to more classical mathematical ones.

The paper is organized as follows. Section 2 describes the tool functionalities. Section 3 provides some experimental results in the haemodyalisis domain. Section 4 compares our tool with related works, and section 5 addresses our concluding remarks.

2 System Functionalities

In this section, we describe the system functionalities introduced in section 1.

Our tool is composed by two main modules:

- a GUI module, which is used to interact with the system;
- a TA ENGINE module, which is responsible for building the orthogonal index structures, and for navigating them in order to support efficient and flexible retrieval.

In particular, our system allows for *multi-level abstractions*. We can abstract time series data at different levels of detail, and users can issue more general as well as more specific queries, according to two *dimensions*, which are formalized by means of two taxonomies: (i) a taxonomy of (trend or state) *TA symbols*, and (ii) a taxonomy of *time granularities*.

Since our tool has been designed and implemented in order to be domain independent and to obtain the maximal generality, all the information which regards the description of the domain application must be taken as an input by the TA ENGINE. This holds for the two taxonomies; obviously, depending on the application domain, the definition of symbols or granules can change and the taxonomies can become wider or higher. In the rest of the paper, we exemplify it by describing a specific application to the haemodialysis domain.

The TA symbol taxonomy is organized as a conventional *is-a* taxonomy. For our experiments in the haemodialysis domain (see section 3), we have defined the taxonomy of trend symbols shown in figure 1, to which we will refer henceforth. The taxonomy is organized as follows: the symbol *Any* is specialized into D (decrease), S (stationary), I (increase), and D is further specialized into D_S (strong decrease) and D_W (weak decrease), according to the slope; I is further specialized into I_W (weak increase) and I_S (strong increase). Moreover we have introduced two special symbols, i.e. U (Unknown), M (Missing). U means that no TA symbols capture the signal behaviour (e.g. due to noise problems), while M means that data are missing (e.g. because the available monitoring instruments are not returning any data).

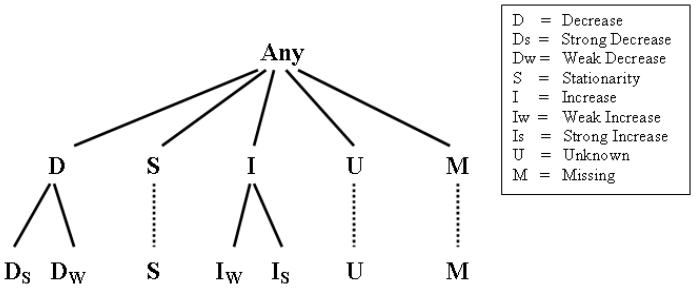


Fig. 1. The symbol taxonomy used in the experiments

The time granularity taxonomy allows one to describe the episodes at increasingly more abstract levels of temporal aggregation, starting from the bottom level; in our example domain, at the bottom level we find 5 minutes long granules, while at top level we aggregate episodes up to 4 hours long granules (see figure 2).

In order to abstract along the temporal dimension, a special function, called the *up* function, is needed in order to scale up from one level to the coarser one in the taxonomy. Again, since abstracting from one granularity to a coarser one is a highly domain-dependent procedure, the definition of the *up* function should be provided by domain experts and is required as an input to the TA ENGINE. Analogously, experts need to provide a proper *distance* function, to compute the distance between two cases. We allow the maximal generality in the *up* and *distance* function definitions, provided that they verify some very general constraints in order to avoid ambiguities (see [21] for the details).

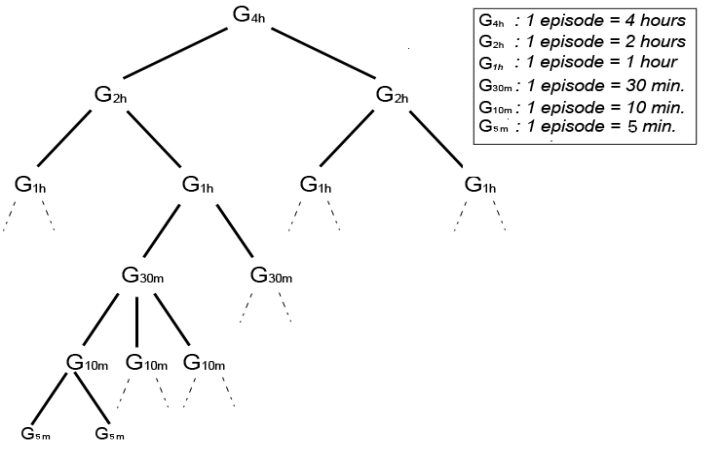


Fig. 2. The time granularities taxonomy used in the experiments

Cases are stored in a specific database. Their features, originally in the form of time series, have been preprocessed through TA to transform them in sequences of abstractions at the lowest level of detail according to both taxonomies [1].

To perform a query, the TA ENGINE module takes advantage of a forest of multi-dimensional indexing trees which is built off-line; the engine also relies on the time and symbols taxonomies and on the *up* function, for the generalization task. Each tree indexes a portion of the case base, accordingly to the trend symbol(s) specified in its root, i.e. at the highest level of granularity in time and in the taxonomy of symbols. For instance, the tree in figure 3 indexes all the cases whose feature shows a global increasing trend. Each node can expand in two dimensions: a leading dimension (e.g. time) and a secondary one (e.g. symbols). In the leading dimension, the children of the current node will index cases compatible with sequences of trends obtained at the same symbol level, but at a lower time granularity; moving in the secondary dimension, the tree will index nodes partitioning the space at the same time granularity, but at a lower symbol level. The root node I in figure 3, for instance, is refined along the time dimension from the 4 hours granularity to the 2 hours granularity, so that the nodes *II*, *IS* and *SI* stem from it, provided that $up(I, I) = I$, $up(I, S) = I$ and $up(S, I) = I$. Moreover the root node I is refined in the nodes I_W and I_S according the symbol taxonomy. Indexes can be incomplete and can grow on demand, on the basis of the available data and of the most frequently issued queries. The domain expert can also define which are the leading and the secondary dimensions before generating the tree. For further details, see [21].

Although the use of a symbol taxonomy and/or of a temporal granularity taxonomy has been already advocated in other works (e.g. in a data warehouse context, see [34]), to the best of our knowledge our system is the first approach attempting to fully exploit the advantages of taxonomical knowledge in order

[1] In the following, for the sake of clarity, we will focus on a single time series feature.

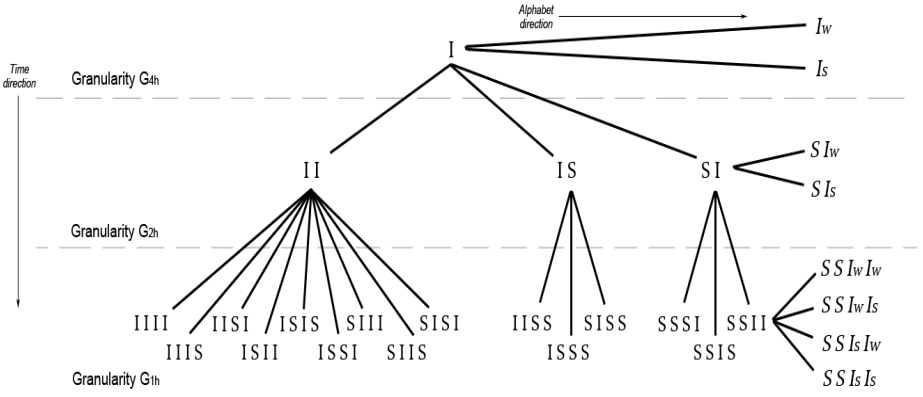


Fig. 3. Part of a multi-level orthogonal index structure

to support (i) case navigation and (ii) flexible case retrieval. For what concerns navigation (i.e. case exploration), our system allows to enter a proper index in the forest and to visit it, visualizing children (in both dimensions), siblings and father of every node. In this way the users can easily analyze cases, whose features can be abstracted as identical or very similar TA strings. The retrieval task asks the user to provide a sequence of TA symbols as a query, specified at any level of detail both in time and in the symbol taxonomy. The system will then elaborate the query and retrieve all the cases compatible with the query. If results are not satisfactory, the user can interact with the system and navigate the multi-dimensional index(es), generalizing or restricting the initial query in either the time or symbol dimensions, in order to find a larger or narrower set of cases, depending on her/his current needs.

Details of query answering are illustrated in section 3 by means of specific examples in the haemodyalisis domain (for other additional technical details about the framework see also [21]).

3 Experimental Results

We executed our experiments in the haemodialysis domain, using a dataset containing data belonging to 37 real patients, for a total of 1475 haemodialysis sessions (i.e. cases), collected at the Vigevano hospital, Italy. In particular, we focused our attention on the analysis of the blood pressure feature. In this section, we report on the difference between the results obtained with a classical method based on a mathematical transform (i.e. DFT), and the ones obtained using our TA-based flexible retrieval.

Diastolic pressure trends are an important indicator of the haemodialysis session performance. The desired overall trend for this feature is a slight, constant decrease from the start to the end of the session, due to the reduction of metabolites and water from blood. Clinical studies state that intradialytic increase in

diastolic pressure (DP hanceforth) can complicate the management of hypertension in haemodialysis patients. Furthermore, DP increasing trend is associated with greater 2-year mortality in these patients [11]. It is therefore important to study this trend, in order to verify which patients are affected by intradialytic increases in DP, and in what haemodialysis sessions this problem arises.

To afford this task, we compared the approach described in this paper with a more classical approach we implemented in the past within the system RHENE [23]. That approach was based on DFT for dimensionality reduction, and on spatial indexing (through TV-trees) for further improving retrieval performances.

As a first consideration, it is worth highlighting the greater *expressiveness* of the TA-based approach, in which we can provide the query as a sequence of symbols describing the abstractions to be searched for, at any level of detail in both dimensions. On the other hand, mathematical methods require to provide a whole time series as a query, built to match the characteristics (e.g. trend) we are interested in. This can be a very complex operation for an unexperienced user. In RHENE we mitigated this difficulty by allowing the user to provide an existing case, whose shape basically showed the desired characteristics, as a query. As an example, we will consider the query case belonging to patient 49, in session 177, taken from our case library. The shape of this signal represents the clinical problem we are going to investigate: in particular, DP basically mantains a stable trend for the first part of the dialysis session, then it slightly increases until the end of the session. The plot is shown in figure 4.

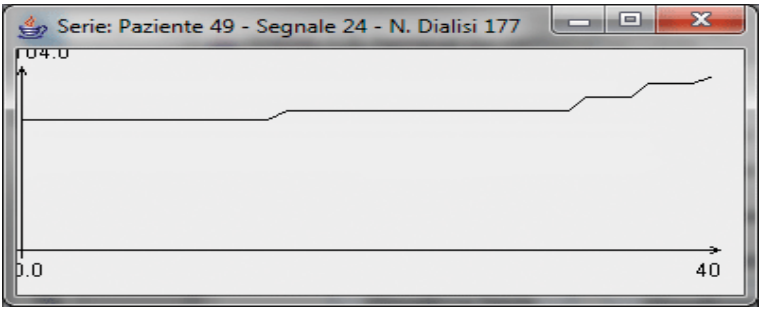


Fig. 4. The query case

The nearest 8 results obtained through DFT are shown in figure 5.

Looking at figure 5, we can see that not all the retrieved cases are compatible with the shape we were searching for. In particular, we have obtained 2 results showing the correct trends (patient 43, session 313, and patient 46, session 272); 3 results showing an increasing trend, but with some instability (patient 39, session 276; patient 61, session 12, and patient 39, session 275), and a result which is almost stable, but with frequent episodes of increasing and decreasing trends (patient 2, session 292). One result even shows an overall decreasing trend (patient 31, session 434).

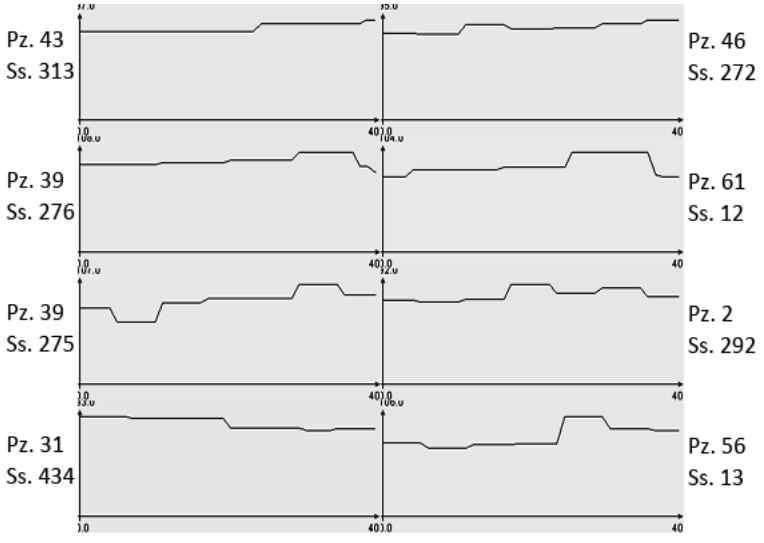


Fig. 5. The best 8 results using DFT

It can be noticed that the results whose trends are not looking similar to the query, are compatible with the retrieval strategy that has been adopted: using the Euclidean distance over the series points, the system retrieves cases that minimize the overall alignment around the points of the query. Considering this point of view, the result showing an overall decreasing trend is correctly retrieved, because its plot is not far from the plot of the query, comparing their values one by one. Figure 6 shows this situation.

Adopting the approach introduced in this paper, on the other hand, the query can be simply issued as a string of trend symbols. The DP problems we are looking for (taking place in the case belonging to patient 47, session 177) can be expressed as the following string: $SSI_W SSI_W I_W I_W$, if choosing a time granularity of 30 minutes. In order to answer this query, the system first progressively

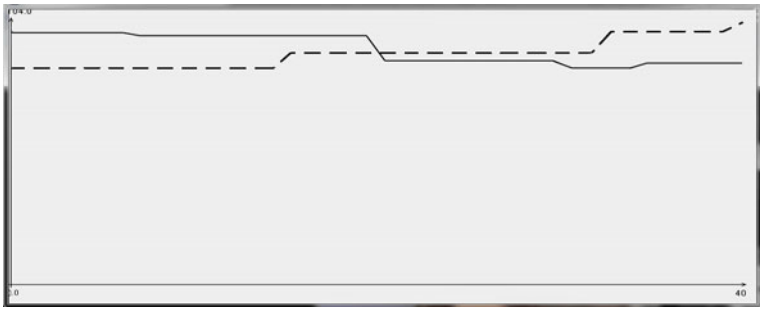


Fig. 6. Comparison between the query case (dotted line) and a result with a decreasing trend

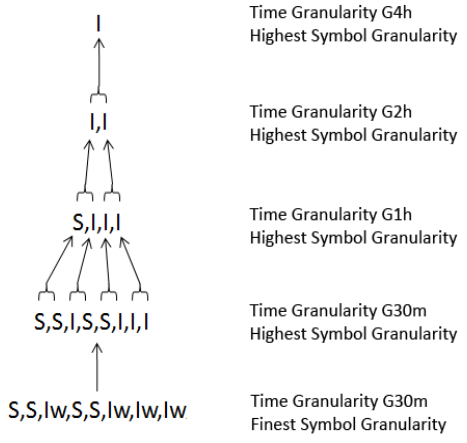


Fig. 7. The generalization steps for the example query

generalizes it in the symbol taxonomy dimension, while keeping time granularity fixed. Then, it generalizes the query in the time dimension as well. The generalization steps are shown in figure 7, and allow to select the index in figure 3 for optimizing retrieval. Following the generalization steps backwards, the user can enter the index from its root, and descend along it, until the node which fits the original query time granularity is reached. If an orthogonal index stems from this node, the user can descend along it, always following the query generalization steps backwards, until the same level of detail in the symbol taxonomy is reached, as in the original query, or at any level desired, depending on the results obtained from time to time.

In the example, 3 cases could be retrieved from node $SSI_W SSI_W I_W I_W$, which exactly matched the query. In order to enlarge the retrieval set, the user can *interactively* relax the query constraints, being guided by the index structure. In this case, we could generalize the initial requirement to a higher level of detail both in the taxonomy of symbols, and in time granularity, thus reaching node $SIII$ at 1 hour granularity level. All the cases indexed by the selected node still have a zero distance from the (relaxed) query; this is because the *up* function preserves the distance and node $SIII$ is the abstraction at 1 hour-granularity of the 30 minutes-granularity query (see details in [21]) . However, the user can also visualize the cases indexed by the siblings of this node, grouped into different sets, depending on the distance from the query itself. The distance from the query to a sibling of this node, for example the node $SIII$, is computed using a suitable distance measure between abstraction sequences as illustrated in [21]. Figure 8 shows the 19 nearest cases retrieved, indexed by node $SIII$ (at distance=0, group A in Figure 8) and by its nearest sibling $SISI$ (at distance=1.5, group B). Please note that, among the cases retrieved at distance=0, we find the case used as the query of the DFT method (patient 49, session 177).

For comparison with the DFT method, we also provide the plots of 8 out of the best retrieved cases (3 at distance=0, and 5 at distance=1.5 from the query)

A) Results from the selected node: S,I,I,I

Patient	Session	TA string at ground level
49	177	S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,Iw,S,S,S,Iw,S,S,S,Iw
2	284	Dw,S,Dw,Iw,S,S,S,S,S,Iw,S,S,Dw,S,Iw,Iw,S,S,S,S,Iw,S,M,M
41	333	S,S,S,S,S,S,S,S,Is,S,S,S,S,S,S,S,S,S,Is,S,S,S,S,Is,S
46	264	S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,Iw,S,S,S,S,S,Iw,S,S,S,M,M

B) Results from the sibling node: S,I,S,I – distance from query: 1.5

Patient	Session	TA string at ground level
2	280	S,S,Is,Ds,S,Dw,S,S,Iw,S,S,Iw,S,S,S,Iw,S,S,S,M,M,M,M,M
27	424	S,S,S,S,S,Is,S,S,S,Iw,S,S,S,S,Iw,S,S,S,S,S,S,S,Is,M
27	442	S,S,S,S,S,S,S,S,S,S,S,Is,S,S,S,S,S,S,Iw,S,S,S,S,M,M
30	451	S,S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,Iw,S,S,S,S,S,S,S,S,S
36	363	Iw,S,S,S,Ds,S,S,S,S,S,Iw,S,Iw,S,S,S,S,S,S,S,M,M,M,M
43	313	S,S,S,S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,Iw,S,M,M,M,M
46	229	S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,S,S,Is,S,S,S,S,M,M
46	234	S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,S,S,Iw,S,S,S,M,M,M,M,M
46	250	S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,M
48	205	S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,S,S,S,Iw,S,M,M,M,M,M
48	230	S,S,S,S,S,S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,Iw,S,S,S,S,M,M
48	241	S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,S,S,Iw,S,S,S,S,M,M,M,M
53	34	S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,S,S,S,S,Iw,S,S,S,S,M,M
53	55	S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,Iw,S,S,S,S,S,M,M,M,M,M
57	21	S,S,S,S,S,S,S,S,Iw,S,S,S,S,S,S,S,Is,S,S,S,S,S,S,S,S,M

Fig. 8. Results using TA-based flexible retrieval

in figure 9. All cases match the required trend. However, observe that only one case (patient 43, session 313) was also obtained by DFT, see figure 5.

From these experimental results, some observations can be drawn:

- considering the 19 best retrieved cases obtained through DFT (the number was chosen for comparison with the results in figure 8), we verified that only 1 case was retrieved by both approaches. This is reasonable, since the two methods work in a different way, giving a different semantics to the query. When adopting mathematical methods, we look for cases whose point-to-point distance from the query is globally minimized, which means that we are searching for the best overall alignment of the resulting case over the plot of the query. The main focus is on values, rather than on the shape. Using TA, instead, we focus on the shape, rather than on values. The choice of the best method to be adopted depends on the domain, and on the specific user needs. Interestingly, the integration of the two different methods in a single system would allow the user to explore the same data from different points of view. Integrating the TA-based approach in RHENE will be object of our future work;
- considering the dynamics of the actions performed to answer a query, it is clear that mathematical methods work in a black box fashion. The user selects the query case, sets the parameters of the query (e.g. the value of K for K-Nearest Neighbour, or a maximum distance for a range query), and then the system performs its retrieval task. Using our TA-based method, instead, the user can drive the query in an interactive fashion, taking control of the retrieval task by navigating the indexing structure, to find the proper level of generalization or refinement of the query. The system is able to help the user in navigating the structure and in selecting the proper navigation direction, providing him/her with quantitative and qualitative information about the cases indexed by sons and siblings of the currently visited node (i.e. the

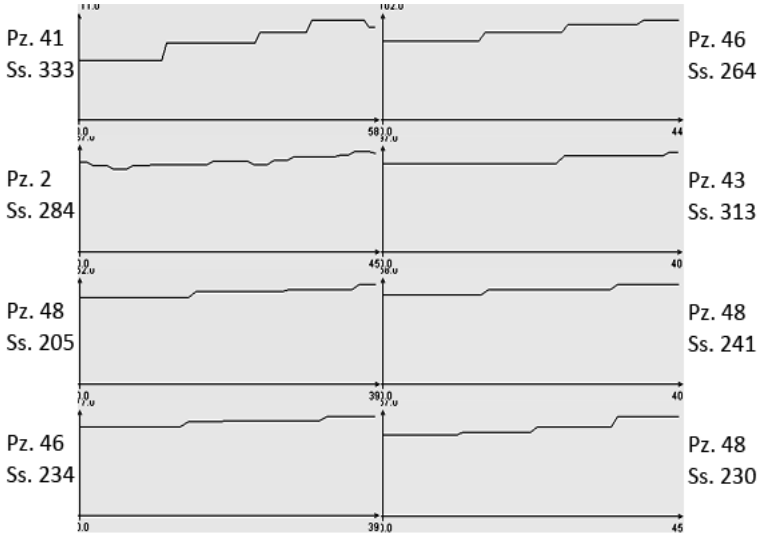


Fig. 9. Plots of the best results obtained using TA-based flexible retrieval

number of indexed cases, the sequence of abstractions representing the cases and the distance from the sequence of abstractions representing the node currently visited by the user).

– the results of the query performed using our TA-based method are easily interpretable compared to the results obtained by mathematical methods. For each retrieved case, we obtain both its distance from the query, as well as its representation as a sequence of abstractions. This sequence gives an immediate understanding of the shape of the result; therefore, in most cases it is not necessary to look at the original plot.

In conclusion, the semantics underlying the two retrieval methods are different; each of them can be more or less suitable for a given search need. However, the use of TA allows the user to perform a flexible and interactive retrieval, with easy-to-understand queries and results. Furthermore, when the focus is on the shape of time series, the use of TA provides better results (see figure 9).

Our TA-based method also proved to be computationally more efficient, in answering the query, than the mathematical method in these experiments. This is because the mathematical method requires to reduce the original data using the DFT transform, to perform the query exploiting the advantages of the indexing tree, and then to post-process the results, in order to filter out the so-called "false alarms" [2]. Even if these steps produce a computationally sub-linear retrieval, the computation to be performed usually requires non neglectible time. The use of TA in the proposed multi-dimensional indexing structure allows the query to be answered by executing the generalization steps from the input TA sequence (using the *up* function), and by using them to navigate the multi-dimensional indexing structure. These steps are very fast; the retrieval performed in our

experiments (using an Intel Core 2 Duo T9400 processor running at 2.53 GHz, equipped with 4 Gb of DDR2 ram) took, on average, only 41 milliseconds. On the other hand, the mathematical method required 3276 milliseconds. Since verification is in an early stage, further tests will be performed in the future, on databases of different sizes, to evaluate the quality of the results and the computational efficiency, also comparing this method with other retrieval techniques.

4 Discussion

CBR has recently been recognized as a valuable decision support technique in time-dependent applications. As a matter of fact, various CBR works dealing with cases with time series features have been recently published in different domains (e.g. robot control [28], process forecast [29], process supervision [9], pest management [6], prediction of faulty situations [13]). Applications in medicine have also been reported, relying on classical mathematical dimensionality reduction techniques, such as DFT [23] and DWT [24]. Moreover, general (e.g. logic-based) frameworks for case representation in time dependent domains have been proposed [22,12,18,5].

As regards TA, they have been extensively resorted to in the literature, especially in the medical field, from diabetes mellitus [31,3], to artificial ventilation of intensive care units patients [19](see also the survey in [33]). However, they have been typically adopted with the aim to solve a data interpretation task, and never as a flexible navigation/retrieval support facility. The goal of our proposal is to try to fill this gap, by supporting data interpretation, as well as case exploration and retrieval; this idea appears to be significantly innovative in the recent literature panorama.

As a final consideration, observe that TA are not the only methodology for reducing dimensionality by transforming a time series into a sequence of symbols. Actually a wide number of symbolic representations of time series have been introduced in the past decades (see [8] for a survey). However, some of them require a extremely specific and hard to obtain domain knowledge [14], since they a priori partition the signal into intervals, naturally provided by the underlying system dynamics, which divide the overall time period into distinct physical phases (e.g. respiration cycles in [10]). Many other approaches to symbolizations are weakened by other relevant issues, like e.g. the fact that the conversion to symbolic representation requires to have access to all the data since the beginning, thus making it not exploitable in a context of data streaming. Rather interestingly, Lin [17] has introduced an alternative to TA, capable to deal with such issues, in which intervals are first obtained through PCA [15], and subsequently labeled with proper symbols. In particular this contribution allows distance measures to be defined on the symbolic approach that lower bound the corresponding distance measures defined on the original data. Such a feature permits to run some well known data mining algorithms on the transformed data, obtaining identical results with respect to operating on the original data, while gaining in

efficiency. Despite these advantages, the approach in [17] is not as simple as TA, which allow a very clear interpretation of the qualitative description of the data provided by the abstraction process itself. As a matter of fact, such a description is often closer to the language of end users (e.g. of clinicians [32]), and easily adapts to domains where data values that are considered as normal at one time, or in a given context, may become dangerously abnormal in a different situation (e.g. in medicine, due to disease progression or to treatments obsolescence). And, of course, the ease and flexibility at which knowledge can be managed and understood by experts is an aspect that impacts upon the suitability and the usefulness of decision support systems in practice.

5 Conclusions

Time series interpretation and retrieval is a critical issue in all domains in which the observed phenomenon dynamics have to be dealt with, like in many medical applications. In this paper, we have described an architecture for the exploration of cases containing time-varying features, which allows time series data interpretation and summarization by means of Temporal Abstractions, a well known data interpretation AI technique.

In our system, we allow end users to issue queries at any level of detail, according to two abstraction dimensions, thus granting a significant degree of *expressiveness*. Abstracted data can also be easily navigated, and similar cases can be retrieved in a *quick*, *flexible* and *interactive* way, relying on proper orthogonal index structures, whose exploitation has been described by means of a concrete case study.

In the future, we plan to extensively test our tool, by analysing its retrieval results, and by evaluating its computational performances and its scalability on large case bases, compared to other, more classical techniques.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations and systems approaches. AI Communications 7, 39–59 (1994)
2. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)
3. Bellazzi, R., Larizza, C., Magni, P., Montani, S., Stefanelli, M.: Intelligent analysis of clinical time series: an application in the diabetes mellitus domain. Artificial Intelligence in Medicine 20, 37–57 (2000)
4. Bellazzi, R., Larizza, C., Riva, A.: Temporal abstractions for interpreting diabetic patients monitoring data. Intelligent Data Analysis 2, 97–122 (1998)
5. Bichindaritz, I., Conlon, E.: Temporal knowledge representation and organization for case-based reasoning. In: Proc. TIME 1996, pp. 152–159. IEEE Computer Society Press, Washington (1996)
6. Branting, L., Hastings, J.: An empirical evaluation of model-based case matching and adaptation. In: Proc. Workshop on Case-Based Reasoning, AAAI 1994 (1994)

7. Chan, K., Fu, A.: Efficient time series matching by wavelets. In: Proc. ICDE 1999, pp. 126–133. IEEE Computer Society Press, Washington (1999)
8. Daw, C., Finney, C., Tracy, E.: Symbolic analysis of experimental data. Review of Scientific Instruments 2002-07-22 (2001)
9. Fuch, B., Mille, A., Chiron, B.: Operator decision aiding by adaptation of supervision strategies. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS (LNAI), vol. 1010, pp. 23–32. Springer, Heidelberg (1995)
10. Funk, P., Xiong, N.: Extracting knowledge from sensor signals for case-based reasoning with longitudinal time series data. In: Perner, P. (ed.) Case-Based Reasoning in Signals and Images, pp. 247–284. Springer, Heidelberg (2008)
11. Inrig, J.K., Patel, U.D., Toto, R.D., Szczech, L.A.: Association of blood pressure increases during hemodialysis with 2-year mortality in incident hemodialysis patients: A secondary analysis of the dialysis morbidity and mortality wave 2 study. American Journal of Kidney Diseases 54(5), 881–890 (2009)
12. Jaczynski, M.: A framework for the management of past experiences with time-extended situations. In: Proc. ACM conference on Information and Knowledge Management (CIKM) 1997, pp. 32–38. ACM Press, New York (1997)
13. Jaere, M., Aamodt, A., Skalle, P.: Representing temporal knowledge for case-based prediction. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 174–188. Springer, Heidelberg (2002)
14. Kadar, S., Wang, J., Showalter, K.: Noise-supported travelling waves in sub-excitable media. Nature 391, 770–772 (1998)
15. Keogh, E.: Fast similarity search in the presence of longitudinal scaling in time series databases. In: Proc. Int. Conf. on Tools with Artificial Intelligence, pp. 578–584. IEEE Computer Society Press, Washington (1997)
16. Keravnou, E.: Modeling medical concepts as time objects. In: Wyatt, J.C., Stefanelli, M., Barahona, P. (eds.) AIME 1995. LNCS (LNAI), vol. 934, pp. 67–90. Springer, Heidelberg (1995)
17. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proc. of ACM-DMKD, San Diego (2003)
18. Ma, J., Knight, B.: A framework for historical case-based reasoning. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 246–260. Springer, Heidelberg (2003)
19. Miksch, S., Horn, W., Popow, C., Paky, F.: Utilizing temporal data abstractions for data validation and therapy planning for artificially ventilated newborn infants. Artificial Intelligence in Medicine 8, 543–576 (1996)
20. Montani, S., Bottrighi, A., Leonardi, G., Portinale, L.: A cbr-based, closed loop architecture for temporal abstractions configuration. Computational Intelligence 25(3), 235–249 (2009)
21. Montani, S., Bottrighi, A., Leonardi, G., Portinale, L., Terenziani, P.: Multi-level abstractions and multi-dimensional retrieval of cases with time series features. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 225–239. Springer, Heidelberg (2009)
22. Montani, S., Portinale, L.: Accounting for the temporal dimension in case-based retrieval: a framework for medical applications. Computational Intelligence 22, 208–223 (2006)
23. Montani, S., Portinale, L., Leonardi, G., Bellazzi, R., Bellazzi, R.: Case-based retrieval to support the treatment of end stage renal failure patients. Artificial Intelligence in Medicine 37, 31–42 (2006)

24. Nilsson, M., Funk, P., Olsson, E., vonScheele, B., Xiong, N.: Clinical decision-support for diagnosing stress-related disorders by applying psychophysiological medical knowledge to an instance-based learning system. Artificial Intelligence in Medicine 36, 159–176 (2006)
25. Nilsson, M., Funk, P., Xiong, N.: Clinical decision support by time series classification using wavelets. In: Chen, C., Filipe, J., Seruca, I., Cordeiro, J. (eds.) Proc. Seventh International Conference on Enterprise Information Systems (ICEIS 2005), pp. 169–175. INSTICC Press (2005)
26. Palma, J., Juarez, J., Campos, M., Marin, R.: A fuzzy approach to temporal model-based diagnosis for intensive care units. In: de Mantaras, R.L., Saitta, L. (eds.) Proc. European Conference on Artificial Intelligence (ECAI) 2004, pp. 868–872. IOS Press, Amsterdam (2004)
27. Portinale, L., Montani, S., Bottrighi, A., Leonardi, G., Juarez, J.: A case-based architecture for temporal abstraction configuration and processing. In: Proc. IEEE International Conference on Tools with Artificial Intelligent (ICTAI), pp. 667–674. IEEE Computer Society, Los Alamitos (2006)
28. Ram, A., Santamaria, J.: Continuous case-based reasoning. In: Proc. AAAI Case-Based Reasoning Workshop, pp. 86–93 (1993)
29. Rougegrez, S.: Similarity evaluation between observed behaviours for the prediction of processes. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) EWCBR 1993. LNCS (LNAI), vol. 837, pp. 155–166. Springer, Heidelberg (1994)
30. Shahar, Y.: A framework for knowledge-based temporal abstractions. Artificial Intelligence 90, 79–133 (1997)
31. Shahar, Y., Musen, M.: Knowledge-based temporal abstraction in clinical domains. Artificial Intelligence in Medicine 8, 267–298 (1996)
32. Stacey, M.: Knowledge based temporal abstractions within the neonatal intesive care domain. In: Proc. CSTE Innovation Conference, University of Western Sidney (2005)
33. Terenziani, P., German, E., Shahar, Y.: The temporal aspects of clinical guidelines. In: Teije, A.T., Miksch, S., Lucas, P. (eds.) Computer-based Medical Guidelines and Protocols: A Primer and Current Trends (2008)
34. Xia, B.: Similarity search in time series data sets. Tech. rep., School of Computer Science, Simon Fraser University (1997)

An Algorithm for Adapting Cases Represented in an Expressive Description Logic

Julien Cojan and Jean Lieber

UHP-Nancy 1 – LORIA (UMR 7503 CNRS-INPL-INRIA-Nancy 2-UHP)
BP 239, 54506 Vandœuvre-lès-Nancy, France
{Julien.Cojan,Jean.Lieber}@loria.fr

Abstract. This paper presents an algorithm of adaptation for a case-based reasoning system with cases and domain knowledge represented in the expressive description logic $\mathcal{ALC}$. The principle is to first pretend that the source case to be adapted solves the current target case. This may raise some contradictions with the specification of the target case and with the domain knowledge. The adaptation consists then in repairing these contradictions. This adaptation algorithm is based on an extension of the classical tableau method used for deductive inference in $\mathcal{ALC}$.

Keywords: adaptation, description logic, $\mathcal{ALC}$, tableau algorithm.

1 Introduction

Adaptation is a step of some case-based reasoning (CBR) systems that consists in modifying a source case in order to suit a new situation, the target case. An approach to adaptation consists in using a belief revision operator, i.e., an operator that modifies minimally a set of beliefs in order to be consistent with some actual knowledge [1]. The idea is to consider the belief "The source case solves the target case" and then to revise it with the constraints given by the target case and the domain knowledge. This has been studied for cases represented in propositional logic in [9]. Then, it has been studied in a more expressive formalism, including numerical constraints and after that extended to the combination of cases in this formalism [3].

In this paper, this approach to adaptation is studied for cases represented in an expressive description logic (DL), namely $\mathcal{ALC}$. The choice of DLs as formalisms for CBR can be motivated in several ways. First, they extend the classical attribute-value formalisms, often used in CBR (see, e.g., [8]) and they are similar to the formalism of memory organisation packets (MOPs) used in early CBR applications [10]. More generally, they are designed as trade-offs between expressibility and practical tractability. Second, they have a well-defined semantics and have been systematically investigated for several decades, now. Third, many efficient implementations are freely available, offering services that can be used for CBR systems, in particular for case retrieval and case base organisation.

The rest of the paper is organised as follows. Section 2 presents the DL $\mathcal{ALC}$, together with the tableau algorithm, at the basis of its deductive inferences for most current implementation. An example is presented in this section, for illustrating notions

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 51–65, 2010.

that are rather complex for a reader not familiar with DLs. This tableau algorithm is extended for performing an adaptation process, as shown in section 3. Section 4 discusses our contribution and relates it to other research on the use of DLs for CBR. Section 5 concludes the paper and presents some future work.

2 The Description Logic $\mathcal{ALC}$

Description logics [2] form a family of classical logics that are equivalent to decidable fragments of first-order logic (FOL). They have a growing importance in the field of knowledge representation. $\mathcal{ALC}$ is the simplest of *expressive DLs*, i.e., DLs extending propositional logic. The example presented in this section is about cooking, in the spirit of the computer cooking contest, but sticks to the adaptation of the ingredient list.

2.1 Syntax

Representation entities of $\mathcal{ALC}$ are concepts, roles, instances, and formulas.

A *concept*, intuitively, represents a subset of the interpretation domain. A concept is either an *atomic concept* (i.e., a concept name), or a conceptual expression of one of the forms $\top$, $\bot$, $\neg\mathtt{C}$, $\mathtt{C} \sqcap \mathtt{D}$, $\mathtt{C} \sqcup \mathtt{D}$, $\exists\mathtt{r}.\mathtt{C}$, and $\forall\mathtt{r}.\mathtt{C}$, where $\mathtt{C}$ and $\mathtt{D}$ are concepts (either atomic or not) and $\mathtt{r}$ is a role. A concept can be mapped into a FOL formula with one free variable x. For example, the concept

$$\mathtt{Tart} \sqcap \exists\mathtt{ing}.\mathtt{Apple} \sqcap \exists\mathtt{ing}.\mathtt{Pastry} \sqcap \forall\mathtt{ing}.\neg\mathtt{Cinnamon} \tag{1}$$

can be mapped to the first-order logic formula

$$\begin{aligned}\mathtt{Tart}(x) \wedge (\exists y, \mathtt{ing}(x,y) \wedge \mathtt{Apple}(y)) \wedge (\exists y, \mathtt{ing}(x,y) \wedge \mathtt{Pastry}(y))\\ \wedge (\forall y, \mathtt{ing}(x,y) \Rightarrow \neg\mathtt{Cinnamon}(y))\end{aligned}$$

A *role*, intuitively, represents a binary relation on the interpretation domain. Roles in $\mathcal{ALC}$ are atomic: i.e., role names. Their counterpart in FOL are binary predicates. The role appearing in (1) is `ing`.

An *instance*, intuitively, represents an element of the interpretation domain. Instances in $\mathcal{ALC}$ are atomic: i.e., instance names. Their counterpart in FOL are constants.

There are four types of formulas in $\mathcal{ALC}$ (followed by their meaning): (1) $\mathtt{C} \sqsubseteq \mathtt{D}$ ($\mathtt{C}$ is more specific than $\mathtt{D}$), (2) $\mathtt{C} \equiv \mathtt{D}$ ($\mathtt{C}$ and $\mathtt{D}$ are equivalent concepts), (3) $\mathtt{C}(\mathtt{a})$ ($\mathtt{a}$ is an instance of $\mathtt{C}$), and (4) $\mathtt{r}(\mathtt{a},\mathtt{b})$ ($\mathtt{r}$ relates $\mathtt{a}$ to $\mathtt{b}$), where $\mathtt{C}$ and $\mathtt{D}$ are concepts, $\mathtt{a}$ and $\mathtt{b}$ are instances, and $\mathtt{r}$ is a role. Formulas of types (1) and (2) are called *terminological formulas*. Formulas of types (3) and (4) are called *assertional formulas*, or *assertions*.

An $\mathcal{ALC}$ knowledge base $\mathtt{KB}$ is a set of $\mathcal{ALC}$ formulas. The *terminological box* (or *TBox*) of $\mathtt{KB}$ is the set of its terminological formulas. The *assertional box* (or *ABox*) of $\mathtt{KB}$ is the set of its assertions.

For example, the following TBox represents the domain knowledge ($\mathtt{DK}$) of our example (with the comments giving the meaning):

$$\begin{aligned}\mathtt{DK} = \{&\mathtt{Apple} \sqsubseteq \mathtt{PomeFruit}, && \textit{An apple is a pome fruit.}\\ &\mathtt{Pear} \sqsubseteq \mathtt{PomeFruit}, && \textit{A pear is a pome fruit.}\\ &\mathtt{PomeFruit} \sqsubseteq \mathtt{Apple} \sqcup \mathtt{Pear}\} && \textit{A pome fruit is either an apple or a pear.}\end{aligned} \tag{2}$$

Note that the last formula is a simplification: actually, there are other pome fruits than apples and pears.

In our running example, the only cases considered are the source and target cases. They are represented in the ABox:

$$\{\texttt{Source}(\sigma), \texttt{Target}(\theta)\} \tag{3}$$

$$\text{with } \begin{cases} \texttt{Source} = \texttt{Tart} \sqcap \exists \texttt{ing}.\texttt{Pastry} \sqcap \exists \texttt{ing}.\texttt{Apple} \\ \texttt{Target} = \texttt{Tart} \sqcap \forall \texttt{ing}.\neg\texttt{Apple} \end{cases} \tag{4}$$

Thus, the source case is represented by the instance σ, which is a tart with the types of ingredients pastry and apple. The target case is represented by the instance θ specifying that a tart without apple is requested.

Reusing the source case without adaptation for the target case amounts to add the assertion $\texttt{Source}(\theta)$. However this may lead to contradictions like here between $\exists \texttt{ing}.\texttt{Apple}(\theta)$ and $\forall \texttt{ing}.\neg\texttt{Apple}(\theta)$: the source case needs to be adapted before being applied to the target case.

2.2 Semantics

An interpretation is a pair $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta_\mathcal{I}$ is a non empty set (the *interpretation domain*) and where $\cdot^\mathcal{I}$ maps a concept $\texttt{C}$ into a subset $\texttt{C}^\mathcal{I}$ of $\Delta_\mathcal{I}$, a role $\texttt{r}$ into a binary relation $\texttt{r}^\mathcal{I}$ over $\Delta_\mathcal{I}$ (for $x, y \in \Delta_\mathcal{I}$, x is related to y by $\texttt{r}^\mathcal{I}$ is denoted by $(x, y) \in \texttt{r}^\mathcal{I}$), and an instance $\texttt{a}$ into an element $\texttt{a}^\mathcal{I}$ of $\Delta_\mathcal{I}$.

Given an interpretation $\mathcal{I}$, the different types of conceptual expressions are interpreted as follows:

$$\begin{aligned} \top^\mathcal{I} &= \Delta_\mathcal{I} & \bot^\mathcal{I} &= \emptyset \\ (\neg\texttt{C})^\mathcal{I} &= \Delta_\mathcal{I} \setminus \texttt{C}^\mathcal{I} & & \\ (\texttt{C} \sqcap \texttt{D})^\mathcal{I} &= \texttt{C}^\mathcal{I} \cap \texttt{D}^\mathcal{I} & (\texttt{C} \sqcup \texttt{D})^\mathcal{I} &= \texttt{C}^\mathcal{I} \cup \texttt{D}^\mathcal{I} \\ (\exists \texttt{r}.\texttt{C})^\mathcal{I} &= \{x \in \Delta_\mathcal{I} \mid \exists y, (x, y) \in \texttt{r}^\mathcal{I} \text{ and } y \in \texttt{C}^\mathcal{I}\} & & \\ (\forall \texttt{r}.\texttt{C})^\mathcal{I} &= \{x \in \Delta_\mathcal{I} \mid \forall y, \text{ if } (x, y) \in \texttt{r}^\mathcal{I} \text{ then } y \in \texttt{C}^\mathcal{I}\} & & \end{aligned}$$

For example, if $\texttt{Tart}^\mathcal{I}$, $\texttt{Apple}^\mathcal{I}$, $\texttt{Pastry}^\mathcal{I}$, and $\texttt{Cinnamon}^\mathcal{I}$ denote the sets of tarts, apples, pastries, and cinnamon, and if $\texttt{ing}^\mathcal{I}$ denotes the relation "has the ingredient", then the concept of equation (1) denotes the set of the tarts with apples and pastries, but without cinnamon.

Given a formula f and an interpretation $\mathcal{I}$, "$\mathcal{I}$ satisfies f" is denoted by $\mathcal{I} \models f$. A model of f is an interpretation $\mathcal{I}$ satisfying f. The semantics of the four types of formulas is as follows:

$$\begin{aligned} \mathcal{I} &\models \texttt{C} \sqsubseteq \texttt{D} & &\text{if } \texttt{C}^\mathcal{I} \subseteq \texttt{D}^\mathcal{I} \\ \mathcal{I} &\models \texttt{C} \equiv \texttt{D} & &\text{if } \texttt{C}^\mathcal{I} = \texttt{D}^\mathcal{I} \\ \mathcal{I} &\models \texttt{C}(\texttt{a}) & &\text{if } \texttt{a}^\mathcal{I} \in \texttt{C}^\mathcal{I} \\ \mathcal{I} &\models \texttt{r}(\texttt{a}, \texttt{b}) & &\text{if } (\texttt{a}^\mathcal{I}, \texttt{b}^\mathcal{I}) \in \texttt{r}^\mathcal{I} \end{aligned}$$

Given a knowledge base KB and an interpretation $\mathcal{I}$, $\mathcal{I}$ satisfies KB –denoted by $\mathcal{I} \models$ KB– if $\mathcal{I} \models f$ for each $f \in$ KB. A *model* of KB is an interpretation satisfying KB. A knowledge base KB entails a formula f –denoted by KB $\models f$– if every model of KB is a model of f. A tautology is a formula f satisfied by any interpretation. "f is a tautology" is denoted by $\models f$. Two knowledge bases are said to be equivalent if every model of one of them is a model of the other one and vice-versa.

2.3 Inferences

Let KB be a knowledge base. Some classical inferences on $\mathcal{ALC}$ consist in checking if KB $\models f$, for some formula f. For instance, checking if KB $\models$ C $\sqsubseteq$ D is called the *subsumption test*: it tests whether, according to the knowledge base, the concept C is more specific than the concept D, and thus is useful for organising concepts in hierarchies (e.g., index hierarchies of CBR systems).

The *concept classification* consists, given a concept C, in finding the atomic concepts A appearing in KB such that KB $\models$ C $\sqsubseteq$ A (the subsumers of C) and the atomic concepts B appearing in KB such that KB $\models$ B $\sqsubseteq$ C (the subsumees of C). The *instance classification* consists, given an instance a, in finding the atomic concepts A appearing in KB such that KB $\models$ A(a). These two inferences can be used for case retrieval in a CBR system.

The *ABox satisfiability* consists in checking, given an ABox, whether there exists a model of this ABox, given a knowledge base KB. Some other important inferences can be reduced to it, for instance:

$$\text{KB} \models \text{C} \sqsubseteq \text{D} \quad \text{iff} \quad \{(\text{C} \sqcap \neg \text{D})(\text{a})\} \text{ is not satisfiable, given KB}$$

where a is a new instance (not appearing neither in C, nor in KB). ABox satisfiability is also used to detect contradictions, e.g. the one mentioned at the end of section 2.1. It can be computed thanks to the most popular inference mechanism for $\mathcal{ALC}$ presented in next section.

2.4 A Classical Deduction Procedure in $\mathcal{ALC}$: The Tableau Method

Let KB be a knowledge base, $\mathcal{T}_0$, be the TBox of KB and $\mathcal{A}_0$, be the ABox of KB. The procedure aims at testing whether $\mathcal{A}_0$ is satisfiable or not, given KB.

Preprocessing. The first step of the preprocessing consists in substituting $\mathcal{T}_0$ by an equivalent $\mathcal{T}_0'$ of the form $\{\top \sqsubseteq \text{K}\}$, for some concept K. This can be done by first, substituting each formula C $\equiv$ D by two formulas C $\sqsubseteq$ D and D $\sqsubseteq$ C. The resulting TBox is of the form $\{\text{C}_i \sqsubseteq \text{D}_i\}_{1 \leq i \leq n}$ and it can be shown that it is equivalent to $\{\top \sqsubseteq \text{K}\}$, with

$$\text{K} = (\neg \text{C}_1 \sqcup \text{D}_1) \sqcap \ldots \sqcap (\neg \text{C}_n \sqcup \text{D}_n)$$

The second step of the preprocessing is to put $\mathcal{T}_0$ and $\mathcal{A}_0$ under negative normal form (NNF), i.e., by substituting each concept appearing in them by an equivalent concept such that the negation sign $\neg$ appears only in front of an atomic concept. It is always

possible to do so, by applying, as long as possible, the following equivalences (from left to right):

$$\neg\top \equiv \bot \qquad \neg\bot \equiv \top \qquad \neg\neg \mathtt{C} \equiv \mathtt{C}$$
$$\neg(\mathtt{C} \sqcap \mathtt{D}) \equiv \neg\mathtt{C} \sqcup \neg\mathtt{D} \qquad \neg(\mathtt{C} \sqcup \mathtt{D}) \equiv \neg\mathtt{C} \sqcap \neg\mathtt{D}$$
$$\neg\exists \mathtt{r}.\mathtt{C} \equiv \forall \mathtt{r}.\neg\mathtt{C} \qquad \neg\forall \mathtt{r}.\mathtt{C} \equiv \exists \mathtt{r}.\neg\mathtt{C}$$

For example, the concept $\neg(\forall \mathtt{r}.(\neg\mathtt{A} \sqcup \exists \mathtt{s}.\mathtt{B}))$ is equivalent to the following concept under NNF: $\exists \mathtt{r}.(\mathtt{A} \sqcap \forall \mathtt{s}.\neg\mathtt{B})$.

The TBox of DK given in equation (2) is equivalent to $\{\top \sqsubseteq \mathtt{K}\}$ under NNF with

$$\mathtt{K} = (\neg\mathtt{Apple} \sqcup \mathtt{PomeFruit}) \sqcap (\neg\mathtt{Pear} \sqcup \mathtt{PomeFruit}) \\ \sqcap (\neg\mathtt{PomeFruit} \sqcup \mathtt{Apple} \sqcup \mathtt{Pear})$$

(technically, to obtain this concept, the equivalence $\mathtt{C} \sqcup \bot \equiv \mathtt{C}$ has also been used).

Main process. Given $\mathcal{T}_0 = \{\top \sqsubseteq \mathtt{K}\}$ a TBox and $\mathcal{A}_0$ an ABox, both under NNF, the tableau method handles sets of ABoxes, starting with the singleton $\mathcal{D}_0 = \{\mathcal{A}_0^{\mathtt{K}}\}$, with

$$\mathcal{A}_0^{\mathtt{K}} = \mathcal{A}_0 \cup \{\mathtt{K}(\mathtt{a}) \mid \mathtt{a} \text{ is an instance appearing in } \mathcal{A}_0\}$$

Such a set of ABoxes $\mathcal{D}$ is to be interpreted as a disjunction: $\mathcal{D}$ is satisfiable iff at least one $\mathcal{A} \in \mathcal{D}$ is satisfiable.

Each further step consists in transforming the current set of ABoxes $\mathcal{D}$ into another one $\mathcal{D}'$, applying some transformation rules on ABoxes: when a rule ϱ, applicable on an ABox $\mathcal{A} \in \mathcal{D}$, is selected by the process, then $\mathcal{D}' = (\mathcal{D} \setminus \{\mathcal{A}\}) \cup \{\mathcal{A}^1, \ldots, \mathcal{A}^p\}$ where the $\mathcal{A}^i$ are obtained by applying ϱ on $\mathcal{A}$ (see further, for the description of the rules).

The process ends when no transformation rule is applicable.

An ABox is *closed* when it contains a *clash*, i.e. an obvious contradiction given by two assertions of the form $\mathtt{A}(\mathtt{a})$ and $(\neg\mathtt{A})(\mathtt{a})$.

Therefore, a closed ABox is unsatisfiable. An *open* ABox is a non-closed ABox.

An ABox is *complete* if no transformation rule can be applied on it.

Let $\mathcal{D}_{\text{end}}$ be the set of ABoxes at the end of the process, i.e. when each $\mathcal{A} \in \mathcal{D}$ is complete. It has been proven (see, e.g., [2]) that, with the transformation rules presented below the process always terminates, and $\mathcal{A}_0$ is satisfiable given $\mathcal{T}_0$ iff $\mathcal{D}_{\text{end}}$ contains at least one open ABox.

The transformation rules. There are four transformations rules for the tableau method applied to $\mathcal{ALC}$: $\longrightarrow_\sqcap$, $\longrightarrow_\sqcup$, $\longrightarrow_\forall$, and $\longrightarrow_\exists^{\mathtt{K}}$. None of these rules are applicable on a closed ABox. The order of these rules affects only the performance of the system, with the exception of rule $\longrightarrow_\exists^{\mathtt{K}}$ that must be applied only when no other rule is applicable on the current set of ABoxes (to ensure termination). These rules roughly corresponds to deduction steps: they add assertions deduced from existing assertions.[1]

[1] To be more precise, each of them transforms a disjunction of ABoxes $\mathcal{D}$ into another disjunction of ABoxes $\mathcal{D}'$ such that given $\mathcal{T}_0$, $\mathcal{D}$ is satisfiable iff $\mathcal{D}'$ is satisfiable.

The rule $\longrightarrow_{\sqcap}$ is applicable on an ABox $\mathcal{A}$ if this latter contains an assertion of the form $(\mathtt{C}_1 \sqcap \ldots \sqcap \mathtt{C}_p)(\mathtt{a})$, and is such that at least one assertion $\mathtt{C}_k(\mathtt{a})$ $(1 \leq k \leq p)$ is not an element of $\mathcal{A}$. The application of this rule returns the ABox $\mathcal{A}'$ defined by

$$\mathcal{A}' = \mathcal{A} \cup \{\mathtt{C}_k(\mathtt{a}) \mid 1 \leq k \leq p\}$$

The rule $\longrightarrow_{\sqcup}$ is applicable on an ABox $\mathcal{A}$ if this latter contains an assertion of the form $(\mathtt{C}_1 \sqcup \ldots \sqcup \mathtt{C}_p)(\mathtt{a})$ but no assertion $\mathtt{C}_k(\mathtt{a})$ $(1 \leq k \leq p)$. The application of this rule returns the ABoxes $\mathcal{A}^1, \ldots, \mathcal{A}^p$ defined, for $1 \leq k \leq p$, by:

$$\mathcal{A}^k = \mathcal{A} \cup \{\mathtt{C}_k(\mathtt{a})\}$$

The rule $\longrightarrow_{\forall}$ is applicable on an ABox $\mathcal{A}$ if this latter contains two assertions, of respective forms $(\forall \mathtt{r}.\mathtt{C})(\mathtt{a})$ and $\mathtt{r}(\mathtt{a},\mathtt{b})$ (with the same $\mathtt{r}$ and $\mathtt{a}$), and if $\mathcal{A}$ does not contain the assertion $\mathtt{C}(\mathtt{b})$. The application of this rule returns the ABox $\mathcal{A}'$ defined by

$$\mathcal{A}' = \mathcal{A} \cup \{\mathtt{C}(\mathtt{b})\}$$

The rule $\longrightarrow_{\exists}^{\mathtt{K}}$ is applicable on an ABox if

(i) $\mathcal{A}$ contains an assertion of the form $(\exists \mathtt{r}.\mathtt{C})(\mathtt{a})$;
(ii) $\mathcal{A}$ does not contain both an assertion of the form $\mathtt{r}(\mathtt{a},\mathtt{b})$ and an assertion of the form $\mathtt{C}(\mathtt{b})$ (with the same $\mathtt{b}$, and with the same $\mathtt{C}$ and $\mathtt{a}$ as in previous condition);
(iii) There is no instance $\mathtt{c}$ such that $\{\mathtt{C} \mid \mathtt{C}(\mathtt{a}) \in \mathcal{A}\} \subseteq \{\mathtt{C} \mid \mathtt{C}(\mathtt{c}) \in \mathcal{A}\}$.[2]

If these conditions are applicable, let $\mathtt{b}$ be a new instance. The application of this rule returns the ABox $\mathcal{A}'$ defined by

$$\mathcal{A}' = \mathcal{A} \cup \{\mathtt{r}(\mathtt{a},\mathtt{b}), \mathtt{C}(b)\} \cup \{\mathtt{K}(\mathtt{b})\}$$

Note that the TBox $\mathcal{T}_0 = \{\top \sqsubseteq \mathtt{K}\}$ is used here: since a new instance $\mathtt{b}$ is introduced, this instance must satisfy the TBox, which corresponds to the assertion $\mathtt{K}(\mathtt{b})$.

Remark 1. After the application of any of these rules on an ABox of $\mathcal{D}$, the resulting $\mathcal{D}'$ is equivalent to $\mathcal{D}$.

Example. Let us consider the example given at the end of section 2.1. Pretending that the source case represented by the instance σ can be applied to the target case represented by the instance θ amounts to identify these two instances, e.g., by substituting σ by θ. This leads to the ABox $\mathcal{A}_0 = \{\mathtt{Source}(\theta), \mathtt{Target}(\theta)\}$ (with $\mathtt{Source}$ and $\mathtt{Target}$ defined in (4)). The figure 1 represents this process. The entire tree represents the final set of ABoxes $\mathcal{D}_{\text{end}}$: each of the two branches represents a complete ABox $\mathcal{A} \in \mathcal{D}_{\text{end}}$. At the beginning of the process, the only nodes of this tree are $\mathtt{Source}(\theta)$, $\mathtt{Target}(\theta)$, and $\mathtt{K}(\theta)$: this corresponds to $\mathcal{D}_0 = \{\mathcal{A}_0^{\mathtt{K}}\}$. Then, the transformation rules are applied. Note that only the rule $\longrightarrow_{\sqcup}$ leads to branching. When a clash is detected in a branch (e.g. $\{\mathtt{Apple}(\mathtt{a}), (\neg\mathtt{Apple})(\mathtt{a})\}$) the branch represents a closed ABox (the clash is symbolised with $\square$). Note that the two final ABoxes are closed, meaning that $\{\mathtt{Source}(\theta), \mathtt{Target}(\theta)\}$ is not satisfiable: the source case needs to be adapted for being reused in the context of the target case.

[2] This third condition is called the *set-blocking* condition and is introduced to ensure the termination of the algorithm.

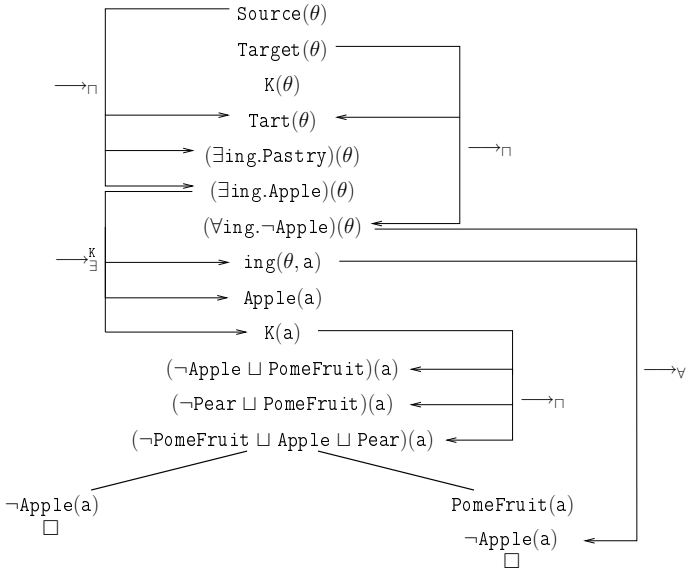


Fig. 1. Application of the tableau method proving that the ABox $\{\texttt{Source}(\theta), \texttt{Target}(\theta)\}$ is not satisfiable, given the TBox $\{\top \sqsubseteq \texttt{K}\}$ (for the sake of readability, the applications of the rules $\longrightarrow_{\sqcup}$ have not been represented; moreover, the order of application of rules has been chosen to make the example illustrative)

3 An Algorithm of Adaptation in $\mathcal{ALC}$

As seen in section 2.1, the reuse of the source case without adaptation may lead to a contradiction between $\texttt{Source}(\theta)$ and $\texttt{Target}(\theta)$. The adaptation algorithm presented in this section aims at solving this contradiction by weakening (generalising) $\texttt{Source}(\theta)$ so as to restore consistency, to apply to the target case θ what can be kept from Source.

3.1 Parameters and Result of the Algorithm

The parameters of the algorithm are DK, $\mathcal{A}^{\sigma}_{\texttt{srce}}$, $\mathcal{A}^{\theta}_{\texttt{tgt}}$, and cost. Its result is $\mathcal{D}$.

DK is a knowledge base in $\mathcal{ALC}$ representing the domain knowledge. In the running example, its ABox is empty, but in general, it may contain assertions.

The source and target cases are represented by two ABoxes that are satisfiable given DK: $\mathcal{A}^{\sigma}_{\texttt{srce}}$ and $\mathcal{A}^{\theta}_{\texttt{tgt}}$, respectively. More precisely, the source case is reified by an instance σ and $\mathcal{A}^{\sigma}_{\texttt{srce}}$ contains assertions about it. In the example above, $\mathcal{A}^{\sigma}_{\texttt{srce}}$ contains only one assertion, $\texttt{Source}(\sigma)$. Similarly, the target case is represented by an instance θ and $\mathcal{A}^{\theta}_{\texttt{tgt}}$ contains assertions about θ (only one assertion in the example: $\texttt{Target}(\theta)$).

The parameter cost is a function associating to a literal ℓ a numerical value $\texttt{cost}(\ell) > 0$, where a literal is either an atomic concept (positive literal) or a concept of the form $\neg\texttt{A}$ where A is atomic (negative literal). Intuitively, the greater $\texttt{cost}(\ell)$ is, the more difficult it is to give up the truth of an assertion $\ell(\texttt{a})$.

The algorithm returns $\mathcal{D}$, a set of ABoxes $\mathcal{A}$ solving the target case by adapting the source case: $\mathcal{A} \models \mathcal{A}^{\theta}_{\mathtt{tgt}}$ and $\mathcal{A}$ reuses "as much as possible" $\mathcal{A}^{\sigma}_{\mathtt{srce}}$. It may occur that $\mathcal{D}$ contains several ABoxes; in this situation, the knowledge of the system, in particular the `cost` function, is not complete enough to make a choice, thus it it up to the user to select an $\mathcal{A} \in \mathcal{D}$ (ultimately, by a random choice).

3.2 Steps of the Algorithm

The algorithm is composed of the following steps:

Preprocessing. Let $\mathcal{T}_{\mathtt{DK}}$ and $\mathcal{A}_{\mathtt{DK}}$ be the TBox and ABox of `DK`. Let `K` be a concept under NNF such that $\mathcal{T}_{\mathtt{DK}}$ is equivalent to $\{\top \sqsubseteq \mathtt{K}\}$. $\mathcal{A}_{\mathtt{DK}}$ is simply added to the ABoxes:

$$\mathcal{A}^{\sigma}_{\mathtt{srce}} \leftarrow \mathcal{A}^{\sigma}_{\mathtt{srce}} \cup \mathcal{A}_{\mathtt{DK}} \qquad\qquad \mathcal{A}^{\theta}_{\mathtt{tgt}} \leftarrow \mathcal{A}^{\theta}_{\mathtt{tgt}} \cup \mathcal{A}_{\mathtt{DK}}$$

Then, $\mathcal{A}^{\sigma}_{\mathtt{srce}}$ and $\mathcal{A}^{\theta}_{\mathtt{tgt}}$ are put under NNF.

Pretending that the source case solves the target problem. Reusing $\mathcal{A}^{\sigma}_{\mathtt{srce}}$ for the instance θ reifying the target case is done by assimilating the two instances σ and θ. This leads to the ABox $\mathcal{A}^{\theta}_{\mathtt{srce}}$, obtained by substituting σ by θ in $\mathcal{A}^{\sigma}_{\mathtt{srce}}$. Let $\mathcal{A}^{\theta}_{\mathtt{srce,tgt}} = \mathcal{A}^{\theta}_{\mathtt{srce}} \cup \mathcal{A}^{\theta}_{\mathtt{tgt}}$. If $\mathcal{A}^{\theta}_{\mathtt{srce,tgt}}$ is satisfiable given `DK`, then the straightforward reuse of the source case does not lead to any contradiction with the specification of the target case, so it just adds information about it. For example, let $\mathcal{A}^{\sigma}_{\mathtt{srce}} = \{\mathtt{Source}(\sigma)\}$ given by equation (3), let $\mathcal{A}^{\theta}_{\mathtt{tgt}} = \{\mathtt{Tart}(\theta), \mathtt{ing}(\theta, \mathtt{p}), \mathtt{FlakyPastry}(\mathtt{p})\}$ (i.e., "I want a tart with flaky pastry"), and the domain knowledge be $\mathtt{DK}' = \mathtt{DK} \cup \{\mathtt{FlakyPastry} \sqsubseteq \mathtt{Pastry}\}$, with `DK` defined in (2). With this example, it can be shown that $\mathcal{A}^{\theta}_{\mathtt{srce,tgt}}$ is satisfiable given $\mathtt{DK}'$ and it corresponds to an apple tart with flaky pastry.

In many situations, however, $\mathcal{A}^{\theta}_{\mathtt{srce,tgt}}$ is not satisfiable given `DK`. This holds for the running example. The principle of the adaptation algorithm consists in repairing $\mathcal{A}^{\theta}_{\mathtt{srce,tgt}}$. By "repairing" $\mathcal{A}^{\theta}_{\mathtt{srce,tgt}}$ we mean modifying it so as to make it complete and clash-free, and thus consistent. Removing clashes is not enough for that, the formulas from which they were were generated should be removed too. This motivates the introduction in section 3.2 of the AGraphs that extend ABoxes by keeping track of the application of rules. Moreover, to have a more fine-grained adaptation, $\mathcal{A}^{\theta}_{\mathtt{srce}}$ and $\mathcal{A}^{\theta}_{\mathtt{tgt}}$ are completed by tableau before being combined.

Applying the tableau method on $\mathcal{A}^{\theta}_{\mathtt{srce}}$ and on $\mathcal{A}^{\theta}_{\mathtt{tgt}}$, with memorisation of the transformation rule applications. In order to implement this step and the next ones, the notion of *assertional graph* (or *AGraph*) is introduced. An AGraph $\mathcal{G}$ is a simple graph whose set of nodes, $\mathit{Nodes}(\mathcal{G})$, is an ABox, and whose edges are labelled by transformation rules: if $(\alpha, \beta) \in \mathit{Edges}(\mathcal{G})$, the set of directed edges of $\mathcal{G}$, then $\lambda_{\mathcal{G}}(\alpha, \beta) = \varrho$ indicates that β has been obtained by applying ϱ on α and, possibly, on other assertions ($\lambda_{\mathcal{G}}$ is the labelling function of the graph $\mathcal{G}$).

The tableau method on AGraphs is based on the transformation rules $\Longrightarrow_{\sqcap}$, $\Longrightarrow_{\sqcup}$, $\Longrightarrow_{\forall}$, and $\Longrightarrow^{\mathtt{K}}_{\exists}$. They are similar to the transformation rules on $\mathcal{ALC}$ ABoxes, with some differences.

The rule $\Longrightarrow_{\sqcap}$ is applicable on an AGraph $\mathcal{G}$ if

(i) $\mathcal{G}$ contains a node α of the form $(\mathtt{C}_1 \sqcap \ldots \sqcap \mathtt{C}_p)(\mathtt{a})$;
(ii) $\mathcal{G} \neq \mathcal{G}'$ (i.e., $\mathit{Nodes}(\mathcal{G}) \neq \mathit{Nodes}(\mathcal{G}')$ or $\mathit{Edges}(\mathcal{G}) \neq \mathit{Edges}(\mathcal{G}')$) with $\mathcal{G}'$ defined by

$$\begin{aligned}
\mathit{Nodes}(\mathcal{G}') &= \mathit{Nodes}(\mathcal{G}) \cup \{\mathtt{C}_k(\mathtt{a}) \mid 1 \leq k \leq p\} \\
\mathit{Edges}(\mathcal{G}') &= \mathit{Edges}(\mathcal{G}) \cup \{(\alpha, \mathtt{C}_k(\mathtt{a})) \mid 1 \leq k \leq p\} \\
\lambda_{\mathcal{G}'}(\alpha, \mathtt{C}_k(\mathtt{a})) &= \Longrightarrow_{\sqcap} \text{ for } 1 \leq k \leq p \\
\lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ for } e \in \mathit{Edges}(\mathcal{G})
\end{aligned}$$

Under these conditions, the application of the rule returns $\mathcal{G}'$.

The main difference between rule $\longrightarrow_{\sqcap}$ on ABoxes and rule $\Longrightarrow_{\sqcap}$ on AGraphs is that the latter may be applicable to $\alpha = (\mathtt{C}_1 \sqcap \ldots \sqcap \mathtt{C}_p)(\mathtt{a})$ even when $\mathtt{C}_k(\mathtt{a}) \in \mathit{Nodes}(\mathcal{G})$ for each k, $1 \leq k \leq p$. In this situation, $\mathit{Nodes}(\mathcal{G}') = \mathit{Nodes}(\mathcal{G})$ but $\mathit{Edges}(\mathcal{G}') \neq \mathit{Edges}(\mathcal{G})$: a new edge $(\alpha, \mathtt{C}_k)$ indicates here that $\alpha \models \mathtt{C}_k(\mathtt{a})$ and thus, if $\mathtt{C}_k(\mathtt{a})$ has to be removed, then α has also to be removed (see further, the repair step of the algorithm).

The rules $\Longrightarrow_{\sqcup}$, $\Longrightarrow_{\forall}$, and $\Longrightarrow_{\exists}^{\mathtt{K}}$ are modified respectively from $\longrightarrow_{\sqcup}$, $\longrightarrow_{\forall}$, and $\longrightarrow_{\exists}^{\mathtt{K}}$ similarly. They are detailed in figure 2.

The tableau method presented in section 2.4 can be applied, given the TBox $\{\top \sqsubseteq \mathtt{K}\}$ and an ABox $\mathcal{A}_0$. The only difference is that AGraphs are manipulated instead of ABoxes, which involves that (1) an initial AGraph $\mathcal{G}_0$ has to be built from $\mathcal{A}_0$ (it is such that $\mathit{Nodes}(\mathcal{G}_0) = \mathcal{A}_0$ and $\mathit{Edges}(\mathcal{G}_0) = \emptyset$), (2) the rules $\Longrightarrow_{\cdot}$ are used instead of the rules $\longrightarrow_{\cdot}$, and (3) the result is a set of open and complete AGraphs (which is empty iff $\mathcal{G}_0$ is not satisfiable given $\{\top \sqsubseteq \mathtt{K}\}$).

Let $\{\mathcal{G}_i\}_{1 \leq i \leq m}$ and $\{\mathcal{H}_j\}_{1 \leq j \leq n}$ be the sets of open and complete AGraphs obtained by applying the tableau method respectively on $\mathcal{A}_0 = \mathcal{A}^{\theta}_{\mathtt{src}}$ and $\mathcal{A}^{\theta}_{\mathtt{tgt}}$. If $\mathcal{A}^{\theta}_{\mathtt{src}}$ and $\mathcal{A}^{\theta}_{\mathtt{tgt}}$ are satisfiable, then $m \neq 0$ and $n \neq 0$. If $m = 0$ or $n = 0$, the algorithm stops with value $\mathcal{D} = \{\mathcal{A}^{\theta}_{\mathtt{tgt}}\}$.

Generating explicit clashes from $\mathcal{G}_i$ and $\mathcal{H}_j$. A new kind of assertion, reifying the notion of clash, is considered: the *clash assertion* $\square\pm\mathtt{A}(\mathtt{a})$ reifies the clash $\{\mathtt{A}(\mathtt{a}), (\neg\mathtt{A})(\mathtt{a})\}$. The rule $\Longrightarrow_{\square}$ generates them. It is applicable on an AGraph $\mathcal{G}$ if

(i) $\mathcal{G}$ contains two nodes $\mathtt{A}(\mathtt{a})$ and $(\neg\mathtt{A})(\mathtt{a})$ (with the same $\mathtt{A}$ and the same $\mathtt{a}$);
(ii) $\mathcal{G} \neq \mathcal{G}'$ with $\mathcal{G}'$ defined by

$$\begin{aligned}
\mathit{Nodes}(\mathcal{G}') &= \mathit{Nodes}(\mathcal{G}) \cup \{\square\pm\mathtt{A}(\mathtt{a})\} \\
\mathit{Edges}(\mathcal{G}') &= \mathit{Edges}(\mathcal{G}) \cup \{(\mathtt{A}(\mathtt{a}), \square\pm\mathtt{A}(\mathtt{a})), ((\neg\mathtt{A})(\mathtt{a}), \square\pm\mathtt{A}(\mathtt{a}))\} \\
\lambda_{\mathcal{G}'}(\mathtt{A}(\mathtt{a}), \square\pm\mathtt{A}(\mathtt{a})) &= \lambda_{\mathcal{G}'}((\neg\mathtt{A})(\mathtt{a}), \square\pm\mathtt{A}(\mathtt{a})) = \Longrightarrow_{\square} \\
\lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ for } e \in \mathit{Edges}(\mathcal{G})
\end{aligned}$$

Under these conditions, the application of the rule returns $\mathcal{G}'$.

The next step of the algorithm is to apply the tableau method on each $\mathcal{G}_i \cup \mathcal{H}_j$, for each i and j, $1 \leq i \leq m$, $1 \leq j \leq n$, using the transformation rules $\Longrightarrow_{\sqcap}$, $\Longrightarrow_{\sqcup}$, $\Longrightarrow_{\forall}$, $\Longrightarrow_{\exists}^{\mathtt{K}}$, and $\Longrightarrow_{\square}$. A difference with the tableau method presented above is that it was useless to apply rules on closed ABoxes (or closed AGraphs). Here, when a rule is applicable to an AGraph containing an assertion clash, it is applied, which may lead to several clashes in the same AGraph.

A necessary condition for $\Longrightarrow_{\sqcup}$ to be applicable on an AGraph $\mathcal{G}$ is that $\mathcal{G}$ contains a node α of the form $(\mathtt{C}_1 \sqcup \ldots \sqcup \mathtt{C}_p)(\mathtt{a})$. If this is the case, then two situations can be considered:

(a) $\mathcal{G}$ contains no assertion $\mathtt{C}_k(\mathtt{a})$ $(1 \leq k \leq p)$. Under these conditions, the application of the rule returns the AGraphs $\mathcal{G}^1, \ldots, \mathcal{G}^p$ defined, for $1 \leq k \leq p$, by

$$\begin{aligned} \mathit{Nodes}(\mathcal{G}^k) &= \mathit{Nodes}(\mathcal{G}) \cup \{\mathtt{C}_k(\mathtt{a})\} \\ \mathit{Edges}(\mathcal{G}^k) &= \mathit{Edges}(\mathcal{G}) \cup \{(\alpha, \mathtt{C}_k(\mathtt{a}))\} \\ \lambda_{\mathcal{G}^k}(\alpha, \mathtt{C}_k(\mathtt{a})) &= \Longrightarrow_{\sqcup} \\ \lambda_{\mathcal{G}^k}(e) &= \lambda_{\mathcal{G}}(e) \text{ for } e \in \mathit{Edges}(\mathcal{G}) \end{aligned}$$

(b) $\mathcal{G}$ contains one or several assertions $\beta_k = \mathtt{C}_k(\mathtt{a})$ such that $(\alpha, \beta_k) \notin \mathit{Edges}(\mathcal{G})$. In this condition, $\Longrightarrow_{\sqcup}$ returns the AGraph $\mathcal{G}'$ obtained by adding to $\mathcal{G}$ these edges (α, β_k), with $\lambda_{\mathcal{G}'}(\alpha, \beta_k) = \Longrightarrow_{\sqcup}$.

The rule $\Longrightarrow_{\forall}$ is applicable on an AGraph $\mathcal{G}$ if

(i) $\mathcal{G}$ contains a node α_1 of the form $(\forall \mathtt{r}.\mathtt{C})(\mathtt{a})$ and a node α_2 of the form $\mathtt{r}(\mathtt{a}, \mathtt{b})$;

(ii) $\mathcal{G} \neq \mathcal{G}'$ with $\mathcal{G}'$ defined by

$$\begin{aligned} \mathit{Nodes}(\mathcal{G}') &= \mathit{Nodes}(\mathcal{G}) \cup \{\mathtt{C}(\mathtt{b})\} \\ \mathit{Edges}(\mathcal{G}') &= \mathit{Edges}(\mathcal{G}) \cup \{(\alpha_1, \mathtt{C}(\mathtt{b})), (\alpha_2, \mathtt{C}(\mathtt{b}))\} \\ \lambda_{\mathcal{G}'}(\alpha_1, \mathtt{C}(\mathtt{b})) &= \lambda_{\mathcal{G}'}(\alpha_2, \mathtt{C}(\mathtt{b})) = \Longrightarrow_{\forall} \\ \lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ for } e \in \mathit{Edges}(\mathcal{G}) \end{aligned}$$

Under these conditions, the application of the rule returns $\mathcal{G}'$.

The rule $\Longrightarrow_{\exists}^{\mathtt{K}}$ is applicable on an AGraph $\mathcal{G}$ if

(i) $\mathcal{G}$ contains a node α of the form $(\exists \mathtt{r}.\mathtt{C})(\mathtt{a})$;

(ii) (a) Either $\mathcal{G}$ does not contain both $\mathtt{r}(\mathtt{a}, \mathtt{b})$ and $\mathtt{C}(\mathtt{b})$, for any instance $\mathtt{b}$;
(b) Or $\mathcal{G}$ contains two assertions $\beta_1 = \mathtt{r}(\mathtt{a}, \mathtt{b})$ and $\beta_2 = \mathtt{C}(\mathtt{b})$, such that $(\alpha, \beta_1) \notin \mathit{Edges}(\mathcal{G})$ or $(\alpha, \beta_2) \notin \mathit{Edges}(\mathcal{G})$;

(iii) There is no instance $\mathtt{c}$ such that $\{\mathtt{C} \mid \mathtt{C}(\mathtt{a}) \in \mathit{Nodes}(\mathcal{G})\} \subseteq \{\mathtt{C} \mid \mathtt{C}(\mathtt{c}) \in \mathit{Nodes}(\mathcal{G})\}$ (set-blocking condition, introduced for ensuring termination of the algorithm).

If condition (ii-a) holds, let $\mathtt{b}$ be a new instance. The application of the rule returns $\mathcal{G}'$ defined by

$$\begin{aligned} \mathit{Nodes}(\mathcal{G}') &= \mathit{Nodes}(\mathcal{G}) \cup \{\mathtt{r}(\mathtt{a}, \mathtt{b}), \mathtt{C}(\mathtt{b}), \mathtt{K}(\mathtt{b})\} \\ \mathit{Edges}(\mathcal{G}') &= \mathit{Edges}(\mathcal{G}) \cup \{(\alpha, \mathtt{r}(\mathtt{a}, \mathtt{b})), (\alpha, \mathtt{C}(\mathtt{b}))\} \\ \lambda_{\mathcal{G}'}(\alpha, \mathtt{r}(\mathtt{a}, \mathtt{b})) &= \lambda_{\mathcal{G}'}(\alpha, \mathtt{C}(\mathtt{b})) = \Longrightarrow_{\exists}^{\mathtt{K}} \\ \lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ for } e \in \mathit{Edges}(\mathcal{G}) \end{aligned}$$

Under condition (ii-b), the application of the rule returns $\mathcal{G}'$ defined by

$$\begin{aligned} \mathit{Nodes}(\mathcal{G}') &= \mathit{Nodes}(\mathcal{G}) \\ \mathit{Edges}(\mathcal{G}') &= \mathit{Edges}(\mathcal{G}) \cup \{(\alpha, \beta_1), (\alpha, \beta_2)\} \\ \lambda_{\mathcal{G}'}(\alpha, \beta_1) &= \lambda_{\mathcal{G}'}(\alpha, \beta_2) = \Longrightarrow_{\exists}^{\mathtt{K}} \\ \lambda_{\mathcal{G}'}(e) &= \lambda_{\mathcal{G}}(e) \text{ for } e \in \mathit{Edges}(\mathcal{G}) \end{aligned}$$

Fig. 2. The transformation rules $\Longrightarrow_{\sqcup}$, $\Longrightarrow_{\forall}$, and $\Longrightarrow_{\exists}^{\mathtt{K}}$

Remark 2. If an assertion clash $\square\pm\mathtt{A}(\mathtt{a})$ is generated, then this clash is the consequence of assertions of both $\mathcal{G}_i$ and $\mathcal{H}_j$, otherwise, it would have been a clash generated at the previous step of the algorithm (since these two AGraphs are complete and open).

Repairing the assertion clashes. The previous step has produced a non-empty set $\mathcal{S}_{ij}$ of AGraphs, for each $\mathcal{G}_i \cup \mathcal{H}_j$. The repair step consists in repairing each of these AGraphs $\Gamma \in \mathcal{S}_{ij}$ and keeping only the ones that minimise the repair cost.[3] Let $\Gamma \in \mathcal{S}_{ij}$. If Γ contains no assertion clash, this involves that $\mathcal{G}_i \cup \mathcal{H}_j$ is satisfiable and so is $\mathcal{A}^{\theta}_{\mathtt{src},\mathtt{tgt}}$: no adaptation is needed. If Γ contains $\delta \geq 1$ assertion clashes, then one of them is chosen and the repair according to this clash gives a set of repaired AGraphs Γ' containing $\delta - 1$ clashes. Then, the repair is resumed on Γ', until there is no more clash.[4] The cost of the global repair is the sum of the costs of each repair. In the following, it is shown how one clash of Γ is repaired.

The principle of the clash repair is to remove assertions of Γ in order to avoid this assertion clash to be re-generated by re-application of the rules. Therefore, the repair of all the assertion clashes must lead to satisfiable AGraphs (this is a consequence of the completeness of the tableau algorithm on $\mathcal{ALC}$). For this purpose, the following principle, expressed as an inference rule, is used:

$$\frac{\varphi \models \beta \qquad \beta \text{ has to be removed}}{\varphi \text{ has to be removed}} \tag{5}$$

where β is an assertion and φ is a minimal set of assertions such that $\varphi \models \beta$ (φ is to be understood as the conjunction of its formulas). Removing φ amounts to forget one of the assertions $\alpha \in \varphi$: when $\mathrm{card}(\varphi) \geq 2$, there are several ways to remove φ, and thus, there may be several AGraphs Γ' obtained from Γ. The relation $\models$ linking φ and β is materialised by the edges of Γ. Therefore, on the basis of (5), the removal will be propagated by following these edges (α, β), from β to α.

Let $\beta = \square\pm\mathtt{A}(\mathtt{a})$, the assertion clash of Γ to be removed. Let $\alpha^+ = \mathtt{A}(\mathtt{a})$ and $\alpha^- = \neg\mathtt{A}(\mathtt{a})$. At least one of α^+ and α^- has to be removed. $\mathcal{H}_j$ being an open and complete AGraph, either $\alpha^+ \notin \mathcal{H}_j$ or $\alpha^- \notin \mathcal{H}_j$. Three types of situation remain:

- If $\alpha^+ \in \mathcal{H}_j$ then α^+ cannot be removed: it is an assertion generated from $\mathcal{A}^{\theta}_{\mathtt{tgt}}$. Then, α^- has to be removed.
- If $\alpha^- \in \mathcal{H}_j$ then α^+ has to be removed.
- If $\alpha^+ \notin \mathcal{H}_j$ and $\alpha^- \notin \mathcal{H}_j$, then the choice of removal is based on the minimisation of the cost. If $\mathtt{cost}(\mathtt{A}) < \mathtt{cost}(\neg\mathtt{A})$ then α^+ has to be removed. If $\mathtt{cost}(\mathtt{A}) > \mathtt{cost}(\neg\mathtt{A})$ then α^- has to be removed. If $\mathtt{cost}(\mathtt{A}) = \mathtt{cost}(\neg\mathtt{A})$, then two AGraphs are generated: one by removing α^+, the other one, by removing α^-.

If an assertion β has to be removed, the propagation of the removal for an edge (α, β) such that $\lambda_{\mathcal{G}}(\alpha, \beta) \in \{\Longrightarrow_{\sqcap}, \Longrightarrow_{\sqcup}, \Longrightarrow^{\mathtt{K}}_{\exists}\}$ consists in removing α (and propagating the removal from α).

[3] In our prototypical implementation of this algorithm, this has been improved by pruning the repair tasks when their cost exceed the current minimum.

[4] Some additional nodes may have to be removed to ensure consistency of the repaired AGraph. They are determined by some technical analysis over the set-bockings ($\Longrightarrow^{\mathtt{K}}_{\exists}$, condition (iii)).

Let β be an assertion to be removed that has been inferred by the rule $\Longrightarrow_\forall$. This means that there exist two assertions such that $\lambda_\mathcal{G}(\alpha_1, \beta) = \lambda_{\mathcal{G}'}(\alpha_2, \beta) = \Longrightarrow_\forall$. In this situation, two AGraphs are generated, one based on the removal of α_1, the other one, on the removal of α_2 (when α_1 or α_2 is in $\mathcal{H}_j$, only one AGraph is generated).

At the end of the repair process, a non empty set $\{\Gamma_k\}_{1 \leq k \leq p}$ of AGraphs without clashes has been built. Only the ones that are the result of a repair with a minimal cost are kept. Let $\mathcal{A}_k = \mathit{Nodes}(\Gamma_k)$. The result of the repair is $\mathcal{D} = \{\mathcal{A}_k\}_{1 \leq k \leq p}$.

Transforming the disjunction of ABoxes $\mathcal{D}$. If $\mathcal{A}, \mathcal{B} \in \mathcal{D}$ are such that $\mathcal{A} \models \mathcal{B}$, then the ABoxes disjunctions $\mathcal{D}$ and $\mathcal{D} \setminus \{\mathcal{A}\}$ are equivalent. This is used to simplify $\mathcal{D}$ by removing such $\mathcal{A}$.[5] After this simplifying test, each $\mathcal{A} \in \mathcal{D}$ is rewritten to remove the instances $\mathtt{i}$ introduced during a tableau process. First, the $\mathtt{i}$'s not related, neither directly, nor indirectly, to any non introduced instance by assertions $\mathtt{r}(\mathtt{a}, \mathtt{b})$ are removed, meaning that the assertions with such $\mathtt{i}$'s are removed (this may occur because of the repair step that may "disconnect" $\mathtt{i}$ from non-introduced instances). Then, a "de-skolemisation" process is done by replacing the introduced instances $\mathtt{i}$ by assertions of the form $(\exists \mathtt{r}.\mathtt{C})(\mathtt{a})$. For instance, the set $\{\mathtt{r}(\mathtt{a}, \mathtt{i}_1), \mathtt{A}(\mathtt{i}_1), \mathtt{s}(\mathtt{i}_1, \mathtt{i}_2), \neg\mathtt{B}(\mathtt{i}_2)\}$ is replaced by $\{(\exists \mathtt{r}.(\mathtt{A} \sqcap \exists \mathtt{s}.\neg\mathtt{B}))(\mathtt{a})\}$. The final value of $\mathcal{D}$ is returned by the algorithm.

Example. Consider the example given at the end of section 2.1. Giving all the steps of the algorithm is tedious, thus only the repairs will be considered.

Several AGraphs are generated and have to be repaired but they all share the same clash $\square\pm\mathtt{Apple}(\mathtt{a})$. Two repairs are possible and the resulting $\mathcal{D}$ depends only on the costs $\mathtt{cost}(\mathtt{Apple})$ and $\mathtt{cost}(\neg\mathtt{Apple})$.

If $\mathtt{cost}(\mathtt{Apple}) < \mathtt{cost}(\neg\mathtt{Apple})$, then $\mathcal{D} = \{\mathcal{A}\}$ with $\mathcal{A}$ equivalent to $(\mathtt{Tart} \sqcap \exists \mathtt{ing}.\mathtt{Pear})(\theta)$. The proposed adaptation is a pear tart.

If $\mathtt{cost}(\mathtt{Apple}) \geq \mathtt{cost}(\neg\mathtt{Apple})$, then $\mathcal{D} = \{\mathcal{A}\}$, with $\mathcal{A}$ equivalent to $\mathcal{A}^\theta_{\mathtt{tgt}}$. Nothing is learnt from the source case for the target case.

3.3 Properties of the Algorithm

The adaptation algorithm terminates. This can be proven using the termination of the tableau algorithm on ABoxes [2]. Repair removes at least one node from finite AGraphs at each step, thus it terminates too.

Every ABox $\mathcal{A} \in \mathcal{D}$ satisfies $\mathtt{Target}$ constraints: $\mathcal{A} \models \mathcal{A}^\theta_{\mathtt{tgt}}$.

Provided that $\mathcal{A}^\theta_{\mathtt{tgt}}$ is satisfiable, every $\mathcal{A} \in \mathcal{D}$ is satisfiable. In other words, unless the target case is in contradiction with the domain knowledge, the adaptation provides a consistent result. When $\mathcal{A}^\theta_{\mathtt{srce}}$ is not satisfiable, $\mathcal{D}$ is equivalent to $\{\mathcal{A}^\theta_{\mathtt{tgt}}\}$. This means that when a meaningless[6] $\mathcal{A}^\theta_{\mathtt{srce}}$ is given, $\mathcal{A}^\theta_{\mathtt{tgt}}$ is not altered.

[5] In our tests, we have used necessary conditions of $\mathcal{A} \models \mathcal{B}$ based on set inclusions, with or without the renaming of one introduced instance. This has led to a dramatic reduction of the size of $\mathcal{D}$, which suggests that the algorithm presented above can be greatly improved, by pruning unnecessary ABox generation.

[6] In a logical setting, an inconsistent knowledge base is equivalent to any other inconsistent knowledge base and thus, it is meaningless.

If the source case is applicable under the target case constraints ($\mathcal{A}^{\theta}_{\mathtt{srce,tgt}} = \mathcal{A}^{\theta}_{\mathtt{srce}} \cup \mathcal{A}^{\theta}_{\mathtt{tgt}}$ is satisfiable) then $\mathcal{D}$ contains a sole ABox which is equivalent to $\mathcal{A}^{\theta}_{\mathtt{srce,tgt}}$: the source case is reused without modification to solve the target case.

The adaptation presented here can be considered as a generalisation and specialisation approach to adaptation. The ABoxes $\mathcal{A} \in \mathcal{D}$ are obtained by "generalising" $\mathcal{A}^{\theta}_{\mathtt{srce}}$ into $\mathcal{A}'$: some formulas of $\mathcal{A}^{\theta}_{\mathtt{srce}}$ are dropped for weaker consequences to obtain $\mathcal{A}'$ thus $\mathcal{A} \models \mathcal{A}'$, then $\mathcal{A}'$ is "specialised" into $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}^{\theta}_{\mathtt{tgt}}$.

4 Discussion and Related Work

Beyond matching-based adaptation processes? There are two types of algorithms for the classical deductive inferences in DLs: the tableau algorithm presented above and the structural algorithms. The former is used for expressive DLs (i.e., for $\mathcal{ALC}$ and all the DLs extending $\mathcal{ALC}$). The latters are used for the other DLs (for which at least some of the deductive inferences are polynomial). A structural algorithm for the subsumption test $\mathtt{KB} \models \mathtt{C} \sqsubseteq \mathtt{D}$ consists, after a preprocessing step, in *matching* descriptors of $\mathtt{D}$ with descriptors of $\mathtt{C}$. This matching procedure is rather close to the matching procedures used by most of the adaptation procedures, explicitly or not (if the cases have a fixed attribute-value structure, usually, the source and target cases are matched attribute by attribute, and the matching process does not need to be made explicit). Structural algorithms appear to be ill-suited for expressive DLs and tableau algorithms are used instead. The adaptation algorithm presented in this paper, based on tableau method principles, has no matching step (even if one can a posteriori match descriptors of source case and adapted target case). From those observations, we hypothesise that beyond a certain level of expressivity of the representation language, it becomes hardly possible to use matching techniques for an adaptation taking into account domain knowledge.

Other work on CBR and description logics. Despite the advantages of using DLs in CBR, as motivated in the introduction, there are rather few research on CBR and DLs. In [7], concepts of a DL are used as indexes for retrieving plans of a case-based planner, and adaptation is performed in another formalism. In [11], a non expressive DL is used for retrieval and for case base organisation. This work uses in particular the notion of *least common subsumer* (LCS) to reify similarity of the concepts representing the source and target cases: the LCS of concepts $\mathtt{C}$ and $\mathtt{D}$ is the more specific concept that is more general than both $\mathtt{C}$ and $\mathtt{D}$ and thus points out their common features. Therefore the LCS inference can be seen as a matching process (that might be used by some adaptation process). In an expressive DL, the LCS of $\mathtt{C}$ and $\mathtt{D}$ is $\mathtt{C} \sqcup \mathtt{D}$ (or an equivalent concept), which does not express anything about similar features of $\mathtt{C}$ and $\mathtt{D}$.

To our knowledge, the only attempts to define an adaptation process for DLs are [5] and [4]. [5] presents a modelling of the CBR life cycle using DLs. In particular, it presents a substitution approach to adaptation which consists in matching source and target case items by chains of roles (similar to chains of assertions $\mathtt{r}(\mathtt{a}_1, \mathtt{a}_2)$, $\mathtt{r}(\mathtt{a}_2, \mathtt{a}_3)$, etc.) in order to point out what substitutions can be done. [4] uses adaptation rules (reformulations) and multi-viewpoint representation for CBR, including a complex adaptation step. By contrast, the algorithm presented in this paper uses mainly the domain

knowledge to perform adaptation: a direction of work will be to see how these approaches can be combined.

5 Conclusion and Future Work

This paper presents an algorithm for adaptation dedicated to case-based reasoning systems whose cases and domain knowledge are represented in the expressive DL $\mathcal{ALC}$. The first question raised by an adaptation problem is: "What has to be adapted?" The way this question is addressed by the algorithm consists in first pretending that the source case solves the target problem and then pointing out logical inconsistencies: these latters correspond to the parts of the source case to be modified in order to suit the target case. These principles are then applied to $\mathcal{ALC}$, for which logical inconsistencies are reified by the clashes generated by the tableau method. The second question raised by an adaptation problem is: "How will the source case be adapted?" The idea of the algorithm is to repair the inconsistencies by removing (temporarily) some knowledge from the source case, until the consistency is restored. This adaptation approach can be classified as a transformational one since it does not use explanations or justifications associated with the source case, as would a derivational (or generative) approach do.

Currently, only a basic prototype of this adaptation algorithm has been implemented, and it is not very efficient. A future work will aim at implementing it efficiently and in an extendable way, taking into account the future extensions presented below. This might be done by reusing available DL inference engines, provided their optimisation techniques do not interfere with the results of this adaptation procedure. It can be noted that the research on improving the tableau method for DLs has led to dramatic gains in term of computing time (see, in particular, [6]).

The second direction of work will be to extend the algorithm to other expressive DLs. In particular, we plan to extend it to $\mathcal{ALC}(\mathtt{D})$, where $\mathtt{D}$ is the concrete domain of real number tuples with linear constraint predicates. This means that cases may have numerical features (integer or real numbers) and domain knowledge may contain linear constraints on these features. This future work will also extend [3].

The algorithm of adaptation presented above can be considered as a generalisation and specialisation approach to adaptation (cf. section 3.3). By contrast, the algorithm of [4] is a rule-based adaptation, a rule (a reformulation) specifying a relevant substitution to a given class of source case. A lead to integrate these two approaches is to use the adaptation rules during the repair process: instead of removing assertions leading to a clash, such a rule, when available, could be used to propose substitutes.

As written in the introduction, this algorithm follows work on adaptation based on belief revision, though it cannot be claimed that this algorithm, as such, implements a revision operator for $\mathcal{ALC}$ (e.g., it does not enable the revision of a TBox by an ABox). In [3], revision-based adaptation is generalised in merging-based case combination. Such a generalisation should be applicable to the algorithm defined in this paper: the ABox $\mathcal{A}^{\theta}_{\mathtt{srce}}$ is replaced by several ABoxes and the repairs are applied on these ABoxes. Defining precisely this algorithm and studying its properties is another future work.

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

References

1. 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, 510–530 (1985)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
3. Cojan, J., Lieber, J.: Belief Merging-based Case Combination. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 105–119. Springer, Heidelberg (2009)
4. d'Aquin, M., Lieber, J., Napoli, A.: Decentralized Case-Based Reasoning for the Semantic Web. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 142–155. Springer, Heidelberg (2005)
5. Gómez-Albarrán, M., González-Calero, P.A., Díaz-Agudo, B., Fernández-Conde, C.: Modelling the CBR Life Cycle Using Description Logics. In: Althoff, K.-D., Bergmann, R., Karl Branting, L. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 147–161. Springer, Heidelberg (1999)
6. Horrocks, I.: Optimising Tableaux Decision Procedures for Description Logics. Ph.D. thesis, University of Manchester (1997)
7. Koehler, J.: Planning from Second Principles. Artificial Intelligence 87, 145–186 (1996)
8. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann, Inc., San Francisco (1993)
9. Lieber, J.: Application of the Revision Theory to Adaptation in Case-Based Reasoning: the Conservative Adaptation. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 239–253. Springer, Heidelberg (2007)
10. Riesbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Inc., Hillsdale (1989)
11. Salotti, S., Ventos, V.: Study and Formalization of a Case-Based Reasoning System Using a Description Logic. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 286–297. Springer, Heidelberg (1998)

Case-Based Plan Diversity

Alexandra Coman and Héctor Muñoz-Avila

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

Abstract. The concept of diversity was successfully introduced for recommender-systems. By displaying results that are not only similar to a target problem but also diverse among themselves, recommender systems have been shown to provide more effective guidance to the user. We believe that similar benefits can be obtained in case-based planning, provided that diversity-enhancement techniques can be adapted appropriately. Our claim is that diversity is truly useful when it refers not only to the initial and goal states of a plan, but also to the sequence of actions the plan consists of. To formalize this characteristic and support our claim, we define the metric of "plan diversity" and put it to test using plans for a real-time strategy game, a domain chosen for the simplicity and clarity of its tasks and the quantifiable results it generates.

Keywords: diversity, similarity, case-based planning.

1 Introduction

"Diversity", the quantifiable variation among retrieved query results, has been explored as a means of improving the performance of recommender systems [1,2,3,4,5]. Results that not only are similar to a user query or adhere to a set of constraints, but are also diverse among themselves are argued to provide genuine and useful alternatives, covering a larger portion of the solution space [1].

We believe that the introduction of diversity considerations in case-based planning [6,7,8,9], while so far insufficiently explored, can prove equally advantageous and have a significant impact on fields such as interactive planning [10,11]. In interactive planning, the user is presented with a set of planning choices. With the assistance of the system, the user chooses one that best accomplishes their goals. Plan diversity could enhance such systems by providing the user with truly distinct choices.

In identifying plan matches, one could supplement the criterion of similarity to a new problem-case by that of diversity between selected plans, with benefits similar to those obtained with recommender systems. Hereinafter, we explore the relative merits of several alternatives that can be taken in integrating diversity in plan retrieval. We formally define and assess two types of diversity, each targeting a different aspect of analyzed and retrieved cases in case-based planning.

- **State diversity** - The "weaker" of the two, as we aim to prove, introduced for comparison purposes. It characterizes cases that are dissimilar in terms of

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 66–80, 2010.

initial and, possibly, final state, but may well be made up of the same (or very similar) plans.

- **Plan diversity** allows the identification of sets of plans with considerably different sequences of intermediary actions, which we hypothesize represent genuine alternatives. Our claim, based on immediate intuition and put to test experimentally, is that diversity is truly useful when it refers not only to the initial and goal states, but rather to the sequence of intermediary steps between them, and is balanced with initial and goal state similarity.

We incorporate both of these forms of diversity into several retrieval methods: two devised specifically for plan retrieval, so as to obtain diversity without sacrificing similarity; and a plan-diversity-aware adaptation of "Bounded Greedy", a technique previously introduced for similar purposes in recommender systems [1,2,3]. For comparison purposes, we also test the standard "Bounded Greedy" algorithm, which is based on what we refer to as "state diversity". The preservation of similarity to the query, while promoting diversity between matches, is a constraint we inherit from previous forays into diversity [2] and which we attempt to address in a manner suitable to the plan retrieval domain.

We test all aforementioned methods on a series of cases containing the usual components (initial state, goal state, plan) of cases in case-based planning [7]. Our testbed is Wargus, a real-time strategy game, chosen because it has been used in previous work (e.g., [8]) and for the possibilities it offers to obtain easily quantifiable results. Diversity, or the lack of it, is immediately observable on watching a game unfold, but also relevantly reflected, for the purpose of analysis, in quantifiers such as game scores and the duration of a game session. Our goal to achieve diversity for case-based planning is motivated by the success obtained with recommender systems, which thus far has been mostly demonstrated for analysis tasks. For our case-based plan diversity methods, the results are very encouraging: adapted plans retrieved based on our case-based plan diversity methods generate game-play instances of significant and discernible variation.

It should be noted that we are, at this stage, only marginally concerned with how "successful" these diverse retrieved plans will, on average, be in solving whatever problem they are meant to solve within any given parameters specific to their domain. By "success", we will, instead, refer to generating a set of adapted plans that produce results running the gamut from low to excellent over a variety of criteria. We consider plan variation to be intrinsically valuable and a goal in itself, although the exact nature of its value, as well as the range of the possibilities it opens up, is bound to vary from domain to domain.

2 Background

McGinty and Smyth [1] enhance the typical recommendation cycle to include the notion of diversity, using a technique called "bounded greedy", which was first introduced in [2]. "Bounded greedy" works by first ranking cases based on their similarity to the query and afterwards repeatedly selecting, out of the ranked list, those that

maximize the weighted sum of similarity to the query and "relative diversity", where relative diversity is defined as (*C* is a set of cases, *n* the number of cases in *C*, *c* a case and *Sim* a similarity metric):

$$RelDiv(c, C) = \frac{\sum_{c_i \in C}(1 - Sim(c_i, c))}{n}, C \neq \{\}. \quad (1)$$

The selected cases are the ones which maximize the quality given by:

$$\alpha Sim(q, i) + (1 - \alpha) RelDiv(i, R) \ , \quad (2)$$

where q is the user query and R the set of cases retrieved so far. Their methods are shown to be an improvement over classical similarity-based retrieval.

While, admittedly, various aspects of the recommendation cycle are not relevant to case-based planning, we can easily retain the diversity metric and adjust it to meet our own purposes. An important difference between recommender systems, and analysis tasks in general, and synthesis tasks such as case-based planning is, however, bound to affect the proper handling of such an adjustment: a classification task stops with the identification of a satisfactory query result or set of results, whereas case-based planning (as synthesis task) must, after identifying a query result, adapt it to produce a solution plan, which, in our context, must be executed.

How do we describe one such satisfactory set of plans with regard to diversity? A successful diversity-aware retrieval algorithm is one that generates plans that, when adapted, produce diverse results.

We follow the usual case-based planning convention: cases are represented as having two components: the problem, composed of initial and goal states, and a plan transforming the initial state into the goal state [7].

3 Example

We use a real-time-strategy game to showcase an example illustrating why diversity based on initial and final states ("state diversity") is insufficient to produce enough variation in the retrieved cases. Approaches to game-play in real-time-strategy games (and, generally, any game genres based on simulating combat) can be categorized by strategies consisting of some combination of "offensive" and "defensive" measures. Any such strategy, however complex, is bound to be reducible to combinations of these two basic approaches, in varying forms and degrees. Furthermore, most actions that can be taken in a game, such as using various types of attacks, building a defensive unit, "healing" one's units or fleeing can usually be categorized as pertaining more to a defensive or to an offensive strategy. Ideally, the system would retrieve plans that are diverse as per these categories. The problem is that a significant effort would be required to annotate the plans according to these categories. Instead, we propose to use plan diversity as the means to identify diverse plans without requiring any such plan labeling to be known beforehand.

Our experiments (described in more detail in Section 5) involve retrieving (based on diversity considerations, as well as similarity to a "new problem" case) gameplay plans, wherein a plan consists of a series of actions (offensive, defensive or balanced) that the player's units can take. Its initial state is the initial structure of our player's combat force i.e. number of units of each kind: in our experiments, these can be "peasants" or "soldiers". We present an example of how plan retrieval would be handled in this context, when using state and plan diversity. Let the plan library contain only the following three cases and let us, for the moment, assume that we are only looking for a pair of maximally diverse cases (ignoring the criterion of similarity to a new problem):

Case 1: The initial state configuration consists of three "soldier" units and two "peasant" units. The plan consists of all units attacking.

Case 2: The initial state configuration consists of four "soldier" units and no "peasant" units. The plan, once again, consists of all units attacking.

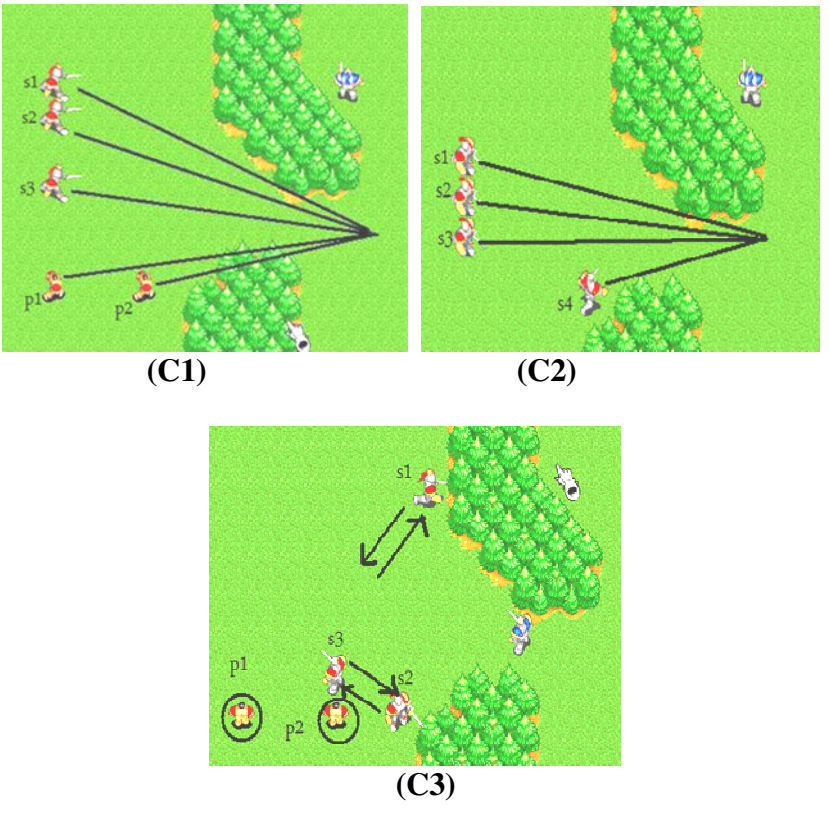


Fig. 1. Case pairs (C1, C2) and (C2, C3) are state-diverse, while pairs (C1, C3) and (C2, C3) are plan-diverse. C1 and C2 are instances of the same offensive strategy (all units attack), while C3 exemplifies a distinct defensive strategy ("soldier" units patrol, "peasant" units stay put).

Case 3: The initial state configuration consists of three "soldier" units and two "peasant" units (it is identical to that of Case 1). The plan consists of the "soldiers" patrolling given areas in expectation of an attack, while the weaker "peasant" units stay put, not "willingly" exposing themselves to damage.

Cases 1 and 2 display purely offensive strategies, whereas Case 3 is largely defensive. An algorithm based on "state diversity" (diversity of initial states only, in our experiment, for reasons explained in Section 5) would select either Case 1 and Case 2 or Case 2 and Case 3 (either Case 1 or Case 3 would always be discarded, as they have identical initial states).

Assuming, therefore, that "tie" situations such as this are handled by choosing one of the multiple cases with identical initial states randomly, the probability of choosing truly diverse plans tends to decrease with the total number of cases. In this simple example, we have a 0.5 probability of obtaining two offensive strategies, which, on being adapted for a new problem case, translate to identical strategies (it is only reasonable that an adaptation of an offensive plan will also be offensive).

If retrieving based on plan diversity, however, we are guaranteed to obtain either Case 1 and Case 3 or Case 2 and Case 3. That is, with a probability of 1, we will (in this simple example) be presented with two plans which are truly different in the strategy that they incorporate and will generate distinct adaptations.

4 State and Plan Diversity

Herein, we describe the diversity-aware plan-retrieval methods we propose and evaluate. For all algorithms below, we assume cases consisting of triples of the type (*initial state*, *goal state*, *plan*) and a new problem described only in terms of the initial and the goal state. The problem of finding a set of cases that are maximally diverse from one another and, at the same time, maximally similar to the problem is computationally expensive [2]. As a result, the algorithms below aim, instead, at finding good approximations of such optimal solutions.

4.1 State Diversity through Similarity Clusters

Below, we show the cluster-based retrieval method for state diversity, which we call *SDSC - State Diversity through Similarity Clusters*. First, cases are sorted in reverse order of their similarity to the query problem (line 1). Cases that are similar to one another are clustered together. To obtain k cases that are likely maximally similar to the new problem, as well as diverse from each other (such that there are no two identical similarity scores in the retrieved set) we need only choose one case from each of the first k clusters, as "state diversity" is also based on the initial and final states (line 4). When we choose a case from a cluster, we do so randomly.

Procedure SDSC(newProb, CB)
Input: newProb: the query problem; CB: the case base
Output: R: the list of recommended cases in CB for newProb
1. CB' ← sort CB in decreasing order of their similarity to newProb
2. nC ← number of clusters in CB'
3. R←{}
4. **for** j←1 **to** nC **do**
   simClust ← select-cluster(j, CB')
   case ← select-random-case(simClust)
   R ← R ∪ {case}
   end-for
5. **return** R

4.2 Plan Diversity through Greedy Selection

Below, we show the retrieval algorithm based on plan-diversity, which we call *PDGS - Plan Diversity through Greedy Selection*. Cases are sorted in reverse order, based on their similarity to the new problem (line 1). We add to R the case in CB' which is the closest to the target problem (line 3). For each case i, starting with the second-highest ranking in the hierarchy, we compute plan diversity (*plDiv*) between it and the cases chosen so far (Formula 3 below). The domain-specific similarity metric *plSim* identifies two plans as similar based on their sequences of actions. If *plDiv* is higher than threshold Δ and *stSim* (similarity to the new problem) is higher than threshold Δ', then c is added to the hierarchy. Otherwise, stop and return the chosen case set (lines 4-6).

$$plDiv(case_i, chosenCaseSet) = \sum_{k=1,|chosenCaseSet|} \frac{1-plSim(case_i,case_k)}{|chosenCaseSet|}. \quad (3)$$

4.3 Plan-Diversity Bounded Greedy

Below, we show our adapted version of the "bounded greedy" algorithm [1], which we call *Plan-Diversity Bounded Greedy - PBGA*. Our version works by selecting, on each step, the case that maximizes the sum of the similarity to the new problem and plan diversity with regard to the states selected so far, according to the following formula (which is a variant of Formula 2):

$$simPlDiv = \alpha * Sim(newProb, c) + (1 - \alpha) * plDiv(c, R), \quad (4)$$

where *plDiv* is plan diversity as described in Formula (3). For the original version of "Bounded Greedy", we compute state diversity according to the formula (simSt is a similarity metric comparing the initial and goal states):

$$stateDiv(case_i, chosenCaseSet) = \sum_{k=1,|chosenCaseSet|} \frac{1-simSt(case,case_k)}{|chosenCaseSet|}. \quad (5)$$

The original "Bounded Greedy" algorithm [1] differs from our variant below in that they use state similarity and diversity to select cases to be added to R (lines 2-3), whereas "Plan-Diversity Bounded Greedy" uses state similarity and plan diversity.

Procedure PDGS(newProb, CB, Δ, Δ')
Input: newProb: the query problem; CB: the case base; Δ, Δ': thresholds
Output: R: the list of recommended cases in CB for newProb
1. CB' ← sort CB in decreasing order of their similarity to newProb
2. R←{}, i←2
3. **add** first case in CB' **into** R
4. **repeat**
   c ←select case i from CB'
   if plDiv(c, R) > Δ **and** stSim(newProb,c) > Δ' **then**
      R ← R ∪ {c}
   end-if
   i←i+1
5. **while** (plDiv(c, R) > Δ **and** stSim(newProb,c) > Δ' **and** i≤$|CB|$)
6. **return** R

Procedure PBGA(newProb, CB, k)
Input: newProb: the query problem; CB: the case base; k: integer
Output: R: the list of k recommended cases in CB for newProb
1. R ←{}
2. **for** i←1 **to** k **do**
   Sort CB by simPlDiv
   c ←first case in CB
   R←R ∪ {c}
   CB ←CB – {c}
3. **end-for**
4. **return** R

5 Experiment

As with other CBR research, we use a game as our testbed [8,12]. Our hypothesis is that plan retrieval by taking into account plan-diversity considerations will result in a wider range of choices than state-diversity-based retrieval. On adapting plan-diverse retrieved cases and running these in a game, we expected to obtain results (measured via game-specific metrics) that are quantifiably more varied than those obtained by running plans retrieved via state-diversity-based methods.

5.1 Experimental Setup

Our experiments are conducted using simple real-time-strategy game plans, run on "Wargus", a clone of "Warcraft II: Tides of Darkness" which uses the free real-time strategy game engine "Stratagus".

The two-player Wargus games we stage take place on a 32x32 tile map, with our player acting out plans against the built-in Wargus enemy AI. We allow only two types of units: "soldiers" (basic fighting units) and "peasants" (used normally for resource harvesting and creating "building" units, but introduced here for the purpose

of illustrating strategy variation by having weaker units engage in battle). We only vary our player's initial configuration. State similarity and, subsequently, state diversity, are based on the initial configuration (our player's units). Plans are not annotated by the specific goals they achieve (e.g., capture the center of the map). This simulates a situation in which successful plans are captured by observing the user's actions, but not knowing their intent. Thus, an initial state is defined by a pair of type *(numSold, numPeas)*, where *numSold* is the number of "soldier" units and *numPeas* the number of "peasant" units our player has at their disposal.

The formula for state similarity, to be utilized in diversity-aware retrieval algorithms as described in the previous section, therefore becomes:

$$simSt(case_1, case_2) = \frac{\frac{min(numSold_1, numSold_2)}{max(numSold_1, numSold_2)} + \frac{min(numPeas_1, numPeas_2)}{max(numPeas_1, numPeas_2)}}{2}. \quad (6)$$

As for the plans themselves, the actions they can include are "AttackMove" (moving to a given spot on the map, while attacking any enemy units encountered on the way), "Patrol" (a defensive attitude, consisting of moving between two locations, prepared to fight in case of an enemy invasion attempt), "Move" (moving the unit to specified coordinates) and the self-explanatory "Attack". For the sake of simplicity, we do not take into account any coordinates associated with these moves when computing plan similarity. Groups of units will, as a rule, move in the same direction, both in our initial "library" plans and in our adapted ones. We, also, do not consider action order for computing plan similarity, as it is neither relevant to this particular task, nor vital in demonstrating the basic concept of plan diversity. Plan similarity between two plans is defined by:

$$plSim(case_1, case_2) = \frac{\frac{min(numAttack_1, numAttack_2)}{max(numAttack_1, numAttack_2)} + \frac{min(numMoveAttack_1, numMoveAttack_2)}{max(numMoveAttack_1, numMoveAttack_2)}}{4} + \frac{\frac{min(numPatrol_1, numPatrol_2)}{max(numPatrol_1, numPatrol_2)} + \frac{min(numMove_1, numMove_2)}{max(numMove_1, numMove_2)}}{4}. \quad (7)$$

The new problem case will be defined by its initial state, a *(numSoldiers, numPeasants)* pair. We use a new problem with the initial state (10, 9): 10 "soldiers" and 9 "peasants" (see Fig. 2).

Fig. 2. Initial configurations of a "library" case and a "new problem". The layout of the terrain and lack of building units ensure the brevity of game-play episodes, making them easily comparable to each other.

Our "plan library" contains 16 plans (Table 1), consisting of all possible state-strategy combinations between 4 start states (with varying numbers of "peasant" and "soldier" units) and a number of plans.

Table 1 shows only a summarization of the plans; the following is an example of an actual plan, as stored in the plan library:

```
m_pb.attackMove(1,13,10);
m_pb.move(2, 7, 8);
m_pb.patrol(2, 9, 7);
m_pb.move(13, 5, 9);
m_pb.patrol(13, 6, 10);
```

It instructs unit 1 to move to coordinates (13,10) on the map, while attacking any enemy unit encountered on the way; and units 2 and 13 to move to coordinates (7,8) and (5,9), respectively, and to start patrolling back and forth between their new location and coordinates (9,7) and (6,10), respectively.

Although the following information is not stored in the plan library, conceptually, there are 4 plan strategies (in each case, adapted to the number of units in the start state). The four strategies are: (1) "Offensive" (all units attack), (2) "Defensive" (all "soldier" units patrol and all "peasant" units stay put), (3) "Balanced Offensive" (75% of "soldiers" attack, 25% patrol), and (4) "Balanced Defensive" (50% of "soldiers" attack, 50% patrol).

Adaptation is performed by building a plan based on the same strategy as the retrieved plan, but adjusted to the number of units in the new problem initial state. For example, if the retrieved plan is Case 2 in Table 1, a "defensive" plan, the adapted plan for the new problem will have all 10 "soldier" units move to a key location and patrol, while all 9 "peasant" units remain where they are.

Our experiments are conducted as follows:

- Each of the four retrieval algorithms ("State Diversity through Similarity Clusters", "Plan Diversity through Greedy Selection", "State Diversity Bounded Greedy" and "Plan Diversity Bounded Greedy") is run on the new problem and set of library cases 4 times (tie-breaking is handled by randomly selecting a case), each time recording the top 4 retrieved plans. For *PDGS* (see Section 4.1.2), we use the thresholds $\Delta = 0.3$ and $\Delta' = 0.5$. For both "Bounded Greedy" variants, α is set at 0.5.
- Retrieved plans (the top 4) are adapted to the new problem and the resulting sequences of actions are run in the game. At the end of each such game (after all enemy units have been destroyed), the values of two metrics are recorded: *number of game cycles* (as recorded by Wargus) and *score* (consisting of the difference between the player's score and the opponent's score, as computed by Wargus. A player's score is incremented when an opponent's unit is destroyed).

Table 1. Each case consists of a state and a plan. The table shows summarizations of the states and of the plans, rather than the actual states and plans stored, for the sake of space.

CASES		
	Initial state	**Plan summarization**
1.	8 s, 3 p	AttackMove x 11
2.	8 s, 3 p	Move x 8, Patrol x 8
3.	8 s, 3 p	AttackMove x 6, Move x 2, Patrol x 2
4.	8 s, 3 p	AttackMove x 4, Move x 4, Patrol x 4
5.	3 s, 2 p	AttackMove x 5
6.	3 s, 2 p	Move x 3, Patrol x 3
7.	3 s, 2 p	AttackMove x 2, Move x 1, Patrol x 1
8.	3 s, 2 p	AttackMove x 1, Move x 2, Patrol x 2
9.	4 s, 0 p	AttackMove x 4
10.	4 s, 0 p	Move x 4, Patrol x 4
11.	4 s, 0 p	AttackMove x 3, Move x 1, Patrol x 1
12.	4 s, 0 p	AttackMove x 2, Move x 2, Patrol x 2
13.	5 s, 5 p	AttackMove x 10
14.	5 s, 5 p	Move x 5, Patrol x 5
15.	5 s, 5 p	AttackMove x 3, Move x 2, Patrol x 2
16.	5 s, 5 p	AttackMove x 2, Move x 3, Patrol x 3

5.2 Results

The results are shown in Fig. 3 (the curves show averaged results of multiple game runs: each point in each of the graphs represents the mean of 4 games). Incorporating the plan diversity criterion in the retrieval process leads to the selection of plans which, after being adapted and run in the game environment, generate significantly varied results (both for score and for game cycles). The variation in results for state diverse plans is negligible in comparison and due to the random factor introduced by the tie-breaking mechanism, as well as to the non-deterministic nature of the game (even when running several games with identical initial configuration and strategies, there will be some variation in game duration and final score). Diversity based on these factors is not satisfactory, as its consistency cannot be guaranteed over multiple runs.

Both tested plan-diversity-aware algorithms, “Plan Diversity through Greedy Selection” and “Plan-Diversity Bounded Greedy” (with retrieval sets of size 4) always retrieve sets of plans containing all four types of strategies, whereas, with state diversity, the retrieval of a highly diverse set is highly unlikely and, if occurring, largely due to chance.

In our State Diversity by Similarity Clusters test runs, at least two results from the state-diverse retrieval set were always instances of the same strategy.

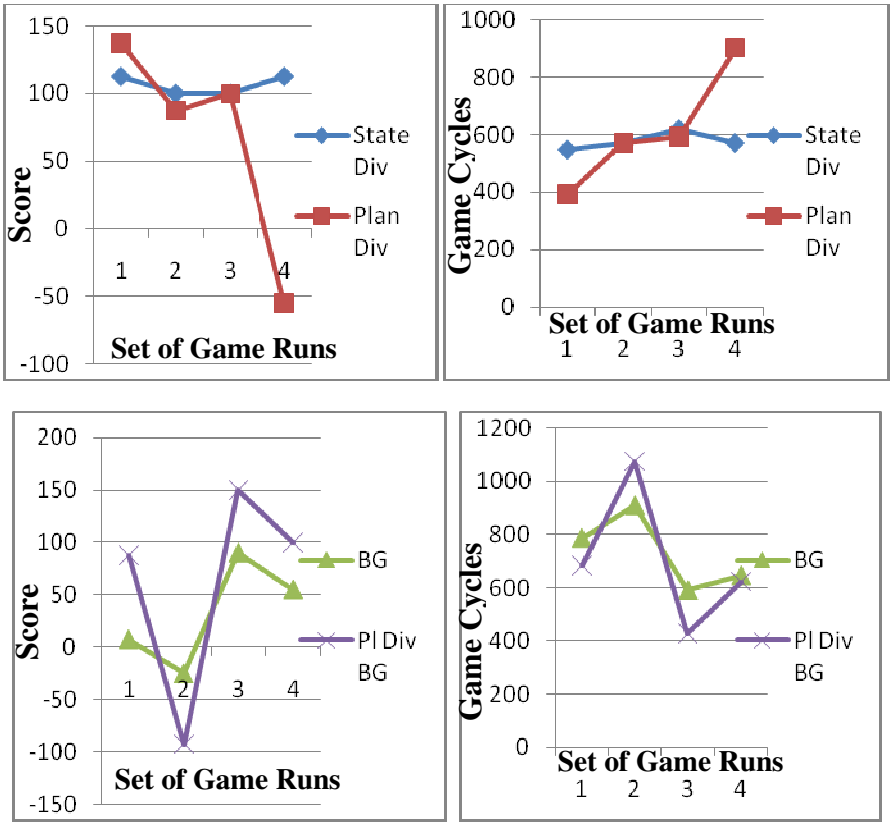


Fig. 3. (a) State Diversity by Similarity Clusters vs. Plan Diversity by Greedy Selection. (b) Bounded Greedy vs. Plan-Diversity Bounded Greedy. Score (left) and game cycles (right) exhibit considerably more variety on adaptations of results based on Plan Diversity. Each point on the x axis represents the average of 4 game runs.

As for the actual results obtained when running these retrieved plans in the game (see Fig. 3), each of the methods based on plan diversity produced results within a larger range than its state-diversity counterpart: the highest and the lowest results, in terms of score, as well as game cycles, were obtained on running plans retrieved with plan-diversity-aware methods.

For the *PDGS* method, the standard deviation was 211.5 for number of game cycles and 84.3 for score (compare with 29.9 for number of game cycles and 7.2 for score in the case of state diversity). In the *PBGA* case, the standard deviation is approximately twice as large as for regular "Bounded Greedy" (271.4 for number of game cycles and 105.9 for score, compared to 142.9 for number of game cycles and 50.8 for score).

We ran the same experiment on an alternative map (shown in Fig. 4) and obtained similar results. The second map is topologically different from the first one in that there are two gaps in the "forest" initially separating the two opponent "camps", offering more passage possibilities for the "enemy army".

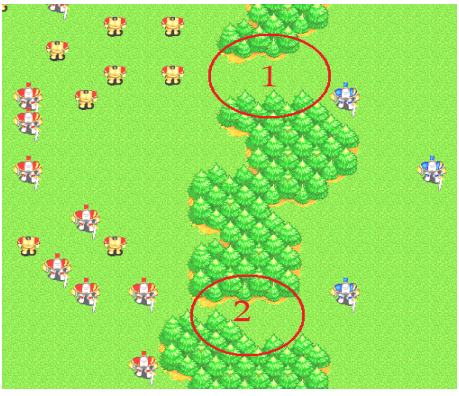


Fig. 4. A second map, topologically-different from that in Fig. 2 (two gaps in the forest between camps), on which we have obtained similar results

6 Related Work

In previous sections, we referred to the use of diversity with recommender systems and adapted a diversity-enhancing technique previously proposed. Recommender systems (with e-commerce as their most common application) [1,2,3,4,5,13] display, to a particular user, frequently in an individualized manner, sets of items from a solution space that would be difficult to navigate without filtering. Conversational recommender systems [3] work over several cycles: they recommend a set of options, obtain user feedback and repeatedly refine their suggestions based on it, producing new sets of recommendations. Feedback may consist of a simple choice between suggested items or of critiquing [14] of particular features of a suggested item. Diversity can be introduced in the recommendation stage, ensuring that the retrieved kNN-set of most similar cases (which satisfy a series of constraints) [13] is also maximally diverse [1]. Various techniques for achieving this have been proposed: Bridge and Kelly [3], for example, incorporate diversity enhancement in collaborative recommender systems [14], which base recommendations not on features of the items themselves but on preferences of neighbors (users assessed as being similar to the user to whom recommendations are being made, based on common item ratings), using "Bounded Greedy" with collaborative-data distance metrics. The trade-off between similarity and diversity is an important consideration in choosing a diversity-enhancement technique [2]. McGinty and Smyth [1] use a method called "adaptive selection" to combine similarity and diversity criteria in accordance with the user's feedback on each consecutive set of recommendations for the same query.

As for related work in planning, a framework for summarizing and comparing HTN plans has been proposed [15]. Maximally different plans are identified using a metric called "plan distance", which is conceptually similar to our notion of "diversity" between two plans (in one of its basic forms, "distance" consists of the sum of the number of features which appear in the first plan, but not the second and the number

of features that appear in the second plan, but not the first, divided by the total number of features). However, the type of plan differentiation proposed is based on high-level, abstracted, semantically-significant features of plans, rather than their low-level, raw components (states and actions). Furthermore, distance-based comparison is being conducted only within pairs of plans, not within sets of multiple plans. Another related work proposes domain-independent methods for generating diverse plans using distance metrics based on actions, intermediary states and causal links between actions [16]. These reflect similar concerns to those behind our "plan" and "state" diversity. The main difference between all the aforementioned previous work and our own is that the former is knowledge-complete, requiring that complete planning domain knowledge is provided, allowing plan generation from scratch, whereas our work is more in line with the knowledge-light CBR approach; we are adapting retrieved plans, rather than generating new ones from scratch. In fact, in our current Wargus framework, we do not have complete knowledge for plan generation; only the cases, the state and plan similarity metrics, and the adaptation algorithm are known.

The concept of plan distance has also been employed for conducting a comparative evaluation of replanning and plan repair [17]. These two methods adapt to unexpected occurrences during plan execution (by constructing a new plan or adapting the existing one to the new conditions, respectively). A new or adjusted plan produced by either method should be "stable", that is, depart from the original plan only insofar as it is necessary in order to successfully adapt to the new conditions. The smaller the distance between a new plan and the original one, the greater the new plan's stability. Distance is computed as the sum of the number of actions that appear in the first plan, but not the second and the actions that appear in the second plan, but not the first. The work reported in [17] is addressing a problem that seems almost the reverse of our similarity/diversity trade-off. They apply their method in a context in which the difference between plans is desirable only within a very narrow set of parameters (as dictated by the new goals or unexpected execution circumstances) and should, otherwise, be kept to a minimum. We, on the other hand, consider diversity to be intrinsically desirable and explore methods for maximizing it, while also maintaining similarity. In addition, the work of Fox et al. is knowledge-complete, as defined in the previous paragraph.

7 Conclusions

We demonstrate how the concept of diversity, as previously explored in the field of recommender systems, can be successfully adapted to help increase diversity within the sets of retrieved plans in case-based planning. To this end, we have formally defined two diversity metrics ("plan" and "state" diversity) and incorporated them into a series of methods for attaining diverse plan retrieval, which we evaluated comparatively. Our experiments show that methods based on "plan diversity", balanced with initial and goal state similarity, retrieve plans that are discernibly varied in the results they produce once they are executed, therefore representing genuine alternatives.

For future work, we intend to explore how the capability of the plan adaptation algorithm influences the diversity of the resulting cases retrieved; if the adaptation algorithm is very powerful, it is conceivable that it could be used to obtain a variety of solution plans adapted from the retrieved plan. On the other hand, these adapted plans might be too far from the query provided by the user. Hence, we would like to explore how the retrieval-centered mechanism developed in this paper would fare versus an adaptation-centered mechanism such as the one reported in [18].

References

1. McGinty, L., Smyth, B.: On the Role of Diversity in Conversational Recommender Systems. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 276–290. Springer, Heidelberg (2003)
2. Smyth, B., McClave, P.: Similarity v's Diversity. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 347–361. Springer, Heidelberg (2001)
3. Bridge, D., Kelly, J.P.: Ways of Computing Diverse Collaborative Recommendations. In: Wade, V., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 41–50. Springer, Heidelberg (2006)
4. McSherry, D.: Increasing Recommendation Diversity Without Loss of Similarity. In: Proceedings of the Sixth UK Workshop on Case-Based Reasoning, Cambridge, UK, pp. 23–31 (2001)
5. McSherry, D.: Diversity-Conscious Retrieval. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 9–29. Springer, Heidelberg (2006)
6. Cox, M.T., Muñoz-Avila, H., Bergmann, R.: Case-based Planning. The Knowledge Engineering Review, 1–4 (2005)
7. Muñoz-Avila, H., Cox, M.T.: Case-Based Plan Adaptation: An Analysis and Review. IEEE Intelligent Systems 23(4), 75–81 (2008)
8. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: On-line Case-Based Planning. Computational Intelligence Journal 26(1), 84–119 (2010)
9. Spalazzi, L.: A Survey on Case-Based Planning. Artificial Intelligence Review 16, 3–36 (2001)
10. Cox, M.T., Veloso, M.M.: Supporting Combined Human and Machine Planning: An Interface for Planning by Analogical Reasoning. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 531–540. Springer, Heidelberg (1997)
11. Muñoz-Avila, H., McFarlane, D., Aha, D.W., Ballas, J., Breslow, L.A., Nau, D.: Using Guidelines to Constrain Interactive Case-Based HTN Planning. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 288–302. Springer, Heidelberg (1999)
12. Aha, D.W., Molineaux, M., Sukthankar, G.: Case-based Reasoning for Transfer Learning. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 29–44. Springer, Heidelberg (2009)
13. McSherry, D.: Completeness Criteria for Retrieval in Recommender Systems. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 9–29. Springer, Heidelberg (2006)
14. Burke, R.: Interactive Critiquing for Catalog Navigation in E-Commerce. Artificial Intelligence Review 18(3-4), 245–267 (2002)

15. Myers, K.L.: Metatheoretic Plan Summarization and Comparison. In: Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006), pp. 182–192. AAAI Press, Menlo Park (2006)
16. Srivastava, B., Kambhampati, S.: T. Nguyen, M. Do, B., Gerevini, A.: Domain independent approaches for finding diverse plans. In: Proceedings of International Joint Conference on AI (IJCAI 2007), pp. 2016–2022. AAAI Press, Menlo Park (2007)
17. Fox, M., Gerevini, A., Long, D., Serina, I.: Plan Stability: Replanning Versus Plan Repair. In: Proceedings of International Conference on Automated Planning and Scheduling (ICAPS 2006), pp. 212–221. AAAI Press, Menlo Park (2006)
18. Lee-Urban, S., Muñoz-Avila, H.: Adaptation Versus Retrieval Trade-Off Revisited: an Analysis on Boundary Conditions. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 180–194. Springer, Heidelberg (2009)

Reducing the Memory Footprint of Temporal Difference Learning over Finitely Many States by Using Case-Based Generalization

Matt Dilts and Héctor Muñoz-Avila

Department of Computer Science and Engineering, 19 Memorial Drive West,
Lehigh University, Bethlehem, PA 18015
{mjd309,hem4}@lehigh.edu

Abstract. In this paper we present an approach for reducing the memory footprint requirement of temporal difference methods in which the set of states is finite. We use case-based generalization to group the states visited during the reinforcement learning process. We follow a lazy learning approach; cases are grouped in the order in which they are visited. Any new state visited is assigned to an existing entry in the Q-table provided that a similar state has been visited before. Otherwise a new entry is added to the Q-table. We performed experiments on a turn-based game where actions have non-deterministic effects and might have long term repercussions on the outcome of the game. The main conclusion from our experiments is that by using case-based generalization, the size of the Q-table can be substantially reduced while maintaining the quality of the RL estimates.

Keywords: reinforcement learning, case similarity, case-based generalization.

1 Introduction

Over the years there has been a substantial interest in combining case-based reasoning (CBR) and reinforcement learning (RL). The potential for integrating these two techniques has been demonstrated in a variety of domains including digital games [1] and robotics [2]. For the most part the integration has been aimed at exploiting synergies between RL and CBR that result in performance that is better than each individually (e.g., [3]) or to enhance the performance of the CBR system (e.g., [4]). Although researchers have pointed out that CBR could help to enhance RL processes [5], comparatively little research has been done in this direction, and the bulk of it has concentrated on tasks with continuous states [6,7,16,17].

In reinforcement learning [8], an agent interacts with its environment in a cyclic pattern. The agent first perceives its state and selects an action to execute. The environment updates the state to reflect changes caused by the agent's action and potentially other actors, and provides the agent with a numerical reward. The reinforcement learning problem is to develop a policy (a mapping from each state to an action that should be taken in that state) that will maximize the sum of rewards the agent will receive in the future.

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 81–95, 2010.

In this paper we use CBR to address a limitation of temporal difference learning (TD learning), a **widely** used form of reinforcement learning [8]. One of the reasons why TD learning has achieved such a widespread use is because it allows the agent to act based on experience in the same episode from when it was learned. This characteristic of TD learning allows it to frequently converge rapidly to an optimal policy faster than Monte Carlo or Dynamic Programming methods [8].

Most implementations of TD learning maintain a Q-table, which is a mapping of the form:

Q-table: States × Actions → Values

That is, the Q-table associates with each state-action pair a value *v*, which represents the expected value of taking the corresponding action in the corresponding state. When an agent takes an action *a* while in an state *s*, the value of the corresponding entry in the Q-table *(s,a)* is updated according to the reward from executing *a*. A drawback of TD learning is that the Q-table can grow very large. For this reason people have suggested generalization methods that reduce the size of the Q-table. For example, neural networks have been used to allow the generalization of states across multiple Backgammon games [10].

In this paper we explore using case-based similarity metrics to reduce the size of the Q-tables when the set of possible states that the agent can visit is finite. In a nutshell, the basic idea is to use a similarity relation $SIM_{state}(s_1,s_2)$ that holds if s_1 and s_2 are very close. Instead of maintaining one entry in the Q-table for each state, the agent maintains one entry for each group of states that are similar enough according to SIM_{state}. Clearly, this will reduce the size of the Q-table. However, this might affect the performance of the TD learning process, possibly reducing the speed of convergence to an optimal policy or making this convergence impossible.

We hypothesize that case-based generalization can attain the reduction of the Q-table while still maintaining the performance of the TD learning process, and potentially even improving it as a result of the reduction in the space of possibilities that the TD learning algorithm must consider. We tested this hypothesis by performing experiments on a gaming testbed. Our experiments confirm our hypothesis pointing towards the potential of case-based similarity to generalize Q-tables while still achieving good performance.

The paper continues as follows: the next section describes our gaming testbed. Section 3 provides a brief overview of TD learning. Then we describe the case-based generalization of Q-tables in Section 4. Then we describe the empirical evaluation. Section 6 describes related work and Section 7 makes some final remarks.

2 Motivation Domain: The Descent Game

We performed experiments on our implementation of Descent, a tabletop, turn-based game where actions have non-deterministic effects that might have long term repercussions on the outcome of the game. This is the kind of game where one would expect temporal difference learning to perform well since episodes last long and, hence, the learning process could take advantage of using the estimates of the Q-values in the same episodes in which they occur.

Descent is our implementation of a tabletop game *Descent: Journeys in the Dark®* in which one to four players control four hero characters cooperating to defeat the overlord, which is controlled by another player [11]. Unlike games like Dungeons & Dragons® where the goal is for the players and the dungeon master to combine efforts to tell a riveting story, in this game the overlord's goal is purely to annihilate the heroes and, as such, he has a fully fleshed-out rule set in this game just like the heroes. Descent is a highly tactical, turn based, perfect information game (each player sees the complete board), and has non-deterministic actions (i.e., actions performed by a player might have multiple outcomes such as the attack action may or may not hit a monster). The entire game is set in a fantasy setting with heroes, monsters, treasures, dragons, and the like. We implemented a digital version of Descent that uses a subset of the original set of rules of the tabletop version, yet is self-contained (i.e., a complete game can be played without referring to rules not implemented).

The goal of the game is for the heroes to defeat the last boss, ***Narthak***, in the dungeon while accumulating as many points as possible. The heroes gain 1200 points for killing a monster, lose 170 points for taking a point of damage, gain 170 points for removing a point of damage, and lose 850 points the hero's for dying. When a hero dies, he respawns at the start of the map with full health. Furthermore, the heroes lose 15 points per turn. This form of point entropy encourages players to finish the game as quickly as possible.

We hard-coded a competent version of the overlord and developed an API that allows an AI agent to control the hero characters, taking the place of the human hero player. This AI agent sends messages to the game server while receiving and evaluating incoming messages from the game server.

Each hero has a number of hit points called wounds, a weapon, armor, a conquest value, a movement speed, 1 special hero ability, and 3 skills (ranging from additional special abilities to basic additional stats). Heroes may move in any direction including diagonals by spending 1 movement point. They may move through their own allies, but may not move through monsters. It takes 2 movement points to open or close a door. Heroes may not move through obstacles (such as the rubble spaces that adorn the map). This means that the AI agent must make a complex decision considering multiple factors: whether to move and if so in what direction, whether it should move forwards and risk attack from a monster or wait for other players (which is always detrimental because of the loss of health per turn). To simplify the AI choices, in our implementation, every turn the heroes can take one of three actions: battle, advance, or run, each of which grants the heroes a different number of attacks and movement points. If the hero declares an advance or battle, it will move closer to the nearest monster and attack whenever possible. If the hero declares a run, it will retreat towards the start of the map.

After all of the heroes have taken their turn, the overlord's turn begins. The current hardcoded overlord AI is set to have each monster pick a random hero on the map and move towards that hero and attack the hero if he is within melee range. Monsters also have special abilities, move speeds (with same restrictions as the heroes), specific attack dice, armor values, and health values.

3 TD Learning

TD learning is a widely used form of reinforcement learning wherein an agent learns a policy which, for every state of the agent's world, maps an estimate of the value of taking each applicable action in that state; the goal of the agent is to maximize the sum of the future rewards it receives.

3.1 Q-Tables and Policies

TD learning algorithms maintain a ***Q-table*** of expected rewards for each state-action pair. A Q-table stores a value for each state-action *(s,a)* pair (*Q(s, a)* → *value*), where the table in this case has game states as row labels, and abstract game action names as column labels. Each entry in the Q-table is called a ***Q-value***.

Given a Q-table, a ***policy*** can be inferred by greedily selecting for each state the action with the highest Q-value. This is called a greedy policy, Π_{greedy}, and is defined as:

$$\Pi_{greedy}(s) = \arg\max_a Q(s,a)$$

3.2 TD Learning Updates

TD learning algorithms balance between exploiting the greedy policy from the current Q-table and exploring other alternative actions even when they do not correspond to the greedy policy. Exploration is done to avoid local minima in which the Q-values converge towards selecting an action *a* for a state *s* even though there is another action *a'* that over the long run will result in a higher Q-value for *s*. An strategy, called ***∈-greedy***, for balancing exploitation and exploration in TD learning is selecting the greedy action, $\Pi_{greedy}(s)$, for state *s* with probability 1-∈, where ∈ is an input parameter in the range [0,1]. This parameter is usually set lower than 0.5 so that most of the time the greedy action for state *s* is selected. With probability ∈ a random selection is made among the set of actions that can be applied in state s.

An alternative to ∈-greedy is called ***softmax*** [8], whereby the probability of selecting an action a for state s is relative to its value Q(s.a). Hence, actions with high Q-values will be more likely to be selected while actions with low Q-values, including those that have a Q-value of 0, will still have a non-zero probability of been selected. The agents we use in our experiments perform a softmax selection.

Regardless of how the action is selected, TD learning uses bootstrapping, in which the agent updates the Q-values based on its own estimates of the Q-value. The following formula is used to update the Q-value Q(s,a) for the action *a* selected in state *s*:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R + \gamma Q(s',a') - Q(s,a)) \quad (1)$$

Here R is the reward obtained after taking action a in state s, and α is the step-size parameter, which determines the extent of the update done to the Q-value; lower values will reduce the extent of the update while larger values will increase it. The value of γ, which is called the discount rate parameter, adjusts the relative influences of current and future rewards in the decision making process. The state s' is the state that was reached after taking action a in state s, and a' is the action that was taken after reaching state s'. Thus, the value of Q(s,a) is updated by looking one step ahead into the estimate of the subsequent state and action pair that the agent visited.

3.3 TD Learning to Control Descent Agents

One of the main challenges of using TD learning for controlling Descent agents is the large number of potential states. It would be impossible to generate and populate a full state table with values given the amount of time it takes to run a single game of Descent. For example, if we assume we would need a different state for each possible monster and hero positioning and a different state for each combination of hero and monster health amounts, given a 26×26 map, 16 monsters, and 4 heroes (and ig noring heroes' health), we would need 4.0×10^{56} states. Because of this, state abstractions are needed to lower the number of possible states; this is a common practice when using reinforcement learning in games [12].

Each state is represented by the following abstraction: the hero's distance to the nearest monster, the number of monsters within 10 (moveable) squares of the hero, the estimated damage those monsters would inflict if they were to all attack the hero, and the hero's current health. In general, the distance to the nearest monster is no more than 20 movable squares. The number of monsters within range is usually no more than 6, the estimated damage taken is typically no more than 18, and the most health any hero has is 12. This reduces our 55 million states problem down to 6500 for each hero. While the reduction is substantial, heroes will visit only dozens of states in an average game. Hence, some form of state generalization is needed.

4 Case-Based Generalization of Q-tables

Frequently, the Q-tables are pre-generated and reside in memory. That is, the agent allocates a memory footprint of the order of O(|S|×|A|), where S is the set of possible states that the agent can visit and A is the set of possible actions that the agent can take.[1] Borrowing ideas from CBR, rather than generating a large table and filling it in with exploration and exploitation choices, what we propose instead is to begin with a blank Q-table and slowly fill it in with new cases, which we view as entries in the Q-table, as the agent encounters them. Furthermore, we propose using a case similarity function to encompass many possible different entries in the Q-table. For example, standing near a monster with 5 health is not much different than standing near a monster with 4 health, so the agent will consider those two to be essentially the same state when generating and using the Q-table. Consider a 2-dimensional map where each point on the map represents a state. Initially there is a completely empty Q-table and the map is not covered at all. When the agent visits the first state, a new entry is made to the Q-table. The state can be thought to cover an area in the map as shown in Figure 1 (left); as usual, the point in the middle of the circle represents the state the agent is currently in and the circle around that point represents the similarity function's coverage of similar states. After visiting 5 different states, the map could be covered as shown in Figure 1 (right).

[1] Actions do not need to be applicable in every state; if an action a is not applicable in an state s, its corresponding Q-Value, Q(s,a), can be initialized with a special value, such as -1, to represent this fact.

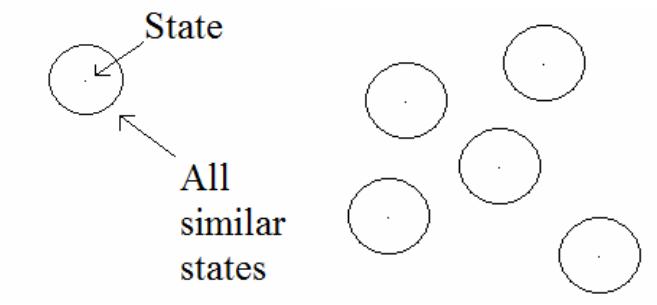


Fig. 1. Graphic depiction of case base coverage

Each circle in Figure 1 (right) represents all states which are close enough to the state s_{first} first visited. Hence when visiting any state s_{new} that is similar to s_{first}, the agent does not need to add a new entry in the Q-table. Instead, s_{first} acts as a proxy for s_{new}. This has the following two consequences:

- For selecting which action to choose from state s_{new}, we do a softmax selection based on the Q-values for the actions in s_{first}, which will result in the selection of an action a.
- For doing the update of the Q-values, the agent updates the entry for $Q(s_{first},a)$ as indicated in Formula 1.

In other words s_{new} and s_{first} are considered to be the same state for the purpose of determining our policy and for the purpose of updating the Q-table. Overlap in the table is guaranteed since similarity does not take action choice into effect. So there will be multiple different state similarity blocks that use different actions. Furthermore, it is possible to generate a state near an already existing state, causing overlap. When overlap occurs the agent is essentially considered to be in the same "state" just with multiple different action choices. Below we present the algorithm, SIM-TD, that takes into account the notion of case-based similarity into the standard temporal difference algorithm. It initializes the Q-table Q with an empty table and runs n episodes, each of which calls the procedure $\text{SIM-TD}_{episode}$, which updates Q.

SIM-TD(α, γ, n)
Input: α: step-size parameter, γ: discount factor, n: number of episodes
Output: Q: the Q-table

Q ← [] // the Q-table is initially empty; no memory allocated for it
S ← [] //current list of states represented in Q
k ← 1
while (k ≤ n) **do**
    Q ← $\text{SIM-TD}_{episode}$(α, γ, Q, S)
    k ← k + 1
end-while
return Q

The procedure SIM-$TD_{episode}$ is shown below. The crucial difference with standard temporal difference occurs at the beginning of each iteration of the while loop. Namely, it checks if there is a state s' similar to the most recently visited state s_{new}. In such a case, s' is used for the selection of the next action and for the TD update of the Q-table Q. If no such a similar state s' exists, then a new entry for state s_{new} is added to the table.

SIM-$TD_{episode}$(α, γ, Q, S)
Input: α: step-size parameter, γ: discount factor, Q: the current Q-table, S: states
Output: Q: the updated Q-table

```
start-episode(G)   //for our experiment G will be one run of the Descent game
s ← null; a ← null;
s_new ← initialState(G)
while not(end-of-episode(G)) do
     s' ← similarState(S, s_new)  //finds a state s' in S similar to s_new
     if (s' = null) then        // no such an s' exists currently in S
          s' ← s_new
          S ←S ∪ {s'}
          make-entry(Q,s')  // creates a new row in Q for state s' and
                            // Q(s',a) is initialized randomly for each action a
     end-if
     a' ← softmax-action-selection(Q,s')
     if (a ≠ null and s ≠ null) then //avoids doing the update in the first iteration
          Q(s,a) ← Q(s,a) + α(R + γQ(s',a') – Q(s,a))   // Same as Formula (1)
     end-if
     a ← a'
     s ← s'
    (R, s_new) ← take-action(a',G) // reward R obtained and the state s_new visited after
                                   // executing action a'
end-while
return Q
```

While we do expect that using case-based generalization will reduce the memory footprint of temporal difference, there is a potential danger: that precision will be lost; this is a common difficulty with generalization techniques. In our context this could result in updates made to wrong entries of the Q-table (e.g., when two conceptually different states are combined into the same entry in the Q-table). This could have a negative effect in the performance of the agent that is using the Q-table. For example, the updates might pull the agent in opposing choices for some crucial state, making it incapable of converging to an optimal policy or even learning a "good" policy. In the next section we present some experiments we performed evaluating both the reduction in memory requirements for temporal difference and the effect of the generalization on the performance of an agent using these case-based generalization techniques.

5 Experimental Evaluation

We performed an experiment to evaluate the effectiveness of our similarity based approach to temporal difference. Specifically we wanted to validate the following hypothesis: the size of the Q-table for the similarity-based temporal difference as implemented in SIM-TD is reduced compared to the size of the Q-table needed for standard temporal difference while still preserving comparative levels of performance. We simulate the standard temporal difference by using SIM-TD with the identity similarity (indicating that two objects are similar only if they are the same). Hence, every new state that is visited will create a new entry in the Q-table.

5.1 Performance Metric

The performance metric is the score of the game is computed formulas follows:

$$\text{Score} = \omega_k * \text{kills} + \omega_h * \text{health-gain} - \omega_d * \text{deaths} - \omega_h * \text{health-lost} - \omega_L * \text{length}$$

Kills refers to the number of monsters killed by the heroes, health-gain is the health that the heroes gain (which can only be gained when the hero performs a run action, in which case they gain roughly 30% of their missing health back), deaths is the number of heroes' deaths (every time a hero dies, he respawns at the starting location), health lost by the heroes during the game and length, which indicates the length of the game (i.e., measured as the number of turns; each turn includes each of the 4 heroes' movements plus the overlord). We ran the experiments on two maps, a small one and a large one. The ranges of these attributes, for each map, are shown in Table 1. The attributes health-gain and health-loss are map independent. The asterisk in front of the ranges indicates that the ranges are unbounded to the right. For example, heroes can die any number of times. Health gain/loss range is 0-12* because each hero has a maximum of 12 health. However, a hero might lose/gain a lot of health. For example, a single hero might lose 60 health in one game because he would lose 12 health, die, lose 12 more health, die again, and so forth. The range is shown for illustration purposes. The ranges for hero's death are per kill; certain heroes are worth more negative points than others upon death.

Table 1. Attributes contributing to scoring formula

Attribute	Range Small map	Range large map	Points earned
Kills	0 to 9	0 to 23	1,200 (per kill)
Health-gain	0 to 12*	0 to 12	170 (per point)
Deaths	0 to 4*	0 to 4*	-1700 to -3400
Health-lost	0 to 12*	0 to 12	-170 (per point)
Length	0 to 15*	0 to 25	-60 (per turn)

5.2 Similarity Metric

We define a similarity relation that receives as input a case's state C and the current state S and returns a Boolean value indicating if C and S are similar or not. Each hero

maintains its own case base to account for the different classes of heroes. States are defined as 4-tuples: (distance to monster, monsters in range, expected damage, health). Distance to monster refers to the Manhattan distance to the nearest monster from the hero's position, monster in range indicate the total number of monsters that can be reached by the hero within one turn (different heroes might have different movement ranges), expected damage is computed based on the maximum damage that the hero can take from the monsters that can reach him within one turn; if no monster is within range the value is set to 0. For the small map, there can be at most 9 monsters in range (13 for the large map) and on average these monsters will do 31 damage (41 for large map). Health is the current health of the hero. This information is sufficient to determine which action to apply. The solution part of the case is the action that the hero must take, which, as detailed in Section 2, is battle, advance, or run.

Table 2 shows the 3 similarity relations we used in our experiments: major similarity, which allows more pairs of states to be similar, minor similarity, which is more restrictive than major similarity, and no similarity, which considers two states to be similar only if they are identical. The rows are for the same attributes indicated in the previous paragraph. They indicate the minimum requirements for two states to be considered similar. Two states are similar if the absolute difference of the attributes is smaller or equal than each of the corresponding entries in the table below. For example, (6,2,5,10) is similar to (3,1,8,5) relative to the major similarity but not relative to the minor similarity. The values in parenthesis in the Major similarity show the ranges for the large and small maps. The current health is independent of map and, hence, only one value is shown. For the minor similarity we consider special values of the attributes that supersede the attribute comparison criteria. For example, if a hero has maximum health, then the case we are comparing against must also have maximum health. We have analogous criteria in place for the other attributes. This makes the minor similarity a much more restrictive criterion than the major similarity.

Table 2. Boundaries for similarity metrics

Attribute	Major similarity	Minor similarity	No similarity
Distance to monster	4 (0-22; 0-29)	4*	0
Monster in range	3 (0-9; 0-13)	3*	0
Expected damage	7 (1-31; 141)	6*	0
Current health	5 (0-12)	4*	0

5.3 Experimental Setup

We ran three variants of SIM-TD: SIM-TD with (1) non similarity (our baseline), (2) minor similarity, and (3) major similarity. We refer as **agents** to any of these three variants, which as explained before, are used to control each hero in the game (i.e., each hero maintains its own Q-table and chooses the actions based on softmax selection of the table). We created two maps. The first map is the original map in the actual Descent board game. The second map is a smaller version of the original map with half the map sawed off.

The two different maps were used to test the different effects of similarity with different scenarios. The smaller map tends to have a much smaller Q-table since the set of situations the heroes can find themselves in is much smaller than with a large map. The large map on the other hand has a much larger set of possible states. For example, the large map has more monsters on it than the small map. Because of this, there is a much wider variance on the Monster in range and Expected damage fields. Also, since the large map is larger it makes sense that the 'closest monster' field could potentially be much larger as well. Using different sets of games can show us different possible results with experimentation. In both maps the boss is located behind a wall from which the boss cannot exit. This is to ensure that the games do not end early by chance because the boss wanders towards the heroes. Since the hardcoded section of the hero AI always attacks the nearest monster and the monster AI is always to run straight for the hero, it is impossible for the hero to kill the last boss before any other monster ensuring that a game does not end abruptly by chance.

For both the small and the large maps, trials of games were run until within each trial the games were fluctuating around a certain score. For the small maps score fluctuated around eight thousand points after 8 games. For the large maps score fluctuated around 9 thousand points after 4 games. Because of this, we ran trials of 8 games each for the small map and 4 games for the large map. The large maps also took a much larger amount of time to run than the small maps. Running each experiment took almost an entire day with a human operator starting each game. Also, while a single trial had multiple games in it to observe the effect of the score increasing over time, with multiple games, multiple trials needed to be run to obtain a reliable estimate of the average score over time. For each set of games, we ran a set of five trials. This was largely a time constraint decision. The game's scoring system tends to fluctuate a lot since combat has a random factor influencing the outcome and other factors such as early decision by a hero to explore instead of attack.

5.4 Results

Figure 2 shows in the y-axis the average number of entries in the Q-table per trial. This table shows the expected effect in regard to the size of the Q-table. Using major similarity, the Q-table had a much smaller number of entries in the end; for a total of about 100 entries (or 25 per hero). For minor similarity, about twice as many were seen, about 225. And for no similarity, about twice as many again were seen, in the 425 range. This shows that case similarity can reduce the size of a Q-table significantly over the course of several games. The no similarity agent used almost five times as many cases as the major similarity agent. The small difference between the number of cases captured in the smaller and in the larger map for each type of similarity is due to the state abstraction explained in Section 3.3, which makes the number of states relatively independent of the size of the map.

Figures 3 and 4 show the scores at the end of each game. Overall with either the major or minor similarity it had a better performance than without similarity on both maps, aside from the game # 3 in the large map, where major similarity performed worst. But in general, the agent performed better with some form of similarity. Even during the first game, the agent managed to learn some strategies that performed better than the other two agents. The anomaly at game #3 can be explained by the

multiple random factors which results in a lot of variation in the score. This randomness is observable even with five game trials. The smaller map had many more trials, so there is less variation. We can draw the same conclusions as in the large map. Again the major similarity agent was better than the other two, occasionally dipping below the minor similarity agent. The no similarity agent performed worse than the other two similarity agents. Even with the fluctuations in the graph, it never surpassed either of the similarity agents past the first game. This shows once again that the notion of similarity helped to make the reinforcement learning agents learn a better solution much faster than without similarity. However, again the Major Similarity Agent was competitive and beat out the Minor Similarity agent at the start and did roughly about as well towards the end.

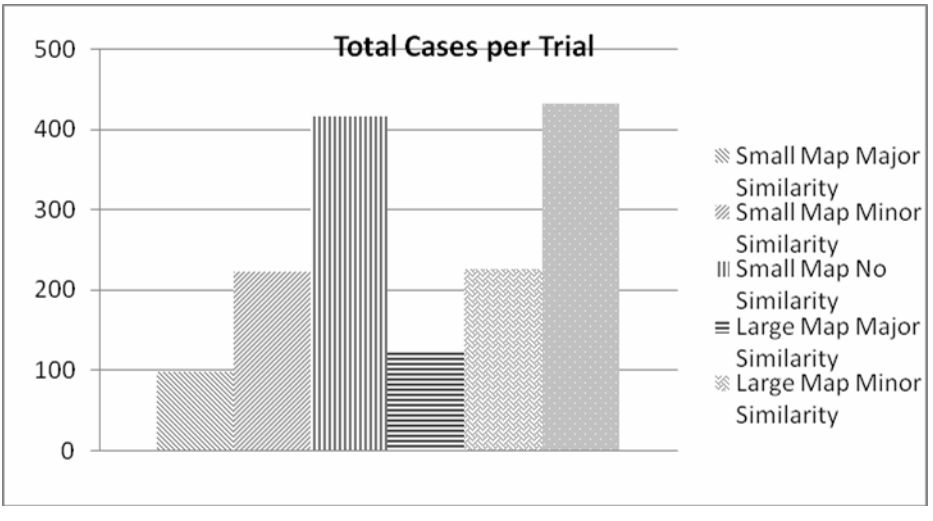


Fig. 2. Total (average) number of cases for small and large maps

We believe that the reason why there is a better performance with some form of generalization than without any generalization is a reflection of the particular case based generalization used working well in this particular domain. Thus, whereas in the non-generalized situation a state s must be visited ns times before it is able to find a good approximation to the value of its actions, in the generalized situation any visit to a similar but not necessarily identical state s' will update the value of the actions. Therefore it will be able to find good values faster.

We performed statistical significance tests with the Student's t-test on the score results obtained. The difference between minor and no-similarity is significant for both maps. The difference between the major and the no-similarity is significant for the small map but not so for the large map (the t-test score was 93.9%). The difference between the major and the minor similarities was significant for the small map but not significant for the large map. The main conclusion from this study is that by using case-based generalization, the size of the Q-table can be substantially reduced while still maintaining at least as good as the performance without case-based generalization, and can even become significantly better.

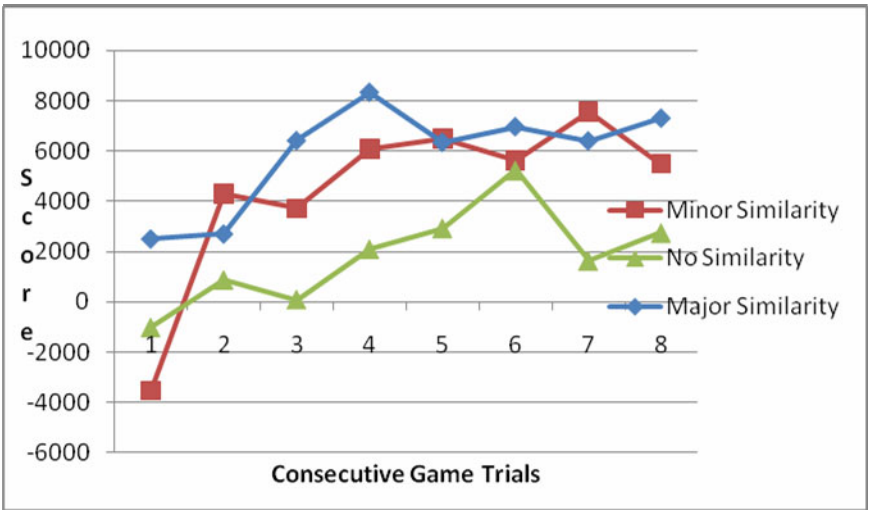


Fig. 3. Average scores after each game for small map

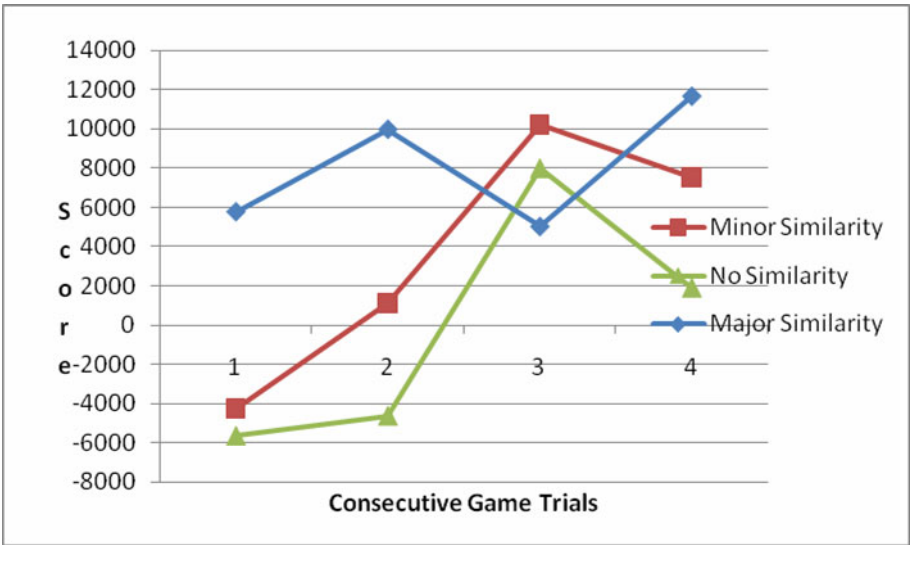


Fig. 4. Average scores after each game for large map

6 Related Work

We also explored using other reinforcement learning methods such as dynamic programming and Monte Carlo methods. It is feasible that case-based generalization could have similar positive effects to those we demonstrated for Temporal Difference.

However, for this particular testbed both were unfeasible to use. Dynamic programming requires that the agent knows the transition probabilities for the actions to be chosen and the expected rewards from those actions. This would require running extensive games to obtain these values. Monte Carlo methods perform the function approximation updates after the episodes ends. This will likely require it to play many more games before it learns capable policies. Games in our testbed are fairly long lasting 15 minutes on the short map and 25 on the larger one. One run of the experiment was lasting one day. Under our time constraints it was not feasible for us to run such experiments.

Researchers have investigated other approaches for reducing the memory footprint requirements of reinforcement learning. TD-Backgammon used neural networks for this purpose [10]. Clustering algorithms have been proposed to group states that are clustered together [13,19]. This requires the system to either know beforehand all states that can be visited or wait until a large sample of states have been visited. In contrast our approach is grouping states as they are visited, which is the classical lazy learning approach in CBR. However, similar to work integrating lazy and non lazy learning approaches [14], one could use our CBR approach until enough states have been visited and at that point run a clustering algorithm. Other works combine gradient-descent methods with RL [9]. Instance-based learning has been used to reduce the number of states needed and showcases with continuous states [18]. The crucial observation here is that the agent does not know the state granularity apriori. Instance-based learning methods allow the agent to refine the granularity as needed. These ideas have been studied in the context of case-based reasoning systems in [16], which also surveys instance-based and case-based reasoning approaches for continuous tasks. As per this survey, our work can be classified as a coarse-coded (since one entry in the table represent multiple states), case-based (since it maintains the Q-value for all actions in that state as a row in the Q-table) function approximation approach.

There has been a large interest in combining CBR and RL over the last years. These include Derek Bridge's ICCBR-05 invited talk where he described potential synergies between CBR and RL [5], the SINS system which performs problem solving in continuous environments by combining case-based reasoning and RL [15], and CBRetaliate, which stores and retrieves Q-tables [3]. Most of these works pursue to improve the performance of an agent by exploiting synergies between CBR and RL or enhance the CBR process by using RL (e.g., using RL to improve the similarity metrics). In contrast, in our work we are using CBR principles to address a well-known limitation of reinforcement learning. Bianchi et al. uses cases as a heuristic to speedup the RL process [7] and Gabel and Riedmiller uses cases to approximate state value functions in continuous spaces [6,17].

7 Conclusions

In this paper we presented an approach for reducing the memory footprint requirement of temporal difference learning when the agent can visit a finite number of states. We use case-based similarity to group the states visited during the reinforcement learning process. We follow a lazy learning approach: cases are grouped in the order in which they are visited. Any new state visited is assigned to an existing entry

in the Q-table provided that a similar state has been visited before. Otherwise a new entry is added to the Q-table. We performed experiments on our implementation of Descent, a turn-based game where actions have non-deterministic effects and might have long term repercussions on the outcome of the game. This is the kind of game where one would expect temporal difference learning to perform well since episodes last long and, hence, the learning process could take advantage of using the estimates of the Q-values in the same episodes in which they occur. The main conclusion from this study is that by using case-based generalization, the size of the Q-table can be substantially reduced while improving the performance compared to without case-based generalization.

As discussed in the related work section, there are a number of closely related works in the literature, CBR-based and otherwise, to tackle RL's memory footprint problem. We used a simple similarity-based approach to tackle this problem and obtained significant gains in the context of a relatively complex game. It is conceivable that the use of recent advances in CBR research, such as case-based maintenance (e.g., [20]), can be used to formulate a robust CBR solution to this problem that can be demonstrated across a wider range of applications domains. It is worthwhile to point out that, as of today, there is no application of RL to a modern commercial game unlike other AI techniques such as induction of decision trees [21] and AI planning [22]. We speculate that part of the reason is the lack of robust generalization techniques for RL that allow rapid convergence towards good policies.

There is a difficulty with our approach that we will like to discuss. As we explained before, when visiting a state s_{new} the agent first checks if there is an entry in the Q-table for a similar state s_{first}. In such a situation, the action *a* to take is selected based on the Q-values for s_{first}. It is possible that the action *a* selected might not be applicable in s_{new}. This situation does not occur with the Descent agents because all actions are applicable in all states. One way to address this is to check if s_{new} and s_{first} have the same applicable actions and if not then make them dissimilar, so that each will have its own entry in the Q-table.

Acknowledgements. This work was supported in part by NSF grant 0642882.

References

1. Sharma, M., Holmes, M., Santamaria, J.C., Irani Jr., A., Ram, A.: Transfer learning in real-time strategy games using hybrid CBR/RL. In: Proceedings of the 20th Int. Joint Conf. on AI (IJCAI 2007), pp. 1041–1046. AAAI Press, Menlo Park (2007)
2. Karol, A., Nebel, B., Stanton, C., Williams, M.A.: Case based game play in the robocup four-legged league part I the theoretical model. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 739–747. Springer, Heidelberg (2004)
3. Auslander, B., Lee-Urban, S., Hogg, C., Munoz-Avila, H.: Recognizing The Enemy: Combining Reinforcement Learning with Strategy Selection using Case-Based Reasoning. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 59–73. Springer, Heidelberg (2008)
4. Juell, P., Paulson, P.: Using reinforcement learning for similarity assessment in case-based systems. IEEE Intelligent Systems, 60–67 (2003)

5. Bridge, D.: The virtue of reward: Performance, reinforcement and discovery in case-based reasoning. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, p. 1. Springer, Heidelberg (2005)
6. Gabel, T., Riedmiller, M.A.: CBR for state value function approximation in reinforcement learning. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 206–221. Springer, Heidelberg (2005)
7. Bianchi, R., Ros, R., Lopez de Mantaras, R.: Improving Reinforcement Learning by using Case-Based Heuristics. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 75–89. Springer, Heidelberg (2009)
8. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
9. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine Learning, 9–44 (1988)
10. Tesauro, G.: Temporal difference learning and TD-Gammon. Communications of the ACM 38(3), 58–68 (1995)
11. http://en.wikipedia.org/wiki/Descent:_Journeys_in_the_Dark (Last checked: February 2010)
12. Vasta, M., Lee-Urban, S., Munoz-Avila, H.: RETALIATE: Learning Winning Policies in First-Person Shooter Games. In: Proceedings of the Innovative Applications of Artificial Intelligence Conference, pp. 1801–1806. AAAI Press, Menlo Park (2007)
13. Fernández, F., Borrajo, D.: VQQL. Applying vector quantization to reinforcement learning. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup 1999. LNCS (LNAI), vol. 1856, pp. 292–303. Springer, Heidelberg (2000)
14. Auriol, E., Wess, S., Manago, M., Althoff, K.-D., Traphöner, R.: INRECA: A Seamlessly Integrated System Based on Induction and Case-Based Reasoning. In: Proceedings of the Int. Conf. on CBR, pp. 371–380. Springer, Heidelberg (1995)
15. Ram, A., Santamaria, J.C.: Continuous case-based reasoning. Artificial Intelligence, 25–77 (1997)
16. Santamaria, J.C., Sutton, R.S., Ram, A.: Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces. Adaptive Behavior, 163–217 (1998)
17. Gabel, T., Riedmiller, M.: An Analysis of Case-Based Value Function Approximation by Approximating State Transition Graphs. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 344–358. Springer, Heidelberg (2007)
18. McCallum, R.A.: Instance-Based State Identification for Reinforcement Learning. In: Advances in Neural Information Processing Systems, NIPS 7 (1995)
19. Molineaux, M., Aha, D.W., Sukthankar, G.: Beating the defense: Using plan recognition to inform learning agents. In: Proceedings of the Twenty-Second International FLAIRS Conference, pp. 257–262. AAAI Press, Menlo Park (2009)
20. Cummins, L., Bridge, D.: Maintenance by a Committee of Experts: The MACE Approach to Case-Base Maintenance. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 120–134. Springer, Heidelberg (2009)
21. Evans, R.: Varieties of Learning. In: AI Game Programming Wisdom, pp. 567–578. Charles River Media, Hingham (2002)
22. Orkin, J.: Applying Goal-Oriented Action Planning to Games. In: AI Game Programming Wisdom, vol. 2, pp. 217–228. Charles River Media, Hingham (2003)

Text Adaptation Using Formal Concept Analysis*

Valmi Dufour-Lussier, Jean Lieber, Emmanuel Nauer, and Yannick Toussaint

LORIA—UMR 7503 (CNRS, INRIA, Nancy-Université)
BP 239, 54506 Vandœuvre-lès-Nancy, France
{valmi.dufour,jean.lieber,emmanuel.nauer,yannick.toussaint}@loria.fr

Abstract. This paper addresses the issue of adapting cases represented by plain text with the help of formal concept analysis and natural language processing technologies. The actual cases represent recipes in which we classify ingredients according to culinary techniques applied to them. The complex nature of linguistic anaphoras in recipe texts make usual text mining techniques inefficient so a stronger approach, using syntactic and dynamic semantic analysis to build a formal representation of a recipe, had to be used. This representation is useful for various applications but, in this paper, we show how one can extract ingredient–action relations from it in order to use formal concept analysis and select an appropriate replacement sequence of culinary actions to use in adapting the recipe text.

Keywords: formal concept analysis, natural language processing, text mining, textual case-based reasoning.

1 Introduction

A case retrieved by a case-based reasoning (CBR) system in order to solve a given problem may need adaptation in order to fit in. Adapting a textual case may be as simple as replacing all the occurrences of a word with another word, but one could want to do better. Contestants in the Computer Cooking Contest[1] (CCC) use case-based reasoning with a recipe book as a case base to propose ingredient substitutions as a solution to cooking problems (adapting a recipe to given constraints) but so far are not making modifications to the recipe text.

This paper shows that using a method based on text mining and machine learning, namely formal concept analysis (FCA), can be of great use for text adaptation. Ingredient preparation *prototypes* are found and used to adapt a recipe. Adapting a recipe by replacing ingredient α with β implies finding out actions performed on α and replacing them with actions performed on β.

* The authors would like to thank the reviewers for their helpful comments, as well as Zeeshan Ahmed for his previous work on a related topic and Jenia Berzak, Le Dieu Thu, and Ripan Hermawan for letting us use their corpus of part-of-speech-tagged recipe texts.

[1] http://vm.liris.cnrs.fr/ccc2010

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 96–110, 2010.

The work was achieved within the TAAABLE project [4,8] and focuses on text adaptation. TAAABLE is a textual case-based cooking system that participated in the first and second (as WIKITAAABLE) CCC. It is built around a case-based inference engine using a (minimal) propositional representation of recipes, a set of known acceptable substitutions, an ontology of ingredients used to build new substitutions on the fly, and a cost function to select the best adaptation for a problem.

In this paper, we shall argue for a more thorough formal representation of recipes, and show how it can be built with natural language processing (NLP) techniques and used with FCA towards a more significant adaptation function.

Presupposing that TAAABLE is able to suggest a recipe from its case base along with some substitution operations that consist in replacing a given ingredient by another given ingredient, our system is able to find an adequate sequence of actions for the new ingredient and modify the recipe text accordingly.

While selecting texts with FCA and reusing them in the adaptation stage of a textual CBR system is to our knowledge a novel approach, it fits within a trend towards the maximal reuse of existing text in providing textual solutions to problems (arguably initiated by [15,14]). Using FCA for information retrieval or CBR in itself is not a totally new idea either (see for instance [7,17] for retrieval, and [10,9] for CBR).

In Sect. 2, we describe the kind of formal representation we expect to create from recipe texts and the process to translate texts into this representation. Then in Sect. 3 we show how FCA is used to adapt recipes, and we detail the algorithms we developed as well as the strategy we used to generate new texts. Finally we discuss our results and future work in Sects. 4 and 5.

2 Linguistic Processing of Recipes

The main idea guiding this work is that some "common uses" of each ingredient, that we call *prototypes,* can be extracted from the case base and used in adapting texts. If for instance we want to substitute *zucchini* with *aubergine* in a recipe, it would be more convenient to prepare the aubergine as done in some aubergine recipes instead of blindly applying the same steps as for preparing the zucchini. A prototype is understood as a sequence of actions applied to an ingredient. To extract it, we need a formal representation of the recipe text. The same linguistic

Easy berry pancakes

Ingredients	Preparation
•Six eggs	Beat the eggs. Add the flour and mix with a fork. Whisk in the milk.Pour batter in warm pan, one ladleful at a time. Add some fruits, then cook for one minute. Flip and cook one minute more.
•2 cups of flour	
•2 cups of milk	
•½ cup of blueberries	
•½ cup of raspberries	

Fig. 1. A sample recipe text

processing that changes a text in its representation is performed on the source recipe (the recipe to be adapted) and on all the recipes of the case base. Those formal representations are then passed on to data mining algorithms to extract the prototypes. The complete process will be illustrated on the recipe in Fig. 1, yielding the representation shown in Fig. 2.

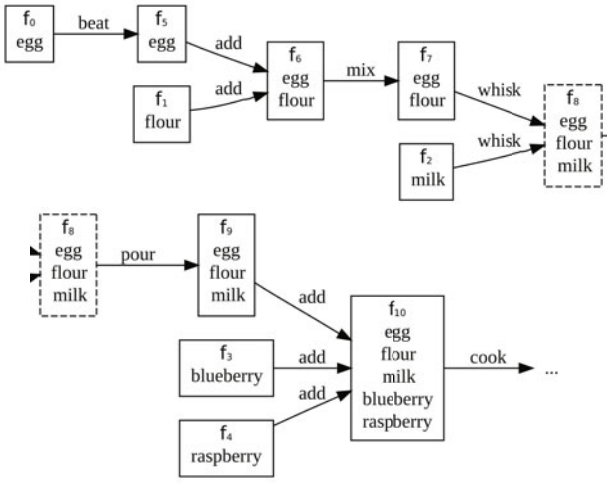


Fig. 2. Tree representation of the recipe of Fig. 1

2.1 Representing Recipes as Trees

Recipes are procedural texts composed of a sequence of actions through which different ingredients are progressively combined in order to obtain one final product, the dish. Each culinary action takes what we call *food components* in entry (as its *arguments*) and produces some other food component in return.

To adapt a recipe, it can help to divide it in smaller "parts" so that some parts can be replaced by new ones. Therefore the formal representation of a recipe must make it easy to identify the different stages in a recipe and the "regularities" across a set of recipes. Viewing actions as functions, it seems only natural to model recipes as trees. Alternatively this can be seen as taking a rather extreme stance in regards to Asher's theory of dynamic semantics [1], considering each verb as simultaneously a destruction and a construction verb (applied onto a food component). Some situations would make trees inappropriate, such as when whole eggs are split between white and yolk. We didn't take this into account in this work, but we think our approach could easily generalised to directed acyclic graph representations.

In a recipe tree, leaves are food components corresponding to the raw ingredients, the root is the finished dish, and the other nodes are the subsequent states of various food components. Trees are labelled (each node has a unique label ℓ) and a function $\mathcal{I}(\ell)$ is defined giving the set of ingredients that went into the food component represented by ℓ. For instance in Fig. 2, giving an example of a

very simple recipe tree, "f_6" is a node corresponding to a food component such that $\mathcal{I}(f_6) = \{\text{egg}, \text{flour}\}$.

This tree structure is what the data mining process is applied to. However it is also needed to solve some of the linguistic problems of recipe texts. The tree is built iteratively: actions found in the text are treated one after the other, each connecting a node to the tree, a process that requires using the information already present in the partially built tree.

2.2 Why Recipe Texts Are Different

While recipe texts have the advantage of exhibiting little fanciness, there exist specific difficulties inherent to their procedural nature:

1. They heavily make use of sentence structures such as imperative clauses that are rare in most other texts, making tools based on machine learning using generic corpora inefficient;
2. They massively exhibit a little-studied linguistic phenomenon known as *evolutive anaphora* wherein a word in a text may be used to refer to an object that exists at some given time and does not (yet or anymore) at some other, requiring a special strategy to find out what this word can refer to at any given moment *e.g.* "mix flour, eggs, and milk; pour *the batter*";
3. In order to avoid tedious repetitions, they usually omit syntactic arguments of verbs when they seem obvious, requiring a strategy to first determine whether a word is missing, and finding out what this word should have been, *e.g.* "cook potatoes; when done, add milk [implicitly: *to potatoes*]" (this is a type of *grammatical anaphora*).

2.3 The Toolchain

The first few steps of the linguistic analysis can be solved using common, well-researched natural language techniques. While we cannot use available annotated corpora as much as we normally would, we still managed to obtain a small corpus of about a hundred recipes annotated with the part-of-speech (*e.g.* verb, noun, adjective) of each word, which was sufficient to train an error-driven, context-sensitive, transformation-based tagger [6] with an accuracy $a > .90$, well below the state of the art for regular texts, but sufficient for a prototype implementation.[2] The other preliminary steps (tokenization, clause segmentation, chunking) were implemented using hand-crafted regular expressions.[3]

[2] Parts-of-speech tend to be even more ambiguous than usual in recipe texts because of words such as "cream" or "salt" that can be both nouns or verbs, so an approach based on a dictionary, even if it were a domain-specific dictionary, would be even less effective.

[3] For instance, the regular expression pattern for matching a noun phrase is "$\text{N}'((\text{Comma N}')^\star(\text{Comma}|\text{Conjunction})^{\{1,2\}}\text{N}')^?$", with "N′" matching "$\text{Predeterminer}^?\ \text{Determiner}^?\ \text{Adjective}^\star\ \text{Noun}^+$".

At this stage we know which words are verbs (and thus actions) and we are able to identify their syntactic arguments, which should be sufficient to get on with the task of building a formal tree representation of a recipe.

2.4 From Text to Tree

The problems described in paragraphs 2.2(2) and 2.2(3) cannot be solved at the sentence level. On the other hand, if they are neglected, the final representation of a recipe will not be a tree, but a set of trees, each representing a part of the recipe, that cannot be connected. A specific post-processing step was developed to handle those.

The idea is that at the beginning of the recipe, all ingredients are available and the very first action will take some of them to produce a new food component. In the same way, at any stage of a recipe, there are certain food components available and the next action picks some of them and produces a new one.

The set of food components available for culinary actions is called *domain.* The initial domain $\mathcal{D}_0$ contain one food component for each ingredient from the recipe listings. The domain changes after each action is performed, hence the domain after t actions, noted $\mathcal{D}_t$, is used to identify the arguments of the $t+1^{\text{th}}$ action.

In the recipe of Fig. 1, the initial domain will look something be:

$$\begin{aligned}\mathcal{D}_0 =&\{\mathrm{f}_0, \mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3, \mathrm{f}_4\},\\ &\mathcal{I}(\mathrm{f}_0) = \{\text{egg}\},\\ &\mathcal{I}(\mathrm{f}_1) = \{\text{flour}\}, \text{etc.}\end{aligned}$$

When an action that takes a single argument is encountered, it creates an arrow with the action name and creates a new node. It also modifies the domain, such that for a verb like "beat X":

$$\begin{aligned}\mathcal{D}_t =&(\mathcal{D}_{t-1} \backslash \{X\}) \cup \{\ell\},\\ &\mathcal{I}(\ell) = \mathcal{I}(X)\ ,\end{aligned} \tag{1}$$

where ℓ is a new label. For instance the first sentence in the example recipe is "beat the eggs", which would have the effect of creating a new "egg" food component (see Fig. 3):

$$\begin{aligned}\mathcal{D}_1 =&(\mathcal{D}_0 \backslash \{\mathrm{f}_0\}) \cup \{\mathrm{f}_5\},\\ &\mathcal{I}(\mathrm{f}_5) = \mathcal{I}(\mathrm{f}_0)\ .\end{aligned}$$

As for actions taking several arguments, they may have either a "union" semantics, like "add X to Y", creating a node with multiple edges,

Fig. 3. Beaten eggs

$$\begin{aligned} \mathcal{D}_t =& (\mathcal{D}_{t-1} \backslash (\{X\} \cup \{Y\})) \cup \{\ell\}, \\ \mathcal{I}(\ell) =& \mathcal{I}(X) \cup \mathcal{I}(Y) \ , \end{aligned} \tag{2}$$

or a "complement" semantics, like "remove X from Y":

$$\begin{aligned} \mathcal{D}_t =& (\mathcal{D}_{t-1} \backslash \{Y\}) \cup \{\ell\}, \\ \mathcal{I}(\ell) =& \mathcal{I}(Y) \backslash \{X\} \ , \end{aligned} \tag{3}$$

where ℓ is a new label. For instance, the next sentence in the example recipe is "Add the flour", which is a union action (see Fig. 4):

$$\begin{aligned} \mathcal{D}_2 =& (\mathcal{D}_1 \backslash (\{\mathrm{f}_1\} \cup \{\mathrm{f}_5\})) \cup \{\mathrm{f}_6\}, \\ \mathcal{I}(\mathrm{f}_6) =& \mathcal{I}(\mathrm{f}_1) \cup \mathcal{I}(\mathrm{f}_5) \ . \end{aligned}$$

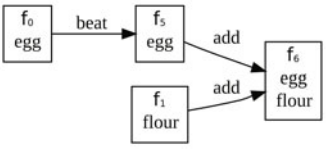


Fig. 4. Some batter

A *subcategorization dictionary* of actions tells which types of arguments are required by each action in order to know whenever one is missing. In that case, the last node added to the tree is assumed to be the missing argument, thus an edge is created from this node to the new node. This is how one can infer that it is to eggs that flour gets added (cf. Fig. 4).

The set-theoretical notation we used for food components' ingredients is useful to resolve the anaphoras. The two most frequent cases are presented below.

Existential References. Expressions such as "beef mixture" refer to some food component that contain at least one specific ingredient, in this case beef, that may have been mixed with others. It is therefore necessary to search the domain for a food component containing this ingredient. A target set $\mathcal{T}$ of ingredients expected in the food component is thus defined and used with a simple operation to retrieve the food component being referred to by the expression:

$$x \in \mathcal{D} : \exists i . i \in \mathcal{I}(x) \wedge i \in \mathcal{T} \ . \tag{4}$$

In the case of "beef mixture", this is trivial: $\mathcal{T} = \{\text{beef}\}$. But some cases are more subtle, such as words similar to "batter". We measured in a corpus of recipes that

the food component referred to by the word "batter" contains either the ingredients egg or flour in over 99% of cases. To find the food component referred to by this word, we will thus use $\mathcal{T} = \{\text{egg}, \text{flour}\}$. Thanks to this the "Pour batter" instruction of the recipe can be dealt with. Before dealing with this instruction, the representation looks like Fig. 5, making it is obvious which of the three food components in the domain is the one the word "batter" refers to.

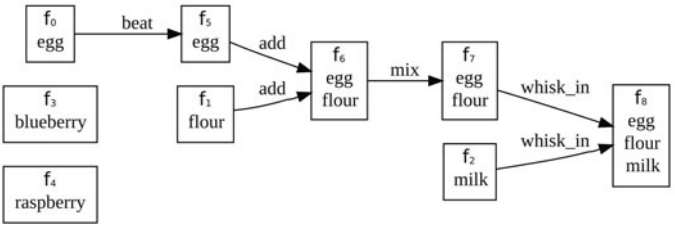


Fig. 5. Where is the batter?

Universal References. Other expressions obviously refer to a set of ingredients that belong to a common class. Such is the case for the word "fruits" when the ingredients listings of a recipe do not actually contain any "fruit" elements, but do contain blueberries and raspberries. Since an ontology of ingredients already exists in TAAABLE, it is used to retrieve the set of food component being referred to by their class name. In any given recipe, the set of food components referred to by the word "fruit" can be defined as:

$$\{x \in \mathcal{D} : \forall i. i \in \mathcal{I}(x) \rightarrow i \sqsubseteq \text{Fruit}\} \ ,$$

where "$i \sqsubseteq$ Fruit" means that ingredient i is a subclass of the "Fruit" class. So when we process the instruction "Add some fruits" in our recipe, we know that it refers to $\{\text{f}_3, \text{f}_4\}$.

3 Adapting Recipes with Formal Concept Analysis

At this stage, with recipes formalized as trees, for each ingredient in each recipe, there exists a path (a sequence of actions) between this ingredient and the final state of the recipe—the dish.

The adaptation process can now be redefined according to this structure. We consider that adapting the preparation with respect to an ingredient substitution consists in replacing a subtree corresponding to the preparation of an ingredient in the retrieved recipe with a subtree that is suitable for the substitution ingredient.

The questions that need to be dealt with now are the selection of a subtree to replace and a subtree to replace it with. This will be demonstrated with an example using genuine recipes from a past CCC recipe base. Suppose the user

wants an aubergine coleslaw, and the system retrieved a zucchini coleslaw recipe and suggested to replace zucchini with aubergine. We need to find the instance of aubergine use in the available recipes that is the closest to the way the zucchini is used in the zucchini coleslaw. FCA provides a conceptualization of the different ways of preparing an ingredient, so that the concept closest to the zucchini can be easily identified.

3.1 Extracting the Relevant Subtree

We now need to extract the subtree referring to any ingredient of particular interest from the representation (this will be done both for the substituted and the substitution ingredient). Three types of actions are considered: actions applied to an ingredient alone, actions applied to many ingredients in parallel, and actions applied to many ingredients together. The distinction between the second and third types is important: while peeled apples and pears for instance are still distinctively apples *and* pears, apple and pears cooked together make up a mixture that has little to do with the original ingredients, and to which a wholly different range of actions may be applied. Additionally, if all the information is taken, the space gets too wide and there is not enough data concentration to allow for efficient learning.

Because no linguistic clues are available to help classify actions between the second and the third category, we simply use a list of possible actions with their most likely category. Considering now a genuine recipe from the case base, represented in Fig. 6, "toss" is an action of the third category hence the actions applied to zucchini are considered as relevant up to "add", making the zucchini prototype: "zucchini $\xrightarrow{\text{cut}} \cdots \xrightarrow{\text{shred}} \cdots \xrightarrow{\text{add}}$". When making the substitution later on, the zucchini prototype will therefore be detached from "mixture_3" and the aubergine prototype chosen in Sect. 3 will be reattached at the same point.

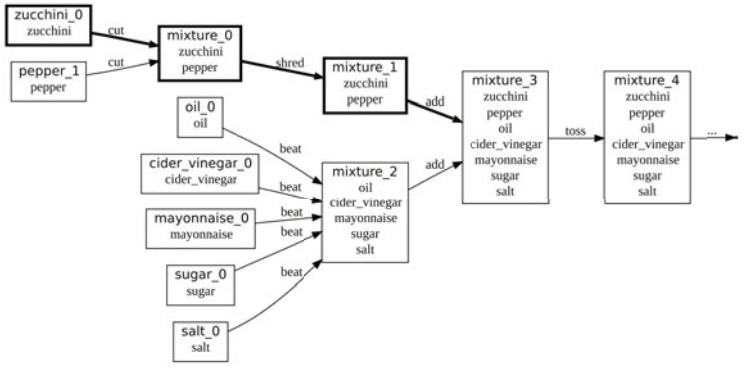


Fig. 6. Relevant "zucchini" subtree identified in bold

3.2 Formal Concept Analysis

FCA [11] takes as entry a binary table (such as the one shown in Table 1) describing a binary relation between objects ("aubergine_509", "aubergine_667"...) and attributes ("add", "arrange"...), called a *formal context,*. Formally, considering a formal context $\mathbb{K} = \langle \mathcal{O}, \mathcal{A}, I \rangle$, where $\mathcal{O}$ is a set of objects, $\mathcal{A}$ is a set of attributes, and $I \subseteq \mathcal{O} \times \mathcal{A}$ is a binary relation such that $\langle o, a \rangle \in I$ means that object o owns attribute a, its Galois connection is defined by the following functions: $f : 2^{\mathcal{O}} \rightarrow 2^{\mathcal{A}}$ and $g : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{O}}$, where a subset of objects is mapped to the subset of attributes it owns in common, and reciprocally.Formal concepts are the $O \times A$ pairs that are closed under the Galois connection, creating the concept lattice $\mathfrak{L}(\mathbb{K})$.

Each node of a lattice such as the one shown in Fig. 7 represents a formal concept, which is a pair $\langle O, A \rangle$ where O (the *extent*) is the set of objects owning all the attributes in A (the *intent*) and A is the set of attributes owned by all the objects in O. The lattice is partially ordered following the inclusion of the extent. Any concept whose extent is included in the extent of another concept is drawn below and connected to the latter—which is said to *subsume* it.

Table 1. The formal context for the query space

	add	arrange	bake	beat	blend	boil	break	...
aubergine_509								
aubergine_667	x		x		x			
aubergine_981		x						
aubergine_1030			x					
aubergine_1387			x					
...								
zucchini	x							

Since an ingredient's mode of preparation is characterized by the culinary actions applied onto it, the formal context that is generated uses culinary actions applied to aubergines as attributes, with each aubergine recipe corresponding to one object. The formal context, which in this case (with a tiny recipe base to maintain lattice readability) has 15 objects and 55 attributes, will thus look as in Table 1 with an object corresponding to the zucchini recipe merged along with the relevant culinary actions. The resulting lattice is shown in Fig. 7.

Formal concept number 1, called "Top", is the maximum of the lattice. Its extent is the set of all objects and its intent is the set of the attributes shared by all objects (in this case, the empty set). Concept number 58, called "Bottom", is the minimum of the lattice. Its intent is the set of all attributes and its extent is the set of the objects that share all attributes (there again the empty set). From top to bottom, the concepts are progressively more "specific", meaning their extent contains less objects but those share more attributes. Concept number 29 is the most specific one containing `zucchini` in its extent, so it shows all attributes of this object. Concept number 16 is said to immediately subsume

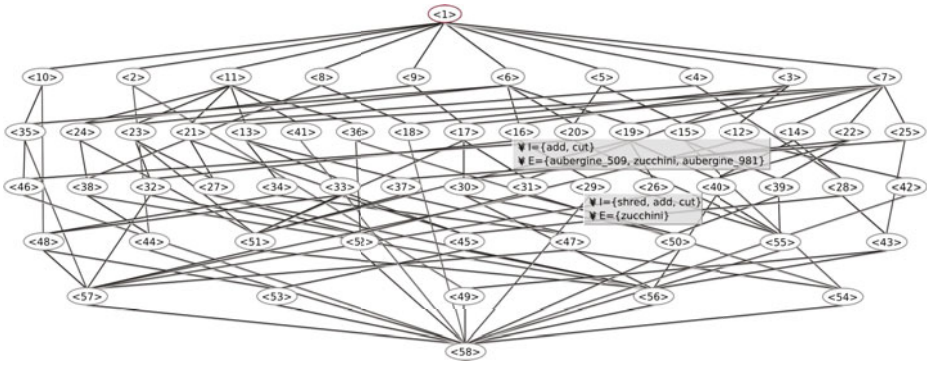


Fig. 7. The 58-concept query lattice

concept number 29: it is the most specific concept having `zucchini` as well as other objects in its extent (in this example there is only one concept immediately subsuming concept number 29, but there could be more). From a certain point of view, those objects are the ones that are the most closely related to `zucchini`, insofar as they are the ones having the most attributes in common with it.

3.3 Answering Queries in a Lattice

In the fashion of Carpineto [7], lattice-based ranking is used to retrieve an adaptation prototype. If one has a recipe with zucchini (say a zucchini coleslaw) and wants to replace this zucchini with aubergine (because they want an aubergine coleslaw), a first step will be to find an aubergine prototype that is compatible with the way the zucchini is prepared, possibly including steps specific to aubergines and excluding steps specific to zucchini. It makes sense to view the zucchini recipe as a query in the document space of aubergine recipes.

Given the formal concept of which our query is part of the *proper* or *reduced extent* (the "lowest", or most specific concept in which the query appears), the candidate answers are selected from the (full) extent of this concept itself, or of the concepts immediately subsuming it, minus the query object. This heuristic is different from Carpineto's (who searches the whole set of subsuming concepts) because the goal here is to reduce the adaptation effort required, hence minimize the difference between the query's and the selected object's attributes. In this very case, the recipe that has the least differences between the attributes of the zucchini z and its replacement aubergine a is:

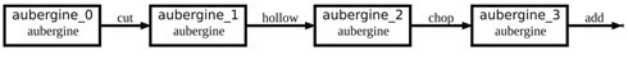
$$\arg\min_{a} \left| \Big(\mathrm{Int}(\mathcal{C}_\mathrm{z}) \backslash \mathrm{Int}(\mathcal{C}_\mathrm{a}) \Big) \cup \Big(\mathrm{Int}(\mathcal{C}_\mathrm{a}) \backslash \mathrm{Int}(\mathcal{C}_\mathrm{z}) \Big) \right| , \tag{5}$$

where $\mathcal{C}_\mathrm{a}$ and $\mathcal{C}_\mathrm{z}$ are the most specific concepts of a and z, and $\mathrm{Int}(\mathcal{C})$ is the intent of $\mathcal{C}$.

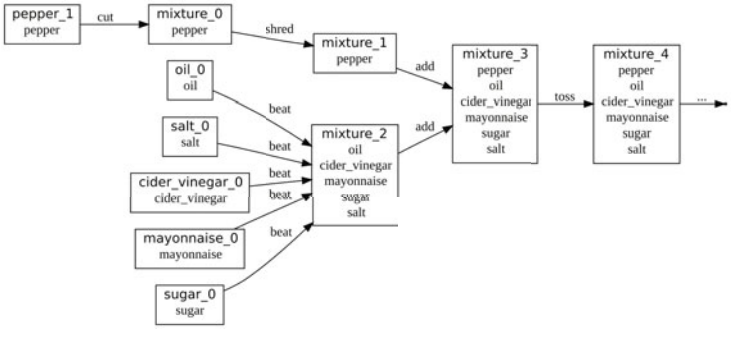
3.4 The Adaptation Algorithm

All that is left to do to complete the tree adaptation is to obtain the zucchini and aubergine subtrees using the technique described in Sect. 2.4, remove the former (keeping parts that are required for the processing of other ingredients), and merge the latter.

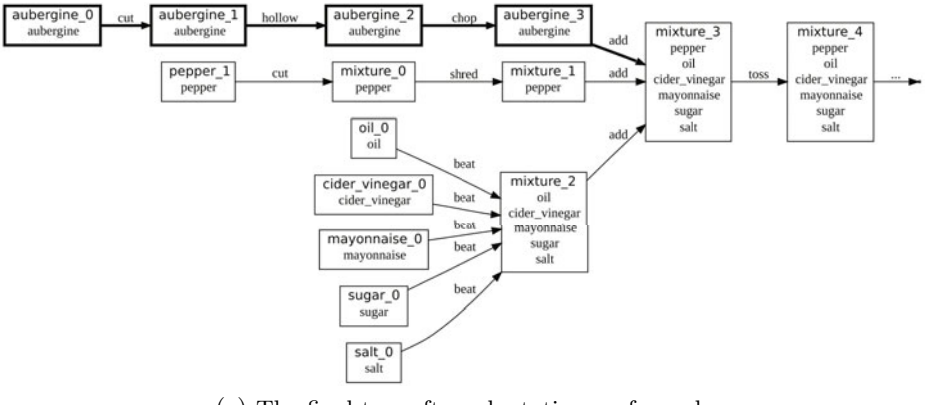
Existing partial-order case-based planning techniques such as [19,13,3] could be reused to integrate the new preparation steps along the rest of the recipe. On the other hand, not keeping the structure of the selected recipe intact would prevent us from reusing its text as is, forcing us to use natural language generation



(a) The `aubergine_981` subtree to be attached.



(b) The tree from Fig. 6 with zucchini subtree pruned.



(c) The final tree after adaptation performed.

Fig. 8. The subtree attachment process

Cut the aubergines in half lengthwise. With a spoon, hollow out the center of each half to make a boat like shell about 1/4 inch thick. Finely chop the aubergine pulp and set it aside. Cut ~~zucchini and~~ red pepper into matchstick thin strips or shred in food processor; add the chopped aubergine pulp. set aside. In large bowl, with wire whisk or fork, beat salad oil, vinegar, mayonnaise, sugar, salt and pepper until mixed. Add vegetables; gently toss to mix well. [...]

Fig. 9. The original recipe text with removed parts stroke through and added parts (copied from the retrieved recipe) underlined

techniques and yielding results of poor textual quality, as argued in [12]. Moreover planning requires advanced knowledge of the domain: action pre- and post-conditions must be known. For instance, a given ingredient may, when heated, create froth that needs removing. In the absence of such knowledge, grafting seems a safer approach. A simpler strategy that consists into connecting the new subtree at only one point in the tree gives results that are satisfactory (as shown in Fig. 8) and allows for easier textual adaptation, at a reduced computational cost.

We propose an algorithm for text adaptation that is very simple and parallel to the tree adaptation process. It copies whole sentences of the selected recipe text as much as possible, causing the resulting text to have a more "natural" quality than if it has been machine generated. Actions related to zucchini are removed: if a clause refers to zucchini only it is removed altogether, otherwise only the word "zucchini" is removed (along possibly with the word "and" or a comma). Then the text related to aubergine preparation in the selected recipe is treated in the same way to remove references to all ingredients other than aubergines. This "aubergine text" is inserted at the beginning of the original text, except for the clause containing the last verb of the prototype (the one that causes it to be merged with other ingredients), which is inserted at the point where those other ingredients have been fully processed. This process is exemplified in Fig. 9. All that is then left to do is minor grammatical (*e.g.* verb agreement) and typographic (*e.g.* punctuation) adjustments.

4 Discussion and Related Work

The adaptation process gives very satisfactory results when the data mined in text was of good quality. The linguistic processing of recipes is thus the weak link. The formal representation of a recipe is usually of high quality if the text contained no spelling mistakes and no "less usual" phrase structures such as negations ("do not peel the potatoes before boiling them") or elaboration ("boil the potatoes—but peel them first"). Those phenomena could be dealt with, with more or less success, but would significantly complicate the implementation.

The worst problem though is the bad performance of the part-of-speech tagger (the module responsible for saying which words are verbs, which are nouns, etc.),

given that it has access to only a very small and low quality training corpus. But such a corpus is expensive to obtain: annotators can only process about 3000 words per hours [16] whereas hundreds of thousands of words are required to achieve good performance as can be seen in [5]. Because of the reference problems described in Sect. 2.2, if one action is missed by the analyzer (because the action verb was not tagged as a verb), it is very likely that many subsequent actions applied to the same ingredient will be missed.

Circa 10% of the formal representations were actual trees, indicating that those were near-perfect and totally usable representations. As for the others, "correctness" is difficult to quantify, but we would say that the analysis is usually very good for the first few actions (the one we actually use most) and decreases as it goes on.

Evaluating a case-based reasoning system is generally very difficult, but can be done through comparisions with other similar systems, which is the very reason why the CCC workshop was created. Therefore the adaptation algorithms described in this paper are being implemented in the TAAABLE system to be evaluated during this year's edition of the CCC.

FCA has already been used in CBR, for instance in [9] and in [2]. In [9], the formal context is a representation of the case base (the objects are the cases and their properties are binary properties of cases). The concept lattice obtained by FCA structures the case base, and the retrieval process is done thanks to a hierarchical classification in this lattice. Moreover, the lattice is used to assist the user query formulation process. By contrast, in our work, FCA is used for adaptation: the formal context is all about the vocabulary for describing cases (the ingredients and the actions performed on ingredients). The objective of [2] is to obtain structured cases from texts, using FCA. The formal context objects are the texts and its properties are relevant terms of these texts. Contrary to our problem though, in the reports they use, one document can be understood as one case. In our case, the boundary is blurred by the presence of multiple ingredients in any recipe, meaning that not all keywords found in a given text are relevant for a given adaptation.

Many of the more interesting types of textual and non-textual cases have a structure similar to recipes: assembly instructions, user's manuals, pharmaceutical directions for preparation are all examples of "procedural" cases that have certain "preconditions" that could be lacking and thus in need of adaptation. We believe that our approach would help solve this problem.

5 Conclusion and Future Work

This paper proposes a solution to adaptation in textual case-based reasoning systems. It uses natural language processing techniques to build a rich formal representation of texts, then data mining algorithms and formal concept analysis to retrieve a text from which some parts are reused.

The formal representation we created is helpful in interesting ways at other stages of the CBR process besides adaptation. For instance, a function using

some physical characteristics, such as the size or the texture, of a food component in its rightmost state of the prototype sequence could be used to assess the quality of the adaptation choices, or contribute to the adaptation cost function. Moreover, the trees could be passed to a learning algorithm to ease the recognition of certain recipe attributes: *e.g.,* the sequence of action–ingredient pairs that makes a dish a soup could be learned, a knowledge which might prove useful to the CBR process.

According to the *adaptation-guided retrieval* principle [18], a source case requiring less adaptation effort should be preferred over the others. In the current implementation of TAAABLE, this adaptation effort is measured using a penalization cost in the ingredient hierarchy. In the future, we will incorporate a cost related to the adaptation of the preparation: if α and β are two ingredients, the more prototypes that FCA shows they have in common, the least the adaptation effort of substituting α with β will be.

References

1. Asher, N.: Events, facts, propositions, and evolutive anaphora. In: Higginbotham, J., Pianesi, F., Varzi, A.C. (eds.) Speaking of events, pp. 123–150. OUP, Oxford (2000)
2. Asiimwe, S., Craw, S., Wiratunga, N., Taylor, B.: Automatically acquiring structured case representations: The SMART way. In: Applications and Innovations in Intelligent Systems, vol. XV, pp. 45–58. Springer, Heidelberg (2007)
3. Muñoz-Avila, H., Weberskirch, F.: Planning for manufacturing workpieces by storing, indexing and replaying planning decisions. In: Drabble, B. (ed.) Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS 1996), Edinburgh, Scotland, May 1996, pp. 158–165 (1996)
4. Badra, F., Bendaoud, R., Bentebitel, R., Champin, P., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J., Meilender, T., Meilender, T., Mille, A., Nauer, E., Napoli, A., Toussaint, Y.: TAAABLE: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In: ECCBR Workshops, Workshop of the First Computer Cooking Contest, pp. 219–228. Springer, Heidelberg (2008)
5. Banko, M., Brill, E.: Mitigating the paucity-of-data problem: Exploring the effect of training corpus size on classifier. In: Proceedings of the first international conference on Human language technology research. Association for Computational Linguistics, Morristown (2001)
6. Brill, E.: Transformation-Based Learning. Ph.D. thesis, Univ. of Pennsylvania (1993)
7. Carpineto, C., Romamo, G.: Order-Theoretical Ranking. Journal of the American Society for Information Science 51(7), 587–601 (2000)
8. Cordier, A., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H., Toussaint, Y.: WIKITAAABLE: A semantic wiki as a blackboard for a textual case-based reasoning system. In: 4th Workshop on Semantic Wikis (SemWiki 2009). 6th European Semantic Web Conference, Heraklion, pp. 88–101 (2009)
9. Díaz-Agudo, B., Gerváz, P., González-Calero, P.A.: Adaptation Guided Retrieval Based on Formal Concept Analysis. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 131–145. Springer, Heidelberg (2003)

10. Díaz-Agudo, B., González-Calero, P.A.: Classification Based Retrieval Using Formal Concept Analysis. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 173–188. Springer, Heidelberg (2001)
11. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Heidelberg (1999)
12. Gervás, P., Hervás, R., Recio-García, J.: The Role of Natural Language Generation During Adaptation in Textual CBR. In: 4th Workshop on Textual Case-Based Reasoning: Beyond Retrieval, ICCBR 2007 (2007)
13. Ihrig, L., Kambhampati, S.: Derivation replay for partial-order planning. In: Proceedings of the 12th National Conference on Artificial Intelligence (AAAI 1994), pp. 992–997 (1994)
14. Lamontagne, L., Bentebibel, R., Miry, E., Despres, S.: Finding Lexical Relationships for the Reuse of Investigation Reports. In: 4th Workshop on Textual Case-Based Reasoning: Beyond Retrieval, ICCBR 2007 (2007)
15. Lamontagne, L., Lapalme, G.: Textual reuse for email response. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 234–246. Springer, Heidelberg (2004)
16. Marcus, M., Santorini, B., Marcinkiewicz, M.: Building a large annotated corpus of English: The Penn Treebank. Computational linguistics 19(2), 313–330 (1994)
17. Messai, N., Devignes, M.D., Napoli, A., Smaïl-Tabbone, M.: Querying a Bioinformatic Data Sources Registry with Concept Lattices. In: Dau, F., Mugnier, M.-L., Stumme, G. (eds.) ICCS, pp. 323–336. Springer, Heidelberg (2005)
18. Smyth, B., Keane, M.T.: Using adaptation knowledge to retrieve and adapt design cases. Knowledge-Based Systems 9(2), 127–135 (1996)
19. Veloso, M. (ed.): Planning and Learning by Analogical Reasoning. LNCS (LNAI), vol. 886. Springer, Heidelberg (1994)

Visualization for the Masses: Learning from the Experts

Jill Freyne[1] and Barry Smyth[2]

[1] Tasmanian ICT Center
CSIRO
GPO Box 1538, Hobart, 7001,
Tasmania, Australia
[2] CLARITY: Centre for Sensor Web Technologies
School of Computer Science and Informatics
University College Dublin,
Dublin, Ireland

Abstract. Increasingly, in our everyday lives, we rely on our ability to access and understand complex information. Just as the search engine played a key role in helping people access relevant information, there is evidence that the next generation of information tools will provide users with a greater ability to analyse and make sense of large amounts of raw data. Visualization technologies are set to play an important role in this regard. However, the current generation of visualization tools are simply too complex for the typical user. In this paper we describe a novel application of case-based reasoning techniques to help users visualize complex datasets. We exploit an online visualization service, ManyEyes, and explore how case-based representation of datasets including simple features such as size and content types can produce recommendations to assist novice users in the selection of appropriate visualization types.

1 Introduction

So called *"knowledge workers"* are defined by the important role that information plays in their day to day work-lives. and over the last 10-15 years the Internet has provided them with a platform for information discovery, collaboration, and communication. Just as Web search engines have played a vital role when it comes to helping the typical user to access relevant information online, we are now witnessing the emergence of a new generation of information tools that will help users to make more sense of an ever-increasing quantity of raw data. Visualization technologies are set to play a key role in this regard, buy helping users to better understand the relationships and messages that are often hidden within complex data.

Great strides have been made to bring a wide range of visualization options and tools to the masses. For example, Microsoft's Excel offers 11 different types of chart (bar, line, pie etc.) and a total of 73 basic variations on these charts. Apple's Numbers spreadsheet is similarly well equipped and even Google's free Spreadsheets programme offers access to about 25 different variations of 6 different chart

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 111–125, 2010.

types. No doubt readers familiar with the popular TED series (www.ted.com) will recognise the added-value that high quality visualization techniques can bring through the inspirational talks of Hans Rosling[1]. In short, visualization techniques can help us to make sense of complex data and by applying the right visualization technique to a given dataset can make all the difference when it comes to helping users to recognise the meaning and message of a given dataset [11,10,9].

Surely then all of this puts powerful visualization capabilities within reach of the average person? In truth, the answer is a negative one. The problem, of course, is that the average user is not a visualization expert and producing the right sort of visualization for a given dataset is far from trivial. How then can we help the average user to take advantage of what state-of-the-art visualization techniques have to offer? How can we provide meaningful assistance when it comes to helping a user to visualize their particular data set so that they too may access its hidden meaning? In this paper we describe a case-based recommender system that is designed to suggest visualizations to users, based on key properties of the dataset that they wish to visualize, by harnessing the past visualization experiences of other users.

Previous work in the area of visualization recommendation includes research into articulated task-orientated systems [3], early data-property based systems [8,6], hybrid task and data based systems, which examine both user intent and the data at hand [14,1]. More recent work aims to discover patterns in user behaviour in preparation of a dataset in order to predict visualization requirements [4]. Finally, closest to our work is work by Mackinlay from 2007 [7] where a rule based approach is taken to visualization recommendation. In that work the characteristics of a dataset including the structure and content determine the type of recommendation presented. This approach is similar to ours in terms of examining the dataset. However, rather than relying on the opinion of experts to determine the rules which are implemented in a non-flexible manner we believe that the ability to harness past visualization experiences can provide valuable insight into the visualization design process. This thinking suggests that there is an ideal opportunity to apply case-based reasoning methods to the visualization recommendation task. It provides opportunity for novice users to learn from the creativity of more experienced visualizers by suggesting imaginative ways to represent their data and also allows the system to increase the type and number of visualizations on offer without the need for additional rules to be added to a rule set.

The starting point for this work is a Web based "social" visualization platform called *ManyEyes* that was created by the Visual Communication Lab in IBM Research's Collaborative User Experience group [13,12]. In brief, ManyEyes is a web-based visualization platform that allows users to upload datasets, choose from a wide variety of visualizations, and make the results available to others. Each "visualization experience" encodes important visualization knowledge about the decisions taken by a user about how to visually represent a given

[1] http://www.gapminder.org/

dataset. These decisions are not explicitly represented within the visualization structures but rather implicitly encoded according to the decisions taken by the user when producing a visualization. Thus, each visualization can be viewed as a concrete *case*, with features of the dataset providing the *case specification* and the resulting visualization configuration providing the *case solution*. In this paper we propose that these visualization cases can be reused to support the visualization of a new dataset, to make suggestions about appropriate visualization choices (e.g. visualization types). We describe a variety of recommendation strategies that explore a number of different ways to represent visualization cases plus a variety of metrics for assessing case similarity. We go on to evaluate these different approaches using real-world ManyEyes data to show how even relatively simple case-based reasoning techniques can inform useful recommendations.

In the next section we review the ManyEyes system, describing the features that are offered, and summarizing the visualization data that has been made available for this work. Section 3 describes the details of our recommender system, detailing a number of options for retrieving and reusing visualization cases. Then, in Section 4 we describe the results of recent evaluation, based on the live ManyEyes dataset, which demonstrate the potential for this recommender system to benefit ManyEyes users, especially casual or novice users. We conclude the paper with discussions of the analysis and an overview of future work.

2 ManyEyes: A Web-Based Visualization Service

ManyEyes (http://manyeyes.alphaworks.ibm.com) is an online browser based visualization tool designed not only to make sophisticated visualization easily accessible to web users, but also explore the potential for visualizations to serve as *social objects* that can be shared and that can serve as a focus of discussion and conversation. As such ManyEyes allows users to freely upload datasets and make them available to the wider public. It allows these datasets to be visualized using a variety of techniques (e.g. bar charts, histograms, line graphs, tag clouds, pie charts etc.); in total, ManyEyes supports 33 different visualization types. Users can try different visualization options on the same dataset and very often the same dataset might be visualized in a variety of different ways to reveal different relationships and concepts. For example, Figure 1 shows an example chart created from FDA survey data about the average lead content of vitamin supplements[2].

The power of ManyEyes is certainly in the sheer range of visualization options that are available to end-users and the flexibility that is offered when it comes to experimenting with, and sharing, different datasets and visualization types. But this power also brings great challenges. ManyEyes is motivated by the desire to bring visualization to the masses but to the novice user choosing the right visualization for the right dataset can be a daunting task, and one that is not

[2] http://manyeyes.alphaworks.ibm.com/manyeyes/visualizations/fda-survey-data-on-lead-in-womens-an

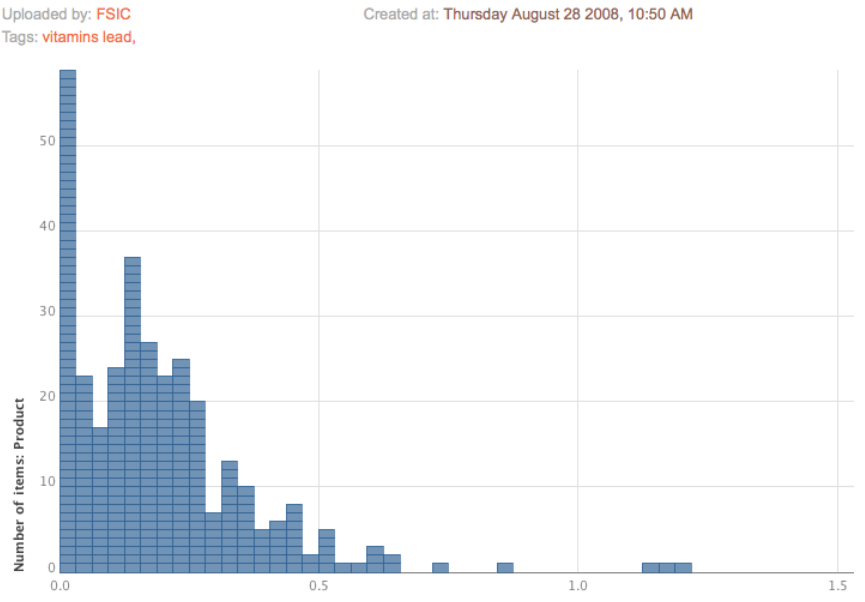


Fig. 1. An example of a ManyEyes histogram representing FDA survey data on fraction of lead contained in women's and children's vitamins

well supported by ManyEyes, beyond the obvious trial-and-error process that it supports by default.

2.1 Data Upload

Raw data is uploaded to ManyEyes as freeform text or as tab-delimited files from popular applications such as Microsoft Excel or Open Office. During upload ManyEyes analyses the upload data to assign data types (e.g. text, numeric, etc.) to different dataset fields. In addition, users can add metadata to their datasets by adding a title, descriptive text, tags, and/or provenance information. In turn, ManyEyes also assigns information about the uploader's user-id and upload date to each uploaded dataset.

2.2 Visualization Creation

ManyEyes supports 6 high-level visualization categories which cover a range of basic visualization functions to provide access to a total of 33 different visualization options. These basic categories include:

1. *Analysing Text* - Word tree, phrase net, and tag cloud type visualizations that are designed for the analysis of textual datasets.
2. *Comparing Sets of Values* - Visualization formats that are designed to provide a comparison of different sets of values, for example, bar charts, histograms, and bubble charts.

3. *Exploring the Relationships among Data Values* - Matrix charts, network diagrams, and scatter plots are used to help users explore the relationships between values.
4. *Visualizing the Parts of a Whole* - Pie charts and tree maps allow users to visualize the parts of a whole.
5. *Geographic Visualizations* - Map-based visualizations allow users to explore geographic datasets.
6. *Tracking Trends* - Line and Stack graphs are used for the visualization of trending data.

At visualization time, the user must select the visualization type that is most appropriate to their particular dataset and needs. While the designers of ManyEyes have provided some useful hints and tips about the type of visualization that might suit a particular dataset, this selection task is non-trivial, and it is one of the main barriers to entry for novice users. When a user selects a visualization type from the options provided, ManyEyes automatically generates the visualization, from the current dataset, automatically assigning chart parameters where possible, and asking for user conformation when multiple options exist.

2.3 Sharing and Discovery

One of the original motivations behind ManyEyes is the crowd-sourcing of different visualizations for a given dataset. It is expected that the right set of visualizations will help to *creatively explore* the dataset in a way that would simply not be possible in the context of a traditional solitary visualization scenario. To the creators of ManyEyes, visualization is a creative act, and datasets and their visualizations, are social objects that provide a focus for sharing and discussion. As such ManyEyes has been designed to support a variety of social interactions providing registered users with publicly accessible profile pages, for example, and supporting the active sharing, discussion and rating of datasets and their visualizations.

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

2.4 The ManyEyes Dataset

In this work we are interested in supporting the ManyEyes user during the creative act of visualization itself, ultimately by suggesting useful visualizations to

users at visualization time. As such we have created a case-based recommender system that harnesses the past visualization experiences of ManyEyes users as the basis for recommendation. We see this as a very appropriate and novel use of case-based reasoning. Visualization experiences are readily available, for example, and they implicitly encode complex visualization decisions that are often made by experienced and creative users. By selecting those experiences that are best suited for a given visualization task we can bring some of this creativity and experience to the novice user.

The dataset (case base) used for this work represents approximately 21 months of usage of ManyEyes from January 2007 and covers 33,656 separate dataset uploads, and 23,940 unique visualizations, from 15,888 registered users. It is worth highlighting that only 43% of uploaded datasets in this repository were successfully visualized; in other words more than half of the datasets do not have an associated visualization associated with it stored in the system, presumably, in part at least, because of the complexity of the selection task. In turn, about 40% of ManyEyes users who uploaded datasets never went on to create and store a visualization. This is surely a telling comment on the challenges faced by ManyEyes users when it comes to choosing and configuring suitable visualizations of their data. These are our target users: they are the novices who are motivated to upload data but for whatever reason did not successfully complete the visualization of their dataset.

In general there are two basic types of dataset in ManyEyes. *Text* datasets are *bag-of-word* type datasets whereas *tabular* datasets are the more traditional column-based datasets, using a mixture of data types. Here we focus on tabular datasets as they present the largest challenge for generating recommendations. The visualization of 9881 tabular datasets resulted in 16,848 different visualizations (1.7 per dataset).

3 A Case-Based Recommender for ManyEyes

The ManyEyes repository of datasets and visualizations is more than a simple collection of raw datasets and charts. It is a representation of visualization experiences, in the sense that each successful visualization represents a deliberate visualization attempt by a user, and is the product of a complex set of choices and decisions. Moreover we can reasonably assume, at least to some extent, that the user has made a good set of choices (considering information content and aesthetics, for instance) in terms of configuring an appropriate and useful visualization. Of course this assumption does not always hold up. For example, there is no doubt that many users create ineffectual visualizations. However, there are mechanisms available to evaluate the likely quality of a particular visualization effort. For example, ManyEyes allows users to rate visualizations. In addition, many datasets attract multiple visualizations and to the extent that different users experiment similar visualization settings we might reasonably assume that a given visualization approach is a good one; for example if a particular dataset has been visualized 10 times and 8 of these use bar charts in various ways then we can assume that the bar chart is a reasonable chart type to apply to this dataset.

In short then, the dataset-visualization combinations in Many-Eyes constitute a set of visualization *experiences* that can be represented as *cases*. To this end we propose to augment the existing ManyEyes system with a case-based recommendation component. When a new dataset is selected the case-based recommender first converts the dataset into a suitable set of features, uses these features to find a set of similar cases from the visualization case base, and then produces a ranked list of visualization types to suggest to the user. In the following sections we will summarize the case representation, retrieval, and ranking techniques that are used by this recommendation system.

3.1 Case Representation

We begin by assuming that each case represents a single dataset and a set of visualizations. Thus, each case, c_i is made up of a dataset component, d_i and a visualization component, v_1,...v_n, as shown in Eq. 1. In fact there is also additional information that is sometimes available such as the rating associated with a particular visualization, r_i. In case-based reasoning parlance the dataset component corresponds to the *specification* part of a case, the visualization component corresponds to the *solution* part of a case, and the rating component can be viewed as the *outcome* of the solution. In this paper we will focus on the specification and solution side of visualizations cases, largely because the ManyEyes dataset is very sparse when it comes to the availability of ratings data.

$$c_i = \{d_i, v_1, ...v_n\} \tag{1}$$

In this work we are focusing on recommending a set of likely visualization types to a users when faced with a new dataset. Thus, the representation of the visualization component is relatively straightforward, each case solution is a set of visualization types used, $chart(v_1)$,.....$chart(v_x)$, ordered by decreasing popularity in the visualizations, v_1,...v_n, created from d_i. Going forward, one can envisage taking this a step further and reasoning about particular features of the visualization, such as the axis placement, label usage etc.

Each dataset is characterised by a set of simple features that relate to the type of data contained in the dataset. Each feature aims to represent the structure, content or metadata description of the dataset. For tabular datasets we extract *structural features* that include the number of textual columns, col_{txt}, the number of numeric columns, col_{num}, the number of data points (rows), $rows$ We compliment these by examining the *descriptive features*, a bag-of-words textual description derived from any metadata associated with the dataset, $desc$ (e.g., column headings, title etc.). In cases where tabular data contains numeric columns we also extract *numeric features* that reflect data contained in the numerical columns such whether the column contains all positive, all negative or mixed values ($num_{pos}, num_{neg}, num_{mixed}$) such that num_{pos} is the number of numeric columns containing only positive values, num_{neg} is the number of numeric columns containing only negative values and num_{mixed} is the number of

numeric columns with mixed values. In this way each case is represented as a feature-based dataset and solution as in Eq. 2.

$$c_i = \{col_{txt}, col_{num}, rows, desc, num_{pos}, num_{neg}, num_{mixed}, chart(v_i)..., chart(v_x) \quad (2)$$

3.2 Similarity and Retrieval

Given a new target case c_T (made up of a particular dataset and set of visualizations) the task of the recommender system is to locate a set of similar cases that can be used as possible visualization recommendations. We use traditional tried and tested similarity techniques using the case specifications above. To be specific, when computing the similarity between cases we use the similarity metric shown in Eq. 3 which calculates the relative difference between two cases by examining the differences between each feature $col_{txt}, col_{num}, rows, desc, num_{pos}, num_{neg}, num_{mixed}$.

The feature differences for all features are determined using Eq. 4 except for the *desc* feature which uses Eg. 5. In this instance uniform weighting is used.

$$sim(c_T, c_i) = 1 - \sum_{f \epsilon \{features\}} w_f \bullet distance(c_T, c_i, f) \quad (3)$$

$$distance(c_T, c_i, f) = \frac{|c_T(f) - c_i(f)|}{max(c_T(f), c_i(f))} \quad (4)$$

$$distance(c_T, c_i, desc) = 1 - \frac{|c_T(desc) \cap c_i(desc)|}{max(c_T(desc), c_i(desc))} \quad (5)$$

In the evaluation section we will demonstrate that even these simple techniques work well when it comes to driving high quality recommendations, while at the same time leaving a number of options open for more sophisticated similarity techniques as part of future work. Thus, given a target case c_T we can use the above similarity techniques to produce a ranked list of n similar cases as the basis for recommendation.

3.3 Generating Recommendations

Each of the n cases retrieved will be associated with a set of visualizations. The same visualization type may occur in more than one case and so we can identify a set of k different visualization types from these n cases. We need a way to rank these visualizations so that those that are associated with more similar cases are preferred over those that are associated with fewer, less similar cases. To achieve this Eq. 6 scores each of the n visualizations, v_i, as the sum of the similarity scores associated with the retrieved parent cases; $visualized(v_i, c_j) = 1$ if v_i is a chart type used to visualize c_j and 0 otherwise. The result is a ranked list of

visualization recommendations, $v_1, ..., v_k$ in descending order of their aggregate scores as per Eq. 6

$$score(v_i, c_T, c_1, ..., c_n) = \sum_{\forall j=1...n} sim(c_T, c_j) \bullet visualized(v_i, c_j) \quad (6)$$

4 Evaluation

This work is motivated by the fact that less than half (43%) of the datasets uploaded to ManyEyes have the resulting visualization saved to the system. We believe that a reasonable number of these "failed visualizations" were due, at least in part, to the confusion of choice that faced the novice first-time uploader while an additional proportion were simply not saved to the ManyEyes database but taken and used by their creators. In this work we implement sophisticated feature modelling and aimed to improve the visualization rate of ManyEyes by making proactive suggestions to the user about which visualization technique might best suit their dataset. In this section we will describe the results of a recent large-scale, off-line, leave-one-out style evaluation using a live ManyEyes dataset.

4.1 Set-Up

The core ManyEyes test-set used in this evaluation covers a total of 14,582 unique datasets that have been uploaded by thousands of different users. These datasets have been variously visualized to produce 23,940 distinct visualizations which are represented as individual visualization cases. The diversity of visualizations created on ManyEyes is high with the most popular chart type being the *Bubble chart* which accounts for 17.2 % of the visualizations. The popularity of each chart type is detailed in Table 1.

Table 1. Popularity of Chart Types

Bubble Chart	Bar Chart	Map	Network Diagram	Treemap
17.2%	13.0%	12.8%	10.5%	7.8%
Line Graph	Scatterplot	Matrix Chart	Pie Chart	Tag Cloud
6.5%	6.3%	5.5%	4.2%	4.2%
Stack Graph	Stack Graph for Categories	Block Histogram	Treemap for Comparisons	Word Tree
3.1%	3.1%	2.3%	1.8%	1.0%

In an effort to eliminate low-quality visualizations from this core test-set we eliminated all those cases which were created by individuals who created only a single visualization. The motivation here is to remove contributions from very novice users as they probably do not reflect expertise in the area. This decision could have eliminated some high-quality visualizations but on average we expect

the overall quality of the test-set to improve as a result. In the end the test-set of visualization cases used for this evaluation comprised of almost 10,000 tabular visualization cases.

For the purpose of evaluating our case-based reasoning approach to visualization recommendation we have developed 5 different recommendation strategies, 2 of which represent simple benchmarks (*Random* and *Popular*) and the remaining 3 are different flavours of case-based reasoning that rely on different types of case specification data as the basis for similarity and retrieval. In summary, these strategies are as follows:

1. *Random* recommend a set of k random visualizations.
2. *Popular* recommend the k most popular visualizations in ManyEyes.
3. *Structure* this CBR strategy uses *structural features* only (such as the number of numeric and/or textual columns in a dataset) to produce a ranked-list of the top k visualizations from a set of n similar cases.
4. *Structure+Description* this more detailed CBR approach exploits both *structural* and *descriptive* (e.g. term-based information derived from dataset metadata and/or column titles) features in its case representation to produce a ranked list of the top k visualizations from a set of n similar cases.
5. *Structure+Numeric Features* this alternative CBR approach exploits *structural* and *content* (e.g.information about individual dataset columns such as whether the contents were *positive, negative or mixed*) features to produce a ranked list of the top k visualizations from a set of n similar cases.

Our evaluation takes the form of a standard leave-one-out test. For each target case, c_T, we use its specification features (whether structural, description or content) as the basis of a new target dataset and generate a set of k visualizations using each of the 5 recommendation strategies above.

4.2 Measuring Recommendation Quality

There are many factors to consider, and different possible approaches to take, when evaluating the quality of a set of recommendations. In this study we look at two basic quality metrics. First, we consider the *accuracy* of the k recommendations produced by counting how frequently a given recommendation strategy produces a recommendation list that contains the *most* popular visualization for the dataset contained in the current target case. Many of these datasets will have been visualized in different ways, using different chart types for example. Our intuition is that if there is one visualization type that dominates then this is likely to be the best way to visualize that particular dataset and hence our focus on looking for these most popular visualization types among our recommendation lists. So an *accuracy* of 60% means that the most popular visualization is present in 60% of the recommendation sets of size k. Of course there are many alternatives to estimating recommendation accuracy and, for what it is worth, we have considered alternative measures such as precision and recall across all visualizations, and the results produce are broadly in agreement with the result we will present for this particular measure of accuracy.

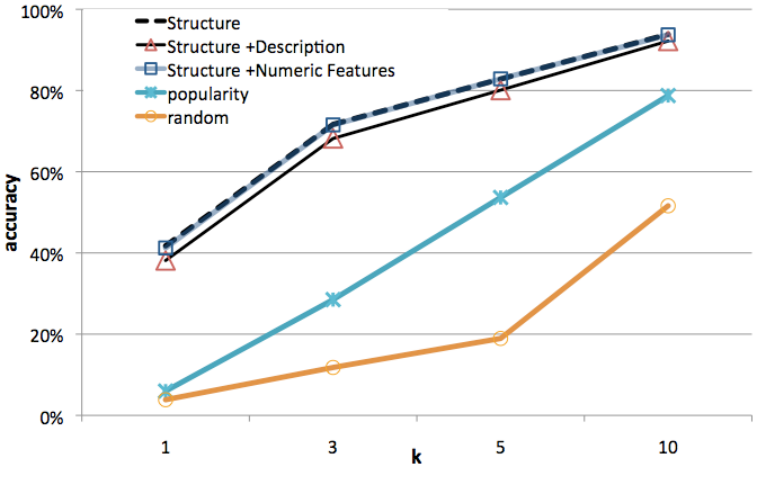


Fig. 2. Average Accuracy

Figure 2 clearly supports the use of a CBR recommendation strategy with all three CBR strategies significantly outperforming the two benchmarks (*Popular* and *Random*) across all values of k. It shows for example, that the *Structure* and $Structure + NumericFeatures$ CBR approaches achieve 42% accuracy with their first recommendations in comparison to only 6% for both benchmarks. In other words, when we just focus on a single recommendation we find that the three CBR techniques suggest the most popular visualization chart almost half of the time, where as the alternative strategies present this recommendation in less that 1 in 10 attempts. Indeed the CBR approaches reach an accuracy level of 70% when $k = 3$ whereas the benchmark techniques never significantly benefit from longer recommendation lists. We note that the three CBR techniques perform comparatively well with a very close coupling seen between the *Structure* and $Structural + NumericFeatures$ algorithms but with the $Structure + Description$ algorithm lagging slightly behind. This indicates that the information contained in the actual data which is being graphed is more reliable than the user specified associated metadata. Figure 3 shows the mean accuracy over all values of k for each strategy and again highlights the advantage of the CBR techniques over the benchmarks and the variation between the three CBR techniques.

The second way we measure recommendation quality is to look for the position of these most popular visualizations among the recommendation lists: the lower the position, the better the recommendation list since the best visualization is appear nearer to the top of the list. Of course using a simple measure of *average position* benefits recommendation lists that do not contain the correct visualization. Thus as an alternative we actually calculate an *adjusted position* by assigning a $k + 1$ penalty to those lists that do not contain a correct (most popular) visualization. This is a conservative penalty because it assumes that the correct visualization is actually in position $k + 1$, which may not be, but it

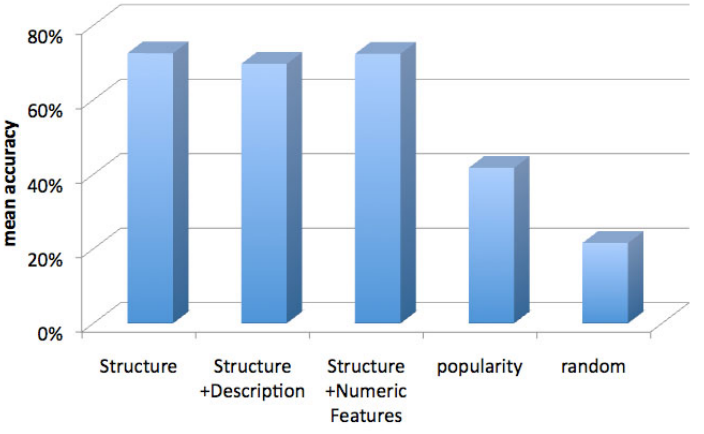


Fig. 3. Mean Average Accuracy

serves to at least remove some of the bias associated with a simpler measure of average position.

The results of the positional analysis of the recommendation techniques are presented in Fig.4 & 5. In terms of the average position statistic the CBR recommendation techniques are mixed in comparison to the benchmarks. For example, we see in Fig.4 the clear difference in performance of the CBR and benchmarks when the adjustment is made with all of the CBR techniques clearly superior to the benchmarks across all values of k. Once again we note the similar performance of each of the CBR approaches with only minor positional benefits observed when $k > 5$ for the $Structure + NumericFeatures$ approach.

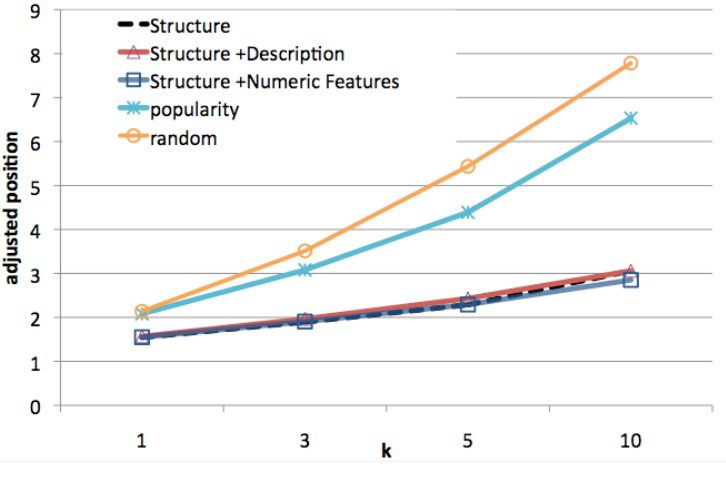


Fig. 4. Adjusted position

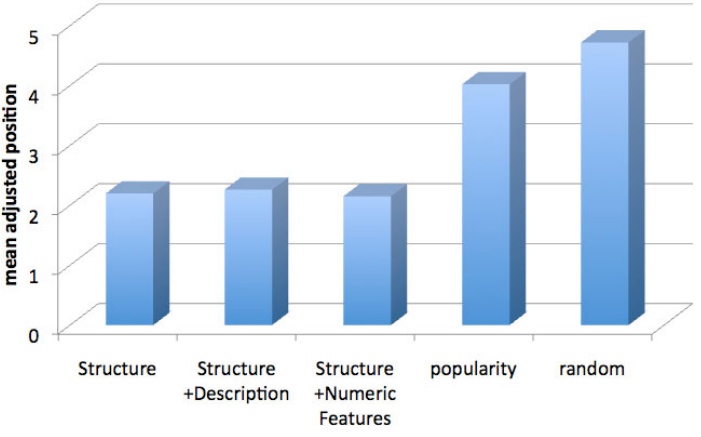


Fig. 5. Mean Adjusted Position

Fig.5 charts the mean adjusted positions for each approach across all values of k and clearly shows the superiority of the CBR approaches once again with the *Structure* and $Structure + NumericFeatures$ approaches out performing the $Structural + Description$ approach.

5 Conclusions

The objective of this work is to help users of a Web-based visualization system to produce better visualizations by harnessing visualizations that have been previously produced for datasets that are similar to their own. This has guided a case-based reasoning approach to recommendation where we view a dataset-visualization pair as a source of visualization knowledge. By reusing these visualization cases we can make high-quality recommendations to novice users.

We have described an evaluation that uses real-world visualization experiences to evaluate a range of different CBR approach, each of which explore the benefits of increasingly sophisticated case representations in order to drive recommendations. We compare these against two simple benchmarks. Overall the CBR approaches have been found to outperform the benchmarks, producing more accurate recommendation lists which contain appropriate visualizations nearer to the top of this list. Interestingly, we have noticed only minor additional benefits are gained from leveraging increasingly sophisticated case representations, at least in the test-set used in this study. Most likely this is a direct result of the large volume of visualization cases that are available. We note that no advantage is seen when the meta-data associated with a dataset is included in the case representation, this could be down to the small amount of data which people add to their datasets or the generic nature of the metadata. We note marginal increases in the performance of the CBR approach which examines the numeric content within each dataset. Examination of the data on across each visualization type did show that the $Structure + NumericFeatures$ outperformed the

simpler *Structure* approach for datasets which resulted in visualizations which fall under the category of *Tracking Trends* in particular when line graphs were generated. This finding highlights the potential for more advanced case representation, possibly combining several other dimensions for comparison as highlighted by [7] such as whether the data is increasing or decreasing, the uniformity of the increase or decrease etc.

We have identified a number of areas where changes could affect the accuracy of our recommender.

1. *Noise and Novice users.* In this work we attempted to remove a certain amount of noise from the dataset by only including visualizations by creators who have created more than one visualization. We will look more closely at the users who have remained and their experience with visualization in order to generate an experience score which can be used to weight visualizations by their creators experience. This score could encompass data on the raw number of visualizations created or more intuitively the number of different visualization types a user has experience with.
2. *Ratings & Provenance.* ManyEyes maintains rating information and information about the creator of the particular visualization. In recent years there has been new work in the area of *provenance* [5] and *reputation* [2] that could be used to improve the recommendation algorithms by harnessing information about the source of a case and the reputation of the creator.
3. *Introducing Adaptation.* There is considerable scope for adaptation in this domain since recommending a visualization type is really just one part of a larger decision support problem. Users will benefit greatly from configuration support when it comes to actually using a particular visualization. This includes deciding which fields are associated with which axes, scale settings, etc. and these all provide opportunities for post-retrieval adaptation.
4. *Comparison to other Classification Techniques.* In this work we have compared our simple CBR technique to very simple alternative measures. With work into creating sophisticated CBR representations planned we can also compare the performance of the CBR method with other classification approaches such as naive bayes, decision trees or neural networks. We also plan to compare our technique to a rule based algorithm [7] to determine its accuracy with its static counterpart.

References

1. André, E., Rist, T.: The design of illustrated documents as a planning task. American Association for Artificial Intelligence, Menlo Park (1993)
2. Briggs, P., Smyth, B.: Provenance, trust, and sharing in peer-to-peer case- based web search. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 89–103. Springer, Heidelberg (2008)
3. Casner, S.M.: Task-analytic approach to the automated design of graphic presentations. ACM Trans. Graph. 10(2), 111–151 (1991)

4. Gotz, D., Wen, Z.: Behavior-driven visualization recommendation. In: IUI 2009: Proceedings of the 13th international conference on Intelligent user interfaces, pp. 315–324. ACM, New York (2009)
5. Leake, D.B., 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)
6. Mackinlay, J.: Automating the design of graphical presentations of relational information. ACM Trans. Graph. 5(2), 110–141 (1986)
7. Mackinlay, J., Hanrahan, P., Stolte, C.: Show me: Automatic presentation for visual analysis. IEEE Transactions on Visualization and Computer Graphics 13(6), 1137 (2007)
8. Roth, S.F., Kolojejchick, J., Mattis, J., Goldstein, J.: Interactive graphic design using automatic presentation knowledge. In: CHI 1994: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 112–117. ACM, New York (1994)
9. Tufte, E.R.: The visual display of quantitative information. American Journal of Physics 53, 1117 (1985)
10. Tufte, E.R.: The cognitive style of PowerPoint. Graphics Press Cheshire, Conn. (2004)
11. Tufte, E.R.: Beautiful evidence. Graphics Press Cheshire, Conn. (2006)
12. Viégas, F.B., Wattenberg, M.: TIMELINES Tag clouds and the case for vernacular visualization. Interactions 15(4), 49–52 (2008)
13. Viegas, F.B., Wattenberg, M., van Ham, F., Kriss, J., McKeon, M.: Manyeyes: A site for visualization at internet scale. IEEE Transactions on Visualization and Computer Graphics 13(6), 1121–1128 (2008)
14. Zhou, M.X., Chen, M.: Automated generation of graphic sketches by example. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 65–74. Morgan Kaufmann, San Francisco (2003)

Imitating Inscrutable Enemies: Learning from Stochastic Policy Observation, Retrieval and Reuse

Kellen Gillespie, Justin Karneeb, Stephen Lee-Urban, and Héctor Muñoz-Avila

Department of Computer Science and Engineering, Lehigh University,
19 Memorial Drive West, Bethlehem, PA 18015 USA
{kjg210,jtk210,sml3,hem4}@lehigh.edu

Abstract. In this paper we study the topic of CBR systems learning from observations in which those observations can be represented as stochastic policies. We describe a general framework which encompasses three steps: (1) it observes agents performing actions, elicits stochastic policies representing the agents' strategies and retains these policies as cases. (2) The agent analyzes the environment and retrieves a suitable stochastic policy. (3) The agent then executes the retrieved stochastic policy, which results in the agent mimicking the previously observed agent. We implement our framework in a system called JuKeCB that observes and mimics players playing games. We present the results of three sets of experiments designed to evaluate our framework. The first experiment demonstrates that JuKeCB performs well when trained against a variety of fixed strategy opponents. The second experiment demonstrates that JuKeCB can also, after training, win against an opponent with a dynamic strategy. The final experiment demonstrates that JuKeCB can win against "new" opponents (i.e. opponents against which JuKeCB is untrained).

Keywords: learning from observation, case capture and reuse, policy.

1 Introduction

Children learn by observing and then mimicking the actions taken by adults or other children. From this psychological motivation, learning from observation has become an important and recurrent topic in AI [1]. In a nutshell, an agent observes the actions performed by another agent while solving a problem and reuses those actions while solving problems. Case-based reasoning systems can be seen as learning from observations; cases retain previously observed episodes and are reused when solving new problems [2].

In the context of agents interacting in adversarial environments, like games, a number of representations have been studied for CBR systems to encode an agent's observations. Table 1 summarizes these representations. We identified four representations for the case (the representations refer to the "solution" part of the case; this is the part of the case that is reused to solve new problems). These representations are listed in order of generality, from the less to the more general (the more general can represent the less general, but the opposite is not necessarily true): (1) a single action/command, (2) a script (i.e., a sequence of commands), (3) a plan (i.e., a sequence

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 126–140, 2010.

Table 1. Representations used for adversarial CBR

Representation	Description	Sample System/Paradigm
action	A single action/command	SIBL
scripts	Sequences of commands	CAT
plans	Sequences of actions	Darmok
policy	A map: States → Actions	CBRetaliate

of actions) and (4) a policy (i.e., a mapping from state to actions, indicating for each state s the action that the agent must take). We distinguish between scripts and plans. The commands in the scripts have no explicit semantics for the CBR system. In contrast, the actions in plans have explicit semantics as indicated by the action's preconditions and effects [3].

Epstein and Shih [4] present an example of a CBR system that uses sequential instance-based learning (SIBL) to decide an agent's next move in playing bridge; these actions are captured from previous instances based on the sequence of states visited so far. Case-based Tactician (CAT) [5] uses scripts to indicate the sequence of commands to execute. CAT records the sequence of commands performed during a game. Darmok [6] stores plans which are sequences of actions. When reusing these plans, a dependency graph between the plan's actions is generated using the action's preconditions and effects. Plans are captured by observing a human expert who also annotates the captured plans with applicability conditions. CBRetaliate [7] stores policies indicating the best action to take for a given state.[1]

In all these situations the CBR systems follow a deterministic path; given the current state of the game once the CBR system has committed to an action/script/plan/policy, the next action/command to be executed is predetermined. This does not mean that subsequent actions/commands are also predetermined. For example, CBRetaliate might alter its current policy to improve performance. But after changing the policy, the next action is once again predetermined. The same holds true for other CBR systems playing games. For example, Darmok can adapt the current plan while executing it in the game in a process called on-line adaptation. Still, once the adaptation ends, the next action to execute is pre-determined by the plan.

In this paper we are interested in extending the representation of the observations for stochastic policies. A stochastic policy π is a mapping: π: State $\times$ Actions → Probabilities. For example, an agent might observe multiple games on the same map and with the same initial conditions, and at the beginning of the game it might observe players rushing to attack the opponent 30% of the time, building a townhall 50% of the time and constructing a defensive tower 20% of the time. An observer might be able to discern the difference in choices made by looking at a complete state. But the state might not be fully observable. Hence, an observer will not be able to fully understand the difference in the choices, leaving it with the need to represent the observed actions of the agent as stochastic policies.

[1] CBRetaliate stores a Q-table. This Q-table yields the greedy policy which is obtained by selecting for every state the action with the highest Q-value. The greedy policy is the solution stored in the case and the Q-table is needed for performing adaptation using Q-learning.

In this paper we study CBR systems learning from observations that can be represented as stochastic policies. While there are a number of works that combine CBR and reinforcement learning (RL) using non-stochastic policies (e.g. [7,8,9,10]), to our knowledge this is the first time a state-action probability-distribution *representation* is being used in the context of CBR systems for adversarial environments. Note that our approach does not perform RL, but is inspired by RL solution representations. This representation is more general than those discussed in Table 1 and, hence, our work advances the state of the art. Being able to reason with stochastic policies can have practical implications for CBR systems learning from observations because frequently there is no explanation for apparently conflicting decisions observed (e.g., an agent observing multiple online games from players world-wide). We describe a general framework which encompasses three steps. The agent: (1) observes agents performing actions, elicits stochastic policies representing the agents' strategies and retains these stochastic policies as cases, (2) analyzes the environment and retrieves a suitable stochastic policy, and (3) then executes the retrieved stochastic policy, which results in the agent mimicking the previously observed agents. We implement and evaluate our framework in a system called JuKeCB that observes and mimics agents playing games.

2 A Framework for Capturing and Reusing Stochastic Policies

A stochastic policy π is a mapping:

$$\pi: \text{States} \times \text{Actions} \rightarrow \text{Probabilities}$$

such that the set of a state's actions $\pi(s) = \{ \pi(s,a) \mid a \in \text{Actions} \}$ is a probability distribution for every state s in States. That is, $\sum_{a \in \text{Actions}} \pi(s,a) = 1$. Our framework captures and reuses stochastic policies. Reusing a policy means that when the agent visits a state s the agent selects an action to take based on the probability distribution π(s). The Algorithm Reuse-StochasticPolicy below implements this.

Reuse-StochasticPolicy(π, G, P, Δ, t)
Input: π: stochastic policy; G: game engine, P: player we are controlling, Δ: time to execute policy; t: time to wait for execution of next action
Output: none

t' ← 0;
while not(t' ≤ Δ) **do**
  s ← currentState(G)
  a ← select-Action(s, π) // selects action based on probability distribution π(s)
  execute(P,a) // P executes action a
  wait(t)
  t' ← t' + t
end-while

Capturing a policy means that the agent observes during each episode the states that are visited and the actions that are taken when those states are visited. Based on the actions taken for every state s, the agent elicits a probability distribution π(s) for

each state's actions. The Algorithm Capture-StochasticPolicy below implements this. It assumes that no action/state can change more than once in one time unit.

Capture-StochasticPolicy(Actions, States, G, P, Δ)
Input: Actions: the possible actions that the agent can take; States: possible states that the agent can visit; G: game engine, P: player we are observing, Δ: time to observe
Output: π: the observed policy

```
for s ∈ States do
   visited(s) ← 0    // a counter of how many times s is visited
   for a ∈ Actions do
      π(s,a) ← 0
t← 0; s ← nil; a ← nil;
while not(t ≤ Δ) do
     wait(1)   // waits one time unit.
     s' ← currentState(G)
     a' ← currentAction(P)
    if (a ≠ a') then
          a ← a'
          π(s',a) ← π(s',a) + 1
    if (s ≠ s') then
          s ← s'
          visited(s) ← visited(s) + 1
          π(s,a') ← π(s,a') + 1
    if (a ≠ a' and s ≠ s') then
          π(s,a) ← π(s,a) − 1    // avoids double-counting
    t ← t + 1
end-while
for s ∈ States do
   for a ∈ Actions do
      π(s,a) ← π(s,a) / visited(s)
return π
```

3 The JuKeCB Problem-Solving Architecture

We developed our ideas for stochastic policy capture and reuse in the JuKeCB architecture. The JuKeCB architecture is one that is typical of the CBR problem solving approach [11]; Fig 1 shows a high-level presentation as it applies to JuKeCB.

Our testbed is DOM, a generic domination game [7] which consists of teams of bots competing to control specific locations on a map called domination points (dom-points). The teams earn points over time for each dom-point controlled, and the first team to reach a specified amount of points wins. A team takes control of a dom-point when one or more of the team's bots is close to it without the other team being in the area for a small period of time. Bots have health points that are lost in dice-roll combat, which is initiated when bots on different teams navigate close to one-another.

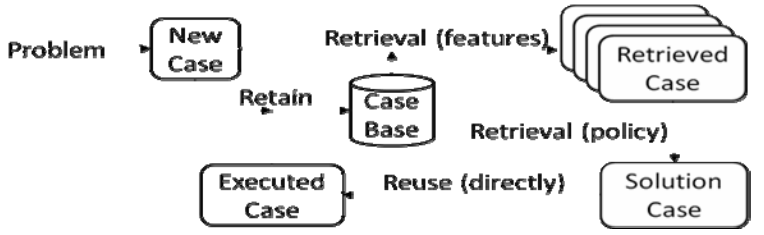


Fig. 1. The JuKeCB architecture

Combat is played to completion (only one team's bots remain), and slain bots are respawned. The dice roll is modified so that the odds of reducing the opponent health points increase with the number of friendly bots in the vicinity. The game then consists of each team trying to hold onto more dom-points than the other team.

Domination games are interesting in that individual deaths have no direct impact on the final score because kills do not give points and any player killed will respawn to continue playing. This allows for overall strategy and organization to have a far larger impact in the final outcome of the game.

Because of the large number of possible game states in DOM, we follow the state abstraction model of [7] which simply tracks ownership of dom-points, and to which dom-points bots are sent. This abstraction reduces the number of states to $d^{(t+1)}$, and the number of actions to $(b \times t)^d$ where d is the number of domination points, t the number of teams, and b the number of bots; one is added to the exponent to account for neutral ownership of dom-points at the beginning of games. We estimate the total number of possible states in the game to be at least $O(2 \times 10^{34})$, a function of: (1) the number of cells in a map of n rows and m columns, (2) the number of bots per team, (3) the remaining health, 0 to 10, of each bot, (4) the ownership of each dom-point (5) a number 0 to 5 for each dom-point, indicating the amount of game ticks since a bot began to attempt the capture; 0 is no attempt, whereas 5 transfers ownership. So the number of states is about $(n \times m)^{(b \times t)} \times 11^{(b \times t)} \times (t+1)^d \times 6^d$. In our experiments, $n = m = 70$, $b = 3$, $t = 2$, and $d = 4$. Hence, without the abstraction, it would be computationally infeasible to observe a sufficient number of actions in each state to have a representative average.

DOM showcases the need to capture and reuse stochastic policies. An agent playing a DOM game may send a bot to a third dom-point in a situation where the agent is already controlling the two other dom-points. However, later on, in the same situation, the same agent might decide to send all bots to defend dom-points it owns without sending any bot to the third locations. An observer might be able to discern the difference in choices made by looking at a complete state. But the state might not be fully observable, or, even if it is fully observable the number of features could be extremely large. In the DOM game, the state is partially observable; whereas features such as the current score and the team owner of a location is fully observable, other features such as the exact locations of the bots and their current trajectories are not. Hence, an observer will not be able to fully understand the difference in the choices, leaving it with the need to represent the observed actions of the agent as stochastic policies. In the face of this complexity we assume that the agents follow a purely

stochastic Markov strategy, in which the action to execute only depends on the current abstract state. In general, this assumption will not always hold – simple averaging can fail when an observed policy is a sophisticated one that depends on a longer history of previous states. However, we empirically found that JuKeCB performs well in DOM; *why* an enemy has chosen to use a specific strategy is not as important as *what* strategy has been observed in the case base that counters it successfully.

Each game episode is observed in real-time, and segmented into windows of observation of size Δ (a pre-determined amount of game-clock cycles). During each Δ, the features characterizing the game state are observed and recorded as the features of the new case; the policies used by each team during Δ is recorded in the case as the solution. Details of the case features and solutions are described in Section 4, as well as the process of retaining cases. Section 5 explains how and when a set of cases similar to the features of the new case is retrieved, and then details how the solutions in this set are searched for the solution to be reused. The directly executed solution is then stored as a new case in the case-base, with features appropriate for the Δ during which it was applied.

4 Case Capture and Retention through Policy Observation

Previous work on learning policies (e.g. [12], [13]) has demonstrated that state abstractions are essential for quickly learning good policies. Consequently, JuKeCB has a representation of game states (used for case features) that focuses on the most important elements for the learning task (namely the ownership of map locations, among others); as gameplay proceeds through different states, JuKeCB observes for each team the actions taken by each team member (or "bot") in these states, and records the observations as cases. That is, over a period of time Δ, JuKeCB observes for each bot b of team t, and for all states visited during Δ, that whenever the game is in state s, bot b takes action a with probability $p_{b,t}(a, s)$. Based on this, the system builds a team policy π^t, which formally represents the *strategy* being followed by each team:

$$\pi^t: \forall \text{ BOT } b \text{ on team t}, \forall \text{ STATE } s, \forall \text{ ACTION } a, s \times a \rightarrow p_{b,t}(s,a)$$

A team's policy π^t indicates for each game state s what is the probability $p_{b,t}(s,a)$ of a team member b taking action a in that state. We therefore define a case as a tuple: $((f_1,f_2,\ldots,f_n), \pi^{t1}, \pi^{t2}, \ldots, \pi^{tm})$, where:

- $(f_1,f_2,\ldots,f_n)$ are the features of the case; these abstract the game state
- $\pi^{t1}, \pi^{t2}, \ldots, \pi^{tm}$ are the team policies for team 1, 2, …, m respectively

In our current testbed, we have two teams (so m = 2). The case features we use are: (1) the number of domination points, (2) the number of bots in each team, (3) the Manhattan distance between all domination points taking walls into consideration, (4) the percentage of time that a team held each domination point, and (5) the point difference observed between winning and losing policies. The elements which define a policy, the case solution, are the following: (1) the percentage of moves in which a given bot went to a given domination point, (2) the percentage of moves in which a given bot went to a domination point not currently owned by its team, and (3) the

percentage of moves in which a given bot went to the closest domination point. Thus a case represents a pair of stochastic policies for a single, abstracted game state.

Case Retention. Over the course of a game (JuKeCB can simultaneously play and observe; for the empirical evaluation, we do not exploit this ability) all observed cases are stored into a temporary file. When the game is over a filtering process takes place to ensure that no redundant cases are added. The filtering process goes through each new case one at a time and compares it to all cases currently stored in the case base. A new case c is added unless there is an existing case with at least 95% similar to c. If the most similar case is over 95% similar, the filtering system will then look at the Delta-score field in the two cases. If the new case has a higher Delta-score, it will replace the existing case otherwise the new case is discarded. In our experiments, JuKeCB did not suffer from dimensionality problems related to the representation of game states, and had roughly 175 cases after full training (when observing, cases are saved every 500 turns and games last about 50,000 turns).

5 Case Retrieval and Reuse

When JuKeCB is playing the game, the case retrieval process follows a 2-step retrieval process. The first step happens once before the start of the game; all cases that do not apply to the current map, as determined by the maps features, are removed from consideration. In the second step JuKeCB compares the observed game state to all remaining cases and choose the most similar one, provided that it is at least as similar as some predefined threshold. If no such a case is found, JuKeCB uses a default case, which implements an equiprobable policy (one in which all actions are equally likely to be taken).

When computing the similarity of two cases, we compute both the similarity of their features and of their stochastic policies. The similarity of features is computed by aggregating local similarities. Given two vectors of features <X> and <Y>, the aggregated similarity metric is defined in the usual way:

$$SIM_{FEATURES}(X_{1..n},Y_{1..n}) = \alpha_1 sim_1(X_1,Y_1) + \ldots + \alpha_n sim_n(X_n,Y_n)$$

The sum of the vector weights, $\alpha_1 + \ldots + \alpha_n$, adds to 1. As a result, $SIM_{FEATURES}()$ returns a value between 0.0 and 1.0 (1.0 being most similar). The most interesting local similarity we compute is that of location ownership; to do so, the percentage of time (ratio) that each team owned a map location is compared.

When computing the similarity of the polices, we compare a policy of the observed case with the losing policy of a case in the case base. Similarity of case solutions is computed according to the following formula:

$$SIM_{sol}(\pi^1,\pi^2) = \sum_{s \in S} \sum_{a \in A} \left(\alpha_{s,a} \times sim(\pi^1_{s,a},\pi^2_{s,a})\right)$$

Thus, the similarity of two policies is a comparison between the frequency that action a was taken by each team in state s ($\pi^x_{s,a}$ is this frequency), for all states and actions.

The retrieval comparison $SIM(C_1,C_2)$ of a case C_2 to the case representing the current situation, C_1, is expressed as the following weighted sum ($\alpha_{sol} + \alpha_{feature} = 1$):

$$\alpha_{sol} SIM_{sol}(\text{ policy}(C_1), \text{policy}(C_2)\text{ }) + \alpha_{feature} SIM_{FEATURES}(\text{ features}(C_1), \text{features}(C_2)\text{ })$$

Case Reuse. The strategy JuKeCB uses during gameplay is to follow the retrieved winning policy according to its probability distribution as explained in Section 4. That is, while the retrieved stochastic policy π is being executed, the selection of the next action is governed according to π by the percentage of moves in which a given bot (1) went to a given domination point, (2) went to a domination point not currently owned by its team, and (3) went to the closest domination point. The probability distribution π' recreated during case reuse is an approximation of the strategy π used by the winning team. Retrieval occurs every 525 turns (games are about 50,000 turns).

6 Empirical Evaluation

We tested the effectiveness of the JuKeCB system in the Dom game. JuKeCB observes and play games against a number of teams and build up a case base of the strategies it observes and captures. Over several experiments the JuKeCB system was tested in such a way to help determine its strengths and weaknesses.

6.1 Setup

In this section, we present the teams that JuKeCB observed, outline the training procedure for the three experiments we conducted, and the parameters used in each. In order to test JuKeCB, teams were created for it to observe and to play against. Each of these teams has a fixed gameplay strategy. Table 1 summarizes each of the fixed strategies.

The training set is slightly different for each experiment but the overall idea remains the same. In order for JuKeCB to be trained to win against a certain opponent, it must observe or by chance play a strategy which wins while the said opponent is playing. For example, if GreedyDistanceTeam plays against DomOneHuggerTeam and does well against it then JuKeCB will have received training against DomOneHuggerTeam. A full training set is one where JuKeCB observes each of the fixed strategy teams playing against every other. Ideally, after that training set JuKeCB will be able to play against any of the fixed strategy teams. For each of the tests run in the following experiments the following game variables were used: 2 teams per game, 3 bots per team, 4 domination points per map, games were until the sum of both team scores is 50,000, and each set was played 5 times (all graphs plot the mean of the set).

6.2 Experiment #1: JuKeCB vs. Fixed Teams

When JuKeCB observes a pair of teams competing against one another, then, assuming that it has observed no other teams before, it is intuitively clear that it should perform well if it plays immediately afterwards against the losing team. However, as it sees more and more teams and the case library grows, it is conceivable that JuKeCB's performance may degrade. The main reason is that JuKeCB is not told against whom it is playing and, hence, it needs to recognize the opponent's strategy and the situation as one similar to one encountered before. As more teams and situations are observed, conflicts may arise in the detection of the "right" opponent. We hypothesize that this will not happen. More precisely we hypothesize that the

Table 1. The fixed strategy teams

Team Name	Static Policy
DomOneHuggerTeam	Sends each of its bots to Domination Point one. (Bot Destination = DomPoint 1)
FirstHalfOfDomPointsTeam	Sends bots to the first half of the Domination Points. If it has bots remaining, they patrol between the first half of the Domination Points. (Bot Destination ≤ #Points/2)
SecondHalfOfDomPointsTeam	Sends bots to the second half of the Domination Points. If it has bots remaining, they patrol between the second half of the Domination Points. (Bot Destination ≤ #Points/2)
GreedyDistanceTeam	Sends bots to the closest unowned Domination Point; if all are owned, goes to a random one. (Bot Destination = Closest Unowned)
Smart OpportunisticTeam	Sends one bot to every unowned Domination Point (without repeating). If not enough unowned Points, it sends multiple to the same point. (Bot Destination = Different Unowned)
EachBotToOneDomTeam	Sends each of its bots to a separate Domination Point until all Domination Points are accounted for. If it has additional bots remaining, it loops. (Bot Destination = A specific Point)

performance of JuKeCB improves as it observes more teams. For this experiment, JuKeCB will be watching and playing games against the fixed strategy teams.

The training set and testing set for this experiment is intertwined. JuKeCB team will play against every fixed strategy team in the training set before, after and during training. The following illustrates the order in which games are played:

Test: JuKeCB plays Opponent1
Train: JuKeCB watches Opponent1 play Opponent1
Test: JuKeCB plays Opponent1
Train: JuKeCB watches Opponent1 play Opponent2
Test: JuKeCB plays Opponent2

This continues until JuKeCB has played against every opponent under all possible training conditions.

On a map with four domination points JuKeCBTeam was able to beat almost every fixed policy team with relative ease. Its performance increase was measured by observing the score difference in the pre-training and post-training games. JuKeCB-Team only fails to beat one team, SmartOpportunisticTeam. It still increases in performance considerably after having trained, however it is not enough to win. Without further training, JuKeCB is unable to counter the strategy that SmartOpportunistic-Team is able to employ. For illustration purposes, Fig 2 shows the mean score of a five game set between GreedyDistanceTeam and JuKeCBTeam. The left graph shows the results of a JuKeCBTeam which had not yet trained against GreedyDistanceTeam. In this game, GreedyDistanceTeam beat JuKeCBTeam by approximately ten thousand points. After having completed the training set, JuKeCBTeam was able

to win by about three thousand points as seen in Fig 2 right. The difference displayed in both graphs is statistically significant (TTest scores: 99%). These results are typical for most of the other games ran. We repeated the experiments on a second map and the results were consistent with these results.

These results are consistent with our hypothesis that as JuKeCB sees more opponents, its performance increases. In this experiment, JuKeCB plays against every team with either an empty case base or one that is not very helpful for the match at hand. During its first match against any team, it is forced to attempt to compare it to fairly dissimilar cases or possibly to no case at all. It must use the ineffective default equiprobable policy. As such, against most teams JuKeCB fails to perform well. However once JuKeCB has been able to watch even just a few games involving the strategies that its enemy is employing its performance increases very quickly. The more cases that JuKeCB has at its disposal the better it performs.

In order to better assess our hypothesis, we tested Retaliate [13], a reinforcement learning agent, and CBRetaliate [7], an extension to Retaliate that uses CBR. Both were trained with the same sets as JuKeCB. Fig 3 shows how Retaliate (left) and CBRetaliate (right) performed after training and under the same game conditions (Number of turns in the game, number of domination points and number of bots). The abstract model makes the game look deceivingly simple but beating some of the hard-coded opponents are difficult for AI agents; the Q-learning agent Retaliate and its case-based extension were beaten by the hard-coded opponent Greedy (Fig 3). In [14] an HTN planning agent was also beaten soundly by Greedy and the reactive planning approach showcased in that paper barely ties with Greedy. Given this evidence, it is remarkable how well JuKeCB did (Fig 2, right).

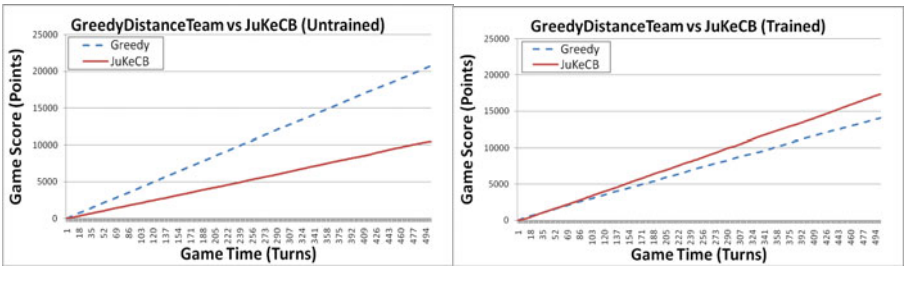


Fig. 2. Results (left) before and (right) after training versus GreedyDistanceTeam

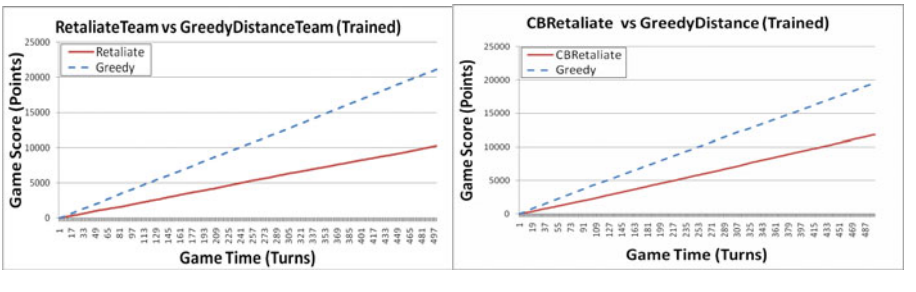


Fig. 3. Performance of Retaliate (left) and CBRetaliate (right), both after training, versus GreedyDistanceTeam

Retaliate and CBRetaliate were clearly beaten in this short game. Retaliate excels at learning another team's strategy, however it requires a lot of game time to fully adapt to its opponent. It is much more effective at long games where its algorithm has more time to work. In this short game (50,000 turns) Retaliate does not have adequate time to learn to adapt to its opponent. CBRetaliate performed better since it can jumpstart the RL process but still was beaten.

JuKeCBTeam is not without flaws however. As the game becomes more complex, such as the games run on a map with more than four domination points, the case base becomes more complicated. There are more features which need to be taken into consideration and the amount of possible cases goes up considerably. This seems to be the main reason that JuKeCBTeam cannot beat SmartOpportunisticTeam. Since SmartOpportunisticTeam has a fairly sophisticated fixed strategy, it takes quite a few set of cases to fully define all strategies that it might employ, especially in larger maps. On the whole this experiment was a success for JuKeCBTeam. It not only showed that the underlying hypothesis was correct, but also that under certain conditions it is able to outperform reinforcement learning techniques.

6.3 Experiment #2: JuKeCB vs. DynamicTeam

In the previous experiment, JuKeCB performs well against different fixed strategy teams. Now we want to test JuKeCB versus a team that has a dynamic strategy. For this purpose we built DynamicTeam, which changes its strategy as follows:

- *First 1/3 of game:* DynamicTeam employs FirstHalfOfDomPoints Strategy
- *Second 1/3 of game:* DynamicTeam employs EachBotToOneDom Strategy
- *Final 1/3 of game:* DynamicTeam employs Greedy Strategy

Our hypothesis is that JuKeCB will quickly adapt to the changes in strategy and will continue to win by exploiting the weaknesses of each fixed strategy employed by DynamicTeam. The JuKeCB agent first plays DynamicTeam completely untrained, and slowly trains and re-plays DynamicTeam until it finally plays is with a fully trained case base.

The results were as follows: during its first match against DynamicTeam, JuKeCB handles well the first and second stage, consistently keeping a higher score than DynamicTeam. Once the Greedy strategy begins, DynamicTeam begins to pull away and eventually wins by a large margin (Fig 4, left). In the graph of the final game, JuKeCB outperforms DynamicTeam in all 3 stages (Fig 4, right).

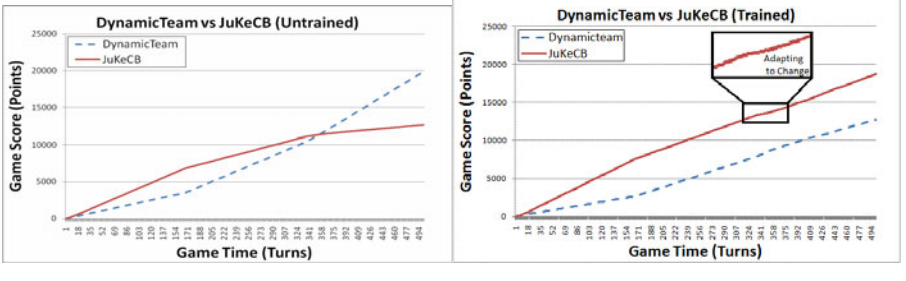


Fig. 4. Results (left) before and (right) after training versus DynamicTeam

From these two graphs we observe the occurrence of learning and adaptation. During the first match, JuKeCB performs well against DynamicTeam's first two strategies. This is expected, as the teams are static and JuKeCB can easily exploit weaknesses in their strategies. However, upon playing Greedy in the third round, JuKeCB performs poorly and ends up losing. Greedy is more dynamic and thus a more challenging opponent for JuKeCB to play for the first time. As the experiment continues and JuKeCB plays and observes more and more games, it begins to perform better; the gap between our score and DynamicTeams' score widens and becomes more dramatic over time. It quickly begins to win over DynamicTeam's first two strategies, and gradually performs better against DynamicTeams third strategy, Greedy. On the final match with DynamicTeam, JuKeCB clearly performs better against all three phases of the DynamicTeam, and wins by a very large margin. This result is statistically significant (TTest score: 99%).

Perhaps the most intriguing point made by the graphs is the quick adaptability of JuKeCB. The score lines form angles at ~350 and ~650 for both teams. This means that when DynamicTeam changed its strategy, JuKeCB immediately adapted and began to exploit the new opponent's weaknesses without delay. Not only has JuKeCB yet again proven its effectiveness as a learning agent, but it has also showcased its ability to adapt to any strategy no matter how quickly it is presented and/or changed.

6.4 Experiment #3: JuKeCB vs. Untrained Team

A trained JuKeCB team performs well against fixed strategy teams after observing them play and it doesn't perform well against them when it has not seen them. However, these experiments do not tell us how JuKeCB's ability to play against teams it has never seen before after it has observed with a large variety of other teams. We designed an experiment in which one of the team is hidden (i.e., it does not appear in the training set) and then play versus that opponent after it has trained with the full training set. For this experiment we selected Greedy because it's the hardest opponent that JuKeCB is able to beat after seen it. Our hypothesis is that JuKeCB will have a good performance versus Greedy after been subject to a large training set and despite not having been trained with Greedy itself before. Fig 5 shows the average score of the match of JuKeCB against Greedy team after it has been trained with all opponents except Greedy itself. JuKeCB's outperforms Greedy at around 130 turns and continues to widen as the game progresses. This is evidence that while JuKeCB has never seen this hidden team, it has recognized a similar situation in which a strategy performs better than the hidden team's (Greedy's) general policy.

This experiment illustrates what is perhaps one the most notable properties of JuKeCBR specifically, and one that has observed with other CBR systems: their capability to generalize from the concrete instances it lazily trains from by means of exploiting similarity metrics and adaptation; in our work the concrete instances are stochastic policies. JuKeCB didn't simply mimic Greedy FixedPolicy team, nor did it guess until something worked (as a reinforcement learner would do). Instead, it looked over its case base (developed during training) and looked for previous cases that looked similar to this new team's strategy. Upon finding a good one, JuKeCB immediately uses the Case(s) and updates current ones. In other words, once JuKeCB

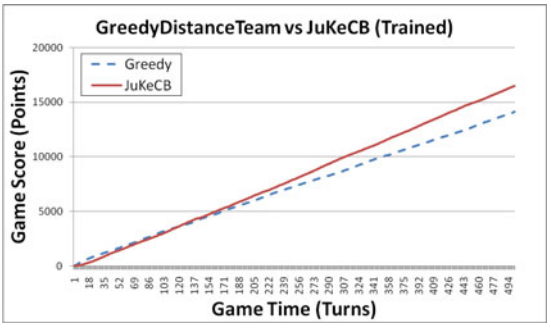


Fig. 5. Average score of JuKeCB against Greedy team after training

finds a good case against this hidden team, it keeps pushing these weaknesses until they can no longer be exploited or it reaches victory.

Fig 6 combines JuKeCB's performances against the Greedy team. The 3 sloping lines represent JuKeCB's score against Greedy for that training set (untrained, hidden, and fully trained). For each of JuKeCB's lines, Greedy Team has a corresponding line in the form of y={MAX_SCORE} representing Greedy's final score against JuKeCB for that training set. Greedy outperforms the untrained JuKeCB agent. Upon training against all teams, including Greedy, JuKeCB performs much better than Greedy, and performs slightly better after a full training set in which it finally has seen Greedy Team play.

7 Related Work

We have discussed some related works in the introduction. Aside from those, our work is related to case-based planning [15]. Case-based planning (CBP) stores and reuse plans for solving new problems. These work frequently assume a complete domain theory is known, and hence the applicability conditions of each action in the plan can be evaluated in the current situation to determine apriori (i.e., deterministically) if it can be executed. Even CBP works that does not make the assumption about a complete domain theory (e..g., CHEF [16]) rely on the outcome of the action been deterministic. For example, CHEF knows apriori that changing one ingredient for another one "works".

Reinforcement learning systems using techniques such as Q-learning (e.g., Retaliate [13]), can be modified to produce a stochastic policy because the underlying reinforcement learning (RL) theory is amenable to stochastic policies [12]. We do not aim at learning by trial-and-error as in RL. In fact, in our experiments JuKeCB learns stochastic policies by observing others play. Unlike RL, adapting to an opponent is not done by trying to adjust the policy but instead by reusing a different policy when the game is not going as expected. In existing approaches combining CBR and RL [8], the policies stored represent state-action values (which reflect anticipated future rewards of taking those actions) rather than observed state-action probability distributions as in our case.

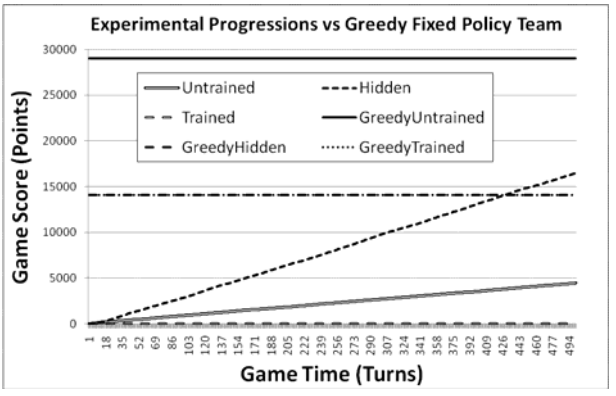


Fig. 6. Combining JuKeCB's performances against the Greedy team

8 Conclusions

We described a general framework for observing, capturing, and reusing stochastic policies. Stochastic policies are a natural representation artifact for situations in which a learning system observes an agent taking different actions when reaching the same state and the reasons behind those choices cannot be discerned. We implemented this framework in JuKeCB, a CBR system that imitates the stochastic policies it has observed in the context of a domination-style game. The efficacy of the approach was tested in three experiments. The first demonstrated that, after observing the play of teams that use a fixed strategy, JuKeCB is able to beat most of them. The second showed that, after training, JuKeCB can beat a team that changed its strategy during the same episode. The final experiment shows JuKeCB can be successful even against opponents it has never observed.

In our experiments we noted that as the number of cases stored in the case base increases significantly, the reduction in speed at which JuKeCBTeam updates its own strategy became more noticeable. Since many teams change their strategies very quickly, JuKeCBTeam must update its strategy often. Scanning through the entire case base every time a strategy change is needed can become quite costly. In the near future we will explore techniques for case base maintenance to reduce the retrieval cost of scanning the case base. For other future work, we will extend our analysis in more quantitative directions, including an analysis of different approaches to compute similarity measures between stochastic policies, such as probabilistic inference.

Acknowledgements. This work was supported in part by NSF grant 0642882.

References

1. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs (1995)
2. López de Mántaras, 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 case-based reasoning. Knowledge Engineering Review 20(03), 215–240 (2005)

3. Fikes, R.E., Nilsson, N.J.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2, 189–205 (1971)
4. Epstein, S.L., Shih, J.: Sequential Instance-Based Learning. In: Mercer, R., Neufeld, E. (eds.) Canadian AI 1998. LNCS, vol. 1418, pp. 442–454. Springer, Heidelberg (1998)
5. Aha, D., Molineaux, M., Ponsen, M.: Learning to win: Case-based plan selection in a real-time strategy game. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 5–20. Springer, Heidelberg (2005)
6. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 164–178. Springer, Heidelberg (2007)
7. Auslander, B., Lee-Urban, S., Hogg, C., Muñoz-Avila, H.: Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 59–73. Springer, Heidelberg (2008)
8. Sharma, M., Holmes, M., Santamaria, J.C., Irani, A., Isbell, C., Ram, A.: Transfer learning in real-time strategy games using hybrid CBR/RL. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 1041–1046 (2007)
9. Bridge, D.: The virtue of reward: Performance, reinforcement and discovery in case-based reasoning. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, p. 1. Springer, Heidelberg (2005)
10. Molineaux, M., Aha, D.W., Sukthankar, G.: Beating the defense: Using plan recognition to inform learning agents. In: The Proceedings of the Twenty-Second International FLAIRS Conference, pp. 257–262. AAAI Press, Sanibel Island (2009)
11. Aamodt, A., Plaza., E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
12. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
13. Vasta, M., Lee-Urban, S., Munoz-Avila, H.: RETALIATE: Learning Winning Policies in First-Person Shooter Games. In: Proceedings of the Seventeenth Innovative Applications of Artificial Intelligence Conference (IAAI 2007), pp. 1801–1806. AAAI Press, Menlo Park (2007)
14. Munoz-Avila, H., Aha, D.W., Jaidee, U., Klenk, M., Molineaux, M.: Applying goal directed autonomy to a team shooter game. In: Proceedings of the Twenty-Third Florida Artificial Intelligence Research Society Conference. AAAI Press, Daytona Beach (to appear, 2010)
15. Muñoz-Avila, H., Cox, M.: Case-based plan adaptation: An analysis and review. IEEE Intelligent Systems 23(4), 75–81 (2008)
16. Hammond, K.J.: Case-based planning: Viewing planning as a memory task. Academic Press, San Diego (1989)

The Utility Problem for Lazy Learners - Towards a Non-eager Approach

Tor Gunnar Houeland and Agnar Aamodt

Department of Computer and Information Science,
Norwegian University of Science and Technology,
NO-7491 Trondheim, Norway
{houeland,agnar}@idi.ntnu.no

Abstract. The utility problem occurs when the performance of learning systems degrade instead of improve when additional knowledge is added. In lazy learners this degradation is seen as the increasing time it takes to search through this additional knowledge, which for a sufficiently large case base will eventually outweigh any gains from having added the knowledge. The two primary approaches to handling the utility problem are through efficient indexing and by reducing the number of cases during case base maintenance. We show that for many types of practical case based reasoning systems, the encountered case base sizes do not cause retrieval efficiency to degrade to the extent that it becomes a problem. We also show how complicated case base maintenance solutions intended to address the utility problem can actually decrease the combined system efficiency.

1 Introduction

A concern for case-based reasoning (CBR) systems that are deployed and will keep running for many years is how the system will change over time. The capability for learning is an important aspect of many such systems, but by its very nature the act of learning will change the system from its current state to something that is partially unknown. There will normally be a desirable improvement from learning, but its effects may also include unwanted changes.

One of these changes is that as the system's knowledge increases, the space needed to store the knowledge and the time it takes to process it also increases. The storage space and time taken to process the knowledge will increase without bounds, and eventually go far beyond the space and time taken by the original system.

Because of this behavior, there will always be some theoretical point where the total performance of the system is degraded by adding additional knowledge. Many different methods have been suggested to address this problem, which is often included under the wider umbrella of case base maintenance.

The maintenance methods used to address this problem can be split in two: maintaining the case base indexes, and maintaining the case base contents.

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 141–155, 2010.

Indexing methods work by quickly identifying the relevant parts of the knowledge base, which can allow a system to examine only a small fraction of its available knowledge.

Methods for maintaining case base contents aim to reduce the size of the system's case base, making it both faster to examine since there is less knowledge, and reducing the storage space needed to store the data.

The *utility problem* in learning systems occurs when knowledge learned in an attempt to improve a system's performance degrades it instead [17,3]. This is most common in speed-up learning systems, where the system's knowledge is used to reduce the amount of reasoning required to solve a problem.

For pure speed-up learners it is assumed that there is already a slower method available for finding an acceptable solution to the problem. From a simplified perspective, cases in a CBR system may be viewed as a form of speed-up knowledge, where storing, retrieving, and adapting cases provides for more efficient problem solving than first-principles or model-based methods [15,9]. The goal is to produce acceptable results more quickly, and hence the time taken to perform the system's reasoning is of primary concern.

Case-based reasoning is also known as a *lazy* approach to learning and problem solving. The very essence of lazy learning is that choices regarding the solution of a problem will be postponed as long as possible, until the problem query is posed and as much information as possible is available.

Building index structures and deleting cases from the case base are both eager methods, and hence somewhat counter-intuitive to the CBR idea. Indexing and deletion reduce the amount of knowledge available, without knowing whether that information could have been useful for solving a future problem. Hence, indexing and deletion methods should only be used when they are really needed. In the work reported here we explore the hypothesis that for a wide range of CBR application systems, addressing real world problems, they may not be needed.

This work is situated within our research on a new architecture for meta-level reasoning and introspective learning [10]. A less eager approach to indexing and case deletion, if feasible, will allow more freedom to the meta-level reasoner.

This paper examines the utility problem in CBR as it applies to most of the CBR systems we have built and are building today, with case bases of reasonable sizes. Based on existing literature and several example utility models for case-based reasoners, we show that neither indexing nor case deletion policies are necessary for a wide range of CBR systems, and in fact can be detrimental to a CBR system's overall goal.

In section 2 the background of the utility problem is summarized, and some earlier research results relevant to our work is discussed. This is followed in section 3 by an analysis of the utility concept and a comparison of three different speed-up learner scenarios. Section 4 discusses the use of indexing strategies for speeding up retrieval. Section 5 discusses the benefit of limiting case base size and shows how the cost of advanced maintenance methods may negate the benefits of a reduced case base through an illustrative experiment. Concluding remarks end the paper.

2 Background and Related Research

A substantial amount of research has addressed the utility problem in CBR [19,13] Over the past few years there has been a broad range of research addressing specific issues of case deletion, addition and efficient indexing [20,25,2]. Wilson and Leake [14] present a thorough examination of the dimensions of maintenance strategies and a survey of maintenance research in terms of those dimensions.

The utility problem is also referred to as the "swamping problem". In their seminal paper on the utility problem in CBR, however, Francis and Ram [9] refer to swamping as one of three types of utility problems in CBR. Swamping is the phenomenon that a single unit of knowledge added to the knowledge base - i.e. a case added to the case base - improves problem solving speed at the individual level, because a more similar case to a query may be found, while the performance over the knowledge base as a whole is degraded due to increased retrieval and matching overhead. Swamping is also referred to as the "core" utility problem, which is probably why the two have become synonyms. The other types of utility problems listed are the "expensive chunks problem" and the "search-space utility problem". The first refers to the problem of performance degradation at the level of individual knowledge units, because of the matching cost of a single unit (a macro operator, for example). The second refers to degradation due to increased complexity of search control knowledge, of particular relevance to the learning of meta-level knowledge. Although all three have relevance for CBR systems, the focus in this paper is on the swamping problem.

The processing power available for modern CPUs continues to increase, continually reducing the problems associated with large amounts of data, and allowing large case bases to be handled which would have been considered impossible for a reasonable budget 10 years ago. However, this is typically only true for polynomial-time algorithms, and especially for algorithms that run in (sub-)linear time. Other algorithms that have an exponential running time are unlikely to ever be practical for large inputs, such as many graph-matching algorithms. This means that the "expensive chunks problem" is unlikely to be greatly affected simply by advances in computer hardware, since they are NP-hard in the worst case [23]. On the other hand these advances affect the degradations experienced due to the swamping problem, since algorithms for searching the knowledge base are typically either O(N) or O(log N).

The main cause of the swamping problem is that retrieval time will increase with a growing case base while adaptation time will decrease. The latter is due to a smaller distance between a query case and the best matching case on average. As the case base grows retrieval time is likely to dominate, however, which leads to a logarithmic or higher increase in processing time. For a speed-up learner this increase may negate the efficiency gains during adaptation and eventually even cause the system to be slower than the underlying "slow" solver. The speed increase has been reported as being substantial in some systems where this logarithmic increase has no significance, for example a thousand-fold increase in CASEY [12]. Other machine learning systems have reported more modest

figures, such as a factor up to six in SOAR and ten in PRODIGY/EBL [9]. Systems with these speed-up figures that are left running and collecting experience for a long time will gradually slow down, and eventually be slower than the unassisted "slow" solver [9].

The trade-off is perhaps most clearly illustrated and explored for some types of control rule learning (CRL) systems, where every individual control rule is guaranteed to have a positive utility (improve performance) but, in concert, also have a negative utility (degrade performance) [9,17].

Francis and Ram [8] describe a framework for modeling CRL and CBR systems, and for analyzing the utility problem for these reasoning models. The authors identified that the retrieval costs for CBR increase without bound, and that in the limit, CBR systems will get swamped and the cost of retrieval will outweigh the benefits of case adaptation. The authors conclude that CBR is nevertheless relatively resistant to the utility problem, compared to CRL systems, because the cases have the potential to greatly reduce the amount of problem solving needed, and that the cost of retrieval is amortized across many adaptation steps.

Smyth and Cunningham [19] examine the utility problem in CBR through experimenting with a path finding system that combines Dijkstra's algorithm with CBR for speed-up. They show how case-base size, coverage and solution quality affect the utility. The authors find that varying these characteristics significantly alters how the system responds to different case base sizes. This indicates that the need for case deletion and indexing is strongly related to requirements for solution quality and retrieval time. We will discuss this issue later in the paper.

A proposed policy for case deletion with minimal effects on case base competence was presented by Smyth and Keane [20]. A footprint-driven method that also accounted for utility gave the best test results. An alternative method was proposed by Zhu and Yang [25] with the emphasis on careful addition of new cases rather than on deleting old ones. Their method performed better than the footprint deletion method, under some conditions.

A deliberate case addition process may be viewed as a form of case deletion as well (i.e. by not adding cases that might otherwise have ended up in the case base). From the perspective of more or less eager case base maintenance methods, assuming an existing case base and only incremental updates, a considerate case addition policy will generally be a more lazy approach than a deletion approach. An even lazier approach is of course to keep all the cases, and rely on the retrieval and adaptation methods to do the "deletion" on the fly.

One solution to the indexing problem is to apply suitable methods for refining indexing features and matching weights. Jarmulak et al. [11] use genetic algorithms to refine indexing features and matching weights. Another approach is to view this problem from a meta level perspective, and use introspective learning techniques to handle the refinement of indexes, triggered by retrieval failures [7,4].

We are developing an introspective architecture for lazy meta-level reasoning characterized by creating and combining multiple components to perform the system's reasoning processes [10]. Each component represents part of a reasoning method and its parameters using a uniform interface, which can be used and modified by the meta-level reasoner. This will enable selecting which reasoning methods to use after a description of the problem to be solved is known.

Although it is known that laziness and indexing strategies impact each other [1], we have not come across work which expressly views indexing also from a maximally lazy perspective. This means to avoid building such indexes at all for the purposes of improving search efficiency, as a way to avoid committing to eager indexing decisions. Indexes may still be built, but in order to improve matching quality only. A typical example is a structure of abstract indexes, which interpret lower-level data in the input cases in order to achieve an improved understanding of the case information and hence more accurate solutions.

Watson [24] discusses case base maintenance for *Cool Air*, a commercially fielded CBR system that provides support for HVAC engineers. The system retrieves cases for similar previous installations, and Watson explains the case-base maintenance required for the system, most notably the removal of redundant cases from a rapidly growing case base. By the nature of the application domain, installing the system in different locations means that the same product will operate under several similar conditions, and result in very similar cases being created. In their client-server design the server selects a small set of cases to be sent to the client, and the client then uses the engineers' custom-tailored similarity measure to rank them. The problem with this design is that many redundant cases are sent to the client, and it was decided to remove the redundant cases from the case base. Although primarily motivated by other purposes than the utility problem, this type of case redundancy avoidance in client-server architectures seems to be generally useful.

3 Describing and Analyzing Utility

In general, the utility of a reasoning system can be expressed as the benefit it brings, minus the costs associated with the system. The benefits are typically achieved over time while the system is in operation, while a large part of the costs of the system are up-front, such as gathering expert knowledge, developing the system and integrating it with the organization that will be using it. Another large source of costs is the continued maintenance of the system, which is often overlooked but should be included when the system is initially planned [24].

The benefit of a generated solution can be measured as its usefulness for addressing the task at hand, which is primarily characterized by the solution accuracy and solution time. When considered in this manner, the time taken to solve a problem can be naturally expressed as a lessened benefit. Then the direct costs associated with the problem solving are related to the resources spent computing them, which are very small for typical systems.

For clarity, we define our use of the terms as follows:

General System Utility: GSU The combined benefit of the reasoning system, minus the associated costs. This consists of the solution usefulness for the solutions generated by the system, the usability of the system for human operators, and the costs associated with developing, running and maintaining the system.

Solution Usefulness: SU The benefit of a generated solution, estimated as a function of solution accuracy, solution time and resource costs, not including human factors.

Solution Accuracy: SA The accuracy of a generated solution, which depends on both the case base and the methods used by the system.

Solution Time: ST The time it takes the system to solve a problem.

Resource Cost: RC The cost of solving a problem. This is primarily the time spent operating the systems, but also includes hardware costs that can potentially be significant for long computations in large systems.

Using the same approach as Smyth and Cunningham [19] for analyzing these concerns, we assume that the solution accuracy increases with a larger case base, and that the solution time is divided into two parts: retrieval time, which increases with a larger case base, and adaptation time, which decreases with a larger case base (or stays the same). By noting that the retrieval time increases with the size of the case base, which is unbounded [17], that the retrieval time is similarly unbounded for any retrieval approach that can potentially reach all the cases, and that the reduction in adaptation time is bounded (since it can never be faster than 0 time units), we see that there will be some point where adding further cases to the case base will slow down the total solution time.

The utility problem is most easily analyzed for speed-up learners, where the solution time is of primary importance. We explore three different scenarios for speed-up learners with different time complexities, and examine the different amounts of utility degradation experienced by modeling the solution time **ST** as a function of case base size N. We use a simplified model of a speed-up learner, where the system will always produce the same correct answer, and the only criterion for the solution's utility will be its solution time. We model the solution usefulness $\mathbf{SU} = 1/\mathbf{ST}$ for such systems, ignoring the solution accuracy.

As reported by Smyth and Cunningham [19], when increasing the size of the case base, the solution time for a case-based speed-up learner will typically consist of an initial rapid improvement, followed by a continuing degradation as the retrieval part begins to dominate the total time spent to solve the problem. For a typical speedup-learner, the total solution time will initially be approximately monotonically decreasing, followed by an approximate monotonic increase, and the optimal solution time and preferred case base size will be where the rate of increase in retrieval time matches the rate of decrease in adaptation time. The exact behavior of the total solution time and the order of magnitude of this preferred case base size depends greatly on the algorithmic complexity classes of the algorithms used, and in general there is no guarantee that the solution time will follow this pattern at all. The exact characteristics also depend on how the case base content is created

and maintained, since a maintained case base can have different case distributions and behave quite differently than an approach retaining all cases.

Fig. 1 shows the solution time for an idealized speed-up learner, where retrieval is a linear search through the case base, adaptation time is proportional to the distance to the retrieved case, and there is no overhead: $\mathbf{ST} = N/5 + 100/N$. In this situation the efficiency of the system initially improves quickly, and then starts degrading slowly as the increased time to perform retrieval eventually becomes greater than the time saved during adaptation. In systems with retrieval and adaptation algorithms displaying this kind of behavior, the solutions will be generated most quickly when retrieval and adaptation times are approximately equal, since that coincides with their derivatives having the same magnitude and opposite signs. Smyth and Cunningham [19] report very similar results to this speed-up learner scenario from experimenting with the PathFinder system.

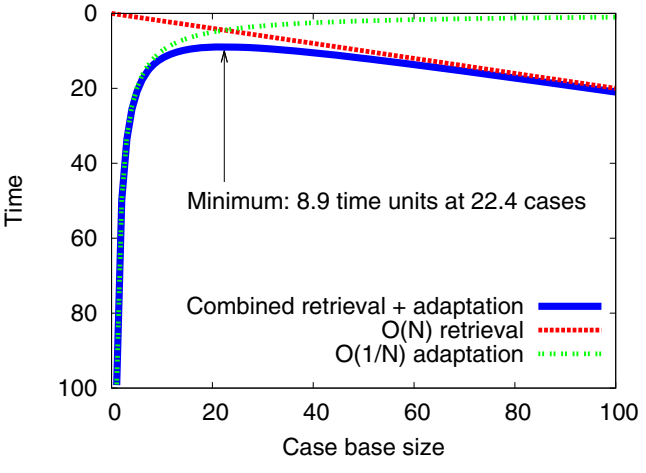


Fig. 1. Retrieval, adaptation and combined solution time compared to case base size for a speed-up learner using an O(N) retrieval algorithm and a O(1/N) adaptation algorithm

Fig. 2 shows the solution time for similar speed-up learning systems, but using different algorithms. The graph on the left uses an indexing scheme for retrieval that causes the retrieval step to run in O(log N) time, and with a comparatively larger constant factor (5 vs 0.2) than the previous example: $\mathbf{ST} = 5 * ln(N) + 100/N$. With this change to the retrieval function, the slowdown associated with adding more cases to the case base happens very slowly, and the total solution time for the full case base is just 20% slower than the minimum solution time, even though the full case base is 5 times larger. The graph on the right shows a much more drastic increase in solution time. A more complicated O(N log N) case retrieval algorithm is shown that compares the retrieved cases against each other, and case adaptation has a significant overhead of 20 time units: $\mathbf{ST} = N * ln(N) + 100/N + 20$. In this situation the combined solution

time increases quickly as more cases are added beyond the optimal amount. For domains where this type of algorithm is desirable, a similar increase in solution quality would be expected, otherwise the case base should be kept relatively small through aggressive case base content maintenance. Due to the very limited number of cases the system can handle before slowing down, these latter types of algorithms appear to be a poor choice for pure speed-up learners, although the constant factors could potentially be of very different magnitudes for some domains. These alternative combinations scale very differently, and illustrate the importance of examining the algorithms used when analyzing the effect of larger case bases.

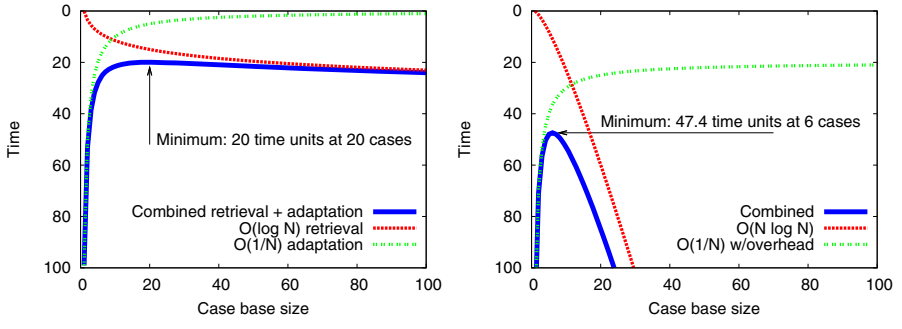


Fig. 2. Retrieval, adaptation and combined solution time compared to case base size for two other speed-up learner scenarios: On the left, using an O(log N) retrieval algorithm and a O(1/N) adaptation algorithm. On the right, using an O(N log N) retrieval algorithm and a O(1/N) adaptation algorithm with overhead. The difference in shapes illustrate the importance of considering the specific algorithms used when examining the utility problem.

For general case-based reasoners, the utility function becomes much more complicated. When still examining only the usefulness of solving one problem while the system is running, and just moving away from the simplified speed-up learner model, we need to also include the accuracy of the solution in our evaluations. The impact the accuracy has on the usefulness of the system will vary greatly based on the domain and the specifics of the application.

Smyth and Cunningham [19] report empirical results from the PathFinder system, where at one point the quality of solutions increased from 94% to 96%, while the solution time increased by 50%. Whether such a trade-off is considered beneficial or not depends on the application and the initial starting values for the retrieval accuracy and solution time. For a speed-up learner this might be unacceptable, while for many other applications a 33% reduction in errors (from 6% to 4%) at the expense of waiting longer or using two computers instead of one would be a great improvement. As in the fielded *Cool Air* system, the solution time might simply be considered acceptable and not be a problem that has to be addressed at all, and then a larger case-base might be purely beneficial without encountering this trade-off.

4 Indexing vs. No Indexing

There are many possible indexing strategies for speeding up retrieval, and their effects are highly dependent on the domain and the similarity measures used. Selecting a good indexing strategy can often require significant expert knowledge, although various automated methods exist [11,7,4] for refining the indexing strategy. This can be particularly helpful for index maintenance, since an appropriately chosen method can potentially handle most indexing maintenance operations without manual intervention.

However, from a lazy learning perspective, an indexing structure that allows the retrieval method to only consider a subset of the case base is an eager optimization, which is made before all the potentially useful information about the target problem is known, and is therefore not always appropriate.

In systems that handle thousands of cases or less, the processing time is not necessarily a critical factor, and might very well increase slower than the increase in processing power available over time, i.e. the solution usefulness **SU** is primarily a function of the solution accuracy **SA**. This is particularly true for knowledge-intensive CBR, where less than a hundred cases is common (but the time required to perform individual similarity measurements is often extensive).

By matching with every case in the case base, the retrieval method is guaranteed not to miss any cases in the case base, and without needing to regularly maintain the indexing structures.

Unlike pure speed-up learners, producing results as quickly as possible is rarely the main concern for fielded CBR systems. The cost of developing and maintaining a system is usually much larger than the cost of using and maintaining the hardware it runs on, and the direct resource cost **RC** can sometimes even be considered negligible. CBR systems with extensive reasoning also do not usually act as speed-up learners, since they can actually produce better solutions with a larger case base. For such systems the utility problem is a trade-off between solution quality and the efficiency degradation associated with a large case base [19].

As an alternative to purely eager indexing, footprint-based retrieval [21] allows for a kind of mix of indexing and similarity measurements. The indexing is eagerly pre-generated based on competence evaluations performed before the input problem is known. During retrieval, the index is used to quickly identify what is believed to be the most relevant knowledge items, which are then evaluated lazily with full similarity-based knowledge. Although the retrieval is less efficient than purely eager methods, this partially lazy approach can produce good results for some domains.

For the commercially fielded *Cool Air* [24] CBR system, processing time was much cheaper for the company than consultancy time for developing the system. The efficiency slow-down associated with an increasing case base did not become a problem, even though the case base doubled over two years.

In another commercially fielded system, the DrillEdge system for fault prediction in oil well drilling [22], case retrieval is an automatic process triggered by a pattern in the continuous stream of drilling data. The cases are indexed by abstract features derived from the numerical drilling data stream. This is done

in order to improve the matching process for retrieval of the most relevant cases. The indexes are not used to improve retrieval efficiency - the case base is always searched sequentially. As long as the number of cases is in the range of hundreds, this is not regarded as a performance problem.

For some applications, like the one just mentioned, solution quality is of utmost importance. In oil well drilling, the costs of time lost due to unwanted events can be huge, since drilling operations typically cost around 200 000 USD per day [18]. In this case the value of the positive utility associated with higher quality solutions is of a different order of magnitude than typical costs of negative utility caused by decreased efficiency. For the knowledge-intensive oil well drilling system, the main cost of a large case base is the amount of expert knowledge required, not the computer systems it runs on.

As an alternative to performing eager indexing at all, a two-step approach [6] to case retrieval has often been employed for systems with expensive retrieval operations, e.g. for knowledge-intensive CBR systems. This consists of first using a fast and resource-efficient scan through the case base to identify relevant knowledge, and then performing more advanced (and comparatively slow) reasoning for this restricted set of cases. This is conceptually very similar to indexing, but is done using a lazy approach, entirely after the input problem query is known.

In this way, there is no need to update indexing structures, and more powerful methods can be performed for identifying relevant knowledge when you already know the problem to be solved. Similarity assessment is usually very important for CBR systems, because there is often no easy way to model the structure of the entire problem space, and there may even be no expert knowledge that directly applies to all problem instances in general. Using similarity measurements to locally identify relevant knowledge for a specific problem is thus likely to produce better results than pre-generated structures.

To perform large numbers of similarity assessments quickly, it might be necessary to increase the amount of computational resources available by examining the cases in the case base in parallel. Many modern distributed computing frameworks available for processing very large data sets in parallel are based around ideas similar to the MapReduce [5] algorithm. MapReduce works by first chopping up a problem into many parts, then distributes each of these parts across a cluster of computers and each node processes only a subset of the problems. The answers are then returned to a master node, which combines them to create a final answer for the entire problem. This is very similar to parallel case retrieval, where each case is assigned a similarity score and then ranked at the end.

While this form of case evaluation producing independent results for every query-to-case comparison does not let us express the most general forms of case retrieval, they are sufficient for most systems that are used in practice. The kind of similarity assessment methods supported by this approach are also typically more flexible than those supported by common indexing schemes. Avoiding the need for additional expert knowledge that is often required to create a good indexing solution is another potential benefit of this approach.

For commercial applications, these kinds of large parallel processing frameworks are typically used to process terabytes or petabytes of data, and provide a possible means to perform full sequential evaluations for the complete case base during retrieval even for very large case bases.

5 Case Base Maintenance

Case-based reasoning system maintenance is important and can involve processes that modify all parts of the system, including updates to each of the knowledge containers. Most often this includes reducing the number of cases in the case base, which is primarily useful for two purposes:

- Reducing the size of the case base used by retrieval methods, which can make retrieval faster.
- Reducing the space required for storing the case base.

Various case base content maintenance algorithms exist for reducing the size of the case base, while optimizing the remaining cases according to some criteria. A fast and simple content maintenance strategy is to delete cases at random, which has been reported to produce good results [16]. Since the case base essentially becomes a random subset of all encountered cases, or effectively just a smaller case base, this strategy also has the added benefit of maintaining the same case distribution as the encountered cases, on average. Other approaches for content maintenance usually examine the relations between cases in the case base, and e.g. attempt to maximize the coverage of the remaining cases in the reduced case base through adding, deleting or combining cases [20].

We conducted a set of experiments to compare these two approaches, using a random set of cases versus the coverage-based case addition algorithm proposed by Zhu and Yang [25] as the content maintenance strategy. The results shown in figs. 3-5 are the average from running each test 10 times. Cases were described by 5 features, each with values ranging from 0 to 1, and new cases were picked uniformly from this 5-dimensional space. Euclidean distance was used as the basis for the similarity measure, and a case was considered to be solvable by another case for the purpose of competence evaluation if the distance between the two cases was less than 0.25. We used the same similarity measure to estimate the solution accuracy **SA** on the basis of the distance between the retrieved case and the query, which is optimistic and more advantageous for the maintained case base strategy than a real world scenario, since the competence evaluations will be flawless. Thus using larger case bases can be expected to usually be at least as good compared to this kind of computationally expensive content maintenance strategy for real-world systems as in the experiments.

Fig. 3 shows the estimated coverage and error for an optimized case base of size N compared to a case base consisting of N random cases. The case base generated by the case addition algorithm has higher resulting coverage (measured as the covered proportion of new queries randomly generated from the underlying domain, which competence-driven maintenance strategies seek

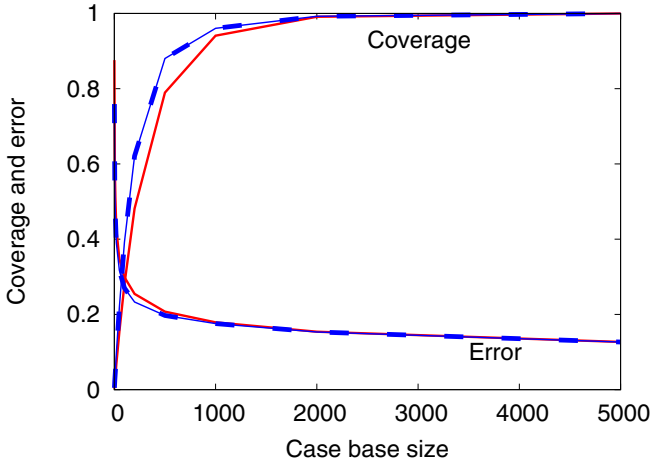


Fig. 3. Coverage and error comparing retaining the full case base of size N (*straight lines*) and using a content maintenance strategy (*dotted lines*) to create a case base of size N based on a larger initial set of 5000 cases. Higher coverage and lower error is better. The time required to perform the maintenance is not considered. In this setting the maintenance strategy outperforms retaining all cases, with higher coverage and slightly lower error.

to optimize), and lower error (measured as the average distance from the best retrieved case to randomly generated new queries). Only the sizes of the case bases are considered, and the computations required to perform the maintenance operations are ignored. Approximately this situation can occur when there are established procedures to run case base maintenance while the system is not being used, e.g. at night, during weekends or during vacations.

However, the computational costs of running case base content reduction algorithms can be extensive. Figs. 4 and 5 show the coverage and error rates for the same two case base content maintenance strategies, but compared according to the time required to perform both maintenance and retrieval. This was examined by running experiments for many different combinations of initial and reduced case base sizes, and choosing the Pareto efficient combinations that gave better results than any faster combinations. The size of the resulting reduced case base size used for retrievals is included in the figures. For each data point the case base maintenance was run only once, and its potentially costly computation was amortized over a large number of retrievals. However, this maintenance cost can still be very high, depending on the number of retrievals performed compared to maintenance operations.

The examples shown in the figures consists of an up-front case maintenance step followed by 1000 and 10000 retrievals respectively (chosen as examples of large numbers of retrievals, since more retrievals favors the maintenance strategy), and shows the combined time for these operations. Even with this relatively large number of retrievals, the simpler strategy of retaining all cases

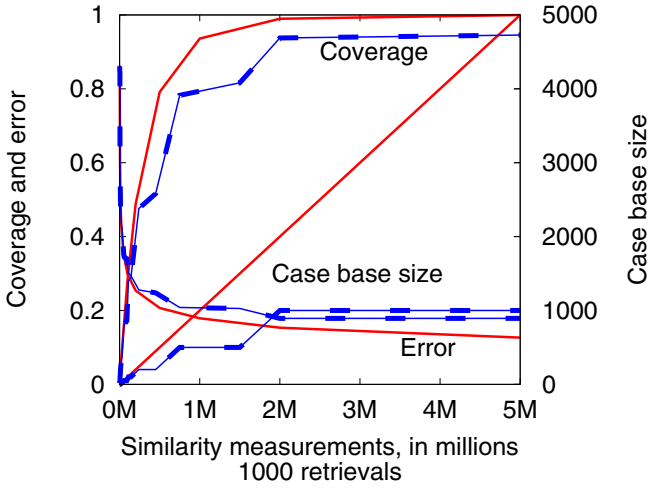


Fig. 4. Coverage and error shown according to the amount of computation required (measured as number of similarity measurements), when retaining the full case base (*straight lines*) and using a content maintenance strategy (*dotted lines*). For the situation with 1000 retrievals, the larger case bases supported by not having a high maintenance cost means that the strategy of retaining all cases performs better, with higher coverage and lower error.

generally performs as well or better than the content maintenance strategy, due to supporting larger case bases in the same time frame. This means that using a maintenance strategy to reduce the case base size for efficiency reasons may sometimes be counter-productive, in addition to size reduction being an eager strategy that limits the potential options available for further problem solving.

The other aspect of reducing the number of cases in the case base is the reduced storage capacity required to hold the case base. Current computer systems intended for personal use can store hundreds of gigabytes of data, which is much much larger than many typical CBR application case bases. Maintaining the set of cases exposed to the retrieval method can be a very useful approach for some applications, but the case base used for retrieval at any given moment does not have to be the full set of cases archived by the system.

Based on this observation, we conclude that many practical CBR system can instead flag the cases as no longer being active and store them in another location that is not searched by the retrieval methods, since conserving disk space is not required for systems that do not generate vast amounts of data. In these situations the archival storage can be done at negligible cost, and provide the advantage that deletions are no longer completely irreversible.

During later system maintenance some time in the future, the reason for the original deletion may no longer be relevant or the algorithms used by the system may have changed, and in such cases it would be beneficial to be able to undo such eager deletion optimizations, in the spirit of lazy learning.

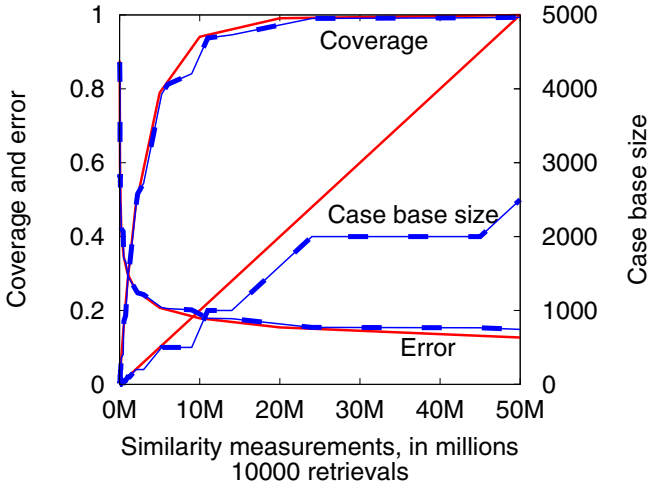


Fig. 5. Even when performing 10000 retrievals, the strategy of retaining all cases generally performs slightly better, with higher coverage and lower error

6 Conclusions

In this paper we have examined the utility problem from a lazy learning perspective, as it applies to speed-up learners and general case-based reasoners. The two primary approaches to addressing the utility problem are through indexing and by reducing the size of the case base itself during case base maintenance.

These approaches are eager compared to the lazy core CBR process, and we have shown how many practical CBR systems do not require the use of these eager optimizations and can be limited by committing to decisions prematurely.

References

1. Aha, D.W.: The omnipresence of case-based reasoning in science and application. Knowledge-Based Systems 11, 261–273 (1998)
2. Bergmann, R., Richter, M., Schmitt, S., Stahl, A., Vollrath, I.: Utility-oriented matching: A new research direction for case-based reasoning. In: 9th German Workshop on Case-Based Reasoning, Shaker Verlag, Aachen (2001)
3. Chaudhry, A., Holder, L.B.: An empirical approach to solving the general utility problem in speedup learning. In: IEA/AIE '94: 7th international conference on Industrial and engineering applications of artificial intelligence and expert systems, USA, pp. 149–158. Gordon and Breach Science Publishers Inc., Newark (1994)
4. Cox, M.T.: Multistrategy learning with introspective meta-explanations. In: Machine Learning: Ninth International Conference, pp. 123–128. Morgan Kaufmann, San Francisco (1992)
5. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: OSDI 2004: 6th conference on Symposium on Operating Systems Design & Implementation, pp. 137–150. USENIX Association, Berkeley (2004)
6. Forbus, K.D., Gentner, D., Law, K.: MAC/FAC: A Model of Similarity-Based Retrieval. Cognitive Science 19(2), 141–205 (1995)

7. Fox, S., Leake, D.B.: Combining case-based planning and introspective reasoning. In: 6th Midwest Artificial Intelligence and Cognitive Science Society Conference, pp. 95–103 (1995)
8. Francis, A., Ram, A.: A comparative utility analysis of case-based reasoning and control-rule learning systems. In: Lavrač, N., Wrobel, S. (eds.) ECML 1995. LNCS, vol. 912, pp. 138–150. Springer, Heidelberg (1995)
9. Francis, A.G., Ram, A.: The utility problem in case based reasoning (1993)
10. Houeland, T.G., Aamodt, A.: Towards an introspective architecture for meta-level reasoning in clinical decision support systems. In: 7th Workshop on CBR in the Health Sciences, ICCBR 2009 (2009)
11. Jarmulak, J., Craw, S., Rowe, R.: Genetic Algorithms to Optimise CBR Retrieval. In: Blanzieri, E., Portinale, L. (eds.) EWCBR 2000. LNCS (LNAI), vol. 1898, pp. 136–147. Springer, Heidelberg (2000)
12. Koton, P.A.: A method for improving the efficiency of model-based reasoning systems. Applied Artificial Intelligence 3(2-3), 357–366 (1989)
13. Leake, D.B., Smyth, B., Wilson, D.C., Yang, Q.: Introduction to the special issue on maintaining case-based reasoning systems. Computational Intelligence 17(2), 193–195 (2001)
14. Leake, D.B., Wilson, D.C.: Categorizing case-base maintenance: Dimensions and directions. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 196–207. Springer, Heidelberg (1998)
15. de Mántaras, R.L., McSherry, D., Bridge, D.G., Leake, D.B., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K.D., Keane, M.T., Aamodt, A., Watson, I.D.: Retrieval, reuse, revision and retention in case-based reasoning. Knowledge Eng. Review 20(3), 215–240 (2005)
16. Markovitch, S., Scott, P.D.: The role of forgetting in learning. In: Fifth International Conference on Machine Learning, pp. 459–465. Morgan Kaufmann, Ann Arbor (1988)
17. Minton, S.: Quantitative results concerning the utility of explanation-based learning. Artif. Intell. 42(2-3), 363–391 (1990)
18. Shokouhi, S.V., Aamodt, A., Skalle, P., Sørmo, F.: Determining root causes of drilling problems by combining cases and general knowledge. In: McGinty, L., Wilson, D.C. (eds.) Case-Based Reasoning Research and Development. LNCS, vol. 5650, pp. 509–523. Springer, Heidelberg (2009)
19. Smyth, B., Cunningham, P., Cunningham, P.: The utility problem analysed - a case-based reasoning perspective. In: Smith, I., Faltings, B.V. (eds.) EWCBR 1996. LNCS, vol. 1168, pp. 392–399. Springer, Heidelberg (1996)
20. Smyth, B., Keane, M.T.: Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In: 13th International Joint Conference on Artificial Intelligence, pp. 377–382. Morgan Kaufmann, San Francisco (1995)
21. Smyth, B., McKenna, E.: Footprint-based retrieval. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 343–357. Springer, Heidelberg (1999)
22. Sørmo, F.: Real-time drilling performance improvement. Scandinavian Oil & Gas Magazine, No. 7/8 (2009)
23. Tambe, M., Newell, A., Rosenbloom, P.S.: The problem of expensive chunks and its solution by restricting expressiveness. Machine Learning 5, 299–348 (1990)
24. Watson, I.: A case study of maintenance of a commercially fielded case-based reasoning system. Computational Intelligence 17, 387–398 (2001)
25. Zhu, J., Yang, Q.: Remembering to add: Competence-preserving case-addition policies for case-base maintenance. In: IJCAI 1999: 16th international joint conference on Artifical intelligence, pp. 234–239. Morgan Kaufmann Publishers Inc., San Francisco (1999)

EGAL: Exploration Guided Active Learning for TCBR

Rong Hu, Sarah Jane Delany, and Brian Mac Namee

Dublin Institute of Technology, Dublin, Ireland
rong.hu@dit.ie, sarahjane.delany@dit.ie, brian.macnamee@dit.ie

Abstract. The task of building labelled case bases can be approached using active learning (AL), a process which facilitates the labelling of large collections of examples with minimal manual labelling effort. The main challenge in designing AL systems is the development of a selection strategy to choose the most informative examples to manually label. Typical selection strategies use exploitation techniques which attempt to refine uncertain areas of the decision space based on the output of a classifier. Other approaches tend to balance exploitation with exploration, selecting examples from dense and interesting regions of the domain space. In this paper we present a simple but effective exploration-only selection strategy for AL in the textual domain. Our approach is inherently case-based, using only nearest-neighbour-based density and diversity measures. We show how its performance is comparable to the more computationally expensive exploitation-based approaches and that it offers the opportunity to be classifier independent.

1 Introduction

A significant barrier to developing case-based reasoning (CBR) systems in certain domains (particularly textual case-based reasoning (TCBR)) is that labelled case bases can be difficult or expensive to obtain. Active learning (AL) can be used to overcome this problem; building labelled case bases by selecting only the *most informative* examples from a larger unlabelled dataset for labelling by an oracle (normally a human expert) and using these to infer the labels for the remainder of the unlabelled data. The most popular *selection strategy* for choosing these most informative examples is *uncertainty sampling* [12]. Typically in uncertainty sampling a ranking classifier is trained using those examples labelled by the oracle so far and is then used to classify the remaining unlabelled examples. Using the output of the ranking classifier as a measure of classification confidence, those examples for which classifications are least confident are selected for labelling by the oracle. This process is repeated until a stopping criterion is reached - typically a limit on the number of labels given by the oracle.

Uncertainty sampling is considered an *exploitation*-based AL selection strategy which attempts to refine the classification decision boundary in uncertain areas of the feature space and can work well if the initial classification boundary is well shaped. However, with small numbers of labelled examples, it can be

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 156–170, 2010.

difficult to reliably estimate the boundary, and it has been suggested that exploitation techniques are prone to querying outliers [20]. Exploitation approaches to selection can also suffer from a lack of exploration of the feature space and may not work well in some scenarios - for example XOR-type problems [2].

Other selection strategies have been developed which attempt to balance exploitation with *exploration*, focussing on examples distant from the labelled set with the aim of sampling wider, potentially more interesting areas of the feature space. These multi-faceted approaches have recently become popular. Existing work has combined uncertainty sampling with density information [7,17,21]; with diversity information [3,5,19,23]; or with both [22,29].

However, we believe that by applying an exploration-only approach to AL selection we can create an AL-based labelling system that is inherently case-based (i.e. based only on features of the case base derived from a similarity measure), and does not suffer from the difficulties associated with exploitation-based approaches. Furthermore, using an exploration-only approach is efficient as it does not require the repeated re-training of a classifier and re-classification of the unlabelled case base associated with exploitation-based approaches.

In this paper we present *Exploration Guided Active Learning* (EGAL), a simple, case-based, computationally efficient, exploration-only AL selection strategy that does not use the output of a classifier in its selection decisions. We compare the performance of this new approach to existing exploitation-based and hybrid selection strategies on a selection of text classification datasets.

The rest of the paper is organized as follows: Section 2 discusses the different selection strategies used in active learning, categorising them into exploration- and exploitation-based methods. Approaches that incorporate uncertainty sampling, density sampling, and diversity sampling; and other related work are discussed. We introduce our exploration-based selection strategy, EGAL, in Section 3 showing how it incorporates simple similarity-based measures of density and diversity. Section 4 describes an evaluation of EGAL using seven textual datasets. We conclude in Section 5 discussing how this approach can be adapted for non case-based classification tasks, offering the opportunity for a classifier-independent selection strategy to get over the reusability problem.

2 Review

AL can be used for two purposes: to build a classifier using the smallest number of manually labelled examples; or to build a fully labelled case base using the smallest number of manually labelled examples. While the difference between these two is subtle, and often ignored, it is important. A labelled case base can be useful for many tasks other than simply building a classification model - for example in [30] an AL-labelled case base was used for information retrieval-like search queries.

The advantages of using case-based classifiers in the AL process were appreciated initially by Hasenjager & Ritter [8] who proposed AL algorithms using local learning models; and by Lindenbaun et al. [14] who developed AL strategies for nearest neighbour classifiers. Although any classifier can be used in the

exploitation-based AL algorithms, the case-based approach to AL is particularly attractive as confidence scores are easily calculated, and the repeated retraining required in AL is especially efficient - new examples are simply added to the case base. More recent examples of case-based AL include index-driven selection sampling for CBR [26]; developing case retention strategies for CBR [18]; semantic labelling of text [16]; supervised network intrusion detection [13] and building classification systems with a weighted k-nearest neighbour classifier [4]. These applications all tend to use exploitation-based selection strategies.

Previous work which uses the underlying structure of the dataset to include exploration in AL selection strategies can be categorised into three approaches: *density-based sampling*, *diversity-based sampling*, and sampling using a combination of both density and diversity. One technique applied frequently is to identify the underlying structure in the dataset by clustering the unlabelled examples. Approaches that use clustering tend to talk about the *most representative* example [24,27], which could either use a local inter-cluster measure which could be considered a density approach, or a global intra-cluster measure which could be considered a diversity approach. For clarity we will avoid the term *most representative*, and the remainder of this section will discuss techniques under the distinctions of density-based and diversity-based sampling.

2.1 Using Density in AL

Uncertainty sampling strategies are prone to querying outliers since outliers are likely to have high uncertainty [20]. To overcome this problem, selection strategies which consider density information have been proposed. The intuition is that an example with high density degree is less likely to be an outlier.

Incorporating density information with uncertainty sampling has been shown to boost the performance of AL in various studies [7,15,21,31]. Labelling an example from a highly dense region of the domain space can increase the confidence of the classifications in its neighbourhood. The density of an example is generally calculated as the average similarity of those neighbours of the example within a specified neighbourhood and has been used, for example, to avoid the selection of outliers [31] and to select the most uncertain examples with maximum density [32]. A common approach is to use *density-weighting* where density is defined explicitly and combined as a function of the uncertainty score [17,21,31]. Other approaches are more implicit, such as those that cluster the unlabelled examples and use the properties of the clusters to select examples for labelling [27].

Novel uses of density information include He *et al.* [9] who make use of nearest neighbours to compare the local density of each example with that of each of its neighbours and select for labelling the example with the highest difference in density; and Fujii *et al.* [7] who use the neighbours of example x to quantify the increase in the utility score (called training utility) of the remaining unlabelled examples if a label is provided for x. The example which is expected to result in the greatest increase in training utility is selected for labelling.

2.2 Using Diversity in AL

Diversity is used in AL selection strategies mainly in an attempt to overcome the lack of exploration when uncertainty sampling is used. A popular approach to incorporating diversity is to include the *Kernel Farthest First* (KFF) algorithm (which selects those examples that are furthest from the current labelled set) as a member of an ensemble of AL processes [2,19] (the other members of the ensemble are typically based on uncertainty sampling).

In the information retrieval literature, several AL heuristics which capture the diversity of feedback documents have been proposed [23,28]. It has been demonstrated in [23] that the performance of traditional relevance feedback (presenting the top k documents according to relevance only) is consistently worse than that of presenting documents with more diversity. Several practical algorithms based on the diversity of the feedback documents have been presented - for example clustering the documents and choosing the cluster centroids to present for labelling [23].

2.3 Using Density and Diversity in AL

Several AL algorithms are proposed in the literature that either explicitly [4,22,29] or implicitly [28] combine both density and diversity with uncertainty sampling to select examples for labelling. These ensemble-based approaches have proven to be particularly successful as they have the advantages of all three approaches.

However, to the best of our knowledge, no approach has been described in the literature that combines density sampling and diversity sampling *without* also using uncertainty sampling. Such an exploration-only approach would be especially efficient as it would not require the repeated building of a classifier, or classification of a large set of unlabelled examples. It would also be particularly suited to the task of building labelled case bases as it would be based only on the properties of the case base and an associated similarity measure. The next section will describe our new EGAL algorithm which takes this approach.

3 The Exploration Guided Active Learning Algorithm

This section describes our exploration-only AL selection strategy: Exploration Guided Active Learning (EGAL). We first discuss how we measure density and diversity, and then explain how they are combined. For this discussion, consider a dataset, $\mathcal{D}$, which consists of a pool of unlabelled examples, $\mathcal{U}$, and a case base of labelled examples, $\mathcal{L}$, which grows as examples, x_i, are selected from $\mathcal{U}$ and presented to the oracle for labelling.

Measuring Density: We measure the density of an unlabelled example x_i by considering the similarity to x_i of the examples that are within a pre-defined neighbourhood N_i of x_i, as given in Equation 1. This neighbourhood N_i (see Equation 2) is set by a similarity threshold α, where $\alpha = \mu - 0.5 \times \delta$; μ and

δ being the mean and standard deviation of the pair-wise similarities of all examples in $\mathcal{D}$ respectively.

$$\text{density}(x_i) = \sum_{x_r \in N_i} \text{sim}(x_i, x_r) \tag{1}$$

$$N_i = \{x_r \in \mathcal{D} | \text{sim}(x_i, x_r) \geq \alpha\} \tag{2}$$

Unlike other density measures such as that in [9], we use the sum of the similarities in the neighbourhood N_i instead of the count of the number of neighbours in N_i. The effect of this is to have fewer *ties* in the density-based ranking, which makes for a more straightforward density-based sampling technique. A selection strategy using density alone will select the example(s) with the highest density to present for labelling.

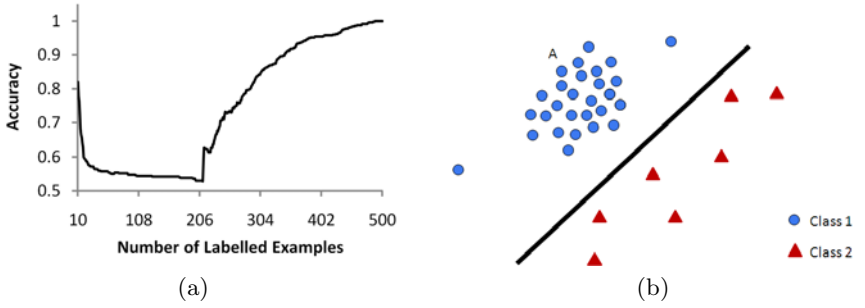
Measuring Diversity: We measure diversity by considering the examples which are most dissimilar to the labelled case base $\mathcal{L}$. Distance being the inverse of similarity, our diversity measure for an example x_i (given in Equation 3) is defined as the distance between x_i and its nearest labelled neighbour. The diversity measure has the advantage of efficient time complexity and it also ensures that the newly selected examples are different from the examples already in $\mathcal{L}$. A selection strategy based on diversity alone would select the example(s) with highest diversity to present for labelling.

$$\text{diversity}(x_i) = \frac{1}{\max_{x_r \in \mathcal{L}} \text{sim}(x_i, x_r)} \tag{3}$$

Combining Density and Diversity: Density and diversity sampling greedily choose examples that optimise locally, which can make them myopic approaches to selection in AL. They can become trapped in local optimums which can result in poor performance globally. An example of density sampling's poor performance is evident in Figure 1(a), which shows the performance of a density-based active learner on a textual dataset of 500 examples starting with 10 initially labelled examples, (details on the selection of the initial case base, the classifier used, and the performance measures used are given in Section 4). This shows a degradation in performance until after 200 or so examples are labelled, at which point performance improves rapidly. Figure 1(b) illustrates how this can happen. With density sampling, examples from class 1 in group A will be repeatedly selected for labelling while examples from class 2 will be ignored, leading to a poorly defined classification boundary during this time. When diversity alone is used, similarly dysfunctional scenarios can arise.

To overcome these problems, we introduce an element of diversity to a density-based sampling approach. Including diversity means that high density examples that are close to labelled examples are not selected for labelling by the oracle.

To determine whether an example should be considered as a candidate for selection, we use a threshold β. If the similarity between an unlabelled example x_i and its nearest neighbour in the labelled case base is greater than β then x_i is



(a) (b)

Fig. 1. Illustrating how density-based sampling can perform badly

not a candidate for selection. We call the set of examples that can be considered for selection the *Candidate Set*, $\mathcal{CS}$, which we define as follows:

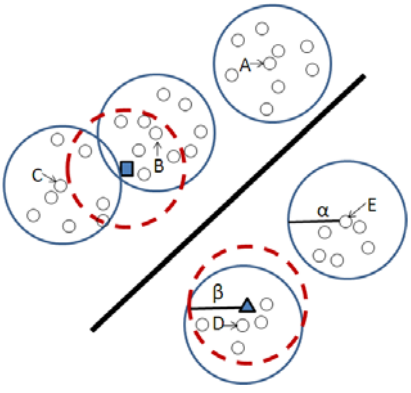
$$\mathcal{CS} = \{\exists x_i \in \mathcal{U} \mid sim(x_i, x_j) \leq \beta, x_j \in \mathcal{L}, \\ sim(x_i, x_j) \geq sim(x_i, x_k), \forall x_k \in \mathcal{L}, j \neq k\}$$

Our EGAL selection strategy ranks the possible candidates for selection (i.e. those in $\mathcal{CS}$) based on their density, and selects those examples with the highest density for labelling first. Thus, examples close to each other in the feature space will not be selected successively for labelling.

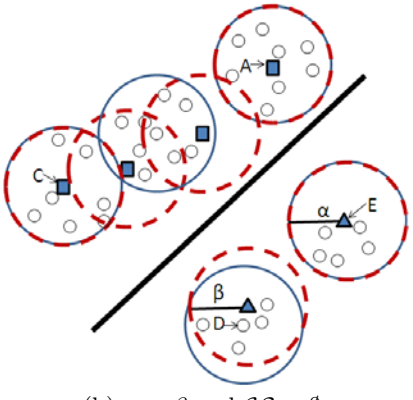
Parameters α and β play an important role in the selection process. α controls the radius of the neighborhood used in the estimation of density, while β controls the radius of the neighbourhood used in the estimation of $\mathcal{CS}$. The values selected for these parameters can significantly impact the overall performance.

Shen et al. [22] use a threshold similar to our β which they set to the average pair-wise similarity of the examples in the whole dataset. Initially, however, we set $\beta = \alpha$ as shown in Figure 2(a), where shaded polygons represent labelled examples in $\mathcal{L}$ and circles represent unlabelled examples in $\mathcal{U}$. The regions defined by α are shown as solid circles for a small number of unlabelled examples (A, B, C, D and E). For clarity of illustration, rather than showing the regions defined by β around every unlabelled example, we show them, as broken circles, around only the labelled examples. The effect, however, is the same: if a labelled example is within the neighbourhood of an unlabelled example defined by β, then the unlabelled example will also be within the neighbourhood of the labelled example defined by β.

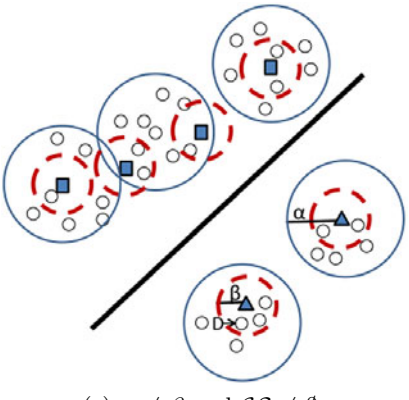
In the example shown in Figure 2(a), since examples B and D have labelled examples in the neighbourhood defined by β, they will not be added to $\mathcal{CS}$. A, C and E, however, will be added. As more examples are labelled, we may reach a stage when there are no examples in the candidate set as there are always labelled examples within the neighbourhood defined by β. This scenario is shown in Figure 2(b). When this happens we need to increase β to shrink this neighbourhood as shown in Figure 2(c). We update β when we have no examples left in $\mathcal{CS}$ - a unique feature of our approach as far as we are aware.



(a) $\alpha = \beta$ and $\mathcal{CS} \neq \emptyset$



(b) $\alpha = \beta$ and $\mathcal{CS} = \emptyset$



(c) $\alpha \neq \beta$ and $\mathcal{CS} \neq \emptyset$

Fig. 2. The relationship between parameters α and β and the candidate set $\mathcal{CS}$

We use a novel method to update β motivated by a desire to be able to set the size of $\mathcal{CS}$. As the size of the $\mathcal{CS}$ is defined by β, a bigger β value gives us a bigger candidate set. We set β to a value which can give us a candidate set with a size proportional to the number of elements available for labelling (i.e. the size of the unlabelled pool $\mathcal{U}$) as detailed below:

(i) Calculate the similarity between each unlabelled example and its nearest labelled neighbour giving the set S, as follows

$$S = \{s_i = sim(x_i, x_j) \mid x_i \in \mathcal{U}, x_j \in \mathcal{L}, \\ sim(x_i, x_j) \geq sim(x_i, x_k), \forall x_k \in \mathcal{L}, j \neq k\}$$

(ii) Choose the value s_w from S that splits S into two, where

$$S_1 = \{s_i \in S \mid s_i \leq s_w\}, \\ S_2 = \{s_j \in S \mid s_j > s_w\} \text{ and} \\ |S_1| = \lfloor (w \times |S|) \rfloor, 0 \leq w \leq 1$$

(iii) Let $\beta = s_w$, which is the similarity value such that w proportion of unlabelled examples will be in diverse neighbourhoods of the feature space.

The proportion parameter, w, allows us to balance the influence of diversity and density in our selection strategy. When $w = 0$, the EGAL algorithm defaults to pure diversity-based sampling discounting any density information. As w increases, the influence of density increases and the influence of diversity decreases with more examples being added to $\mathcal{CS}$. When $w = 1$ the EGAL algorithm becomes purely a density-based sampling algorithm. We explore the effect of changing the value of the proportion parameter w in Section 4.2.

Our combined strategy can be implemented very efficiently. At the start the pair-wise similarity matrix for the entire dataset and the individual density measure for every example are calculated and cached. At each iteration of the selection algorithm, the updated diversity measure for each example in the unlabelled set, $\mathcal{U}$, is the only calculation necessary. Computationally this is very efficient, especially considering the rebuilding of a classifier and the classification of every unlabelled example required by uncertainty sampling based methods at each iteration of the selection algorithm.

4 Evaluation

To assess the performance of our EGAL algorithm, we performed a comparative evaluation with other AL selection strategies. The objective of our evaluation was firstly to see whether the performance of combining density and diversity information in our EGAL approach was better than density or diversity sampling alone. In addition, we compared EGAL to uncertainty sampling which is the most commonly used AL selection strategy, and density-weighted uncertainty sampling which is the most common approach to combining density and uncertainty. After describing the datasets used, the implementation details of our EGAL approach and the evaluation measures used; this section will describe the results of these experiments.

4.1 Experimental Setup

In our evaluations we used seven balanced text-based classification datasets: a spam dataset [6]; four binary classification datasets derived from the 20-Newsgroup collection[1]; and two binary classification datasets from the Reuters collection[2]. The properties of each dataset, and the average accuracy achieved in five iterations of 10-fold cross validation using a 5-NN classifier, are shown in Table 1 (accuracies are included as an indication of the difficulty of each classification problem). Each dataset was pre-processed to remove stop-words and stemmed using Porter stemming.

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/
[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Table 1. Details of datasets used in the evaluation experiments

Dataset	Task	Examples	Feat.	Accu.
20NG-WinXwin	comp.os.ms-windows.misc vs. comp.windows.x	496	8557	91.14%
20NG-Comp	comp.sys.ibm.pc.hardware vs. comp.sys.mac.hardware	500	7044	85.56%
20NG-Talk	talk.religion.misc vs. alt.atheism	500	9000	93.92%
20NG-Vehicle	rec.autos vs. rec.motorcycles	500	8059	92.96%
Reuters-1	acq vs. earn	500	3692	89.56%
Reuters-2	g151 vs. g158	500	6135	95.36%
Spam	spam vs. non-spam	500	18888	96.80%

As the datasets used in our evaluations are fully labelled, the labelling process can be simulated without the need for a human oracle. At each iteration one example from the unlabelled pool, $\mathcal{U}$, is selected for labelling and its label is applied. This process is repeated until the oracle's label budget expires. In order to monitor the performance of the EGAL algorithm, and compare it to other approaches, after each labelling a k-NN classifier is built from the labelled case base, $\mathcal{L}$, and classifications are made for every example remaining in the unlabelled pool, $\mathcal{U}$. These classifications are compared with the actual labels in each dataset and the accuracy of this labelling is used to evaluate the performance of the selection strategy. Accuracy is calculated as $Accuracy = C/|\mathcal{D}|$, where C is the number of correctly labelled examples. Both manually and automatically labelled examples are included in this calculation so as to avoid large fluctuations as new labels are added in the latter stages of the process [10]. Using the accuracy recorded after each manual labelling, a learning curve is constructed to plot the accuracy as a function of the number of labels provided (for example Figure 3(a)). It is important to note that the classifications of the unlabelled pool made after each manual labelling are only for evaluation purposes and are not required by the EGAL algorithm.

In all of the experiments described in this section the same AL process is used. The initial case base contains 10 examples selected for labelling by the oracle using a deterministic clustering approach, as we have found it to be a successful approach to initial case base selection [11]. The same initial case base is used by each AL algorithm for each dataset. When classifiers are used, these are 5-NN classifiers using distance weighted voting. Finally, the stopping criteria used by all algorithms is a labelling budget which assumes that the oracle will provide 110 labels for each dataset.

4.2 Exploration of the Effect of the Balancing Parameter w

The density neighbourhood parameter, α, is set to $\mu - 0.5 \times \delta$ (as discussed in Section 3), as preliminary experiments showed it to be a good choice. In order to set the diversity neighbourhood parameter β, a value of w which controls the

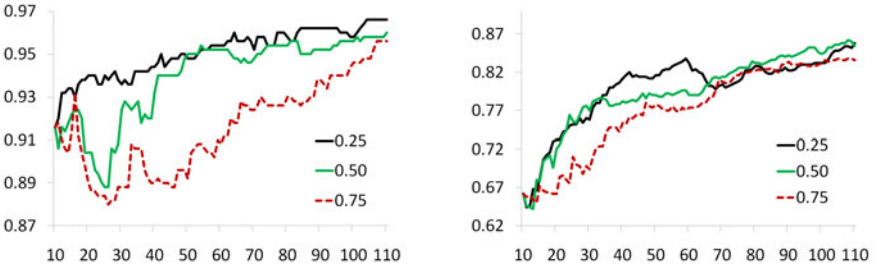


Fig. 3. The effect of the balancing parameter w on the EGAL algorithm

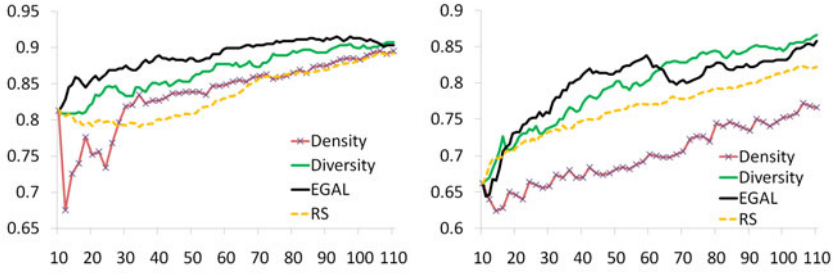
balance between density and diversity in the EGAL selection process is required. Intuition would suggest that diversity is more important than density, and in order to investigate this experiments were performed with w set to $0.25, 0.50$ and 0.75 on the datasets described previously. Results on two of these datasets are shown in Figure 3. Across the seven datasets it was clear that $w = 0.25$ gave the best results (indicated by the fact that the learning curve for $w = 0.25$ dominates the others) and this value was used in all further experiments. This experiment supports the intuition that diversity is more important than density in the selection process.

4.3 EGAL Evaluation Results

The results of comparisons between our proposed approach (labelled EGAL), density sampling (labelled Density) and diversity sampling (labelled Diversity) across the seven datasets are summarised in Figure 4. A random sampling strategy (labelled RS), which randomly picks examples for labelling, is also included as a baseline. The results show that density sampling doesn't perform well but that diversity sampling performs consistently better than the baseline random sampling. In addition, incorporating density information with diversity sampling in our EGAL algorithm improves the performance of diversity sampling consistently on all datasets.

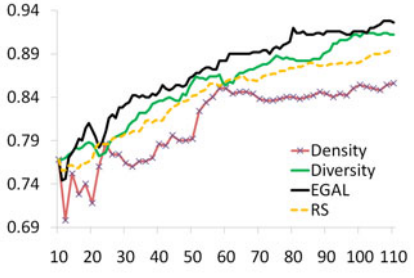
We also compared EGAL to the more frequently used uncertainty sampling (US) using Hu *et al.*'s implementation [10] which is based on a k-NN classifier and density-weighted uncertainty sampling (DWUS) where uncertainty is multiplied with the density measure and examples with the highest resulting ranking score are selected for labelling. The results are shown in Figure 5.

Previous work on density weighted uncertainty sampling has shown an improvement over uncertainty sampling [17,21]. Interestingly, the results in Figure 5 agree with that conclusion for datasets where density sampling alone also improves performance. However, for datasets where density sampling performs badly (see Figures 5(e), 5(f) and 5(g)) DWUS does not improve performance over US indicating that the density information is having a negative effect on the AL process.

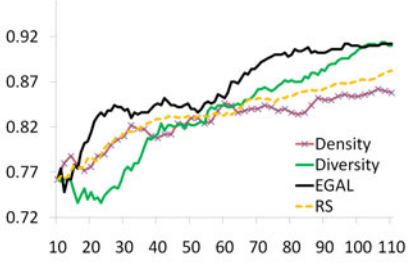


(a) 20NG-WinXwin Dataset

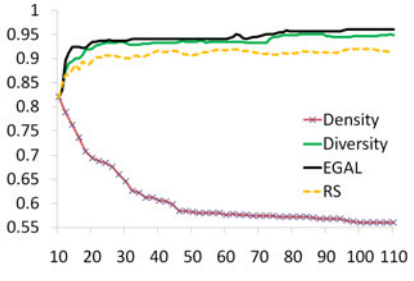
(b) 20NG-Comp Dataset



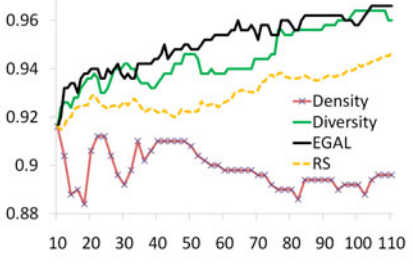
(c) 20NG-Talk Dataset



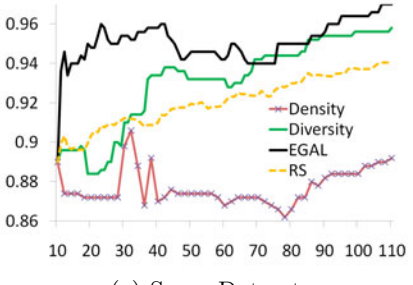
(d) 20NG-Vehicle Dataset



(e) Reuters-1 Dataset

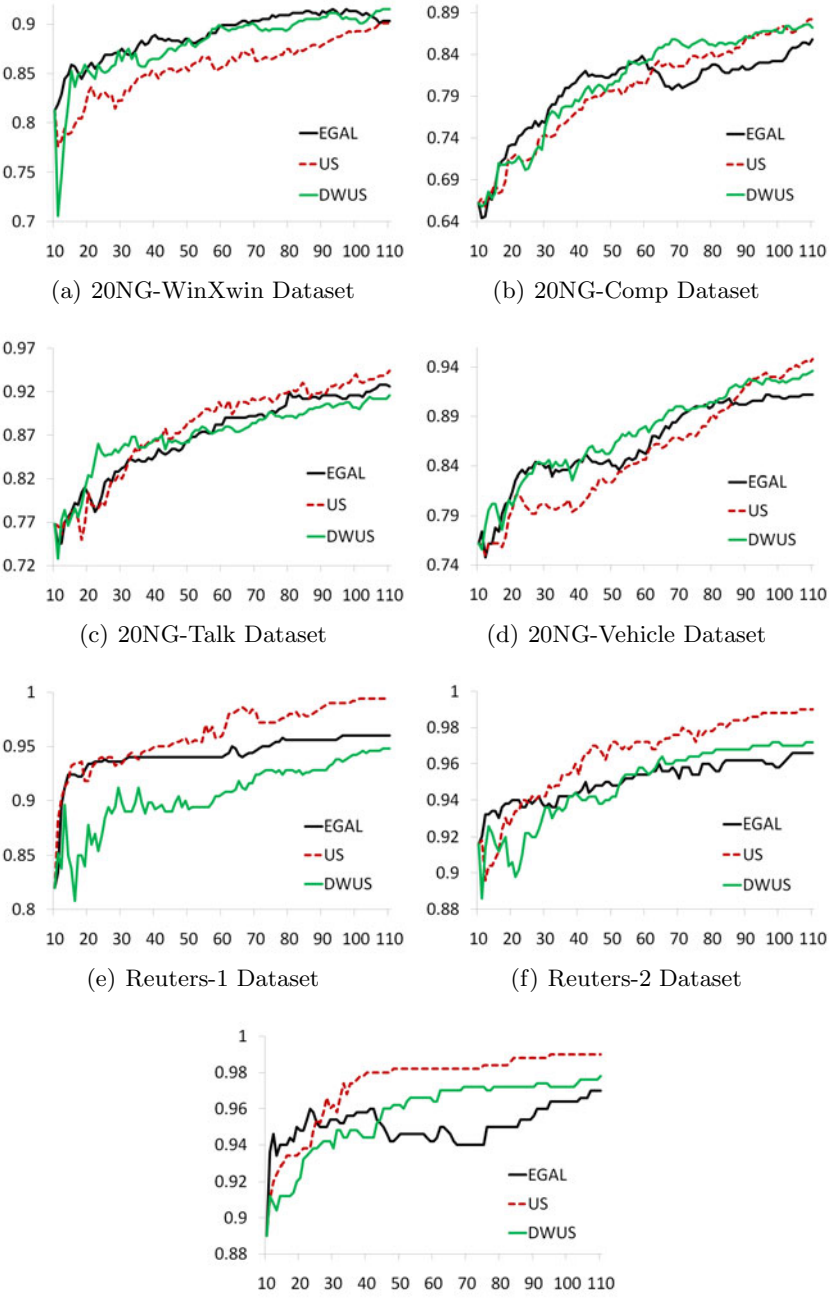


(f) Reuters-2 Dataset



(g) Spam Dataset

Fig. 4. Comparison of Density, Diversity, EGAL and RS selection strategies



(a) 20NG-WinXwin Dataset

(b) 20NG-Comp Dataset

(c) 20NG-Talk Dataset

(d) 20NG-Vehicle Dataset

(e) Reuters-1 Dataset

(f) Reuters-2 Dataset

(g) Spam Dataset

Fig. 5. Comparison of EGAL, US and DWUS selection strategies

The more interesting benefit of EGAL is in the early stage of the AL process, the first 20 to 30 labellings, where it outperforms both US and DWUS. A detailed analysis of the Area under the Learning Curve (ALC) for learning curves up to a varying number of labels was performed. Illustrative examples of ALC values are given in Table 2. The difference between US and EGAL was found to be significant (at $\alpha = 0.05$) using the Wilcoxon signed-rank test at 30 labels and below. There was no significant difference between DWUS and US at any number of labels.

Table 2. Illustrative ALC values for learning curves up to the specified number of labels. The best values across the three approaches are highlighted in bold.

Dataset	30 Labels			60 Labels			110 Labels		
	US	DWUS	EGAL	US	DWUS	EGAL	US	DWUS	EGAL
20NG-WinXwin	16.24	16.69	**17.09**	41.79	42.95	**43.58**	85.72	88.07	**88.97**
20NG-Comp	14.02	14.04	**14.36**	37.47	37.97	**38.69**	79.87	**80.90**	79.89
20NG-Talk	15.74	**16.24**	15.89	41.92	**42.22**	41.56	**87.94**	87.04	86.95
20NG-Vehicle	15.66	**16.17**	16.14	40.20	**41.81**	41.42	85.26	**87.38**	86.35
Reuters-1	**18.49**	17.28	18.42	**47.03**	44.16	46.62	**96.21**	90.63	94.32
Reuters-2	18.53	18.30	**18.71**	**47.42**	46.60	47.11	**96.49**	94.92	95.09
Spam	18.76	18.49	**18.92**	**48.10**	47.15	47.43	**97.41**	95.75	95.12

These results point towards an interesting empirical property of the EGAL algorithm: it can improve the labelling accuracy fastest in the beginning stages of active learning. This would be beneficial in domains where labelling cost is high.

5 Conclusions and Future Work

In this work, we have proposed EGAL, an exploration-only approach to AL-based labelling of case bases. EGAL is inherently case-based as it uses only the notions of density and diversity, based on similarity, in its selection strategy. This avoids the drawbacks associated with exploitation-based approaches to selection. Furthermore, in contrast to most active learning methods, because EGAL does not use a classifier in its selection strategy it is computationally efficient. We have shown empirical results of EGAL's viability as a useful tool for building labelled case bases, especially in domains where it is desirable to front-load the AL process so that it performs well in the earlier phases - a feature of EGAL demonstrated in our evaluation experiments.

It is on the absence of any particular classifier in EGAL that we intend to focus our future work. AL methods that use a classifier in their selection strategy are tuned to that particular classifier, resulting in poor reusability of the labelled data by other classifiers. This is known as the *reusability problem* in active learning [1,25]. Tomanek et al. [25] argued that by using a committee-based active

learner, the dataset built with one type of classifier can reasonably be reused by another. Another possible solution to the reusability problem is our EGAL algorithm as a classifier-free AL framework. Our future work in this area will check the reusability of the resultant labelled examples from EGAL at training different types of classifier.

Acknowledgments. This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/RFP/CMSF718.

References

1. Baldridge, J., Osborne, M.: Active learning and the total cost of annotation. In: Proc. of EMNLP 2004, pp. 9–16 (2004)
2. Baram, Y., El-Yaniv, R., Luz, K.: Online choice of active learning algorithms. Journal of Machine Learning Research 5, 255–291 (2004)
3. Brinker, K.: Incorporating diversity in active learning with support vector machines. In: Proc. of ICML 2003, pp. 59–66 (2003)
4. Cebron, N., Berthold, M.R.: Active learning for object classification: from exploration to exploitation. Data Mining and Knowledge Discovery 18(2), 283–299 (2009)
5. Dagli, C.K., Rajaram, S., Huang, T.S.: Combining diversity-based active learning with discriminant analysis in image retrieval. In: Proc. of ICITA 2005, pp. 173–178 (2005)
6. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems 18(4-5), 187–195 (2005)
7. Fujii, A., Tokunaga, T., Inui, K., Tanaka, H.: Selective sampling for example-based word sense disambiguation. Computational Linguistics 24(4), 573–597 (1998)
8. Hasenjäger, M., Ritter, H.: Active learning with local models. Neural Processing Letters 7(2), 107–117 (1998)
9. He, J., Carbonell, J.G.: Nearest-neighbor-based active learning for rare category detection. In: Proc. of NIPS 2007 (2007)
10. Hu, R., Mac Namee, B., Delany, S.J.: Sweetening the dataset: Using active learning to label unlabelled datasets. In: Proc. of AICS 2008, pp. 53–62 (2008)
11. Hu, R., Mac Namee, B., Delany, S.J.: Off to a good start: Using clustering to select the initial training set in active learning. In: Proc. of FLAIRS 2010 (to appear, 2010)
12. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: Proc. of SIGIR 1994, pp. 3–12 (1994)
13. Li, Y., Guo, L.: An active learning based TCM-KNN algorithm for supervised network intrusion detection. Computers and Security 26, 459–467 (2007)
14. Lindenbaum, M., Markovitch, S., Rusakov, D.: Selective sampling for nearest neighbor classifiers. Machine Learning 54(2), 125–152 (2004)
15. McCallum, A., Nigam, K.: Employing EM and pool-based active learning for text classification. In: Proc. of ICML 1998, pp. 350–358 (1998)
16. Mustafaraj, E., Hoof, M., Freisleben, B.: Learning semantic annotations for textual cases. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 99–109. Springer, Heidelberg (2005)

17. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: Proc. of ICML 2004, pp. 623–630 (2004)
18. Ontañón, S., Plaza, E.: Collaborative case retention strategies for CBR agents. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 392–406. Springer, Heidelberg (2003)
19. Osugi, T., Kun, D., Scott, S.: Balancing exploration and exploitation: A new algorithm for active machine learning. In: Proc. of ICDM 2005, pp. 330–337 (2005)
20. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: Proc. of ICML 2001, pp. 441–448 (2001)
21. Settles, B., Craven, M.: An analysis of active learning strategies for sequence labeling tasks. In: Proc. of EMNLP 2008, pp. 1069–1078 (2008)
22. Shen, D., Zhang, J., Su, J., Zhou, G., Tan, C.L.: Multi-criteria-based active learning for named entity recognition. In: Proc. of ACL 2004, p. 589 (2004)
23. Shen, X., Zhai, C.: Active feedback in ad hoc information retrieval. In: Proc. of SIGIR 2005, pp. 59–66. ACM, New York (2005)
24. Tang, M., Luo, X., Roukos, S.: Active learning for statistical natural language parsing. In: Proc. of ACL 2002, pp. 120–127 (2002)
25. Tomanek, K., Wermter, J., Hahn, U.: An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In: Proc. of EMNLP 2007, pp. 486–495 (2007)
26. Wiratunga, N., Craw, S., Massie, S.: Index driven selective sampling for CBR. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 637–651. Springer, Heidelberg (2003)
27. Xu, Z., Yu, K., Tresp, V., Xu, X., Wang, J.: Representative sampling for text classification using support vector machines. In: Sebastiani, F. (ed.) ECIR 2003. LNCS, vol. 2633, pp. 393–407. Springer, Heidelberg (2003)
28. Xu, Z., Akella, R.: Active relevance feedback for difficult queries. In: Proc. of CIKM 2008, pp. 459–468 (2008)
29. Xu, Z., Akella, R., Zhang, Y.: Incorporating diversity and density in active learning for relevance feedback. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECiR 2007. LNCS, vol. 4425, pp. 246–257. Springer, Heidelberg (2007)
30. Zhang, Q., Hu, R., Namee, B.M., Delany, S.J.: Back to the future: Knowledge light case base cookery. In: Workshop Proc. of 9th ECCBR, pp. 239–248 (2008)
31. Zhu, J., Wang, H., Tsou, B.: Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In: Proc. of COLING 2008, pp. 1137–1144 (2008)
32. Zhu, J., Wang, H., Tsou, B.K.: A density-based re-ranking technique for active learning for data annotations. In: Li, W., Mollá-Aliod, D. (eds.) ICCPOL 2009. LNCS, vol. 5459, pp. 1–10. Springer, Heidelberg (2009)

Introspective Knowledge Revision in Textual Case-Based Reasoning

Karthik Jayanthi[1], Sutanu Chakraborti[1], and Stewart Massie[2]

[1] Department of Computer Science and Engineering, Indian Institute of Technology, Chennai-600036, India
[2] School of Computing, The Robert Gordon University, Aberdeen AB25 1HG, Scotland, UK
jkarthik@cse.iitm.ac.in, sutanuc@iitm.ac.in, sm@comp.rgu.ac.uk

Abstract. The performance of a Textual Case-Based Reasoning system is critically dependent on its underlying model of text similarity, which in turn is dependent on similarity between terms and phrases in the domain. In the absence of human intervention, term similarities are often modelled using co-occurrence statistics, which are fragile unless the corpus is truly representative of the domain. We present the case for introspective revision in TCBR, whereby the system incrementally revises its term similarity knowledge by exploiting conflicts of its representation against an alternate source of knowledge such as category knowledge in classification tasks, or linguistic and background knowledge. The advantage of such revision is that it requires no human intervention. Our experiments on classification knowledge show that revision can lead to substantial gains in classification accuracy, with results competitive to best-in-line text classifiers. We have also presented experimental results over synthetic data to suggest that the idea can be extended to improve case-base alignment in TCBR domains with textual problem and solution descriptions.

1 Introduction

Humans learn through introspection and interaction. When we fail to achieve a desired outcome, we introspect and attempt to explain the failure. Typically, we revise our models or beliefs based on introspection. There are times, however, when introspection alone may not be adequate and we may have to seek advice from an expert. Let us take an example of a child learning trigonometry. She may be confused between the definitions of sine and cosine, and thus arrive at a wrong result when solving a problem. If she is provided with the correct answer to the problem and her answer turns out to be wrong, it would be natural for her to question those concepts she is not clear about. In case of the confusion between sine and cosine definitions, she may try using the sine definition instead of the cosine, or vice versa. If she succeeds, she may revise her concept accordingly for future use. In case she is confused between several possible choices however, it may not be a worthwhile effort to try all of these alternatives; rather it would

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 171–185, 2010.

make sense to elicit help from an expert. The first case is one of introspective concept revision, the second of interactive concept revision. It is intuitively appealing to extend these ideas to Machine Learners.The current work is based on the former idea, while Mixed Initiative Systems [1] address the latter problem.

This paper extends the idea of learning driven by introspection to Case-Based Reasoning (CBR). In particular, we focus on Textual Case-Based Reasoning (TCBR) which aims at reusing episodes of previously solved problems recorded in the form of unstructured text. Each episode represented as a problem-solution pair is called a case. Given a new problem, TCBR aims at identifying cases whose problem components are most similar to the presented problem. The performance of a TCBR system is critically determined by the procedure to estimate similarity between texts. This is a hard problem as the surface level representations at the term level fail to capture deep semantics. Most TCBR systems borrow techniques from Information Retrieval (IR) which are typically based on shallow representations of cases like a bag of terms. A richer representation is one which is augmented with the knowledge of "semantic relatedness" between terms as estimated from their co-occurrences observed in a corpus.

Despite improvements based on statistical learning, TCBR retrieval effectiveness in most real world tasks leaves enough room for improvement, when benchmarked against human level performance. In the TCBR context, the challenge is to strike the right trade-off between knowledge acquisition overhead and retrieval effectiveness. Our initial goal in the current work was to explore ways of using minimal feedback from the expert to generate revised representations of textual cases that can in turn lead to improved competence in problem solving. Textual case representations are critically dependent on the similarity (alternately referred to as semantic relatedness [1]) between terms in the problem space vocabulary. Interaction with humans can help the system identify similarity relations which could have adversely affected system performance, and to make systematic corrections over these relations. In the course of our experiments, however, we realized that there was a second interesting possibility: we could achieve substantial improvements by doing away with the expert altogether. The key idea involves introspectively trying out the effect of corrections on suspected similarity relations by measuring the effectiveness of the revised representation of training data using cross validation and retaining those corrections that lead to improvements. While this approach may not scale up to performance levels obtainable with more elaborate human intervention, it is still attractive in that it is fully automated, and can be a plug-in to any existing TCBR system which has numeric representations of term and/or case similarities. It may be noted that unlike earlier work [2] which presented introspective *knowledge acquisition*, the focus of our work is on introspective *revision*.

The structure of the paper is as follows. Section 2 presents the general framework and outlines the scope of our work. The background for our work in terms of our choices of formalisms and representations is described in Section 3.

[1] In this paper we use the term similarity and the phrase "semantic relatedness" interchangeably, though the latter is more reflective of what we mean.

A detailed description of our methods is presented in Section 4. Evaluation methodology and the empirical results obtained are discussed in Section 5. In Section 6 we position our work in the context of related work. Section 7 summarizes our contributions and discusses possible extensions of our work.

2 Goals, Conflict Sources, Representations and Introspection

In this section we lay out a framework for introspective revision within the TCBR context. Goals, conflict sources and representations are three dimensions that define a context for an introspective learning task. The goal of the child learning trigonometry is to improve her competence at solving problems, measured by the proportion of test problems she correctly solves. Conflicts can arise from errors in her results over practice problems on which correct results are known. Representations refer to the underlying model of her concepts, and are important because a learning task driven by knowledge of goals and conflicts makes changes to these representations.

Figure 1 shows goals, conflicts and representations in TCBR. Many TCBR systems have problems represented as unstructured text and solutions drawn from a set of symbols which could be viewed as category labels. The goal of learning in such a context would be to improve performance with respect to measures such as classification accuracy. In TCBR systems which have both problems and solution represented as unstructured text; a corresponding goal would be to improve retrieval effectiveness. It has been shown in earlier work [3] that "alignment" between problems and solution components has a strong bearing on effectiveness of retrieval. A case-base is well-aligned if similar problems have similar solutions; it is poorly aligned otherwise. Alignment measures position a case-base in this spectrum, and are used for predicting retrieval effectiveness. We can thus treat improving alignment as an indirect goal. Representations in the case of TCBR systems refer to how attribute values and associated similarity measures are captured within the system. TCBR systems could involve

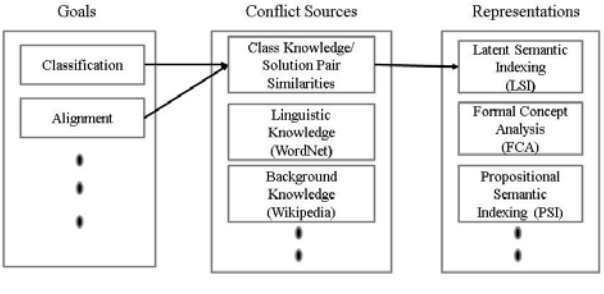


Fig. 1. Pictorial Representation of the Framework

Information Extraction for extracting attribute values from text. This typically involves significant manual intervention for encoding domain-specific knowledge. Many TCBR systems are knowledge light, in that they represent text in terms of bag of terms (or phrases) and use statistical approaches to infer similarity between terms from a given corpus. An example of such a system is [4] which uses Latent Semantic Indexing (LSI) for arriving at term similarities. A second system would be one which uses Formal Concept Analysis [5] to arrive at case or feature clusters. The systems mentioned above have the effect of using knowledge from a corpus of existing cases to learn better estimates for semantic relatedness between terms. Introspective learning in such representations will have the effect of learning better models of semantic relatedness.

Conflicts arise out of differences between knowledge inferred from a corpus, and knowledge from some source external to the corpus. In case of TCBR, an external knowledge source could be lexical knowledge of English, as encoded in WordNet, or background knowledge of a domain as can be obtained from a collection of documents in Wikipedia. In classification tasks, external knowledge could be the category labels associated with a case. In interaction, external knowledge would be explicit human feedback.

The above discussion shows that goals, conflicts and representations can be viewed as three dimensions that characterize an introspective revision task. In this paper, introspective revision is tried out on two goals: improving classification accuracy and improving case-base alignment. LSI is used to represent the associations between terms and cases as mined from the corpus. In the first case, the class knowledge is the source of conflict; in the second, it is the knowledge of solution pair similarities.

3 Background

In this section we outline the retrieval formalism and case representations, which form the basis for algorithms and evaluation reported in this paper.

3.1 Case Retrieval Network Architecture for Case Representation

A Case Retrieval Network (CRN) is a framework for facilitating efficient retrieval in CBR. It was originally presented in the doctorial thesis of Mario Lenz [6]. Figure 2 shows a CRN schematic for a textual case-base having 4 cases C_1 through C_4 and 5 terms W_1 through W_5. The rectangles represent the terms and ovals represent the cases. The basic idea is very similar to that of an inverted file representation where terms are used to index cases except that a CRN also captures the knowledge of similarity between terms. These real valued similarities are shown alongside circular arcs in the figure. Section 3.2 describes one method in obtaining this knowledge. Thus a CRN has two broad knowledge containers: relevance weights which associate terms with cases and similarity values indicating the strength of semantic relatedness between terms. In Figure 2, the relevance weights take binary values; a value of 1 indicates the presence of a term in a case

Cases	W_1	W_2	W_3	W_4	W_5
C_1	1	1	0	0	0
C_2	1	0	1	0	0
C_3	0	0	0	1	1
C_4	0	0	1	1	0

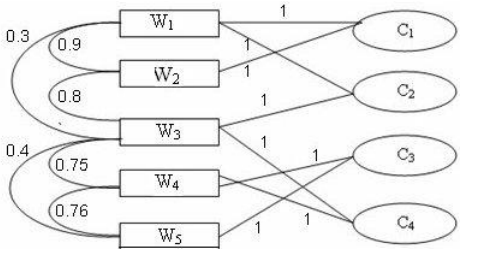


Fig. 2. An example for a CRN for Text Retrieval

and 0 its absence. Retrieval in CRN involves two phases of spreading activation. In the first phase terms in the query pass activations to similar terms leading to query revision. In the second phase the terms as activated in the first phase pass on their activations to case nodes via relevance weights.

In the current work we have used CRNs to two different ends. The first is to arrive at new representations for cases that incorporate the knowledge of term similarities. This is realized as follows. We start with a case representation as a vector of binary values representing its constituent terms. This case is treated as a query and its constituent terms are allowed to activate related terms via similarity arcs. At each term node the incoming activations are aggregated and the revised case representation is the row vector comprising the aggregated activations at each term node. For example the initial representation of C_1 is {1, 1, 0, 0, 0} in the vector space spanned by the terms W_1 through W_5. The term similarities between the terms are as given by Figure 2. For arriving at the revised representation of C_1, W_1 and W_2 activate the only term related to them, namely W_3. If the aggregation function at each term node is a simple summation, the resulting representation of C_1 turns out to be {1.9, 1.9, 1.1, 0, 0}. This new representation can be seen as a result of a matrix operation. Let R_{ini} be the initial representation of a case. Let T be a symmetric matrix of term pair similarities. The new representation of the case R_{new} is given by

$$R_{new} = R_{ini}T \tag{1}$$

The second use of CRN in our work is to facilitate retrieval. We have used the cosine measure for our implementations since it is tolerant to varying lengths of cases when treated as vectors over a space of terms. We had to extend the CRN so that the similarities could still be computed without compromising on the efficiency. We observe that the CRN as described above realizes a dot product of the query vector with the case vector. The case norms are computed in advance, so the similarities of the query to the case can be computed if the dot product and query norms are computed at run time.

3.2 Modeling Similarity Knowledge Using Latent Semantic Indexing

Latent Semantic Indexing can be used to mine similarity between terms based on their co-occurrence patterns in a corpus (case-base). In estimating similarity,

we end up facing the problem of circularity: cases are similar if they have similar terms, but terms in turn are similar if they occur in similar cases. LSI uses a dual mode factor analysis in the form of Singular Value Decomposition to resolve this circularity. Both cases and terms are represented in terms of a set of underlying concepts. These concepts can be viewed as linear combination of terms, or as a linear combination of cases. For example the concept of *motor vehicle* can be expressed as a weighted combination of terms like *tyres*, *steering wheel*, *brakes* etc.

Terms and cases can be expressed as linear combination of these concepts as well as of each other. These concepts are expected to be more representative of the underlying semantics since they weed out noise due to term choice variability. In particular, LSI ranks the mined concepts according to their importance; concepts with very low importance do not play a role in explaining the case-term distribution and hence can be ignored.

LSI is often viewed as a dimensionality reduction technique. A subset of the concepts mined using LSI which are deemed to be important are used to define a lower dimensional space in which the cases and terms are represented. Term and case similarity can be computed based on their representations in this reduced space. For mathematical formulations in terms of SVD decompositions, see [7]. Term similarities mined using LSI capture higher order co-occurrences between terms [8]. For example, the terms *car* and *automobile* may never co-occur in any case but they are inferred to be similar since there is considerable overlap in the set of terms they co-occur with (say *gear*, *drive*, *chassis*). This is an example of second order co-occurrence.

4 Introspective Learning Algorithms

In this section we present our algorithms for introspective learning. We present two classes of algorithms corresponding to two learning goals. The first is one of classification, where the goal is straightforward: to improve the accuracy of classification. The second is one where both problem and solution components are textual, the goal here is to improve the alignment between problem and solution side representations. The algorithms are based on the idea of revising similarities between cases based on the conflicts between introspectively acquired knowledge and external knowledge, so as to improve effectiveness. In our experiments, LSI was used for introspective acquisition. External knowledge, alternately referred to as source of conflict, was modeled using class knowledge in the case of classification and solution text (along with their similarities) in alignment.

The algorithms are based on two steps. The first step involves identifying those pairs of cases which may have contributed negatively to the classification/alignment measure. The second step involves trying out changes on these "potentially spurious" similarities and repairing them. In supervised settings, a part of the training data can be used as a hold out set, and this is used to validate the effectiveness of repair actions and choosing between repair actions. It may be noted that given the duality of terms and cases, it is equally possible to try out a similar approach on modifying similarity between term pairs as well.

One advantage of using term pairs is that conflicts at a lexical level, based on resources like WordNet, can be conveniently modeled. Also, using terms make sense in an interaction setting where the external knowledge is human feedback, since it seems intuitive that humans would be more comfortable in assessing term pair similarities rather than case pair similarities. However, the downside of using term pairs in an introspective setting for classification is that, unlike cases, terms do not have class labels attached to them. So, additional heuristics would be needed to compensate for this lack of knowledge.

4.1 Improving Classification Effectiveness

We revise case representations with the goal of improving classification performance. The revision involves the following steps: identification of a list of potential case-pairs with spurious similarities, cross-validation to narrow down the list and identification of a combination of case-pairs that need repair, and finally revision of those case associations and creating a new representation of all cases based on the repair.

Detecting conflicts: The supervised classification setting has a set of labelled cases as part of the training data; another set of cases whose class labels are not known is used as test data. Similarity between textual problems is often modelled using co-occurrence statistics. Similarity estimated by the number of terms in common between the texts is the first order co-occurrence measure of similarity and that modelled by LSI (used in our work) are reflective of higher order co-occurrences. The differences between higher order association values and the first order association values is the first input to the algorithm. The second input is the class labels of the training cases. Let $sim_{first}(x_i,x_j)$ and $sim_{high}(x_i,x_j)$ be the similarities of the problem components of cases x_i and x_j obtained using first order co-occurrences and higher order co-occurrences respectively. Let the solutions of x_i and x_j belong to class c_i and c_j respectively. Using this notation, the procedure for identifying potentially spurious similarities is described in Table 1. The candidate list of potential spurious associations is ranked based on the change parameter *simDiff* as computed in Table 1.

Table 1. Identification of candidates with spurious similarities

Step 1: Calculate simDiff = $sim_{high}(x_i,x_j)$ - $sim_{first}(x_i,x_j)$
Step 2: If simDiff is positive and $c_i \neq c_j$ then $sim_{high}(x_i,x_j)$ is a potential spurious association
Step 3: If simDiff is negative and $c_i == c_j$ then $sim_{high}(x_i,x_j)$ is a potential spurious association

Cross Validation and Case Revision: In this step we consider batches of case similarity pairs from the ranked list of cases having potentially spurious similarities. The similarity values of these case pairs are now modified according

to one of two policies. The first policy, which we call the *OldValuesPolicy*, replaces $sim_{high}(x_i,x_j)$ by $sim_{first}(x_i,x_j)$ for a case pair x_i,x_j having potentially spurious similarity. This amounts to undoing the effect of LSI, and reverting back to an estimate of case similarity based on first order co-occurrences. The second policy is *MinMaxPolicy*, which assigns a revised value for all case pairs, x_i and x_j, with potential spurious associations as follows:

$$sim_{revised}(x_i, x_j) = \begin{cases} min & \text{if } simDiff > 0 \\ max & \text{if } simDiff < 0 \end{cases}$$

where *min* and *max* are the minimum and maximum values of all possible values of $sim_{high}(x_i,x_j)$ for any pair of cases x_i,x_j. The key intuition is to promote the value of similarity in the case *simDiff* is less than 0 and demote it otherwise. The choice of *min* and *max* maximally emphasize the effects due to these changes. *OldValuesPolicy* is a relatively less aggressive policy.

These modifications will now be used to change the case representations. To achieve this, we recognize that the CRN as shown in Figure 2 can be used alternately to find terms relevant to a set of given cases. In such a scenario, the cases will map onto IE nodes and similarity arcs will capture similarity between cases. The modified case similarity values, $sim_{revise}(x_i,x_j)$, obtained above, can thus be used to construct revised representations for terms and term similarities. The effectiveness of the revised representation of cases, obtained using the new term similarities, is measured using cross-validation (10 fold in all our experiments). The objective of cross validation is to determine the optimal number of candidates from the initial ranked list of candidates. To achieve this, the size of the batch is incremented by a fixed amount (500 in our experiments). The combination of potentially spurious pairs whose change resulted in the best performance over cross validation was recorded. It is intuitive that considering candidates too low in the ranked list do not contribute significantly, so the improvements in accuracy stabilize after incorporating changes to a certain number of case pair similarities; this is corroborated by our experiments reported in the next section.

Cases from the test data are now transformed using the new term representations resulting from the revised value of similarities corresponding to set of case pairs identified as above. The classification accuracy before and after the revision are compared.

4.2 Improving Alignment in TCBR

In many TCBR systems, both problem and solution components of cases are recorded in the form of text. Unlike the classification case, the goal of revision in such situations would be to improve the effectiveness of retrieval. Ideally, we would need recourse to a large collection of human relevance judgments, which unfortunately, are hard to come by. Case-base alignment is thus used in TCBR as a surrogate for retrieval effectiveness. Case-base alignment measures the extent to which similar problems have similar solutions. One approach to estimating alignment, as presented in [9], involves listing down the pairs of cases in a case-base in decreasing order of their problem side similarity. The correlation of the

problem side similarities as ordered above, and the corresponding solution side similarities gives an estimate of how strongly aligned the problem and solution components are. Another observation is relevant here: while high alignment requires that similar problems do have similar solutions, it does not require that dissimilar problems have dissimilar solutions. To take this asymmetry into account, a weighted correlation is used to define alignment in place of a simple correlation. The weights are problem side similarities, so cases with very similar problem components play a bigger role in influencing the measure. The weighted correlation, *wtCorr*, for any set of values x,y and weights w is defined as follows:

$$wtCorr(x, y, w) = wtCov(x, y, w)/\sqrt{wtCov(x, x, w) * wtCov(y, y, w)} \quad (2)$$

where *wtCov* is a weighted covariance function defined as :

$$wtCov(x, y, w) = (\sum_i w_i * (x_i - \bar{x}) * (y_i - \bar{y}))/\sum_i w_i \quad (3)$$

where in $\bar{x}$ and $\bar{y}$ denote the weighted means of x and y respectively. For calculating alignment using *wtCorr*, x is set as the problem pair similarities and y for solution side similarities with weights w being same as x. We use the above measure as the basis for our study on introspective revision for improving alignment. As in the classification case, this process also involves the following steps: identification of list of potential case-pairs with spurious similarities and identification of a combination of case-pairs that need to be modified.

Identifying Conflicts: The external knowledge source used for modelling conflicts in this case is solution side similarities. While we have estimated solution side similarities using LSI and first order co-occurrences, ideally they should evolve through system usage by interacting with an external agent. As in the classification case, let $P_{first}(i, j)$ and $P_{high}(i, j)$ be the similarities of problem components of cases i,j obtained from first order co-occurrences and higher order co-occurrences respectively. Similarly $S_{first}(i, j)$ and $S_{high}(i, j)$ be correspondingly defined for the solution components of cases i,j. Let P'_{high} be the matrix obtained by normalizing the values of P_{high} in the closed range of 0 and 1. Similarly $S'_{high}(i, j)$ is derived from S_{high}. Let PS be the set of all problem pair similarity values from the matrix P'_{high} arranged in decreasing order. Consider SS be the corresponding solutions component similarities from matrix S'_{high}. The procedure for asserting if a case-pair i,j is a potential candidate with spurious association is given in Table 2.

Identifying Candidates for Modification: In this step, we take one case pair at a time from the list of case pairs, which are potential candidates with spurious associations identified previously, and modify its similarity based on the policy to replace $P_{high}(i,j)$ with $P_{first}(i,j)$ for that case pair i,j. The effect of this modification is measured by re-computing the alignment. All the case pairs whose modifications improved the alignment are combined together.

Table 2. Identification of Potentially Spurious Associations for Alignment

Step 1: The initial correlation is calculated using iniCorrelation = wtCorr(PS,SS,PS) Step 2: New Problem and solution similarity lists are calculated using PS' = PS - $\{P'_{high}(i,j)\}$ SS' = SS - $\{S'_{high}(i,j)\}$ Step 3: Weighted correlation for the new problem and solution similarity lists is calculated using newCorrelation = wtCorr(PS',SS',PS') Step 4: If newCorrelation > iniCorrelation then $P_{high}(i,j)$ is a potential candidate with spurious association.

5 Evaluation and Experimental Results

We evaluated the effectiveness of our proposed introspective learning mechanisms with respect to two goals: text classification and improving alignment. The former was done on realistic textual data, and the latter on synthetic data.

5.1 Evaluation on Text Classification

Evaluation for classification effectiveness is done using accuracy as the performance metric since this is known to be appropriate for cases with single class label in datasets with equal class distributions. Cases constructed from the 20 Newsgroups corpus [10] was used for classification. One thousand messages from each of the 20 newsgroups were chosen at random and partitioned by the newsgroup name [10]. The following four sub-corpuses were created: SCIENCE from 4 science related groups; REC from 4 recreation related groups; HARDWARE from 2 problem discussion groups on Mac and PC and RELPOL from 2 groups on religion and politics. Each data set contains 15 test train pairs and average accuracy is reported for each data set for each method.

Weighted k Nearest Neighbor is the classifier used for all the experiments with the number of neighbors (k) fixed to 3. Hold out sets are constructed out of training data using 10 fold cross validation for identifying candidates whose similarities would be repaired. The limit on number of potential candidates considered for cross validation is set to 40,000. LSI dimensions are chosen by considering a candidate set of values and choosing the one that performs best over 10 fold transductive cross validation [11].

The results of the experiments are shown in Table 3. The column Method indicates the method used for classification. BaseLine is the case where no modification was done to the LSI-based case similarities. *MinMaxPolicy* and *OldValuesPolicy* are the two methods as described in Section 4.1. Also included are average test accuracies (over the 15 test train pairs) of a Nave Vector Space Model (VSM) approach (no LSI), Support Vector Machines over a linear kernel, Propositional Semantic Indexing (PSI) [12] and LogitBoost [13]. We have also included in our comparison Sprinkled LSI [4], which can be viewed as an

extension of LSI that biases the concepts mined based on class knowledge. Paired t-test was carried out between accuracies reported by each pair of methods over the 15 train test pairs; the values in bold indicate those that outperform the others at p = 0.05. PSI and LogitBoost were not subjected to the statistical test since we did not have access to the 15 values for each train-test pair, but the average values were available. A linear kernel was chosen for Support Vector Machines since it was reported in [14] that linear kernels work best over textual data. SVM is inherently a binary classifier and its multi-class extension is yet to be tried out on the four-class datasets.

There are several interesting observations based on the results presented in Table 3. Firstly, introspective revision achieves statistically significant improvements over baseline in all four datasets except RELPOL. On both binary classification datasets, our approach recorded higher averages compared to SVM, which is regarded as one of the best off-the-shelf text classifiers [15]. The Hardware dataset is known to be the hardest of the four datasets; this has been confirmed by alignment studies comparing all four datasets, as reported in [3]. Revision based on *MinMaxPolicy* significantly outperforms all other classifiers except Sprinkled LSI on this dataset. This is because of the lack of separability between messages related to Mac and PC domains, resulting from a large overlap of vocabulary between these classes. Introspective revision is clearly effective in remodeling the term similarities and improving separability. The performances gains using revision are not so illustrative in RELPOL, which is a relatively easy domain with a lot less scope of improvement; this is clear from the good accuracy figure in baseline. PSI is seen to be quite effective on HARDWARE, but is outperformed by introspective revision in the multi-class datasets, by a wide margin. *MinMaxPolicy* outperforms *OldValuesPolicy* on all datasets.

The effect of number of potential candidates considered on accuracy obtained in cross validation for training data of HARDWARE and REC for both MinMaxPolicy and OldValuesPolicy is shown in Figure 3. This illustrates the fact that performance stabilizes beyond a certain number of candidate pairs, and revisions over just a few thousand pairs gives a considerable improvement over the baseline.

Table 3. Empirical Results for the method of modifying case similarities

Method	**HARDWARE**	**RELPOL**	**SCIENCE**	**REC**
Naive VSM	59.51	70.51	54.89	62.79
BaseLine	70.28	93.42	81.98	83.46
MinMaxpolicy	**80.24**	93.49	**84.40**	**87.43**
OldValuesPolicy	74.55	93.52	82.82	85.22
Sprinkled LSI	**80.42**	93.89	80.60	**86.99**
SVM	78.82	91.86	-	-

Method	**HARDWARE**	**RELPOL**	**SCIENCE**	**REC**
LogitBoost	77.99	79.67	73.77	87.15
PSI	80.1	91.2	76.2	66.28

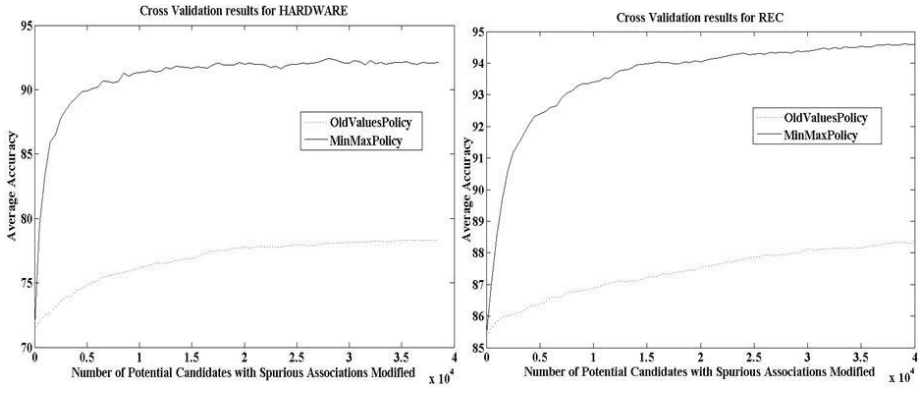


Fig. 3. Cross Validation Results for HARDWARE and REC

5.2 Evaluation on TCBR over Textual Problems and Solutions

The Alignment improvement technique proposed in Section 4.2 is evaluated on a synthetic dataset. We are not sure of how the experiments would scale in realistic TCBR systems, so the results should be viewed as a proof-of-concept. The synthetic dataset comprising cases with textual problem and solution components is created using the Deerwester toy dataset [7]. It is a collection of nine cases, five of which are about human-computer interaction and the remaining four about graphs. Instead of using the class label as solution, randomly selected terms from the problems are used as their corresponding solutions. The dataset constructed is shown in Figure 4. The complexity of this dataset is varied iteratively by modifying the solutions of a pair of problems. Given a pair of cases, the terms common to their solutions are replaced by new terms in one of the cases. This has the effect of reducing the solution side similarity of the two cases. At each iteration, a new pair of cases is subjected to this change, in addition to the changes done in the earlier iterations. It is intuitive to expect that each iteration will result in a case-base that is poorly aligned relative to the one generated in the previous iteration. The results of these experiments are shown in Figure 5. The *initial alignment* computed using the *wtCorr* method along with *modified alignment* computed using the method proposed are reported. Figure 5 shows the plots of *initial alignment* and *modified alignmen*t for number of solution pairs that are modified.

Figure 5 shows the performance of the introspective alignment improvement method for various number of solution pairs modified on the synthetic dataset. It is observed that as the number of solutions pairs which are modified is increased the *initial alignment* decreases from the baseline case. And for the varied complexity of the system, there is an improvement in the alignment measured by *modified alignment*. These initial results on the synthetic dataset are encouraging for the fact that the improvement observed is in the order of 0.1.

Problem	Solution
Human Machine interface for Lab ABC Computer Applications	Human
A survey of user opinion of computer system response time	System, response
The EPS user interface management system	System
System and human system engineering testing of EPS	Human, system
Relation of user perceived response to time error measurement	Response
The generation of random, binary, unordered trees	Trees
The intersection graph of paths in trees	Trees, Graph
Graph Minors 4: Widths of trees and well quasi ordering	Trees, Graph
Graph Minors: A survey	Graph

Fig. 4. Deerwester Dataset

Solution PairsModified	Initial Alignment	Modified Alignment
0	0.6670	0.7134
1	0.2317	0.3187
2	0.0129	0.1393
3	-0.2183	-0.1098
4	-0.2183	-0.1098
5	-0.3056	-0.2296

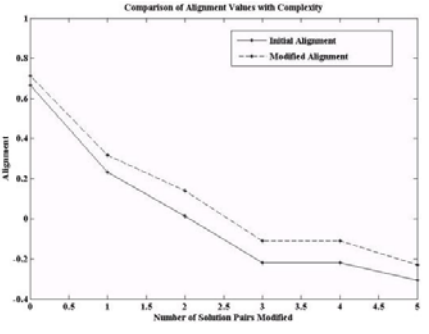


Fig. 5. Performance of Alignment Measures for various Complexities of System

6 Related Work

The current work can be viewed as an instance of Knowledge Revision. One of the early systems that incorporated learning by interaction with an expert was Protos [16]. In contrast to most previous work where knowledge revision has been applied in more formal settings over rule based systems, this work shows that TCBR systems with numeric representations of their underlying knowledge can benefit by revising their knowledge based on conflicts. A significant advantage is that no human intervention is involved in generating conflicts and triggering revision.

Several works in text classification have attempted to compensate for the fact that the training corpus is not adequately representative of domain knowledge. Zelikovitz et al. [17] use external background text in conjunction with training data. This is particularly effective when the available training data set is small. Chakraborti et al. [4] incorporates the class knowledge into LSI by appending the class labels as additional terms into the term-case matrix. Sam Scott and Stan Matwin [18] use WordNet to improve classification performance, and Gabrilovich and Malkovitch [19] use revised representations based on Wikipedia to achieve a similar goal. Our work significantly differs from these approaches in that it is more general in scope and can be easily extended to cater to widely different learning

goals. None of the above- mentioned approaches model the conflict between internal and external sources. The key contribution in our work is an approach that selectively modifies regions of representation that need revision, and thus combines the best of what is known and what needs to be known. This is cognitively intuitive. As humans we are conservative in throwing away things we know, in presence of a conflict or dilemma. Also the learning goal in our approach is flexible, as it can be directly used to drive the cross validation process for identifying candidate pairs for revision, starting from any representations. This is not true of approaches like Sprinkling, for example, which have been specifically targeted to improve classification performance, and is closely tied to LSI-based representations.

7 Conclusion and Outlook

The main contribution of this paper is a novel learning framework based on identifying potential sources of conflict between introspectively acquired knowledge and external knowledge, and repairing the case representations accordingly, with the goal of improving effectiveness in retrieval or classification. The experimental results over classification domains suggest that this approach succeeds in making substantial improvements on datasets which are hard to classify. This is significant since the results show that instance based learners could achieve performance comparable to the best in line classifiers such as SVM, while retaining advantages in terms of incremental and lazy learning, and more interpretable representations. Given the trade-off between knowledge acquisition overhead and effectiveness of retrieval, completely automated solutions often fail to be effective, and those involving substantial human intervention have large deployment lead times and may have poor user acceptance. This paper has attempted to position introspective knowledge revision as a middle ground between these two extremes. Though we have presented the idea of revising case similarities mined using LSI, the approach is fairly general and can be extended to develop a plug-in for potentially improving the performance of any knowledge light TCBR system with numeric representations of term and/or case similarities.

There are several interesting avenues for extending the current work. Firstly, we intend to explore other applications that involve different choices of background (WordNet) and linguistic knowledge (Wikipedia) as additional conflict sources, which model inter-term semantic relatedness, to realize learning over a richer set of conflicts. In the current work, we have treated each change to case pair similarity as independent of any other change. While the performance gains are substantial even under this assumption, it is interesting to wonder if we can benefit by modelling interactions between case pairs. Solving the full-blown constrained optimization problem taking into account all possible interactions between case-pairs is indeed a holy grail.

The idea of detecting spurious similarities can be used for eliciting feedback from an expert in an interaction setting. Unlike in introspective revision, it would be important to ensure that the number of candidate pairs on whom feedback is sought, is limited to as few as possible. An observation in that context is that

introspective revision compensates for the lack of reliable human judgements, by exploiting changes over a large number of candidate suspicions.

References

1. Aha, D.W., Muoz-Avila, H.: Introduction: Interactive case-based reasoning. Applied Intelligence 14(1), 7–8 (2001)
2. Bonzano, A., Cunningham, P., Smyth, B.: Using introspective learning to improve retrieval in cbr: A case study in air traffic control. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 291–302. Springer, Heidelberg (1997)
3. Chakraborti, S., Beresi, U., Wiratunga, N., Massie, S., Lothian, R., Khemani, D.: Visualizing and evaluating complexity of textual case bases. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 104–119. Springer, Heidelberg (2008)
4. Chakraborti, S., Lothian, R., Wiratunga, N., Watt, S.: Sprinkling: Supervised latent semantic indexing. In: Lalmas, M., MacFarlane, A., Rüger, S.M., Tombros, A., Tsikrika, T., Yavlinsky, A. (eds.) ECIR 2006. LNCS, vol. 3936, pp. 510–514. Springer, Heidelberg (2006)
5. Diaz-agudo, B., Gonzalez-Calero, P.A.: Abstract formal concept analysis as a support technique for cbr. Knowledge Based Systems 14(3-4), 163–171 (2001)
6. Lenz, M.: Case Retrieval Nets as a Model for Building Flexible Information Systems. PhD dissertation, Humboldt University Berlin. Faculty of Mathematics and Natural Sciences (1999)
7. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990)
8. Kontostathis, A., Pottenger, W.M.: Detecting patterns in lsi term-term matrix. In: Proceedings of IEEE ICDM 2002 workshop, pp. 243–248 (2002)
9. Raghunandan, M.A., Chakraborti, S., Khemani, D.: Robust measures of complexity in tcbr. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 270–284. Springer, Heidelberg (2009)
10. Mitchell, T.: Machine Learning. Mc Graw Hill International (1997)
11. Vapnik, V.N.: Statistical Learning theory. Wiley, Chichester (1998)
12. Wiratunga, N., Lothian, R., Chakraborti, S., Koychev, I.: A propositional approach to textual case indexing. In: Proc. of 9th European Conference on Principles and Practice of KDD, pp. 380–391 (2005)
13. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. Annals of Statistics 28(2), 337–407 (1988)
14. Joachims, T.: Making large-scale svm learning practical. In: Advances in Kernel Methods: Support Vector Learning, pp. 169–184 (1999)
15. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
16. Porter, B.W., Bareiss, R., Holte, R.C.: Concept learning and heuristic classification in weak-theory domains. Artificial Intelligence 45(1-2), 741–758 (1993)
17. Zelkovitz, S., Hirsh, H.: Using lsi for text classification in the presence of background text. In: Finin, T.W., Yesha, Y., Nicholas, C. (eds.) CIKM 1992. LNCS, vol. 752, Springer, Heidelberg (1993)
18. Sam, S., Stan, M.: Text classification using wordnet hypernyms. In: Proc. of Workshop on usage of WordNet in NLP systems, pp. 45–51 (1998)
19. Gabrowich, E., Markovith, S.: Computing semantic relatedness using wikipedia based explicit semantic analysis. In: Proc. of 20th International Joint Conference on AI, pp. 1606–1611 (2007)

A General Introspective Reasoning Approach to Web Search for Case Adaptation

David Leake and Jay Powell*

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

Abstract. Acquiring adaptation knowledge for case-based reasoning systems is a challenging problem. Such knowledge is typically elicited from domain experts or extracted from the case-base itself. However, the ability to acquire expert knowledge is limited by expert availability or cost, and the ability to acquire knowledge from the case base is limited by the the set of cases already encountered. The WebAdapt system [20] applies an alternative approach to acquiring case knowledge, using a knowledge planning process to mine it as needed from Web sources. This paper presents two extensions to WebAdapt's approach, aimed at increasing the method's generality and ease of application to new domains. The first extension applies introspective reasoning to guide recovery from adaptation failures. The second extension applies reinforcement learning to the problem of selecting knowledge sources to mine, in order to manage the exploration/exploitation tradeoff for system knowledge. The benefits and generality of these extensions are assessed in evaluations applying them in three highly different domains, with encouraging results.

1 Introduction

The World Wide Web provides an unprecedented knowledge resource. The availability of Web resources has led to optimism for a transformation of AI through harnessing Web knowledge sources [16], both in the form of large bodies of formally encoded knowledge, such as OpenCyc [10], and of less formal knowledge sources, such as Wikipedia [28]. The WebAdapt project [20] aims to exploit such knowledge to help address the classic CBR problem of knowledge acquisition for case adaptation. Previous work on WebAdapt conducted initial explorations on just-in-time mining of large-scale, freely available Web-based resources for knowledge to support case adaptation, as a first step towards the goal of developing a flexible, extensible knowledge-planning framework with the generality to enable it to be easily applied to new domains. WebAdapt starts from minimal pre-coded knowledge and applies domain independent search strategies to acquire adaptation-relevant knowledge on demand from Web sources. This paper presents research on extending WebAdapt's initial framework in two ways:

* This material is based on work supported by the National Science Foundation under Grant No. OCI-0721674. We thank the anonymous reviewers for their comments.

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 186–200, 2010.

Applying introspective reasoning to aid error recovery and learning to select knowledge sources to search.

When mining pre-existing Web knowledge sources to address an open-ended set of adaptation problems, some failures are inevitable. However, the initial WebAdapt framework had no capability for failure recovery; the system would simply report that it was unable to suggest an adaptation. This paper presents an extension in which WebAdapt's model has been extended to include an introspective reasoning component (cf. [3,7]) to diagnose and respond to failures in the adaptation process.

As new knowledge sources become available, or as the framework is applied to new task domains, it will not initially be clear which sources are best for different types of queries (cf. [22]). To address this we have added a reinforcement learning component to guide exploration of candidate knowledge sources.

Our previous work illustrated the system's performance for the single domain of travel itinerary planning, drawing on the formalized knowledge of OpenCyc [10], the informal natural language text of Wikipedia [28], and the geographical information of the Geonames GIS database [14]. Both Wikipedia and OpenCyc are large, comprehensive sources that cover a wide variety of domains. Because a goal of the WebAdapt project is to develop a general framework for applying case adaptation which can be used in multiple domains, the paper tests the system's generality by evaluating its overall performance for three disparate domains: travel itinerary planning, menu planning, and software recommendation. For our evaluation, the system draws not only on OpenCyc, Wikipedia, and Geonames, but also on the USDA database of nutritional information [1] and the Macintosh OS X dashboard widget web pages [2][1]. The combination of the three task domains and five knowledge sources enable assessing the system's ability to manage problems with sources of varying applicability.

Initial results also show that a combination of introspective reasoning and problem dispatching knowledge reduce the reasoning failures produced when the system is still learning about the coverage of sources at its disposal, and that the learning approach is effective at guiding source choice.

Section 3 provides an overview of WebAdapt's adaptation process and the components that guide this process. Section 4 describes the introspective model used by the WebAdapt system, and Section 5 discusses the system's new problem dispatching model. Section 6 presents experimental results showing the effectiveness of WebAdapt's introspective model and new problem dispatching model.

2 Motivations for Web Mining to Support Case Adaptation

Case adaptation and knowledge capture are classic problems for case-based reasoning (CBR). Acquiring the knowledge necessary for automated adaptation is

[1] The software recommendation domain could be expanded to include, e.g. Google Apps. Tests were limited to Mac dashboard widgets to examine system performance for a domain covered by a single knowledge source.

a difficult task, and it is not uncommon for systems to leave adaptation to the user [4,19]. Much work has been done to exploit knowledge already captured in the local case base for adaptation (for a survey, see [23]). However, these approaches are inherently limited by the knowledge in the case base, which may be difficult or prohibitively expensive to acquire and maintain by hand. For example, for a software recommendation system, the frequent introduction of new packages and ongoing updates to specifications of existing packages would require constant maintenance of internal knowledge sources. In contrast, once general procedures are defined to access a Web source, just-in-time Web mining for needed information enables the system to profit immediately from external updates to sources and to process external information only as needed, rather than attempting to anticipate which information may be needed in the future.

3 WebAdapt's Adaptation Process

This section summarizes the WebAdapt framework, as described more fully in [20]. The work focuses on substitution adaptations, in which the system must replace a role-filler in a case with another which satisfies a given set of constraints.

WebAdapt's adaptation process (illustrated in Figure 1) begins when a user makes a request to find substitutions for a role-filler of a case (e.g., for 'Louvre' as one of the sights in a Paris sight-seeing itinerary). WebAdapt begins by generating top-level knowledge goals (step 1 in Figure 1). The system may generate lower-level knowledge goals during the knowledge planning process as the need arises. A top-level knowledge goal describes the knowledge needed to respond to a user's query, while a lower-level knowledge goal describes required intermediate knowledge. For example, when finding substitutions for 'Louvre' using Wikipedia, WebAdapt will generate a top-level knowledge goal for finding a ranked set of candidate substitutions and several low-level knowledge goals such as: (1) *find a Wikipedia entry for 'Louvre'* , (2) *hypothesize constraints for 'Louvre'*, (3) *find unranked candidate substitutions for 'Louvre'*.

Once a goal has been formulated, WebAdapt selects knowledge sources expected to satisfy its goals (step 2). Knowledge sources are selected based on (1) prior cases for satisfying similar knowledge goals, and (2) source profiles reflecting a source's performance in several categories.

After a knowledge source (or set of sources) has been selected, WebAdapt passes the knowledge goal and source(s) to a planning component (step 3). The component begins by attempting to retrieve a prior plan satisfying the given knowledge goals, and generates one from scratch using the planner UCPOP [25] if no plan is found. A plan is then executed (step 4), where any empty role-fillers in the plan are instantiated with the knowledge acquired from each source. For example, if the plan calls for Wikipedia to be mined for substitutions for 'Louvre', WebAdapt begins by generating a Google query to find the Wikipedia page for 'Louvre'. The Wikipedia entry returned by Google is parsed for a set of links to category pages which become the set of hypothesized constraints (e.g., 'Visitor attractions in Paris' and 'Art museums in Paris'). The URLs for the categories

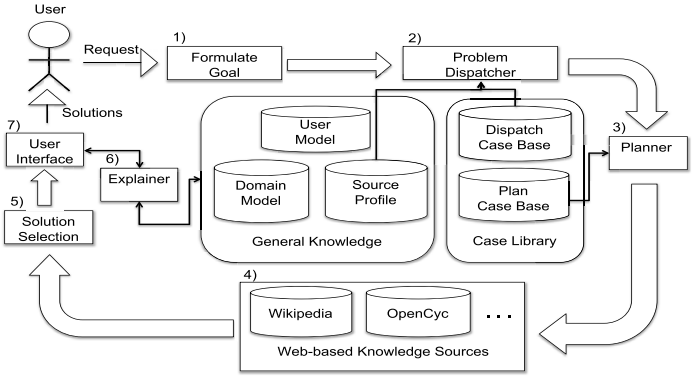


Fig. 1. WebAdapt's adaptation process and knowledge sources

are followed and parsed. The entries discovered under each category form the set of candidate substitutions (step 5). These may be refined using source specific techniques (e.g., searching for the word 'Paris' in a candidate's Wikipedia entry) and displayed to the user. The system's domain model is then updated with the constraints and candidates discovered during the search process.

3.1 Knowledge Planning Operators

WebAdapt's operators are described by a set of pre- and post-conditions, where an operator's preconditions specify the knowledge necessary for execution, and the postconditions specify the knowledge generated by an operator. Operators are defined in terms of a vocabulary of roles filled during the planning process, either from the initial knowledge goal or based on intermediate results. Role-fillers are typically frames describing structured knowledge discovered from each source (e.g., the URL addresses and titles of Wikipedia pages). Because the methods required to interact with each source are different, WebAdapt's library of general domain independent search and transformation operators calls upon a small set of source-specific operators to form operational plans (e.g., to search a source's abstraction hierarchy).

3.2 WebAdapt's Internal Knowledge

WebAdapt relies on several sources of knowledge to guide its adaptation process.

Source Profiles: WebAdapt maintains statistical information about each knowledge source, updated each time a source is accessed, to guide source selection. This includes average access times, uptimes, estimated coverage (measured by percent of queries that generate substitutions), and estimated diversity of results (by a method similar to that in [26]).

Cases for Dispatching and Adaptation: Source selection is also influenced by a case-base describing the results of prior knowledge planning episodes. After

attempting to generate candidate substitutions, the system stores a case containing the adapted item, the constraints hypothesized for the item, the knowledge sources searched, and a Boolean value indicating whether candidates were found using the source. WebAdapt uses this information to select sources which satisfied similar prior requests.

Domain Model: The domain model reflects the abstraction hierarchies of mined sources. Nodes and leaves discovered during the mining process and traversed links between constraints and candidates are added to the domain model.

3.3 WebAdapt's External Knowledge Sources

Each external knowledge source used is defined by an explicit abstraction hierarchy, which is traversed to find viable substitutions. Nodes in the abstraction hierarchy are viewed as constraints while leaves are candidate substitutions. The system begins by discovering an item to adapt in a source's hierarchy. The set of abstract nodes the item falls under form the set of hypothesized constraints which are expanded to discover candidate substitutions.

A central goal of the WebAdapt project is to provide a general domain-independent Web mining method applicable to multiple knowledge sources and domains. WebAdapt can process any source with a hierarchical knowledge organization; for each source, a small set of basic operators for traversing the hierarchy must be defined and provided to the system (see Section 6 for specific examples). No other knowledge modeling is required to use a new source. To demonstrate its applicability to unstructured sources, WebAdapt currently uses its own procedures to mine textual information, but the system's application will be facilitated by efforts such as the Open Data Movement and the structured information available in sources such as ontologies, DBpedia (`dbpedia.org`), Freebase (`freebase.com`), and Search Monkey (`developer.yahoo.com/searchmonkey`).

4 WebAdapt's Introspective Failure Recovery

Reasoning failures are inevitable when reasoning with incomplete domain knowledge. When reasoning failures occur in WebAdapt, they are often caused by missing background knowledge or by the system querying the wrong source. The goal of WebAdapt's introspective model is to detect reasoning failures and automatically update WebAdapt's background knowledge to avoid future reasoning failures. The introspective model used by WebAdapt is domain independent—it focuses on reasoning about the knowledge acquired by WebAdapt and how that knowledge influences the adaptation process, not on individual domains.

WebAdapt's adaptation process (shown in Figure 1, and represented as a cloud in Figure 2) is modeled as a plan, described in terms of knowledge planning operators described in section 3. The introspective process, shown in Figure 2, is divided into three stages: (1) monitoring, (2) blame assessment, and (3) recovery. Monitoring (Step 1 in Figure 2) keeps a trace of the executing plan in WebAdapt's adaptation process for a single problem solving episode. It also

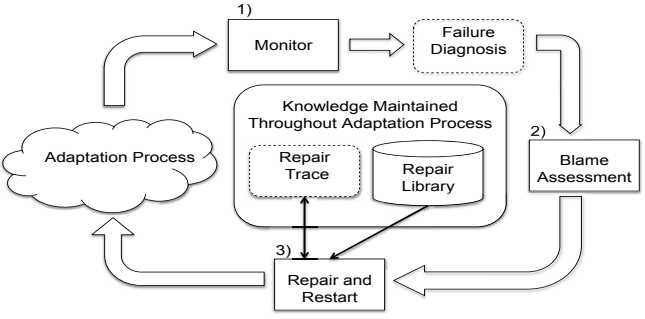


Fig. 2. WebAdapt's introspective process and knowledge

monitors the state of the plan's role-fillers during execution and interacts with the plan if expectations about the states of role fillers are violated. Role-fillers include the case being adapted, sources used, the plan used to mine sources, and results. A role-filler expectation violation may occur when no solution can be found (e.g., if a source is temporarily unavailable), if the knowledge sought by the system cannot be found in the knowledge source, or if the user rejects a proposed solution (e.g., if a travel suggestion fails to satisfy user preferences).

After identifying a reasoning failure, WebAdapt generates a failure diagnosis from its reasoning trace which is used to identify the primary cause of the failure. This failure diagnosis becomes input for the blame assessment stage (Step 2 in Figure 2). A taxonomy based on [8] is used to categorize reasoning failures (cf., [3]). WebAdapt focuses on reasoning failures concerning knowledge states (i.e., the acquisition and application of background knowledge). An executing plan contains a set of knowledge goals for every action in the plan. Each type of knowledge goal is associated with potential reasons for its failure. The types of unsatisfied knowledge goals at the time of a failure are used to diagnose the failure's cause, and the taxonomy categories suggest associated fixes. For example, if the plan execution stage fails (Step 4 in Figure 1) WebAdapt examines the unsatisfied knowledge goals of the executing plan. Table 1 shows examples of common reasoning failures and their sources. The taxonomy suggests deficiencies in the system's problem dispatching knowledge for the first two failures (for which the repair is for the dispatching stage to be re-executed), while the third suggests either a problem with the system's dispatching knowledge or its choice of strategy for mining a source (for which a new strategy is created by knowledge planning).

The recovery stage (Step 3 in Figure 2) takes the cause(s) of failures identified in Step 2 and attempts to re-execute the plan at the point of failure. For example, after WebAdapt chooses to use the USDA nutritional database to suggest candidate substitutions for the Louvre, which cannot be found in the database. Here the *Discover item in source* knowledge goal failed. The taxonomy suggests that the source of the failure is the problem dispatching stage. A case is added to the dispatch case base indicating that the USDA database cannot suggest substitutions for the Louvre. The adaptation process is then restarted

Table 1. Examples of knowledge goals and possible sources of failure

Failed Knowledge goal	Failure Point(s)	Explanation
Discover item in source	Problem dispatching	wrong source chosen
Hypothesize seed constraints	Problem dispatching	wrong source chosen
	Planning stage	wrong strategy chosen
Hypothesize candidate substitutions	Problem dispatching	wrong source chosen
	Planning stage	wrong strategy chosen

at the problem dispatching stage, where the system draws on its background knowledge to suggest a new source to mine.

5 Learning Which Knowledge Sources to Use

Different knowledge sources have different capabilities; some are large and comprehensive while others are small and specialized. For example, the Geonames GIS database can provide destinations close to the 'Louvre', but has a poor semantic description of the 'Louvre' (i.e., 'Building'). Wikipedia has a rich semantic description of the 'Louvre' (e.g., 'Visitor attraction in Paris', 'Archaeological museum in France', 'Art museum in Paris'), but contains little geographical information. Wikipedia can suggest substitutions for more problems than Geonames can, but is not the best choice for adaptation constraints based on location.

In accordance with the goal to minimize WebAdapt's dependence on pre-coded knowledge, the framework does not assume *a priori* knowledge of which sources to use for particular problems. Instead, WebAdapt builds up two knowledge sources for guiding dispatching of problems to sources: (1) "dispatching cases," storing information about the performance of a source for a particular query, and (2) "source profiles," storing generalized information about source performance.

Prior experience with particular sources is often a good predictor of future performance. However, the relative value of sources may change, as previously-tried sources become obsolete or are updated, or if new sources (potentially more comprehensive or more specialized) come on line. Likewise, early successful experience with a source might bias the system towards reusing that source, despite other sources having even higher quality. In fact, this happened with the experience-based dispatching model originally used by WebAdapt, which ended up using Wikipedia for over 90% of its queries [20].

Source selection presents a classic "exploration versus exploitation" problem: whether to expend effort to explore new sources or to exploit sources which have already proven useful. To address this question in a principled way, a Q-learning component has been added to WebAdapt's dispatching model.

Q-learning is a form of reinforcement learning (RL) [27]. RL techniques focus on learning what actions to take in particular states, where an agent is not informed of what the best actions are, but receives eventual payoff information. An RL agent learns a utility value for each state-action pair through trial and error. Reinforcement learning algorithms do not require a model of the agent's environment in order to reason.

WebAdapt relies on Q-learning to guide exploration/exploitation as it populates its dispatching case base and builds its source profiles. WebAdapt's Q-learning estimates the utility of taking a particular dispatching action in a given state. The possible states are:

1. WebAdapt cannot suggest a source to mine due to having neither dispatching cases satisfying similar knowledge goals nor source profile knowledge.
2. WebAdapt cannot suggest sources to mine based on dispatching cases but can suggest sources based on general source profile information.
3. WebAdapt can suggest sources to mine based on both prior cases satisfying similar knowledge goals and source profile information.

In state (1) the system randomly chooses a source to mine. In state (2) WebAdapt must rely on source profile information. This may occur when the dispatch case base is empty or when presented with a novel problem, which provides a coarse-grained approach to source selection. A Q-value is associated with each knowledge source and reflects the probability that the source will produce desirable results; the source with the maximal probability is chosen according to the Q-learning policy. State (3) occurs when the system has previously mined one or more sources for knowledge related to the given problem and can use content information to guide source selection. Here a Q-value is associated with each dispatching case, and the maximal case is chosen according to the Q-learning policy. This presents a finer-grained approach to source selection.

Figure 3 describes WebAdapt's algorithm for ranking and selecting sources. When presented with a set of knowledge goals, WebAdapt first ranks the available sources based on (1) prior cases recording sources used to satisfy similar knowledge goals, and (2) source profile information. WebAdapt takes a weighted

1: $rankedSources \leftarrow \phi$
2: {calculating each source's rank}
3: **for all** $s \in KnowledgeSources$ **do**
4: $\quad contentValue \leftarrow retrieveContentRating(s, goal)$
5: $\quad profileValue \leftarrow retrieveProfileRating(s)$
6: $\quad rankedSources \leftarrow rankedSources \cup \{s, \frac{0.5*contentValue+0.5*profileValue}{2}\}$
7: **end for**
8: {select a source to use}
9: **for all** $s \in Sort(rankedSources)$ **do**
10: $\quad \epsilon \leftarrow 1 - getSourceUtlity(s)$
11: $\quad$ **if** $random(0,1) < \epsilon$ **then**
12: $\quad\quad$ **return** s
13: $\quad$ **else**
14: $\quad\quad rankedSources \leftarrow rankedSources - \{s\}$
15: $\quad$ **end if**
16: **end for**
17: **return** $Sort(rankedSource)$

Fig. 3. Algorithm for ranking knowledge sources

average of the similarity of the content of the closest previous knowledge goal successfully addressed by the source in the past, and of a general "source quality" value based on its aggregate profile statistics (see Figure 3). Sources that have returned acceptable results are expected to have a greater probability of future success and are ranked higher than sources unable to satisfy similar knowledge goals, or sources which have not been considered for satisfying similar knowledge goals. After ranking all available sources, WebAdapt uses an ϵ-greedy policy to select the source(s) to mine (where ϵ is a source's normalized *utility*). WebAdapt will choose the top-ranked source with probability ϵ, and consider the next highest ranked source with probability $1 - \epsilon$, where the value of ϵ is dependent on the source currently under consideration. After WebAdapt selects a source (or sources) to mine, it tracks the outcome of the knowledge discovery effort. A positive reward is given when a user selects one of the system's candidate substitutions, and a negative reward otherwise. The reward is then propagated to each dispatch case that influenced source selection and to each source profile, using the standard Q-learning policy formula [27].

6 Evaluation

Our evaluation addresses the following questions:

1. *Benefits of introspective reasoning:* How does the addition of introspective reasoning affect WebAdapt's ability to solve problems?
2. *Usefulness of exploration:* How does the addition of reinforcement learning for sources affect WebAdapt's ability to successfully exploit a variety of knowledge sources?
3. *Generality of approach:* How does WebAdapt's performance vary for different domains?

To explore the generality of the methods, the experiments tested adaptation suggestions for cases taken from the domains of travel planning, menu planning, and software recommendation. The travel planning cases were taken from Frommer's Paris Travel Guide [13], the menu planning cases from Epicurious [12], and the software recommendation cases from a set of Mac OS X dashboard widgets [2]. Problems from these domains addressed using information mined from five Web knowledge sources, as described in the following section. Problems were considered "solved" if the system was able to propose a solution satisfying the desired constraints. The average problem solving time was one minute. We are currently studying processing time behavior and the role of search case retention strategies in reducing search time [21].

Knowledge for the experimental problems was drawn from two general sources, one informal (Wikipedia) and one formal (OpenCyc), and three specialized sources, Geonames, USDA SR20, and an Apple Downloads library. For each source, simple processing was performed to develop a usable hierarchical structure. Table 3 summarizes the results of this processing.

Table 2. Query method, hierarchy nodes, and leaves for each source

Source	Query method	Nodes	Leaves
Wikipedia	Google[2]	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[3]	Widget category	Widgets

Table 3. Applicability of knowledge sources

Domain	Wikipedia	OpenCyc	Geonames	USDA SR21	Mac Widgets
Travel planning	Yes	Yes	Yes	No	No
Menu planning	Yes	Yes	No	Yes	No
Software recommendation	No	No	No	No	Yes

In the following experiments all failures were triggered automatically within the reasoning process; none were triggered by a user.

Experimental Design for Question 1 (Benefits of introspective reasoning): An ablation study was performed, with the following configurations: (1) No introspective reasoning, learned problem dispatching, (2) Exhaustive search, and (3) Introspective reasoning, learned problem dispatching.

In configuration (1) WebAdapt did not repair reasoning failures but could learn from successful reasoning episodes; in configuration (2) WebAdapt would iterate over the set of possible sources until a suitable one was found. In configuration (3) WebAdapt was able to repair failures and learn dispatching knowledge. Thirteen randomly chosen features were selected for adaptation (five itinerary items, five ingredients, and three widgets, respectively). Each item was adapted ten times using each configuration (WebAdapt picks a source at random when it has no prior dispatching knowledge). This was repeated for ten rounds of testing resulting in 520 adaptations.

Results for Question 1: Configurations (1) and (2) present a baseline to compare the efficacy of introspective reasoning and problem dispatching knowledge. The system was only able to suggest substitutions 48% of the time in configuration (1), while it was able to suggest substitutions 100% of the time for configurations (2) and (3). With introspection, it was able to suggest substitutions 100% of the time, regardless of whether dispatching knowledge learning was enabled. Configuration (3) resulted in a 76% decrease in reasoning failures over configuration (2).

Experimental Design for Question 2 (Usefulness of exploration): An ablation study was also used to evaluate question 2. Four system configurations were used to better understand the effects of the knowledge states described in Section 5: (1) No exploration, (2) Exploration using source profile knowledge only, (3)

[2] e.g., *site:en.wikipedia.org "Louvre Paris France"*

[3] e.g., *site:www.apple.com/downloads/dashboard "iTunes Widget"*

Table 4. Average reasoning failures per adaptation

Configuration	Average Reasoning Failures
(1) No exploration	0.24
(2) Exploration using source profiles only	0.84
(3) Exploration with learning	0.34
(4) Exploration with full dispatching knowledge	0.07

Exploration using learned source profile knowledge and dispatching cases, and (4) Exploration with full dispatching knowledge.

Introspective reasoning was enabled in each of the configurations. Initially, sources were selected at random. By case-based dispatching, the source discovered to contain knowledge related to a problem was always used for similar problems encountered in the future. This approach favors Wikipedia, which was able to solve 94% of the given problems, at the cost of ignoring potentially interesting candidates from other sources. In configuration 2, only source profile knowledge was used to dispatch problems. In configuration 3, source profile knowledge and the dispatching case base were used, where the dispatching case base was built from scratch. In configuration 4, the system was given a dispatching case base with cases describing how every problem could be solved and source profile knowledge acquired from a prior round of testing; here the goal was a worst-case test of the potential degradation in exploring new sources. Fifty-two case features were randomly chosen for adaptation. These items were adapted ten times (to identify trends when sources were selected randomly when dispatching knowledge was unavailable) under each configuration, resulting in 2,080 adaptations.

Results for Question 2: Table 4 shows the average number of reasoning failures incurred by the four configurations for each of the thirteen problems solved. The system was able to present candidate substitutions for each problem presented, though some problems were more difficult to solve than others (e.g., one could only be solved by Wikipedia).

Configuration (4) resulted in the fewest reasoning failures. When given full dispatching knowledge the system learns to explore only a small percentage of the time. Maintaining some exploration is advantageous when source coverage may grow and the system needs to check if sources contain new knowledge. Configuration (2) resulted in the highest failure rate while there was a 42% increase in the failure rate of configuration (3) over configuration (1). Although the system could always use the first source found capable of solving a problem as in configuration (1), there is not a significant amount of overhead associated with learning dispatching knowledge from scratch.

Experimental Design for Question 3 (Generality of approach): A random set of forty-eight features from each domain were chosen for adaptation; this same set of features was used for each trial. This was repeated ten times to establish trends when sources are initially picked at random for a total of 480 adaptations. For this experiment the system learned both problem dispatching and source profile knowledge from scratch during each round of testing, with introspection

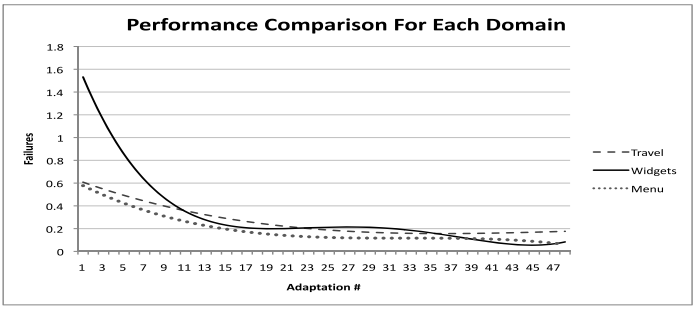


Fig. 4. Reasoning failures for individual domains

enabled to repair reasoning failures. The goal of this experiment was to see if WebAdapt's performance in each domain was similar.

Results for Question 3: Figure 4 shows failure trends for each individual domain. The standard deviations for the travel, widget, and menu domains were 0.44, 0.53, and 0.39, respectively. The failure rates for the menu and travel domains are very similar. Each of these domains can be solved by three of the five available knowledge sources. The initial failure rate for the Mac Widget domain is high because this domain can only be solved by one knowledge source. After eleven adaptations the failure rate for this domain is comparable for both the menu and travel domains. Failures are dependent on the number of sources to explore. Even when given no a priori knowledge about each source the system learned quickly to reduce failures. The final failure rate in each domain of 0.2 was the result of noise from the Q-learning algorithm. Source selection was based on an ϵ-greedy policy where sources are chosen with probability $1 - \epsilon$. In the experiments, the value of ϵ for a source never reached zero, resulting in occasional incorrect source choices.

Overall Discussion: Without introspective reasoning, WebAdapt is unable to reason effectively when presented with problems from a variety of domains. Adding introspective reasoning and the ability to learn dispatching knowledge dramatically improves problem solving ability. WebAdapt was able to satisfy 100% of the given adaptation requests using introspective reasoning even when it did not acquire dispatching knowledge. Introspection and learning lead to a 76% decrease over the exhaustive search method, which always iterated over all possible sources until a suitable one was discovered.

WebAdapt's original profile-based dispatching model [20] favored sources that quickly developed competent source profiles. The introduction of source learning increases the system's overhead by a relatively small amount. When both source profile knowledge and dispatch cases were used, the system averaged less than half a failure per adaptation. When WebAdapt relied on a purely case-based approach for source selection, less than half a failure per problem occurred on average. Unfortunately the system did not attempt to mine every relevant source, ignoring potentially interesting substitutions from other sources. The addition of the Q-learning component provided an exploration mechanism, at the cost of

increasing reasoning failures. However, results show that a combination of case-based dispatching and exploration-based dispatching lead to only a small increase in reasoning failures. We expect Q-learning to be worthwhile when multiple sources have unique information about a domain.

The experiment for question 3 supports the generality of WebAdapt's approach. The failure rates were similar in the menu and travel planning domains, and slightly higher initially for the software domain. This higher rate can be attributed to the software domain being covered by only one knowledge source, so there is no benefit to exploration once the best source is identified. Overall, given no a priori knowledge of the applicability of any source, WebAdapt learned quickly to exploit the sources, regardless of the domain.

7 Related Work

Our knowledge planning approach is in the spirit of Hunter's early work on planful knowledge acquisition, which applied this to two external knowledge sources in the medical domain [18]. However, Hunter's work did not learn source coverage or how best to apply each source.

Cordier et al.'s IAKA [6] also acquires case adaptation knowledge on-line rather than at design time. The knowledge acquisition process is triggered when IAKA encounters a failure. IAKA makes use of an "oracle" to identify the reasoning failure and correct the knowledge that led to the failure. In WebAdapt, the knowledge acquisition process begins when the user makes a request for an adaptation. Failures in WebAdapt's reasoning process are diagnosed and fixed by the system itself through introspective reasoning. Several methods exist for acquiring adaptation knowledge in advance from sources with known content and structure [29,24,15,9,11]. However, such an a priori approach is not feasible for large-scale Web sources, due to their comprehensive coverage. Consequently, WebAdapt acquires this knowledge by interacting with the sources at its disposal.

Work by Kushmerick et al. [17,5] studies inductive learning of semantic descriptions of small specialized knowledge sources, given descriptions of similar sources. WebAdapt does not learn semantic descriptions of sources but attempts to estimate their coverage.

8 Conclusion

We have presented two extensions to the WebAdapt adaptation framework to increase its generality. An introspective reasoning component has been added to the WebAdapt framework to identify and repair failures in WebAdapt's adaptation process. In addition, a Q-learning component was added to WebAdapt's problem dispatching model. When used in conjunction with the introspective component, the exploration component facilitates the acquisition of knowledge about which Web-based resources are best suited to search for information.

Experimental results show that WebAdapt can automatically correct failures in its adaptation process. The results also show that WebAdapt is able to acquire

knowledge of the coverage of the sources at its disposal when this knowledge is not available *a priori*. Acquiring this knowledge only leads to a small increase in reasoning failures over an approach that only uses a single reliable source, and provides the benefit of being able to learn to exploit the unique contributions of multiple sources. In addition, tests in multiple domains are encouraging for the generality of the approach.

References

1. U.S. Department of Agriculture, A.R.S.: USDA national nutrient database for standard reference, release 21 (2008), nutrient Data Laboratory Home Page, http://www.ars.usda.gov/ba/bhnrc/ndl
2. Apple: Mac os x dashboard widgets (2008), http://www.apple.com/downloads/dashboard/ (accessed October 1, 2008)
3. 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)
4. Barletta, R.: Building real-world CBR applications: A tutorial. In: Haton, J.-P., Manago, M., Keane, M.A. (eds.) EWCBR 1994. LNCS, vol. 984. Springer, Heidelberg (1995)
5. Carman, M.J., Knoblock, C.A.: Learning semantic descriptions of web information sources. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 2695–2700. Morgan Kaufmann, San Mateo (2007)
6. Cordier, A., Fuchs, B., de Carvalho, L.L., Lieber, J., Mille, A.: Opportunistic acquisition of adaptation knowledge and cases - the IAKA approach. In: Althoff, K., Bergmann, R., Minor, M., Hanft, A. (eds.) Advances in Case Based Reasoning: 9th European Conference. Springer, Berlin (2008)
7. Cox, M., Raja, A.: Metareasoning: A manifesto. Technical Memo 2028, BBN (2008)
8. Cox, M., Ram, A.: Introspective multistrategy learning: On the construction of learning strategies. Artificial Intelligence 112(1-2), 1–55 (1999)
9. Craw, S., Jarmulak, J., Rowe, R.: Learning and applying case-based adaptation knowledge. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 131–145. Springer, Heidelberg (2001)
10. Cycorp: OpenCyc (2007), http://www.opencyc.org/ (accessed February 17, 2007)
11. 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 2007), pp. 750–755. Morgan Kaufmann, San Mateo (2007)
12. Epicurious: Epicurious.com: Recipes, menus, cooking articles and food guides (2008), http://www.epicurious.com/ (accessed July 15, 2008)
13. Frommer's: Frommer's Paris 2006. Frommer's (2006)
14. Geonames: Geonames (2007), http://www.geonames.org (accessed February 17, 2007)
15. Hanney, K., Keane, M.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS (LNAI), vol. 1266, Springer, Heidelberg (1997)
16. Hendler, J.: Knowledge is power: A view from the semantic web. AI Magazine 26(4), 76–84 (2005)

17. Heß, A., Kushmerick, N.: Learning to attach semantic metadata to web services. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 258–273. Springer, Heidelberg (2003)
18. Hunter, L.: Knowledge acquisition planning for inference from large datasets. In: Shriver, B. (ed.) Proceedings of the Twenty Third Annual Hawaii International Conference on System Sciences, Kona, HI, pp. 35–45 (1990)
19. Kolodner, J.: Improving human decision making through case-based decision aiding. AI Magazine 12(2), 52–68 (Summer 1991)
20. 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, Springer, Berlin (2008)
21. Leake, D., Powell, J.: On retaining web search cases (2010) (submitted)
22. Leake, D., Scherle, R.: Towards context-based search engine selection. In: Proceedings of the 2001 International Conference on Intelligent User Interfaces, pp. 109–112 (2001)
23. 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)
24. Patterson, D., Anand, S., Dubitzky, W., Hughes, J.: Towards automated case knowledge discovery in the M^2 case-based reasoning system. Knowledge and Information Systems: An International Journal, 61–82 (1999)
25. Penberthy, J., Weld, D.: UCPOP: A sound, complete, partial order planner for ADL. In: Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning, pp. 103–114. Morgan Kaufmann, San Francisco (1992)
26. Smyth, B., McClave, P.: Similarity vs. diversity. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 347–361. Springer, Heidelberg (2001)
27. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)
28. Wikimedia Foundation: Wikipedia (2007), http://www.wikipedia.org (accessed February 17, 2007)
29. Wilke, W., Vollrath, I., Althoff, K.D., Bergmann, R.: A framework for learning adaptation knowledge based on knowledge light approaches. In: Proceedings of the Fifth German Workshop on Case-Based Reasoning, pp. 235–242 (1997)

Detecting Change via Competence Model

Ning Lu, Guangquan Zhang, and Jie Lu

Decision Systems & e-Service Intelligence (DeSI) Lab
Centre for Quantum Computation & Intelligent Systems (QCIS)
Faculty of Engineering and Information Technology, University of Technology, Sydney
P.O. Box 123, Broadway, NSW 2007, Australia
{philiplu,jielu,zhangg}@it.uts.edu.au

Abstract. In real world applications, interested concepts are more likely to change rather than remain stable, which is known as *concept drift*. This situation causes problems on predictions for many learning algorithms including case-base reasoning (CBR). When learning under concept drift, a critical issue is to identify and determine "when" and "how" the concept changes. In this paper, we developed a *competence-based empirical distance* between case chunks and then proposed a change detection method based on it. As a main contribution of our work, the change detection method provides an approach to measure the distribution change of cases of an infinite domain through finite samples and requires no prior knowledge about the case distribution, which makes it more practical in real world applications. Also, different from many other change detection methods, we not only detect the change of concepts but also quantify and describe this change.

Keywords: Case-based Reasoning, Competence Model, Concept Drift.

1 Introduction

In recent years, with the rapid development of information, modern organizations are accumulating data at unprecedented rates. Examples of such data streams include customer purchase logs, telephone calling records, credit card transactional flows. While these data may contain valuable knowledge, the distribution or pattern underlying the data is more likely to change over time rather than remain stable, which is also known as concept drift [1, 2]. As a result, when a certain learning algorithm considers all the past training data or makes assumption that the training data is a random sample drawn from a stationary distribution, the induced pattern may not relevant to the new data. In practical terms, this means an increasing error in classifying new records with existing models [3, 4].

Generally there are three approaches for handling concept drift: 1) instance selection (window-based); 2) instance weighting (weight-based); 3) ensemble learning [5, 6]. In instance selection, the key idea is to select the most relevant instances to the current concept. The typical technique of this category is to pick up the training dataset within a fixed or dynamic window that moves over recently arrived instances to construct a model [2, 3, 7]. Many case-base editing strategies in case-based reasoning (CBR) that delete noisy, irrelevant and redundant cases are also a form of instance selection [8]. In

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 201–212, 2010.

Instance weighting, each instance is assigned a weight to represent the decreasing relevance of existing training examples. And learning algorithms are adopted to process these weighted instances, such as Support Vector Machines (SVMs) [9]. Instances can be weighted according to their age, and their competence with regard to the current concept [5]. Ensemble learning deals with concept drift by utilizing multiple models and by voting or selecting the most relevant one to construct a proper predictive model [10-12]. Generally, there are two ensemble frameworks: 1) horizontal ensemble, which builds on a set of buffered data chunks; 2) vertical ensemble, which builds on the most recent data chunk only. More recently, an aggregate ensemble framework, which could been seen as a hybrid approach of the two, has been proposed [13].

All these proposed methods reported great improvement for learning under concept drift. However, most of current solutions implicitly assume that concept drift is ubiquitous and global. This causes problem when change in the concept or data distribution occur in some regions of instance space only, which is known as local concept drift [14]. So instead of directly assigning a weight to each classifier or chunk of training set, Tsymbal, Pechenizkiy, Cunningham and Puuronen [14] gave a weighted strategy from instance level, which estimated the local performance of each base classifier for each instance of the coming instance set. However, their method is not able to determine whether there is a concept drift happened. On one hand, when concept remains, clearly old training examples can help to achieve a more robust model. But on the other hand, when concept drift occurs, old training data do not always help produce a more accurate hypothesis than using the most recent data only [15]. As a result, further information about when and where the change has occurred is needed, so that a learner can distinguish whether there is a concept drift and make better use of existing training data. Addressing to this issue, we propose a new change detection method for CBR system, which compares the distribution of existing case base and newly available cases. Our method not only decides whether concept drift occurs, but also provides a meaningful explanation about where and how the underlying distribution change is.

The remaining of this paper is organized as follows. Section 2 reviews the related works concerning change detection for data stream and a competence model for CBR. In Section 3, a competence-based empirical distance between case chunks is introduced with a simple example. Then we present our change detection method in more details. The results of experimental evaluation are shown in Section 4. Finally, conclusions and future works come in Section 5.

2 Related Work

In this section, we first introduce a change detection method for data streams. Following that, a competence model for CBR systems will be discussed.

2.1 A Change Detection Method

A natural approach of detecting concept drift is to compare the distribution of the data. However, in real world applications, the data that one typically encounters may not arise from any standard distribution, which makes non-parametric tests more practical. Moreover, the data may contain several dimensions. As a result, traditional

non-parametric tests like the Wilcoxon and Kolmogorov-Smirnov cannot be easily adopted. Kifer, Ben-David and Gehrke [16] proposed a change detection method by employing a notation of distance which could be seen a generalization of Kolmogorov-Smirnov statistic (Def. 1). Two probability distributions are considered as ε-close if their distance is no greater than ε.

Definition 1. [16] Fix a measure space and let $\mathcal{A}$ be a collection of measurable sets. Let P and P′ be probability distributions over this space.

- The *$\mathcal{A}$-distance* between P and P′ is defined as

$$d_{\mathcal{A}}(P, P') = 2 \sup_{A \in \mathcal{A}} |P(A) - P'(A)| \tag{1}$$

- For a finite domain subset S and a set $A \in \mathcal{A}$, let the *empirical weight* of A with regard to (*w.r.t.*) S be

$$S(A) = \frac{|S \cap A|}{|S|} \tag{2}$$

- For finite domain subsets, S_1 and S_2, the *empirical distance* is defined as

$$d_{\mathcal{A}}(S_1, S_2) = 2 \sup_{A \in \mathcal{A}} |S_1(A) - S_2(A)| \tag{3}$$

They also provided a variation of notion of the $\mathcal{A}$-distance, called *relativized discrepancy*, which takes the relative magnitude of a change into account. But for this work, we only show how our method works with the *$\mathcal{A}$-distance* in a CBR system and leave the discussion of *relativized discrepancy* for future work. Interested readers please refer to the original work [16] for the details.

Although there exist many other change detection methods [17-20], there is a reported advantage for us to choose Kifer, Ben-David and Gehrke's [16] method. That is being able to quantify and describe the change it detects, which makes it more appropriate for handling local concept drift.

2.2 A Competence Model

Competence is a measurement of how well a CBR system fulfils it goals. As CBR is a problem-solving methodology, competence is usually taken to be the proportion of problems faced that it can solve successfully [21]. According to Smyth and Kenna [22], the local competence of an individual case is characterized by its coverage and reachability. The coverage of a case is the set of target problems that it can be used to solve. The reachability of a target problem is the set of cases that can be used to provide a solution for the target. Since it is impossible to enumerate all possible future target problems, in practice Smyth and Kenna [22] estimated the *coverage set* of a case by the set of cases that can be solved by its retrieval and adaption. And the *reachability set* of a case is estimated by the set of cases that can bring about its solution. Smyth and McKenna [23] extended this competence model. They defined the *related set* of a case as the union of its *coverage set* and *reachability set*, and said the *shared coverage* of two cases exists if and only if the intersection of the *related sets* of two different cases is not empty. Definition 2 gives a overall view of this competence model based on a survey provided by Smyth and McKenna [24].

Definition 2. [24] For a case base $\mathbb{C} = \{c_1, c_2, \cdots, c_n\}$, given a case $c \in \mathbb{C}$

$$\text{CoverageSet}(c) = \{c' \in \mathbb{C}: \text{Solves}(c, c')\} \tag{4}$$

$$\text{ReachibilitySet}(c) = \{c' \in \mathbb{C}: \text{Solves}(c', c)\} \tag{5}$$

$$\text{RelatedSet}(c) = \text{CoverageSet}(c) \cup \text{ReachabilitySet}(c) \tag{6}$$

Further, based on Smyth and McKenna's competence model [24], we defined a *competence closure* as the maximal set of cases linked together though their *related set* in our previous research (Def. 3).

Definition 3. [25] For $G = \{c_1 \cdots c_m\} \subseteq \mathbb{C}$,

$$\begin{gathered} \text{CompetenceClosure}(G), \text{iff } \forall c_i, c_j \in G, \text{if } c_i \neq c_j, \exists \{c_{i_1}, c_{i_2}, \cdots, c_{i_k}\} \subseteq G, \\ \text{st. SharedCoverage}\left(c_{i_p}, c_{i_{p+1}}\right) \neq \emptyset \ (p = 0, \cdots, k) \\ \text{where } c_i = c_{i_0}, c_j = c_{i_{k+1}} \\ \text{and } \forall c_k \in \mathbb{C} - G, \nexists c_l \in G, \text{st. SharedCoverage}(c_k, c_l) \neq \emptyset \end{gathered} \tag{7}$$

3 Competence-Based Change Detection Method

When mining concept drifting data, a common assumption is that the up-to-date data chunk and the yet-to-come data chunk share identical or considerable close distributions [26]. In CBR, this means the newly available cases represent the concept that we may interested in the future. Obviously, cases in existing case base and the newly available cases could be considered as two samples drawn from two probability distributions. Thus by detecting possible distribution change between existing case base and newly available case chunk, we are able to identify whether there is a concept drift. However, there are two difficulties that prevent us from applying Kifer, Ben-David & Gehrke's detecting algorithm [16] directly. First, we have no prior knowledge about the probability distributions of either the existing case base or the new case chunk. Second, the cases may come from an infinite domain. As a result, we cannot estimate the distance through the cases directly.

As the competence measures the problem solving capabilities of a CBR system, the probability distribution change of its cases should also reflects upon its competence. This inspired our research of detecting change via competence model. The key idea is to measure the distribution change of cases with regarding to their competence instead of their real distribution. This section will illustrate how to detect change via competence model for CBR systems.

3.1 Competence-Based Empirical Distance

Similar as Smyth and McKenna's work [23], we refer the *related set* of a case to represent a local area of target problems. A visible benefit of adopting their competence

model is that it transfers the infinite case domain into a finite domain of *related sets*. This solves our difficulties of measuring the statistic distance between two case samples.

Definition 4. Given a case $c \in \mathbb{C}$, denote the *related set* of c with regard to $\mathbb{C}$ as $R^{\mathbb{C}}(c)$

- We define the *related closure* of c *w.r.t.* $\mathbb{C}$ as

$$\mathcal{R}^{\mathbb{C}}(c) = \{R^{\mathbb{C}}(c_i): \forall c_i \in \mathbb{C}, \exists R^{\mathbb{C}}(c_i) \text{ st. } c \in R^{\mathbb{C}}(c_i)\} \tag{8}$$

- For a case sample set $\mathbb{S} \subseteq \mathbb{C}$, we define the *related closure* of $\mathbb{S}$ *w.r.t.* $\mathbb{C}$ as

$$\mathcal{R}^{\mathbb{C}}(\mathbb{S}) = \bigcup_{c \in \mathbb{S}} \mathcal{R}^{\mathbb{C}}(c) \tag{9}$$

To be more clear, $\mathcal{R}^{\mathbb{C}}(c)$ is the set of all *related sets*, with regard to $\mathbb{C}$, which contain the case c. Since the *related set* measures the local competence of a case, the intuitive meaning of the *related closure* is the maximum set of local competence that a case or a group of cases could stand for.

Theorem 1. For a case base of finite size $\mathbb{C}$, and a case sample set $\mathbb{S} \subseteq \mathbb{C}$, $\mathcal{R}^{\mathbb{C}}(\mathbb{S})$ is a finite set and we have:

$$|\mathcal{R}^{\mathbb{C}}(\mathbb{S})| \leq |\mathcal{R}^{\mathbb{C}}(\mathbb{C})| \leq |\mathbb{C}| \tag{10}$$

Since each case in $\mathbb{C}$ corresponding to a *related set*, the proof of Theorem 1 is obvious. Therefore, over a case base of finite size $\mathbb{C}$, for two case samples of $\mathbb{S}_1, \mathbb{S}_2 \subseteq \mathbb{C}$, we obtain two finite *related closures*, $\mathcal{R}^{\mathbb{C}}(\mathbb{S}_1)$ and $\mathcal{R}^{\mathbb{C}}(\mathbb{S}_2)$. Intuitively we could measure distance between $\mathbb{S}_1$ and $\mathbb{S}_2$ as the *empirical distance* between $\mathcal{R}^{\mathbb{C}}(\mathbb{S}_1)$ and $\mathcal{R}^{\mathbb{C}}(\mathbb{S}_2)$. However, it will only represent the distance between the competences covered by these two samples. The relative distribution discrepancy within the competence is missing. This introduces problem when we are comparing two samples of similar *related closures*, but with dramatic different distribution. To address this problem, we assign a weight for each element in $\mathcal{R}^{\mathbb{C}}(\mathbb{S}_1)$ and $\mathcal{R}^{\mathbb{C}}(\mathbb{S}_2)$ to represent the relative density of the cases distributed over their *related closures*.

Definition 5. Denote the i^{th} element in $\mathcal{R}^{\mathbb{C}}(\mathbb{S})$ as $r_i^{\mathbb{C}}(\mathbb{S})$, let $\mathcal{R}_i^{\mathbb{C}}(\mathbb{S}) = \{r_i^{\mathbb{C}}(\mathbb{S})\}$, we defined the *density* of $r_i^{\mathbb{C}}(\mathbb{S})$ *w.r.t* $\mathbb{S}$ be

$$w^*\left(r_i^{\mathbb{C}}(\mathbb{S})\right) = \frac{1}{|\mathbb{S}|} * \sum_{\substack{j=1 \\ c_j \in \mathbb{S}}}^{n=|\mathbb{S}|} \frac{\left|\mathcal{R}_i^{\mathbb{C}}(\mathbb{S}) \cap \mathcal{R}^{\mathbb{C}}(c_j)\right|}{\left|\mathcal{R}^{\mathbb{C}}(c_j)\right|} \tag{11}$$

The *density* weights each *related set* in a *related closure* by the degree to which the sample cases distributed.

Theorem 2. For a case base of finite size $\mathbb{C}$, and a case sample set $\mathbb{S} \subseteq \mathbb{C}$, the sum of the *densities* of all elements in $\mathcal{R}^{\mathbb{C}}(\mathbb{S})$ equals to 1.

$$\sum_{i=1}^{n=|\mathcal{R}^{\mathbb{C}}(\mathbb{S})|} w^*\left(r_i^{\mathbb{C}}(\mathbb{S})\right) = 1 \tag{12}$$

Proof. Substitute Equation 11 into Equation 12, we have the left side as

$$\frac{1}{|\mathbb{S}|} * \sum_{i=1}^{|\mathcal{R}^{\mathbb{C}}(\mathbb{S})|} \sum_{\substack{j=1 \\ c_j \in \mathbb{S}}}^{|\mathbb{S}|} \frac{\left|\mathcal{R}_i^{\mathbb{C}}(\mathbb{S}) \cap \mathcal{R}^{\mathbb{C}}(c_j)\right|}{\left|\mathcal{R}^{\mathbb{C}}(c_j)\right|} = \frac{1}{|\mathbb{S}|} * \sum_{\substack{j=1 \\ c_j \in \mathbb{S}}}^{|\mathbb{S}|} \sum_{i=1}^{|\mathcal{R}^{\mathbb{C}}(\mathbb{S})|} \frac{\left|\mathcal{R}_i^{\mathbb{C}}(\mathbb{S}) \cap \mathcal{R}^{\mathbb{C}}(c_j)\right|}{\left|\mathcal{R}^{\mathbb{C}}(c_j)\right|} \tag{13}$$

According to definition of *related closure* (Def. 4), $\mathcal{R}^{\mathbb{C}}(c_j) \subseteq \mathcal{R}^{\mathbb{C}}(\mathbb{S})$, therefore we have:

$$\sum_{i=1}^{|\mathcal{R}^{\mathbb{C}}(\mathbb{S})|} \frac{\left|\mathcal{R}_i^{\mathbb{C}}(\mathbb{S}) \cap \mathcal{R}^{\mathbb{C}}(c_j)\right|}{\left|\mathcal{R}^{\mathbb{C}}(c_j)\right|} = 1 \tag{14}$$

From practical point of view, this means, for all cases in sample $\mathbb{S}$, they are equally important with regarding to the contribution of the total *density* of elements in $\mathcal{R}^{\mathbb{C}}(\mathbb{S})$, no matter what their *related sets* are.

Finally, we define the distance with regard to the competence of two case samples.

Definition 6. Given a case base of finite size $\mathbb{C}$, and a case sample set $\mathbb{S} \subseteq \mathbb{C}$, let $\mathcal{R}^{\mathbb{C}}(\mathbb{C})$ be the measure space and a set $A \subseteq \mathcal{R}^{\mathbb{C}}(\mathbb{C})$

- We define the competence-based empirical weight of $\mathbb{S}$ w.r.t. A over $\mathbb{C}$ as

$$S^{\mathbb{C}}(A) = \sum_{\substack{i=1 \\ r_i^{\mathbb{C}}(\mathbb{S}) \in A \cap \mathcal{R}^{\mathbb{C}}(\mathbb{S})}}^{i=\left|A \cap \mathcal{R}^{\mathbb{C}}(\mathbb{S})\right|} w^*\left(r_i^{\mathbb{C}}(\mathbb{S})\right) \tag{15}$$

- For two case sample sets $\mathbb{S}_1, \mathbb{S}_2 \subseteq \mathbb{C}$, we define the *competence-based empirical distance* as

$$d^{\mathbb{C}}(\mathbb{S}_1, \mathbb{S}_2) = 2 \sup_{A \subseteq \mathcal{R}^{\mathbb{C}}(\mathbb{C})} \left|S_1^{\mathbb{C}}(A) - S_2^{\mathbb{C}}(A)\right| \tag{16}$$

For all sets A which cause the *competence-based empirical distance* to be greater than ε, we say that there is a concept drift. Similar to Kifer, Ben-David & Gehrke's work [16], the set A also depicts a local area where the largest distribution discrepancy between two samples lies.

Now, we consider a simple example to illustrate how to measure the *competence-based empirical distance* between two case sample sets. Suppose there is a case base $\mathbb{C} = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ aims to determine whether a cup of milk turns bad after some hours taken out of a fridge. Assume we have c_1 represents the milk is still good after 4 hours, and c_2 (7hs, good), c_3 (12hs, bad), c_4 (16hs, bad), c_5 (19hs, bad), c_6 (21hs, bad). Also we assume two cases can be used to retrieve each other if they were both taken out within 5 hours. Constructing the competence model over $\mathbb{C}$, we have the *related set* for each case in $\mathbb{C}$ is as follows: $R^{\mathbb{C}}(c_1) = R^{\mathbb{C}}(c_2) = \{c_1, c_2\}$, $R^{\mathbb{C}}(c_3) = \{c_3, c_4\}$, $R^{\mathbb{C}}(c_4) = \{c_3, c_4, c_5, c_6\}$, $R^{\mathbb{C}}(c_5) = R^{\mathbb{C}}(c_6) = \{c_4, c_5, c_6\}$. Our goal is to measure the *competence-based empirical distance* between two sample sets drawn from the case base $S_1 = \{c_1, c_4, c_5\}$ and $S_2 = \{c_2, c_3, c_6\}$. The *related closure* of c_3

with regard to $\mathbb{C}$ is the set of all *related sets* which contain c_3. Thus we have $\mathcal{R}^{\mathbb{C}}(c_3) = \{\{c_3, c_4\}, \{c_3, c_4, c_5, c_6\}\}$. Also we have the *related closure* of S_1 and S_2 as $\mathcal{R}^{\mathbb{C}}(S_1) = \mathcal{R}^{\mathbb{C}}(S_2) = \{\{c_1, c_2\}, \{c_3, c_4\}, \{c_3, c_4, c_5, c_6\}, \{c_4, c_5, c_6\}\}$. The *density* of $\{c_3, c_4\}$ with regard to S_1 is calculated as 1/3*(0/1+1/3+0/2) = 1/9. And the *density* of $\{c_1, c_2\}$, $\{c_3, c_4, c_5, c_6\}$, $\{c_4, c_5, c_6\}$ with regard to S_1 is 1/3, 5/18, 5/18 respectively. Accordingly, the *density* of $\{c_1, c_2\}$, $\{c_3, c_4\}$, $\{c_3, c_4, c_5, c_6\}$, $\{c_4, c_5, c_6\}$ with regard to S_2 is 1/3, 1/6, 1/3, 1/6. In this case, when $A = \{\{c_3, c_4\}, \{c_3, c_4, c_5, c_6\}\}$ or $\{c_4, c_5, c_6\}$, we get 1/9 as the *competence-based empirical distance* between S_1 and S_2.

3.2 A Detection Algorithm

In this section we discuss how to determine whether a concept drift occurs through competence-based empirical distance. Assume we are running a CBR system, and a new case chunk becomes available. Before retaining these new cases, we could like to detect whether there is a concept drift. We measure the *competence-based empirical distance* between current case base and the new case chunk and say a concept drift exists if the distance is large enough ($> \varepsilon$).

If we deem all cases of current case base follow a certain distribution, we may say that there is no significant distribution change between two case samples drawn randomly from the case base. Therefore, the distance between these samples provides a reference for determining ε.

To minimize the error inferred due to sample bias, we incorporate two mechanisms. First, we do multiple experiments and choose ε as the maximum distance rather than rely on a single test. Actually, the number of experiments plays the role of tradeoff between miss detection and false detection error. Second, we split the whole case base into several *competence closures* and draw samples within each *competence closure* respectively. A major concern for us to use *competence closure* but not other methods, such as dividing the case base according to feature values, is that a *competence closure* represents a group of local competence measurements that related to each other. Sampling based on *competence closures* consists with the sense of our competence-based change detection method, also facilitates the work of doing experiments within a certain local competence area when desired. In additional we determine the sample size according to the size of each *competence closure*. That is to draw a larger sample for larger competence closures, and vice versa. By doing this, we expect the case sample set represents the distribution of the original case base to the greatest extend.

The overall process of our change detection method is shown in Figure 1. When new cases become available, we merge these new cases with existing case base. We use this merged case base to represent the whole case domain, and construct competence model on it. Then we draw samples randomly from the original case base and measure the *competence-based empirical distance* between samples. The maximum distance is selected as the bound (ε) for determining whether a concept drift occurs, after a certain number of experiments (n).

Take our previous example again, but this time we change the environment by a little, thus the milk can last longer. And we get some new cases c_7 (8hs, good), c_8 (15hs good) and c_9 (17hs bad). Merging these new cases into the case base $\mathbb{C}$, we

reconstruct the competence model over the merged case base $\mathbb{C}'$. We have $R^{\mathbb{C}'}(c_1) = R^{\mathbb{C}'}(c_2) = R^{\mathbb{C}'}(c_7) = \{c_1, c_2, c_7\}$, $R^{\mathbb{C}'}(c_3) = \{c_3, c_4, c_9\}$, $R^{\mathbb{C}'}(c_4) = R^{\mathbb{C}'}(c_9) = \{c_3, c_4, c_5, c_6, c_9\}$, $R^{\mathbb{C}'}(c_5) = R^{\mathbb{C}'}(c_6) = \{c_4, c_5, c_6, c_9\}$, $R^{\mathbb{C}'}(c_8) = \{c_8\}$. We measure the *competence-based empirical distance* between the new case set $S_{new} = \{c_7, c_8, c_9\}$ and a sample set S_1 drawn from the original case base $\mathbb{C}$. By comparing this distance with distance between two sample sets, S_1 and S_2 for example, drawn from the original case base $\mathbb{C}$, we find that concept drift happens, with a decreasing trend for bad cases, and an increasing trend for good cases especially around 15 hours (c_8).

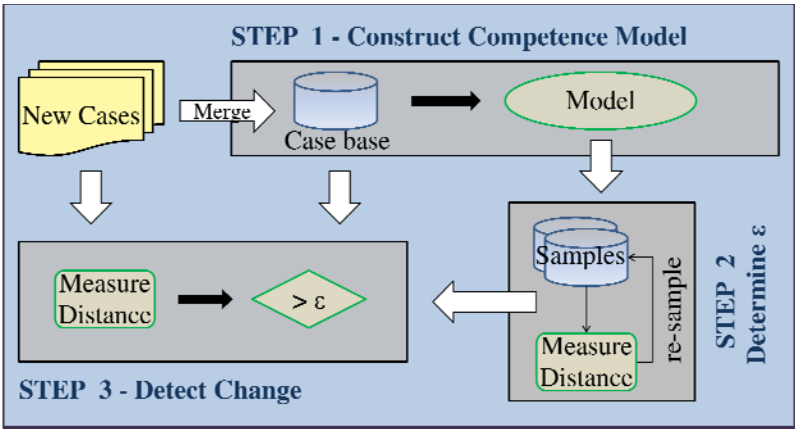


Fig. 1. Competence-based change detection flow chart

4 Experimental Evaluation

In order to evaluate the proposed competence-based change detection method, it is necessary to use simulated data so that the change in generating distributions is known. We use four synthetic datasets based on data used in another paper concerning concept drift [18]. All the datasets have two classes.

- STAGGER (1S). sudden, noise free. The dataset has three nominal attributes: size (small, medium, large), color (red, green) and shape (circle, non-circle), and has three concepts: 1) [size = small and color = red], 2) [color = green or shape = circle] and 3) [size = medium or large]. Data was randomly generated within the domain and labeled according to current concept.
- MIXED (2M). sudden, noise. The dataset is a mixture of data generated according to two different but overlapped geometric distributions, $f_p(x) = \begin{cases} p(1-p)^{x-1} & 1 \le x < 20 \\ (1-p)^{x-1} & x = 20 \end{cases}$ and $f_p'(x) = f_p(21 - x)$. There are two concepts for this dataset. In both concept the positives are generated by $f_{0.25}(x)$, while the negatives are changed from $f_{0.25}'(x)$ to $f_{0.33}'(x)$. In addition the proportion of the negatives is changed from 1/3 to 3/7. In both concept, we consider a sample as positive if $x \le 11$. The overlapping can be considered as noise. Although the

condition of classifying the samples remains the same, the distribution of the data changes, thus concept drift occurs. Figure 2 shows the data distributions before and after concept drift.

- CIRCLES (3C). sudden, noise free. The examples are label according to the condition: if an example is inside the circle, then its label is positive. The change is achieved by displacing the centre of the circle $\big((0.2,0.5) \rightarrow (0.4,0.5)\big)$ and growing its radius $(0.15 \rightarrow 0.2)$. We assume the problem space is $([0,1],[0,1])$, and two cases are considered as similar if their Euclidean distance is not greater than 0.1. Being different from Nishida and Yamauchi [18], we still consider this dataset as sudden concept drift, and create another dataset CIRCLES-2 based on Stanley's definition on gradual concept drift [27] to compare the results.
- CIRCLES-2 (4C). gradual, free noise. The concept is the same as CIRCLES, but gradually changed over Δx samples. We assume the probability of the coming instance being in the first concept declines linearly as the probability of an instance being in the second concept increases until the first concept is completely replaced by the second concept. That means for the i^{th} new instance c_i, it still has the probability of $\frac{\Delta x - i}{\Delta x}$ to follow the first concept, when $i \geq \Delta x$ the second concept will completely replace the first one. We continuously detect whether there is a concept drift each time a certain number of instances (samples) become available. We assume the size of the case base is ten times the size of the samples. And all previous samples containing both concepts are retained and used for future detection.

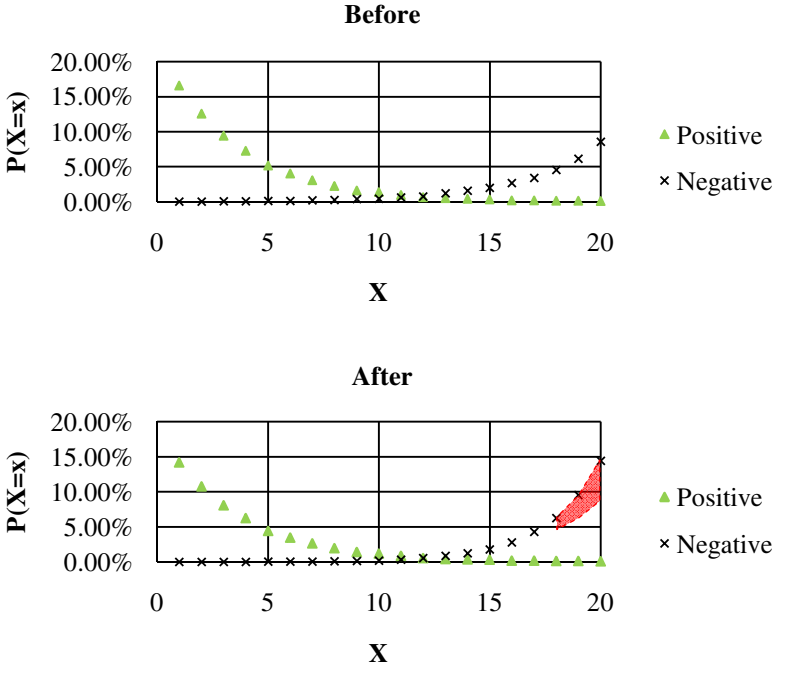


Fig. 2. Distributions of the mixed dataset (2M) before and after concept drift

The experiment results are shown in Table 1. We varied the sample size (N) for each concept and number of experiments (n) used to determine the upper bound (ε) to see how our detection method is affected. We evaluate all results by two types of error rate, false positive (miss detection) and true negative (false detection). All error rates were calculated based on 5K tests.

It can be seen that for all these dataset, our method can detect change with very low error rates. However, one thing to note is that the false positive error rate increases dramatically with relative a small sample size. This is probably due to the sample bias, considering that this size of samples may not fully represent the distribution of the data. When the sample size is large enough, the error rate remains stable. Second, the number of experiments (n) balances between the two types of error. An increasing of n will lead to an increasing of miss detection error but lower the false detection error and vice versa. Third, by comparing the results of the last two datasets, we found that our method performs better for sudden concept drift, even during the last stage when the first concept has been completely replaced (20.05% vs. 38.43% and 2.1% vs. 7.67%). This is due to the retention of previous cases, which could actually been considered as adding some noise. Finally, being a specialty of our change detection method, it is able to quantify and describe the change detected. For example, we detect a dramatic increase of negative samples when $17 \leq x \leq 20$ (Fig. 2).

Table 1. Experiment Results of Competence-based Change Detection

Data Set	Δx	N	n	FP	TN
1S		500	100	0.00%	0.73%
		500	20	0.00%	3.52%
		100	100	0.34%	0.70%
		100	20	0.10%	3.55%
2M		500	100	0.50%	0.80%
		500	20	0.02%	4.36%
		100	100	56.59%	0.97%
		100	20	38.23%	4.50%
3C		100	10	20.05%	10.43%
		200	10	2.1%	9.51%
4C	300	001-100	10	77.32%	10.74%
		101-200	10	62.39%	9.31%
		201-300	10	42.58%	9.13%
		301-400	10	38.43%	9.96%
	600	001-200	10	78.07%	9.33%
		201-400	10	45.65%	8.54%
		401-600	10	15.34%	9.41%
		601-800	10	7.67%	10.99%

5 Conclusions and Future Works

We present a method for detecting change in the distribution of samples. The method requires no prior knowledge about distribution but measures it through a competence model. The competence-based change detection method can be applied to CBR

systems where new case chunks are available sequentially over time. Empirical experiments show that the competence-based change detection method works well for large samples and is not too sensitive to noises. Being special, the competence-based change detection method is able to quantify and describe the change it detects, which makes it more suitable for handling local concept drift.

For future works, the proposed approach for detecting concept drift requires a sample data which is large enough to represent the true data distribution. How to track concept drift with very little data is still a remaining issue. Second, how to construct the competence model and detect change for non-classification problems will be another issue. Finally, detecting change is only the first step towards handling concept drift. Successive methods that take advantage of change detection are needed to improve the final learning performance.

Acknowledgment. The work presented in this paper was supported by Australian Research Council (ARC) under Discovery Project DP0880739. We also wish to thank the anonymous reviewers for their helpful comments.

References

1. Widmer, G., Kubat, M.: Effective learning in dynamic environments by explicit context tracking. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 227–243. Springer, Heidelberg (1993)
2. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23(1), 69–101 (1996)
3. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97–106. ACM Press, San Francisco (2001)
4. Cohen, L., Avrahami, G., Last, M., Kandel, A.: Info-fuzzy algorithms for mining dynamic data streams. Applied Soft Computing 8(4), 1283–1294 (2008)
5. Tsymbal, A.: The Problem of Concept Drift: Definitions and Related Work. Technical Report TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Ireland (2004)
6. Tsai, C.-J., Lee, C.-I., Yang, W.-P.: Mining decision rules on data streams in the presence of concept drifts. Expert Syst. Appl. 36(2), 1164–1178 (2009)
7. Maloof, M.A., Michalski, R.S.: Incremental learning with partial instance memory. Artificial Intelligence 154(1-2), 95–126 (2004)
8. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems 18(4-5), 187–195 (2005)
9. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. Intell. Data Anal. 8(3), 281–300 (2004)
10. Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 377–382. ACM Press, San Francisco (2001)
11. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235. ACM Press, Washington (2003)

12. Kolter, J.Z., Maloof, M.A.: Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. J. Mach. Learn. Res. 8, 2755–2790 (2007)
13. Zhang, P., Zhu, X., Shi, Y., Wu, X.: An Aggregate Ensemble for Mining Concept Drifting Data Streams with Noise. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 1021–1029. Springer, Heidelberg (2009)
14. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Dynamic integration of classifiers for handling concept drift. Information Fusion 9(1), 56–68 (2008)
15. Fan, W.: Systematic data selection to mine concept-drifting data streams. In: 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 128–137. ACM Press, Seattle (2004)
16. Kifer, D., Ben-David, S., Gehrke, J.: Detecting change in data streams. In: 13th International Conference on Very Large Data Bases. VLDB Endowment, Toronto, Canada, pp. 180–191 (2004)
17. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with Drift Detection. In: 17th Brazilian Symposium on Artificial Intelligence, pp. 286–295. Springer, Sao Luis (2004)
18. Nishida, K., Yamauchi, K.: Detecting Concept Drift Using Statistical Testing. In: 10th International Conference on Discovery Science, pp. 264–269. Springer, Heidelberg (2007)
19. Song, X., Wu, M., Jermaine, C., Ranka, S.: Statistical change detection for multi-dimensional data. In: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 667–676. ACM Press, San Jose (2007)
20. Dries, A., Rückert, U.: Adaptive concept drift detection. Statistical Analysis and Data Mining 2(5-6), 311–327 (2009)
21. Massie, S., Craw, S., Wiratunga, N.: What is CBR competence? BCS-SGAI Expert Update 8(1), 7–10 (2005)
22. Smyth, B., Keane, M.T.: Remembering To Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems. In: 14th International Joint Conference on Arti-ficial Intelligence, pp. 377–382. Morgan Kaufmann, Montreal (1995)
23. Smyth, B., McKenna, E.: Footprint-Based Retrieval. In: 3rd International Conference on Case-Based Reasoning and Development, pp. 343–357. Springer, Seeon Monastery (1999)
24. Smyth, B., McKenna, E.: Competence Models and the Maintenance Problem. Computational Intelligence 17(2), 235–249 (2001)
25. Lu, N., Lu, J., Zhang, G.: Maintaining Footprint-Based Retrieval for Case Deletion. In: Leung, C.S., Lee, M., Chan, J.H. (eds.) ICONIP 2009, Part II. LNCS, vol. 5864, pp. 318–325. Springer, Heidelberg (2009)
26. Gao, J., Fan, W., Han, J.: On Appropriate Assumptions to Mine Data Streams: Analysis and Practice. In: 7th IEEE International Conference on Data Mining, pp. 143–152. IEEE Computer Society, Omaha (2007)
27. Stanley, K.O.: Learning concept drift with a committee of decision trees. Technical Report UT-AI-TR-03-302, Department of Computer Science, University of Texas at Austin, USA (2003)

CBTV: Visualising Case Bases for Similarity Measure Design and Selection

Brian Mac Namee and Sarah Jane Delany

DIT AI Group, Dublin Institute of Technology, Dublin, Ireland
brian.macnamee@dit.ie, sarahjane.delany@dit.ie

Abstract. In CBR the design and selection of similarity measures is paramount. Selection can benefit from the use of exploratory visualisation-based techniques in parallel with techniques such as cross-validation accuracy comparison. In this paper we present the Case Base Topology Viewer (CBTV) which allows the application of different similarity measures to a case base to be visualised so that system designers can explore the case base and the associated decision boundary space. We show, using a range of datasets and similarity measure types, how the idiosyncrasies of particular similarity measures can be illustrated and compared in CBTV allowing CBR system designers to make more informed choices.

1 Introduction

Similarity measures are widely used in instance-based learning and more specifically in nearest neighbour classification. The importance of using the appropriate similarity measure in a k-NN classifier is well known and there has been significant work on proposing, learning and evaluating new similarity measures [28,21,16,22,25,3].

In most cases, the selection of the best measure is based on a comparison of similarity measure performance in a cross-validation evaluation. Our proposal in this paper is to provide a tool, the Case Base Topology Viewer (CBTV), for case base visualisation. This will allow case-based system designers to explore different similarity measures through visualisations in order to assist them in choosing the most appropriate measure for the case base in question.

CBTV is based on the force-directed graph algorithm, similar to other work on case base visualisation [20,19], and projects a case base onto a two dimensional plane such that cases that are similar to one-another appear closer together. Changing from one similarity measure to another, users can examine the class separability possible with different measures, which assists in their decision as to which similarity measure is most appropriate for their current task.

The contributions of this paper are the presentation of the CBTV system and the description of a number of illustrative examples that show how the differences between similarity measures can be explored through visualisations. The paper is organised as follows: Section 2 gives an overview of existing work, both in the context of designing or choosing the best similarity measure to use in a case-based system, and in the context of visualisation of case bases. Section 3 then

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 213–227, 2010.

describes the CBTV system and how it can be used to visualise the impact of using different similarity measures. Section 4 then provides a series of evaluation examples of differing visualisations of various case bases - including numeric only datasets, textual datasets and heterogeneous datasets. Finally, in Section 5 we draw conclusions on some of the advantages and disadvantages of using a system such as CBTV and present a number of avenues for further research.

2 Related Work

A variety of similarity measures are proposed in the literature for use in case-based systems. Cunningham [5] provides a comprehensive taxonomy of strategies for similarity assessment in CBR, grouping measures into four different categories. The most well known of these is the *direct* category where cases are represented as feature vectors and similarity is assessed directly from these features. Less frequently used categories include *transformation-based* measures where similarity between cases is based on the effort required to transform one case into another case (e.g. Edit Distance and Earth Mover Distance); *information-theoretic* measures where the measure is based on the information content of the case, the most dramatic of these being Compression-Based Similarity; and, finally, *emergent measures* such as Random Forests which exploit the considerable computational power now available.

Most of the existing research into choosing similarity measures has focussed on (i) proposing or (ii) learning new *direct* measures. By and large the most commonly used *direct* similarity measure is the generalised weighted similarity measure where the similarity between two cases is a function (often linear) of individual weighted local feature similarities. The task is then to set the individual feature weights and Wettschereck *et al.* [27] introduced a five dimensional framework for categorising automated feature weight setting methods and included a comprehensive review of early work in this area.

Aside from setting feature weights, there has been work presenting a formal framework for the specification and construction of a wide range of heterogeneous similarity measures [22] and proposing new heterogenous and hybid measures [21,16]. More recently learning similarity measures has received more attention. A large proportion of research in this area uses incremental or evolutionary algorithms. These approaches can use system performance feedback [13,9], introspective and reinforcement learning [2,23] or qualitative information such as the ranking of cases [3,4]. Xiong & Funk [29] introduce a novel learning approach with the advantage that the importance of a feature is built into a local similarity measure and there is no need to learn individual feature weights.

In most, if not all, of this previous work, measures such as classification accuracy in cross validation are used as a measure of the performance of the similarity measures, showing which similarity measure is most appropriate for the case base in question. Our approach to selecting the most appropriate similarity measure is to provide a visualisation of the case base for a specific similarity measure which shows how the cases are distributed and where decision boundaries are

located. It is not intended that this approach replace the use of performance measures, but rather that it complements these techniques and allows a system designer to further explore a case base and associated decision boundary space.

Previous work in case base maintenance has used the visualisation of the case base and its competence model to improve the efficiency of the authoring process and the quality of the resulting case base [20]. Case-base visualisation has also been used to provide dynamic visualisations of case base usage over the life-cycle of a case base [19]. Both approaches use the force-directed graph drawing algorithm known as the *spring model* [10] which allows the display of n-dimensional data on a 2-dimensional plane.

A second category of visualisation focusses more on showing the relationships between cases by visualising the connections between feature values. Falkman [11] visualises clinical experience as a three dimensional cube based on the idea of 3D parallel co-ordinate plots. A case feature can be viewed as a two-dimensional plot with the cases plotted in some order on the x-axis and the possible feature values on the y-axis. The Cube is a collection of planes, each plane representing a feature in the case representation and each case is represented as a line connecting individual values in the different feature planes. Massie *et al.* [18] adopts a similar approach to visualisation for explanation in pharmaceutical tablet formulation. They use a two-dimensional version with each feature represented as a vertical line rather than a plane, with feature values plotted along this line and each case represented by connecting the feature lines at positions reflecting the case feature value. Clusters of similar cases can be observed easily with this parallel co-ordinate plot approach and it facilitates the identification of similarities and differences between cases and groups of cases.

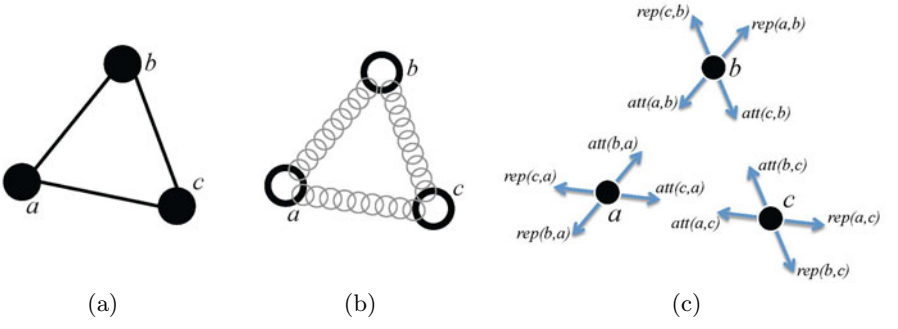
The limitation of these approaches is that they do not support the visualisation of large dimensional case bases. While some of the approaches mentioned can deal with high-dimensional data (e.g. [20] [19]) Kontkanen *et al.* directly address this problem [14]. Visualisations are created in which, rather than basing case positions on their pairwise Euclidean distances, they use the output of a Bayesian network - the intuition being that cases that result in similar network output should appear close together in the visualisation. This, they argue, gives better visualisations than the use of Euclidean distance measures. However, they fail to acknowledge the fact that Euclidean distance is not the only similarity measure available. We examine whether it is possible to create more interesting visualisations using more sophisticated similarity measures.

3 The CBTV System

This section will describe the CBTV system and how it can be used to visually explore the suitability of using various similarity measures with particular datasets. Firstly, we will describe how the system generates visualisations using a force directed graph drawing algorithm, or the spring model. Then we will describe how these visualisations can be fine tuned using transformations of the calculated similarity values. Finally, we will describe the technique we use to evaluate how well a particular visualisation matches an underlying dataset.

3.1 Representing Case Bases Using a Force-Directed Graph Drawing Algorithm

Similarly to Smyth *et al.* [24], we create visualisations of case bases using a force directed graph drawing algorithm [10]. We consider a case base as a maximally inter-connected graph in which each case is represented as a node that is linked to every other case (or node) in the case base. An example is shown in Fig. 1(a) in which cases a, b and c are shown as a maximally inter-connected graph. The layout of the graph should be such that cases most similar to each other appear close together in the graph.



(a) (b) (c)

Fig. 1. Case base representations used by the force-directed graph drawing algorithm

To allow similarities between cases dictate the arrangement of the graph the metaphor of springs is used. Each case is imagined as a steel ring with springs connecting it to every other case (illustrated in Fig. 1(b)). The strength of the spring between any two cases is proportional to the similarity between the two cases, i.e. cases that are similar are linked by stronger springs than cases that are dissimilar. In order to create a visualisation, the cases in a case base are initially placed in random positions on a two-dimensional plane (a three dimensional visualisation could also be created in the same way, although this is not something that we explore in this work). The cases are then allowed to move according to the forces applied to them until equilibrium is reached. As the system is allowed find its own equilibrium the stronger springs will draw together those cases that are most similar to each other. So that all cases do not form a single small group, a repulsive force between each case is also introduced.

The *attractive* forces that pull cases together and *repulsive* forces that push cases apart are illustrated in Fig. 1(c), in which each case is shown as having an attractive and repulsive force due to each other case. Thus the total force exerted on any case v, $f(v)$, can be calculated as follows:

$$f(v) = \sum_{u \in N(v)} att(u, v) + \sum_{u \in N(v)} rep(u, v) \tag{1}$$

where $att(u, v)$ is the attractive force exerted on case v by case u; $rep(u, v)$ is the repulsive force exerted on case v by case u; and $N(v)$ is the set of vertices

emanating from case v. Attractive forces are dictated by the strength of the springs between cases and are calculated using Hooke's law - i.e. $att(u, v)$ is proportional to the distance between u and v and the *zero energy length* of the spring. The zero energy length of a spring is the length at which the spring will exert no attractive forces and is directly proportional to the similarities between the cases it connects. The repulsive forces between cases are modelled as Newtonian gravitational forces, and so follow an inverse square law.

Thus, Equation 1 can be expanded as follows (only the x component of the forces is shown - the y component is calculated similarly):

$$f_x(v) = \sum_{u \in N(v)} k_{att} * \frac{(dist(u,v) - zero(u,v)) * (v_x - u_x)}{dist(u,v)} + \sum_{u \in N(v)} k_{rep} * \frac{(u_x - v_x)}{dist(u,v)^3} \tag{2}$$

where $dist(u, v)$ is the distance on the graph between nodes u and v; $zero(u, v)$ is the zero energy length of the spring between u and v; u_x and v_x are the x components of the positions of cases u and v respectively; k_{rep} is the repulsive force coefficient; and k_{att} is the attractive force coefficient. k_{att} and k_{rep} are used to balance the effects of the attractive and repulsive forces. Based on recommendations from [12], and some preliminary experiments, we set $k_{att} = \frac{1}{3}$ and $k_{rep} = 3$.

The zero energy length of each spring should be proportional to the similarity between the cases it connects, and is calculated as follows:

$$zero(u, v) = (1 - sim(u, v)) * maxZeroEnergyLength \tag{3}$$

where $sim(u, v)$ is the similarity between cases u and v, and $maxZero-EnergyLength$ is the distance apart that two cases should appear in the visualisation graph if they are totally dissimilar to each other (i.e. the zero energy length of a very weak spring). $maxZeroEnergyLength$ is a constant that, again following recommendations from [12], we set to $4,900$. In effect, $maxZeroEnergyLength$ determines the size of the visualisation created.

Equation 2 calculates the forces applied to each case in a visualisation at a specific moment in time. To animate the way in which a visualisation finds equilibrium, the forces are repeatedly calculated and small movements are applied to the cases until equilibrium is reached. In order to avoid large jumps across the graph space these small movements are limited as follows:

$$f_x(v) = \begin{cases} -5 & \text{for} \quad f_x(v) \leq -5 \\ f_x(v) & \text{for} \quad -5 < f_x(v) < 5 \\ 5 & \text{for} \quad f_x(v) \geq 5 \end{cases} \tag{4}$$

The system at present is capable of handling case bases including thousands of cases. However, further work will be required in order to make it scalable to case bases beyond this size.

3.2 Using Similarity Transformations to Aid Visualisations

All of the similarity measures described in this paper are designed to produce similarity scores in the range $[0, 1]$. However, with certain case bases some similarity measures can return collections of similarity values that have very skewed distributions. This can result in visualisations where cases tend to be maximally separated (when the distributions are skewed towards zero) or bunched very tightly together which does not allow patterns to be seen (when similarity distributions are skewed towards one). In these scenarios it is useful to perform a transformation on the similarity values in order to create more useful visualisations. The CBTV system uses power law transformations to allow fine tuning by a user. Power laws transformations were chosen as they are straightforward to implement and were shown empirically to be effective. Each similarity value in the pair-wise similarity matrix used to create visualisations is transformed according to $s' = s^{\gamma}$ where s is the original similarity value, s' is its transformed equivalent, and γ is a value chosen by the user.

Fig. 2 shows an example for a small classification dataset. It is difficult to interpret the visualisation shown in Fig. 2(a) as all of the cases are overly clustered together due to the distribution of the similarities being skewed, evident from the similarity histogram shown in Fig. 2(a). However, by applying a power law transformation with $\gamma = 4$ the skew can be corrected, leading to a much more useful visualisation, as shown in Fig. 2(b).

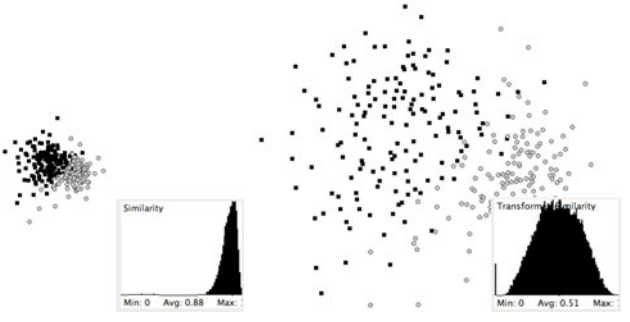


Fig. 2. The effect on a visualisation of a similarity transformation

However, these sorts of transformations must be used with care, particularly when comparing different similarity measures. One may draw the conclusion that one similarity measure creates a better separation of classes than another, when in fact it is the case that a more appropriate transformation has been applied to the first visualisation than to the second. For this reason in the scenarios presented in Section 4 the most appropriate transformations are applied in each individual case. The appropriateness of a transformation is measured by how well the similarities in the original feature space and the distances on the resulting visualisation correlate. This is discussed in the Section 3.3.

3.3 Measuring the Quality of Visualisations

Creating a force directed graph model of a case base involves projecting a series of multi-dimensional cases to a two dimensional spatial representation. There is a danger in this projection that the lower dimensional version will not be capable of suitably representing the higher dimensional information. In fact, we should *not* expect the lower dimensional representation to contain all of the information in the higher dimensional version. However, unless the lower dimensional version is a reasonable approximation of the higher dimensional version the visualisations created will not be useful.

In order to evaluate this projection correlation tests following Smyth *et al.* [24] are performed. The correlation tests evaluate the Pearson correlation between the pair-wise similarity matrix generated for the case base and the pair-wise distance matrix calculated from the visualisation. All correlations are negative because distances and similarities are being correlated. As long as a reasonable correlation exists between the two matrices, it is acceptable to interpret the potential of a similarity measure from the visualisation which uses it. However, if a reasonable correlation does not exist, then the visualisation is not useful. Preliminary experiments have suggested that correlations above $|0.6|$ indicate that a visualisation is a useful representation of the case base.

4 Demonstration and Discussion

In this section we will demonstrate how CBTV can be used to explore the suitability of different similarity measures for different case bases. Before presenting the actual visualisations, the datasets used and the similarity measures available for comparison when visualising each of these datasets are described.

4.1 Datasets and Similarity Measures

The datasets used in this work are listed in Table 1. Also shown are the size of each dataset and the similarity measures that have been used for each dataset type. Three different types of dataset have been used: textual datasets, numeric datasets, and heterogeneous datasets.

Table 1. The datasets used in the demonstration of the CBTV system with their sizes and the range of similarity measures available for each type

Type	**Similarity Measures**	**Name**	**#Instances**	**#Features**
Textual	Euclidean, Cosine, Jacard, Normalized Compression Distance	Spam-1	500	4,449
		Spam-2	500	3,219
		Reuters	500	828
Numeric	Euclidean, Manhattan, Chebyshev, Mahalanobis	Breast Cancer	569	30
Heterogeneous	Basic, Designed	Camera	210	19

Three textual classification datasets are included: two spam detection datasets [7], and a binary classification dataset generated from the Reuters collection[1]. For the textual datasets we have experimented with three feature-based similarity measures: normalised Euclidean distance [5], cosine similarity [1], and the Jaccard index [5]; and one feature-free similarity measure: Normalized Compression Distance (NCD) [15]. For the three feature-based measures, texts are tokenised at the word level and feature values are recorded as unit length normalised word frequencies. Stop-word removal and document frequency reduction (removing all words that occur in less than 3 documents in the dataset) was performed on each dataset. Normalized Compression Distance was implemented using the gzip compressor as described in [6].

The numeric dataset is the Breast Cancer classification dataset [26] from the UCI Machine Learning Repository[2]. Four similarity measures are considered: normalised Euclidean distance, normalised Manhattan distance, Chebyshev distance [5] and Mahalanobis distance [17].

Although all of the classification datasets used represent binary classification problems, the system works equally well for classification problems with any number of classes. Table 2 shows 5*10 fold cross validation average class accuracies using a 3 nearest neighbour classifier for each dataset using each similarity measure (the best result for each dataset is shown in bold).

Table 2. Cross validation accuracies for each dataset

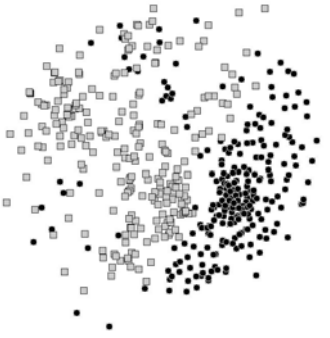
Dataset	**Cosine**	**Euclidean**	**Jaccard**	**NCD**
Spam-1	96.96	97.56	94.6	**98.4**
Spam-2	96.12	96.84	95.4	**97.92**
Reuters	95.36	91.64	94.88	**95.8**
Dataset	**Chebyshev**	**Euclidean**	**Manhattan**	**Mahalanobis**
BreastCancer	94.8	96.66	**96.9**	81.98

The final dataset considered in this paper, the Camera dataset[3], is a heterogeneous case base which contains both numeric and discrete features and examines choices made by customers in selecting a camera to purchase. Two types of similarity measure are considered for this dataset. The first is a straightforward direct type distance metric and the second type is a *designed* direct measure which takes into account domain knowledge to create feature-level utility-type similarity measures such as those in [8].

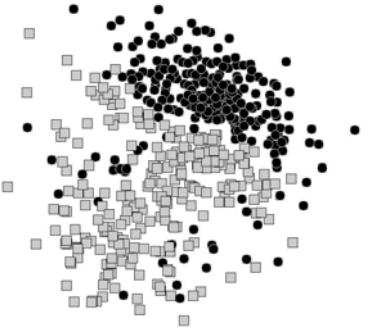
4.2 Exploring Datasets Using CBTV

Fig. 3 shows visualisations (and correlation scores) of the Spam-1 dataset using the four specified similarity measures (with γ equal to 1, 0.3, 0.3 and 0.3 for

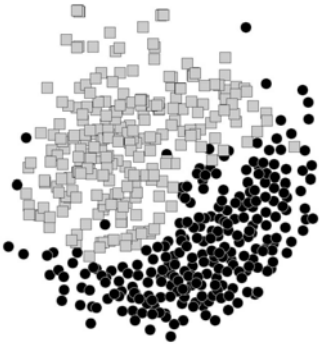
[1] Available at: http://www.daviddlewis.com/resources/testcollections/reuters21578/
[2] Available at: http://archive.ics.uci.edu/ml/
[3] Available at: http://cbrwiki.fdi.ucm.es/wiki/index.php/Case_Bases



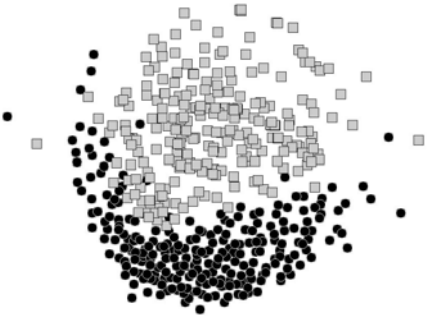
(a) Cosine Similarity (corr: −0.846)



(b) Euclidean Distance (corr: −0.854)



(c) Jacard Index (corr: −0.748)



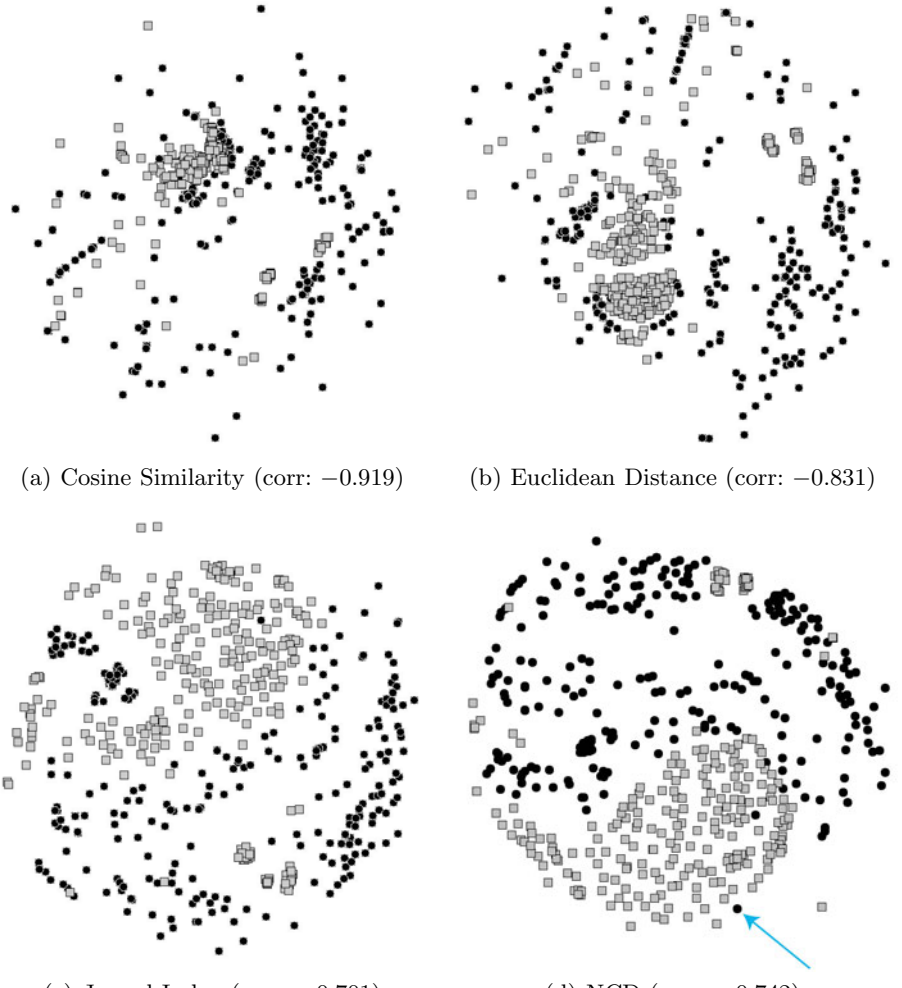
(d) NCD (corr: −0.781)

Fig. 3. Visualisations of the Spam-1 dataset using different similarity measures

the Cosine, Euclidean, Jacard and NCD measures respectively) and tells an interesting story. Figs. 3(a), 3(b) and 3(c) show that when the Cosine, Euclidean and Jacard measures are used, the two classes do not appear to separate well and there is an amount of confusion at the class boundary. However, in Fig. 3(d) much better class separation is apparent. This would suggest to us that either the NCD similarity measure is the most appropriate for this problem - which is backed up by the accuracies in Table 2.

Fig. 4 shows a set of visualisations for the Spam-2 dataset (no transformations are used) and tells a similar, if less pronounced, story. The visualisation showing the use of the NCD similarity measures, Fig. 4(d), displays the cleanest class separation which again suggests that this is likely to be the most successful similarity measure for this problem. This is in agreement with the results in Table 2 and confirms previous work on feature-free similarity measures in [6].

Fig. 4(d) also illustrates how CBTV can be used for noise detection. A lone spam example, denoted with the arrow, is seen amongst a number of non-spam examples. CBTV allows users to select an example and view the underlying case (in this case the original spam email). Investigation showed that this email is a one of a number of new journal issue notifications that the user originally classified as non-spam but over time began to classify as spam.



(a) Cosine Similarity (corr: −0.919)

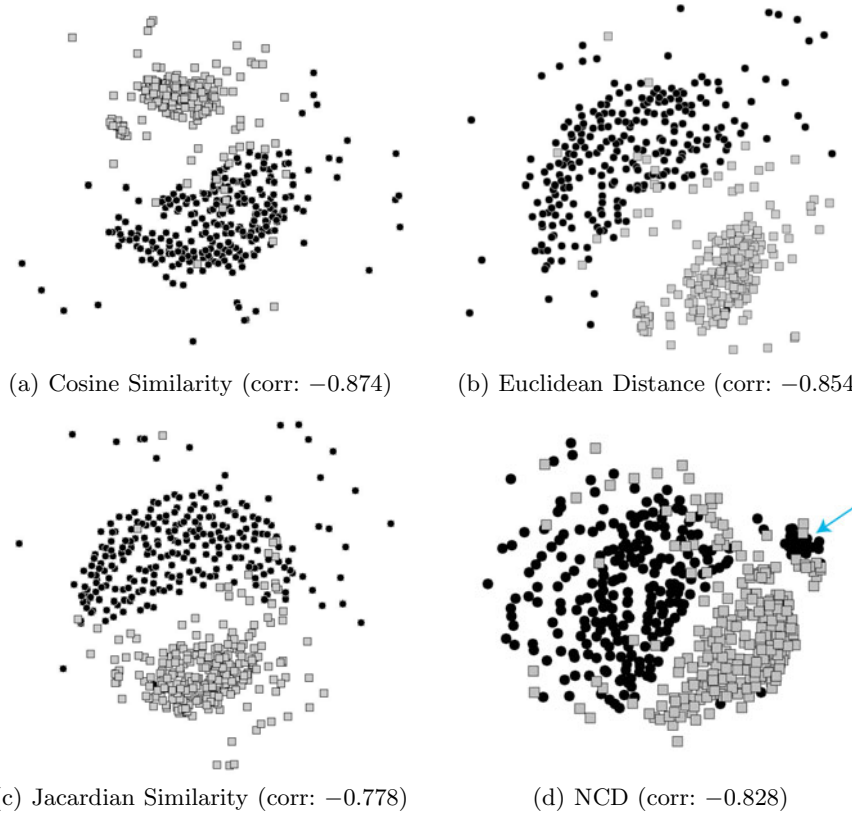
(b) Euclidean Distance (corr: −0.831)

(c) Jacard Index (corr: −0.701)

(d) NCD (corr: −0.742)

Fig. 4. Visualisations of the Spam-2 dataset using different similarity measures

Fig. 5 shows the set of visualisations for the Reuters text dataset. In all of these visualisations a power law transformation with $\gamma = 0.3$ is used. This time it is particularly interesting to note the clustering in the region marked with the arrow in Fig. 5(d). This small cluster, which includes cases from each class, represents documents that have been truncated and appended with the text "*blah blah blah*". The NCD similarity measure responds particularly strongly to this repeated text, a connection that is not evident in the other visualisations. This is a clear example of how visualising different similarity measures can illustrate how each responds to particular characteristics of the underlying dataset. The visualisations of the Reuters dataset also suggest the existence of possible



(a) Cosine Similarity (corr: −0.874)

(b) Euclidean Distance (corr: −0.854)

(c) Jacardian Similarity (corr: −0.778)

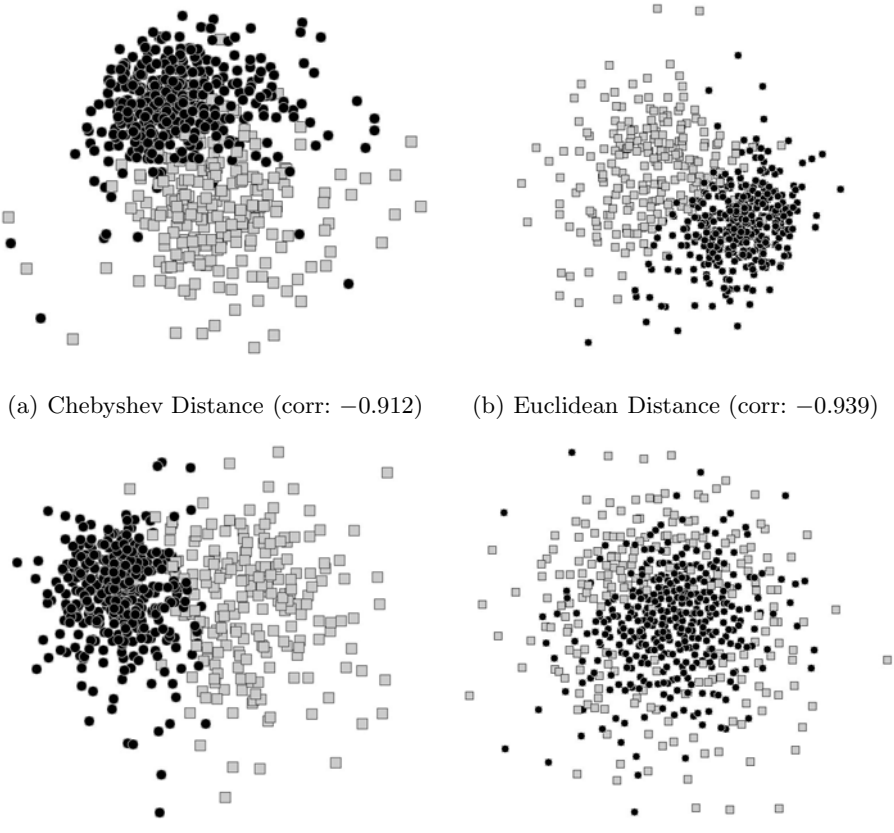
(d) NCD (corr: −0.828)

Fig. 5. Visualisations of the Reuters dataset using different similarity measures

outliers, or unusual cases, particularly in Figs. 5(a), 5(b) and 5(c). This is another useful feature of creating visualisations.

Fig. 6 shows the visualisations created for the numeric BreastCancer dataset (power-law transformations with γ equal to 5, 5, 3 and 2 are used for the Euclidean, Manhattan, Chebyshev and Mahalanobis measures respectively). These visualisations tell a slightly different story than those of the text-based datasets. The best class separation appears using the Manhattan and Euclidean measures, which is not surprising as these are the predominant measures used for this kind of dataset. The visualisation based on the Chebyshev measure (Fig. 6(a)) shows reasonable separation, while for the Mahalanobis measure (Fig. 6(d)) class separation is poor, although an interesting and very distinctive pattern is present. This visualisations match the cross validation results shown in Table 2.

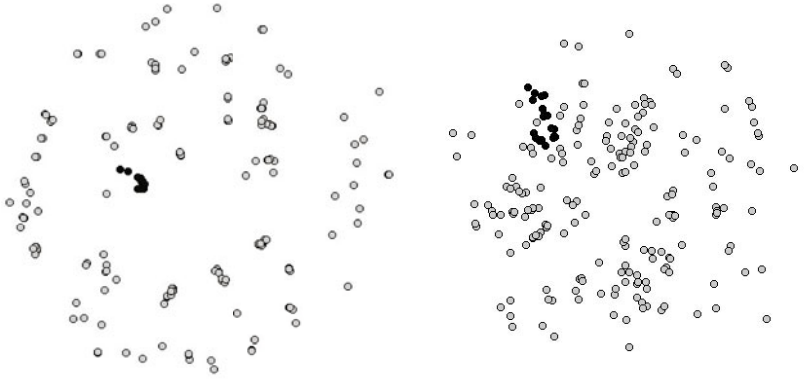
Finally, Fig. 7 shows the visualisations for the heterogeneous Camera dataset - the only non-classification dataset used in this paper. We created visualisations for a number of heterogeneous datasets using basic and the designed measures and in general found that differences were minimal. This might suggest that the



(a) Chebyshev Distance (corr: −0.912) (b) Euclidean Distance (corr: −0.939)

(c) Manhattan Distance (corr: −0.948) (d) Mahanalobis Distance (corr: −0.741)

Fig. 6. Visualisations of the Breast Cancer dataset using different similarity measures



(a) Basic Similarity (corr: −0.66) (b) Designed Similarity (corr: −0.65)

Fig. 7. Visualisations of the Camera dataset using different similarity measures

extra effort of designing a similarity measure to incorporate domain knowledge is not worthwhile. However, the Camera dataset proved quite interesting and illustrates the benefits of domain-focussed similarity measures. Fig. 7(a) shows the visualisation of the basic direct measure where there is a small number of very tightly packed clusters of cases. In Fig. 7(b) cases are much better spread out across the visualisation space as a result of the domain knowledge introduced by the utility-type similarity measure used. We illustrate this by highlighting in black one of the clusters in Fig. 7(a) and highlighting in black the same cases in Fig. 7(b). It is clear that in Fig. 7(b) the cases are better spread out.

5 Conclusions and Future Work

In CBR the design and selection of similarity measures is paramount. While techniques such as the comparison of cross-validation accuracies can be used for this, it is also useful to have more exploratory techniques available and the CBTV system has been built for this purpose. Using a force-directed graph drawing algorithm visualisations of cases bases using different similarity measures can be created and compared. Furthermore, by allowing the user to query the underlying cases in areas of particular interest, possible noisy cases and probable outliers can be investigated. In this paper we have presented the CBTV system and illustrated a range of visualisations that can be created using it for a variety of case base and similarity measure types. We believe that these worked examples illustrate the usefulness of the system.

In the future we intend to extend this work in a number of potentially interesting directions. The first of these is to look at how the visualisations can be given broader coverage and made more interactive. For example, we will allow interactive selection of particular features to use within a similarity measure and interactive adjustment of feature weights. Focusing on text, we will allow for pre-processing tasks such as stop-word removal and stemming to be activated or deactivated at run-time and, finally, allow for visualisation of interactive machine learning tasks such as active learning. Finally, there is a random element present in the creation of visualisations (cases are initially placed randomly in the visualisation space) and there can be small differences in repeated visualisations of a dataset. We will investigate techniques such as repeatedly running the visualisation to an equilibirum state and using an aggregation of these results in the final visualisation in order to address this issue.

Acknowledgments. This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/RFP/CMSF718.

References

1. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press/Addison-Wesley (1999)

2. Bonzano, A., Cunningham, P., Smyth, B.: Using introspective learning to improve retrieval in CBR. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 291–302. Springer, Heidelberg (1997)
3. Cheng, W., Hüllermeier, E.: Learning similarity functions from qualitative feedback. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 120–134. Springer, Heidelberg (2008)
4. Coyle, L., Cunningham, P.: Improving recommendation ranking by learning personal feature weights. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 560–572. Springer, Heidelberg (2004)
5. Cunningham, P.: A taxonomy of similarity mechanisms for case-based reasoning. IEEE Transactions on Knowledge and Data Engineering 21, 1532–1543 (2008)
6. Delany, S.J., Bridge, D.: Textual case-based reasoning for spam filtering: A comparison of feature-based and feature-free approaches. Artificial Intelligence Review 26(1-2), 75–87 (2006)
7. Delany, S.J., Cunningham, P., Coyle, L.: An assessment of case-based reasoning for spam filtering. Artificial Intelligence Review Journal 24(3-4), 359–378 (2005)
8. Doyle, D., Cunningham, P., Bridge, D., Rahman, Y.: Explanation oriented retrieval. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 157–168. Springer, Heidelberg (2004)
9. Dubitzky, W., Azuaje, F.J.: A genetic algorithm and growing cell structure approach to learning case retrievel structures. In: Soft computing in Case Based Reasoning, pp. 115–146 (2001)
10. Eades, P.: A heuristic for graph drawing. Congressus Nutnerantiunt 42, 149–160 (1984)
11. Falkman, G.: The use of a uniform declarative model in 3D visualisation for case-based reasoning. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 103–117. Springer, Heidelberg (2002)
12. Huang, M.L., Eades, P., Lai, W.: On-line visualization and navigation of the global web structure. Journal of Software Eng. and Knowledge Eng. 13(1), 27–52 (2003)
13. Jarmulak, J., Craw, S., Crowe, R.: Genetic algorithms to optimise cbr retrieval. In: Blanzieri, E., Portinale, L. (eds.) EWCBR 2000. LNCS (LNAI), vol. 1898, pp. 136–147. Springer, Heidelberg (2000)
14. Kontkanen, P., Lahtinen, J., Myllymäki, P., Silander, T., Tirri, H.: Supervised model-based visualization of high-dimensional data. Intelligent Data Analysis 4(3-4), 213–227 (2000)
15. Li, M., Chen, X., Li, X., Ma, B., Vitanyi, P.: The similarity metric. In: Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 863–872 (2003)
16. Liao, T.W., Zhang, Z., Mount, C.: Similarity measures for retrieval in case-based reasoning systems. Applied Artificial Intelligence 12(4), 267–288 (1998)
17. Maesschalck, R.D., Jouan-Rimbaud, D., Massart, D.L.: The mahalanobis distance. Chemometrics and Intelligent Laboratory Systems 50(1), 1–18 (2000), http://www.sciencedirect.com/science/article/B6TFP-3Y8VGYK-1/2/92ac3e8ac922320df9a62b096aca9bee
18. Massie, S., Craw, S., Wiratunga, N.: Visualisation of case-base reasoning for explanation. In: Procs. ICCBR 2004 Workshop on Long-Lived CBR Systems, pp. 135–144 (2004)
19. McArdle, G., Wilson, D.C.: Visualising case-base usage. In: Procs. of ICCBR 2003 Workshop on Long-Lived CBR Systems (2003)
20. McKenna, E., Smyth, B.: An interactive visualisation tool for case-based reasoners. Applied Intelligence 14(1), 95–114 (2001)

21. Núñez, H., Sànchez-Marrè, M., Cortés, U., Comas, J., Martínez, M., Rodríguez-Roda, I., Poch, M.: A comparative study on the use of similarity measures in case-based reasoning to improve the classification of environmental system situations. Environmental Modelling and Software 19(9), 809–819 (2004)
22. Osborne, H., Bridge, D.: Similarity metrics: A formal unification of cardinal and non-cardinal similarity measures. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS (LNAI), vol. 1266, pp. 235–244. Springer, Heidelberg (1997)
23. Ricci, F., Avesani, P.: Learning a local similarity metric for case-based reasoning. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS (LNAI), vol. 1010, pp. 301–312. Springer, Heidelberg (1995)
24. Smyth, B., Mullins, M., McKenna, E.: Picture perfect: Visualisation techniques for case-based reasoning. In: Procs of ECAI 2000, pp. 65–72 (2000)
25. Stahl, A., Gabel, T.: Using evolution programs to learn local similarity measures. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 537–551. Springer, Heidelberg (2003)
26. Street, W., Wolberg, W., Mangasarian, O.: Nuclear feature extraction for breast tumor diagnosis. In: IS&T/SPIE 1993, pp. 861–870 (1993)
27. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artif. Intell. Rev. 11(1-5), 273–314 (1997)
28. Wilson, D., Martinez, T.: Improved heterogeneous distance functions. Journal of Artificial Intelligence Review 6, 1–34 (1997)
29. Xiong, N., Funk, P.: Building similarity metrics reflecting utility in case-based reasoning. Journal of Intelligent and Fuzzy Systems 17(4), 407–416 (2006)

Goal-Driven Autonomy with Case-Based Reasoning

Héctor Muñoz-Avila[1], Ulit Jaidee[1], David W. Aha[2], and Elizabeth Carter[1]

[1] Department of Computer Science & Engineering
Lehigh University; Bethlehem, PA 18015
[2] Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory (Code 5514); Washington, DC 20375
ulj208@lehigh.edu, munoz@cse.lehigh.edu, david.aha@nrl.navy.mil,
eec209@lehigh.edu

Abstract. The vast majority of research on AI planning has focused on automated plan recognition, in which a planning agent is provided with a set of inputs that include an initial goal (or set of goals). In this context, the goal is presumed to be static; it never changes, and the agent is not provided with the ability to reason about whether it should change this goal. For some tasks in complex environments, this constraint is problematic; the agent will not be able to respond to opportunities or plan execution failures that would benefit from focusing on a different goal. *Goal driven autonomy* (GDA) is a reasoning framework that was recently introduced to address this limitation; GDA systems perform anytime reasoning about what goal(s) should be satisfied [4]. Although promising, there are natural roles that case-based reasoning (CBR) can serve in this framework, but no such demonstration exists. In this paper, we describe the GDA framework and describe an algorithm that uses CBR to support it. We also describe an empirical study with a multiagent gaming environment in which this CBR algorithm outperformed a rule-based variant of GDA as well as a non-GDA agent that is limited to dynamic replanning.

1 Introduction

One of the most frequently cited quotes from Helmuth von Moltke, one of the greatest military strategists in history, is that "no plan survives contact with the enemy" [1]. That is, even the best laid plans need to be modified when executed because of (1) the non-determinism in one's own actions (i.e., actions might not have the intended outcome), (2) the intrinsic characteristics of adversarial environments (i.e., the opponent might execute unforeseen actions, or even one action among many possible choices), and (3) imperfect information about the world state (i.e., opponents might be only partially aware of what the other side is doing).

As a result, researchers have taken interest in planning that goes beyond the classic deliberative model. Under this model, the state of the world changes solely as a result of the agent executing its plan. So in a travel domain, for example, a plan may include an action to fill a car with enough gasoline to follow segments (A,B) and (B,C) to drive to location C from location A. The problem is that the dynamics of the environment might change (e.g., segment (B,C) might become unavailable due to some road damage). Several techniques have been investigated that respond to contingencies which

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 228–241, 2010.

may invalidate the current plan during execution. This includes *contingency planning* [2], in which the agent plans in advance for plausible contingencies. In the travel example, the plan might include an alternative subplan should (B,C) becomes unavailable. One such subplan might call to fill up with more gasoline at location B and continue using the alternative, longer route (B,D), (D,C). A drawback of this approach is that the number of alternative plans required might grow exponentially with the number of contingencies that need to be considered. Another alternative suggested is *conformant planning* [3], where the generated plan is guaranteed to succeed. For example, the plan might fill up with enough gasoline at B so that, even if it has to go back to B after attempting to cover the segment (B,C), it can continue with the alternative route (B,D), (D,C). The drawback is that the plan might execute many unnecessary steps for contingencies that do not occur (such as obtaining additional gasoline while initially in location B).

Recently, another alternative, called *Goal Driven Autonomy* (GDA), was proposed to solve this problem [4, 5]. GDA agents continuously monitor the current plan's execution and assess whether the actual states visited during the current plan's execution match expectations. When mismatches occur (i.e., when the state does not meet expectations), a GDA monitor will suggest alternative goals that, if accomplished, would fulfill its overarching objectives. In our travel example, the GDA monitor might suggest that the agent first drive to location D and then to C.

In this paper we introduce and assess CB-gda, the first GDA system to employ case-based reasoning (CBR) methods [6]. CB-gda uses two case bases to dynamically generate goals. The first case base relates goals with expectations, while the latter's cases relate mismatches with (new) goals. We describe an empirical study of CB-gda on the task of winning games defined using a complex gaming environment (DOM). Our study revealed that, for this task, CB-gda outperforms a rule-based variant of GDA when executed against a variety of opponents. CB-gda also outperforms a non-GDA replanning agent against the most difficult of these opponents and performs similarly against the easier ones. In direct matches, CB-gda defeats both the rule-based GDA system and the non-GDA replanner.

The rest of the paper continues as follows. Section 2 describes a testbed that we use for our experiments and for motivating some of the GDA concepts. Section 3 presents the GDA framework. In Section 4, we discuss related work. In Section 5, we introduce our case-based GDA algorithm and give an example of its behavior in Section 6. Finally, Section 7 presents our empirical study and Section 8 discusses the results and plans for future work.

2 DOM Environment

Domination (DOM) games are played in a turn-based environment in which two teams (of *bots*) compete to control specific locations called *domination points*. Each time a bot on team *t* passes over a domination point, that point will belong to *t*. Team *t* receives one point for every 5 seconds that it owns a domination point. Teams compete to be the first to earn a predefined number of points. Domination games have been used in a variety of combat games, including first-person shooters such as Unreal Tournament and online role-playing games such as World of Warcraft.

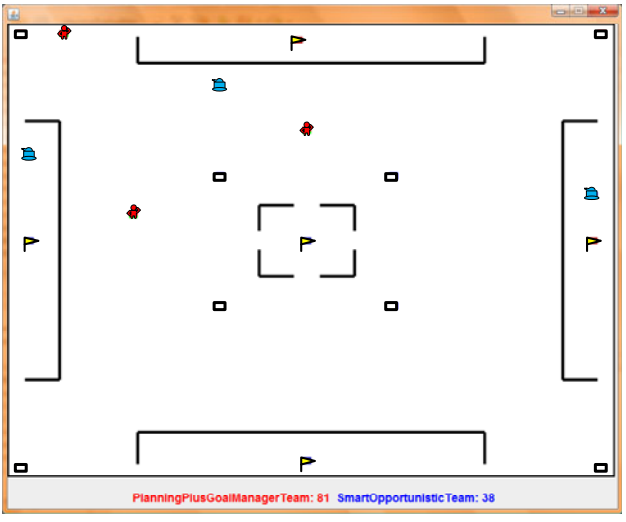


Fig. 1. An example DOM game map with five domination points (yellow flags), where small rectangles identify the agents' respawning locations, and the remaining two types of icons denote each player's agents

Domination games are popular because they reward team effort rather than individual performance. No awards are given for killing an opponent team's bot, which *respawns* immediately in a location selected randomly from a set of map locations, and then continues to play. Killing such bots might be beneficial in some circumstances, such as killing a bot before she can capture a location, but the most important factor influencing the outcome of the game is the strategy employed. An example strategy is to control half plus one of the domination locations. A location is captured for a team whenever a bot in that team moves on top of the location and within the next 5 game ticks no bot from another team moves on top of that location. Figure 1 displays an example of DOM map with five domination points [7].

Bots begin the game and respawn with 10 health points. Enemy encounters (between bots on opposing teams) are handled by a simulated combat consisting of successive die rolls, each of which makes the bots lose some number of health points. The die roll is modified so that the odds of reducing the opponent health points increase with the number of friendly bots in the vicinity. Combat finishes when the first bot health points decreases to 0 (i.e., the bot dies). Once combat is over, the death bot is respawned from a spawn point owned by its team in the next game tick. Spawn point ownership is directly related to domination point ownership, if a team owns a given domination point the surrounding spawn points also belong to that team.

The number of possible states in DOM is a function of (1) the number of cells in the map, which is $n * m$ where n is the number of rows and m is the number of columns, (2) the number, b, of bots in each team, (3) for each bot the number of health points (between 0 and 10), (4) the number, t, of teams, (5) the number, d, of domination locations, (6) a number between 0 and 5 for each domination location; 0 indicates than no other team is trying to capture the location and 1 to 5 indicates the number of

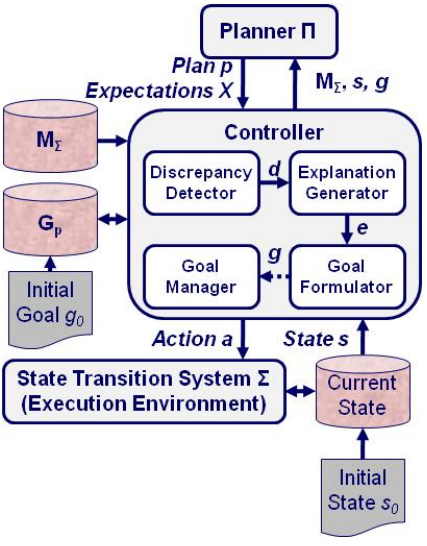


Fig. 2. The GDA conceptual model

game ticks since a bot has attempted to capture the location belonging to a different team. The total number of states is $(n \times m)^{(b \times t)} \times 11^{(b \times t)} \times 6^{d} \times (t+1)^{d}$. The exponent $(t+1)$ accounts for the beginning of the game in which the domination locations do not belong to any team. In our experiments, $n = m = 70$, $b = 3$, $t = 2$, and d = 4. Hence, the number of possible states is 2×10^{34}.

Because of the large number of possible states, we follow the state abstraction model of Auslander *et al.* (2008) for decision making by the AI agents in DOM games [7]. Since there is no reward for killing an opponent, emphasis is made into controlling the domination locations. Hence the states in the abstract model simply indicate which team owns the domination locations, reducing the number of states to $d^{(t+1)}$. The decisions that the agent makes is to decide to which domination location to send each bot. Thus, in the abstract model the number of actions is $(b \times t)^{d}$.

DOM is a good testbed for testing algorithms that integrate planning and execution because domination actions are *non-deterministic*; if a bot is told to go to a domination location the outcome is uncertain because the bot may be killed along the way. Domination games are also *adversarial*; two or more teams compete to control the domination points. Finally, domination games are *imperfect information* games; a team only knows the locations of those opponent bots that are within the range of view of one of the team's own bots.

3 Goal Driven Autonomy

Goal driven autonomy permits autonomous agents to direct the focus of their planning activities, and thus become more self-sufficient.

> **Definition**: *Goal Driven Autonomy* (GDA) is a goal reasoning method for problem solving in which autonomous agents dynamically identify and self-select their goals throughout plan execution.

The GDA conceptual model has four steps, as shown within the Controller component in Figure 2. This model extends the conceptual model of online classical planning [8], whose components include a Planner, a Controller, and a State Transition System $\Sigma = (S,A,E,\gamma)$ with states S, actions A, exogenous events E, and state transition function γ: $S\times(A\cup E)\rightarrow 2^S$. In the GDA model, the Controller is centric. It receives as input a planning problem (M_Σ,s_c,g_c), where M_Σ is a model of Σ, s_c is the current state (e.g., initially s_c), and $g_c\in G$ is a goal that can be satisfied by some set of states $S_g\subseteq S$. It passes this problem to the Planner Π, which generates a sequence of actions $A_c=[a_c,\ldots,a_{c+n}]$ and a corresponding sequence of expectations $X_c=[x_c,\ldots x_{c+n}]$, where each $x_i\in X_c$ is a set of constraints that are predicted to hold in the corresponding sequence of states $[s_{c+1},\ldots,s_{c+n+1}]$ when executing A_c in s_c using M_Σ. The Controller sends a_c to Σ for execution and retrieves the resulting state s_{c+1}, at which time it performs the following knowledge-intensive (and GDA-specific) tasks:

1. *Discrepancy detection*: GDA detects unexpected events before deciding how to respond to them. This task compares observations s_{c+1} with expectation x_c. If one or more discrepancies (i.e., unexpected observations) $d\in D$ are found in s_{c+1}, then explanation generation is performed to explain them.
2. *Explanation generation*: This module explains a detected discrepancy d. Given also state s_c, this task hypothesizes one or more explanations $ex\in Ex$ of their cause.
3. *Goal formulation*: Resolving a discrepancy may warrant a change in the current goal(s). This task generates goal $g\in G$ given a discrepancy d, its explanation ex, and current state s_c.
4. *Goal management*: New goals are added to the set of pending goals $G_P\subseteq G$, which may also warrant other edits (e.g., removal of other goals). The Goal Manager will select the next goal $g'\in G_P$ to be given to the Planner. (It is possible that $g=g'$.)

The GDA model makes no commitments to the choice of Planner or algorithms for these four tasks. For example, Muñoz-Avila *et al.* (2010) describe GDA-HTNbots [4], a system that implements a simple GDA strategy in which Π=SHOP [9], the Discrepancy Detector triggers on any mismatch between the expected and current states, a rule-based reasoner is used for the Explanation Generator and Goal Formulator, and the Goal Manager simply replaces the current goal with any newly formulated goal. However, CBR is not employed in GDA-HTNbots.

4 Related Work

Cox's (2007) investigation of self-aware agents inspired the conception of GDA [10], with its focus on integrated planning, execution, and goal reasoning. Some of the terms we adopt, such as expected and actual states, are directly borrowed from that work.

In the introduction we discussed two alternatives to GDA: contingency planning and conformant planning. Their main drawback is that, before plan execution, they require the a priori identification of possible contingencies. In DOM games, a plan would need to determine which domination points to control, which locations to send

a team's bots, and identify alternative locations when this is not possible. An alternative to generating contingencies beforehand is performing plan repair. In plan repair, if a mismatch occurs during plan execution (i.e., between the conditions expected to be true to execute the next action and the actual world state), then the system must adapt the remaining actions to be executed in response to the changing circumstances [11, 12]. The difference between plan repair and GDA is that plan repair agents retain their goals while GDA agents can reason about which goals should be satisfied. This also differentiates GDA from replanning agents, which execute a plan until an action becomes inapplicable. At this point, the replanning agent simply generates a new plan from the current state to achieve its goals [13, 14, 15].

There has been some research related to reasoning with goals. Classical planning approaches attempt to achieve all assigned goals during problem solving [16]. Van den Briel *et al.* (2004) relax this requirement so that only a maximal subset of the goals must be satisfied (e.g., for situations where no plan exists that satisfies all the given goals) [17]. Unlike GDA, this approach does not add new goals as needed. Formulating new goals has been explored by Coddington and Luck (2003) and Meneguzzi and Luck (2007), among others [18, 19]. They define *motivations* that track the status of some state variables (e.g., the gasoline level in a vehicle) during execution. If these values exceed a certain threshold (e.g., if the gasoline level falls below 30%), then the motivations are triggered to formulate new goals (e.g., fill the gas tank). In contrast, we investigate the first case-based approach for GDA, where goals are formulated by deriving inferences from the game state and the agent's expectations using case-based planning techniques.

5 Case-Based Goal Driven Autonomy

Our algorithm for case-based GDA uses two case bases as inputs: the *planning* case base and the *mismatch-goal* case base. The planning case base (PCB) is a collection of triples of the form (s_c, g_c, e_c, pl), where s_c is the observed state of the world (formally, this is defined as a list of atoms that are true in the state), g_c is the goal being pursued (formally, a goal is a predicate with a task name and a list of arguments), e_c is the state that the agent expects to reach after accomplishing g_c starting from state s_c, and pl is a plan that achieves g_c. The mismatch-goal case base (MCB) is a collection of pairs of the form (m_c, g_c), where m_c is the mismatch (the difference between the expected state e_c and the actual state s_c) and g_c is the goal to try to accomplish next. In our current implementation both PCB and MCB are defined manually. In Section 6 we will discuss some approaches we are considering to learn both automatically.

CB-gda(D, A, g_{init}, PCB, $\text{SIM}_g()$, th_g, MCB, $\text{SIM}_s()$, th_s, $\text{SIM}_m()$, th_m) =
// Inputs:
// D: Domain simulator (here, *DOM*) *A:* The CBR intelligent agent
// g_{init}: Initial goal PCB: Planning case base
// $\text{SIM}_g()$: True iff goals are similar MCB: Mismatch-goal case base
// $\text{SIM}_s()$:True iff states are similar $\text{SIM}_m()$: True iff mismatches are similar
// $th_{g/s/m}$: Thresholds for defining whether goals/state/mismatches are similar
// Output: the final score of simulation D

1. **run**(D,A,g_{init})
2. **while** status(D)=Running **do**
3. | s_i ← currentState(D) ; g_i ← currentGoal(A,D)
4. | **while** SIM_g(currentTask(A),g_i) **do**
5. | | **wait**(t)
6. | | e_c← retrieve(PCB, g_i, s_i, $SIM_s()$, th_s, $SIM_g()$, th_g)
7. | | s_D ← currentState(D)
8. | | **if** $e_c \neq s_D$ **then**
9. | | | $g_{c'}$ ← retrieve(MCB, e_c, mismatch(e_c,s_D), $SIM_m()$, th_m)
10. | | | **run**(D,A,$g_{c'}$)
11. **return** game-score(D)

The algorithm above displays our CBR algorithm for GDA, called *CB-gda*. It runs the game D for the GDA-controlled agent A, which is ordered to pursue a goal g_{init}. Our current implementation of A is a case-based planner that searches in the case base PCB for a plan that achieves g_{init}. The call to run(D,A, g_{init}) represents running this plan in the game. (Line 1). While the game D is running (Line 2), the following steps are performed. Variables s_i and g_i are initialized with the current game state and agent's goal (Line 3). The inner loop continues running while A is attempting to achieve g_i (Line 4). The algorithm waits a time t to let the actions be executed (Line 5). Given the current goal g_i and the current state s_i, agent A searches for a case (s_c, g_c, $e_{c,\,pl}$) in PCB such that the binary relations $SIM_s(s_i,s_c)$ and $SIM_g(g_i,g_c)$ hold and returns the expected state e_c. $SIM(a,b)$ is currently an equivalence relation. (Line 6). We follow the usual textbook conventions [20] to define $SIM(a,b)$, which is a Boolean relation that holds true whenever the parameters a and b are similar to one another according to a similarity metric $\text{sim}(\cdot)$ and a threshold th (i.e., $\text{sim}(a,b) \geq th$). Since the similarity function is an equivalence relation, the threshold is 1. The current state s_D in D is then observed (Line 7). If the expectation e_c and s_D do not match (Line 8), then a case (m_c, g_c) in MCB is retrieved such that mismatch m_c and mismatch(e_c,s_D), are similar according to $SIM_m(\cdot)$; this returns a new goal $g_{c'}$ (Line 9), Finally, D is run for agent A with this new goal $g_{c'}$ (Line 10). The game score is returned as a result (Line 11).

From a complexity standpoint, each iteration of the inner loop is dominated by the steps for retrieving a case from PCB (Line 6) and from MCB (Line 9). Retrieving a case from PCB is of the order of O(|PCB|), assuming that computing $SIM_s(\cdot)$ and $SIM_g(\cdot)$ are constant. Retrieving a case from MCB is of the order of O(|MCB|), assuming that computing $SIM_m(\cdot)$ is constant. The number of iterations of the outer loop is O(N/t), assuming a game length of time N. Thus, the complexity of the algorithm is $O((N/t) \times \max\{|PCB|,|MCB|\})$.

We claim that, given sufficient cases in PCB and MCB, CB-gda will successfully guide agent A in accomplishing its objective while playing the DOM game. To assess this, we will use two other systems for benchmarking purposes. The first is HTNbots [13], which has been demonstrated to successfully play DOM games. It uses Hierarchical Task Network (HTN) planning techniques to rapidly generate a plan, which is executed until the game conditions change, at which point HTNbots is called again to generate a new plan. This permits HTNbots to react to changing conditions within the game. Hence, it is a good benchmark for CB-gda. The second benchmarking system is GDA-HTNbots [4], which implements a GDA variant of HTNbots using a rule-based

approach (i.e., rules are used for goal generation), in contrast to the CBR approach we propose in this paper.

6 Example

We present an example of CB-gda running on the DOM game. Suppose there are 3 domination points in the current instance of the game: dom_1, dom_2, and dom_3. As we explained before, the possible states that are modeled by the case-based agent is the Cartesian product $\prod_i o_i$ of the owner o_i of the domination point i. For instance, if there are 3 domination points, the state (E,F,F) denotes the state where the first domination point is owned by the enemy and the other two domination points are owned by our friendly team. Suppose that the case base agent was invoked with the goal g_{init} = *control-dom*$_1$, which sets as its goal to control dom_1. Suppose that this is the beginning of the game, so the starting state is (N,N,N), indicating that no team controls any of the domination points. Suppose that the case-based agent A retrieves a case (s_c, g_c, e_c, pl) in PCB such that g_c = *control-dom*$_1$ and s_c = (N,N,N). Thus, pl is executed in Line 1 and $s_i = (N,N,N)$ and g_i = *control-dom*$_1$ in Line 3.

After waiting for some time t, the PCB case base is consulted for a similar case (Line 6). Assume we retrieve the same case as before: (s_c, g_c, e_c, pl), where $s_c = s_i$, $g_c = g_i$, and $e_c = (F,N,N)$. This case says that with this state and with this goal, the ***expected*** state is one where the controlled team owns dom_1. Now suppose that the ***current*** state s_D as obtained in Line 7 is (E,N,N), which means that s_D differs from s_c (Line 8). At this point, a case is searched in the MCB case base (Line 9). Suppose that a case (m_c, g_c) exists (and is retrieved) such that $m_c = (F/E,\_,\_)$, which means there is only a mismatch in the ownership of dom_1. Suppose that g_c = *control-dom*$_2$*-and-dom*$_3$, a goal that tells the case-based agent to control dom_2 and dom_3. This goal is then pursued by the case-based agent in Line 10.

Table 1. Domination Teams and Descriptions

Opponent Team	Description	Difficulty
Dom1 Hugger	Sends all agents to domination point 0.	Trivial
First Half Of Dom Points	Sends an agent to the first half +1 domination points. Extra agents patrol between the 2 points.	Easy
2nd Half Of Dom Points	Sends an agent to the second half +1 domination points; extra agents patrol between the two points.	Easy
Each Agent to One Dom	Each agent is assigned to a different domination point And remains there for the entire game.	Medium-easy
Greedy Distance	Each turn the agents are assigned to the closest domination point They do not own.	Hard
Smart Opportunistic	Sends agents to each domination point The team doesn't own. If possible, it will send multiple agents to each un-owned point.	Very hard

Table 2. Average Percent Normalized Difference in the Game AI System vs. Opponent Scores (with average Scores in parentheses)

Opponent Team (controls enemies)	**Game AI System** (controls friendly forces)		
	HTNbots	**HTNbots-GDA**	**CB-gda**
Dom1 Hugger	**81.2%**† (20,002 vs. 3,759)	80.9% (20,001 vs. 3,822)	81.0% (20,001 vs. 3,809)
First Half Of Dom Points	**47.6%** (20,001 vs. 10,485)	42.0% (20,001 vs. 11,605)	45.0% (20,000 vs. 10,998)
2nd Half Of Dom Points	**58.4%** (20,003 vs. 8,318)	12.5% (20,001 vs. 17,503)	46.3% (20,001 vs. 10,739)
Each Agent to One Dom	**49.0%** (20,001 vs. 10,206)	40.6% (20,002 vs. 11,882)	45.4% (20,001 vs. 10,914)
Greedy Distance	-17.0% (16,605 vs. 20,001)	0.4% (19,614 vs. 19,534)	**17.57%** (20,001 vs. 16,486)
Smart Opportunistic	-19.4% (16,113 vs. 20,001)	-4.8% (19,048 vs. 20,001)	**12.32%** (20,000 vs. 17,537)

†**Bold face** denotes the highest average measure in each row.

7 Empirical Study

We performed an exploratory investigation to assess the performance of CB-gda. Our claim is that our case-based approach to GDA can outperform our previous rule-based approach (GDA-HTNbots) and a non-GDA replanning system (HTNbots [13]) in playing DOM games. To assess this hypothesis we used a variety of fixed strategy opponents as benchmarks, as shown in Table 1. These opponents are displayed in order of increasing difficulty.

We recorded and compared the performance of these systems against the same set of hard-coded opponents in games where 20,000 points are needed to win and square maps of size 70 x 70 tiles. The opponents above were taken from course projects and previous research using the DOM game and do not employ CBR or learning. Opponents are named after the strategy they employ. For example, Dom 1 Hugger sends all of its teammates to the first domination point in the map [7]. Our performance metric is the difference in the score between the system and opponent while playing DOM, divided by the system's score. The experimental setup tested these systems against each of these opponents on the map used in the experiments of GDA-HTNbots [4]. Each game was run three times to account for the randomness introduced by non-deterministic game behaviors. Each bot follows the same finite state machine. Thus, the difference of results is due to the strategy pursued by each team rather than by the individual bot's performance.

The results are shown in Table 2, where each row displays the normalized average difference in scores (computed over three games) against each opponent. It also shows the average scores for each player. The results for HTNbots and GDA-HTNbots are the same as reported in [4], while the results for CB-gda are new. We repeated the

Table 3. Average Percent Normalized Difference in the Game AI System vs. Opponent Scores (with average Scores in parentheses) with Statistical Significance

Opponent	CB-gda – Map 1	CB-gda - Map 2
Dom 1 Hugger	80.8% (20003 vs. 3834)	78.5% (20003 vs. 4298)
	81.2% (20001 vs. 3756)	78.0% (20000 vs. 4396)
	80.7% (20001 vs. 3857)	77.9% (20003 vs. 4424)
	81.6% (20002 vs. 3685)	77.9% (20000 vs. 4438)
	81.0% (20003 vs. 3802)	78.0% (20000 vs. 4382)
Significance	3.78E-11	1.92E-11
First Half of Dom Points	46.0% (20000 vs. 10781)	53.1% (20000 vs. 9375)
	45.8% (20001 vs. 10836)	56.7% (20002 vs. 8660)
	44.9% (20001 vs. 11021)	54.6% (20002 vs. 9089)
	46.1% (20000 vs. 10786)	52.0% (20001 vs. 9603)
	43.4% (20001 vs. 11322)	53.7% (20001 vs. 9254)
Significance	4.98E-08	1.38E-07
Second Half of Dom Points	45.6% (20002 vs. 10889)	60.6% (20000 vs. 7884)
	47.2% (20002 vs. 10560)	61.7% (20000 vs. 7657)
	44.1% (20001 vs. 11188)	61.7% (20000 vs. 7651)
	45.1% (20000 vs. 10987)	61.0% (20001 vs. 7797)
	45.8% (20000 vs. 10849)	60.8% (20002 vs. 7848)
Significance	4.78E-08	7.19E-10
Each Agent to One Dom	46.1% (20001 vs. 10788)	54.9% (20002 vs. 9019)
	46.2% (20000 vs. 10762)	53.7% (20002 vs. 9252)
	44.7% (20002 vs. 11064)	56.8% (20001 vs. 8642)
	44.6% (20000 vs. 11077)	55.4% (20000 vs. 8910)
	47.6% (20002 vs. 10481)	57.7% (20002 vs. 8469)
Significance	6.34E-08	7.08E-08
Greedy Distance	6.4% (20001 vs. 18725)	95.6% (20003 vs. 883)
	8.3% (20001 vs. 18342)	92.7% (20002 vs. 1453)
	5.0% (20000 vs. 18999)	64.6% (20004 vs. 7086)
	9.0% (20001 vs. 18157)	94.9% (20004 vs. 1023)
	12.7% (20001 vs. 17451)	98.0% (20004 vs. 404)
Significance	1.64E-03	6.80E-05
Smart Opportunistic	4.5% (20000 vs. 19102)	13.4% (20001 vs. 17318)
	11.5% (20000 vs. 17693)	13.9% (20001 vs. 17220)
	11.5% (20000 vs. 17693)	1.0% (20001 vs. 19799)
	10.6% (20000 vs. 17878)	10.7% (20002 vs. 17858)
	13.4% (20009 vs. 17333)	12.0% (20003 vs. 17594)
Significance	1.23E-03	1.28E-03

same experiment with a second map and obtained results consistent with the ones presented in Table 2 except for the results against *Greedy*, for which we obtained inconclusive results due to some path-finding issues.

In more detail, we report the results of additional tests here designed to determine whether the performance differences between CB-gda and the opponent team strategies are statistically significant. Table 3 displays the results of playing 10 games over two maps (5 games per map) against the hard-coded opponents. We tested the difference in score between the opponents using the Student's t-test. For the significance value p of each opponent, the constraint $p < 0.05$ holds. Hence, the score difference is statistically significant.

Table 4. Average Percent Normalized Difference for the Dynamic Game AI Systems vs. CB-gda Scores (with average scores in parentheses)

Opponent Team (controls enemies)	**Game AI System** (controls friendly forces)
	CB-gda's Performance
HTNbots	**8.1%** (20,000 vs. 18,379)
GDA-HTNbots	**23.9%** (20,000 vs. 15,215)

We also ran games in which the two dynamic opponents (i.e., HTNbots and GDA-HTNbots) competed directly against CB-gda using the same setup as reported for generating Table 2. As shown in Table 4, CB-gda easily outperformed the other two dynamic opponents. Again, we repeated this study with a second map and obtained results consistent with the ones we present in Table 4.

8 Discussion and Future Work

Looking first at Table 2, CB-gda outperformed GDA-HTNbots, alleviating some of the weaknesses that the latter exhibited. Specifically, against the easier and medium difficulty-level opponents (the first 4 in Table 2), HTNbots performed better than GDA-HTNbots (i.e., GDA-HTNbots outperformed those easy opponents but HTNbots did so by a wider margin). The reason for this is that the rule-based GDA strategy didn't recognize that HTNbots had already created an immediate winning strategy; it should have not suggested alternative goals. CB-gda still suffers from this problem; it suggests new goals even though the case-based agent is winning from the outset. However, CB-gda's performance is very similar to HTNbots's performance and only against the third opponent (2nd Half Of Dom Points) is HTNbots's performance observably better. Against the most difficult opponents (the final two in Table 2), GDA-HTNbots outperformed HTNbots, which demonstrated the potential utility of the GDA framework. However, GDA-HTNbots was still beaten by Smart Opportunistic (in contrast, HTNbots did much worse), and it managed to draw against Greedy (in contrast, HTNbots lost). In contrast, CB-gda clearly outperforms these two opponents and is the only dynamic game AI system to beat the Smart Opportunistic team. Comparing CB-gda to GDA-HTNbots, CB-gda outperformed HTNbots-GDA on all but one opponent, Dom 1 Hugger, and against that opponent the two agents recorded similar score percentages. From Table 4 we observe that CB-gda outperforms both HTNbots and GDA-HTNbots.

One reason for these good results is that CB-gda's cases were manually populated in PCB and MCB by an expert DOM player. Therefore, they are high in quality. In our future work we want to explore how to automatically acquire the cases in PCB and MCB. Cases in the PCB are triples of the form (s_c, g_c, e_c, pl); these could be automatically captured in the following manner. If the actions in the domain are defined as STRIPS operators (an action is a grounded instance of an operator), then e_c can be automatically generated by using the operators to find the sequence of actions that achieves g_c from s_c (i.e., e_c is the observed state of the game after g_c is satisfied). This sequence of actions will form the plan pl. Cases in MCB have the form (m_c, g_c). These cases can be captured by observing an agent playing the game. The current

state s_c can be observed from the game, and the expectation e_c can be obtained from PCB. Thus, it is possible to compute their mismatch m_c automatically.

We plan to investigate the use of reinforcement learning to learn which goal is the best choice instead of a manually-coded case base. Learning the cases could enable the system to learn in increments, which would allow it to address the problem of dynamic planning conditions. We also plan to assess the utility of GDA using a richer representation of the state. As explained earlier, states are currently represented as *n*-tuples that denote the owner of each domination point. Thus, the total number of states is $(t+1)^d$. In our maps d=4 domination points and there were only two opponents (i.e., $t = 2$), which translates into only 81 possible states. For this reason, our similarity relations were reduced to equality comparisons. In the future we plan to include other kinds of information in the current state to increase the granularity of the agent's choices, which will result in a larger state space. For example, if we include information about the locations of CB-gda's own bots, and the size of the map is $n \times m$, then the state space will increase to $(t+1)^d \times (n \times m)^b$. In a map where $n = m = 70$ and the number of bots on CB-gda's team is 3, then the state space will increase from size 81 to 81×4900^6 states. This will require using some other form of state abstraction, because otherwise the size of PCB would be prohibitively long.

We plan to use continuous environment variables and provide the system represent and reason about these variables. The explanation generator aspect of the agent could be expanded to dynamically derive explanations via a comprehensive reasoning mechanism. Also, we would like to incorporate the reasoning that some discrepancies do not require goal formulation.

9 Summary

We presented a case-based approach for goal driven autonomy (GDA), a method for reasoning about goals that was recently introduced to address the limitation of classical AI planners, which assume goals are static (i.e., they never change), and cannot reason about nor self-select their goals. In a nutshell, our solution involves maintaining a case base that maps goals to expectations given a certain state (the planning case base - PCB) and a case base that maps mismatches to new goals (the mismatch-goal case base - MCB). We introduced an algorithm that implements the GDA cycle and uses these case bases to generate new goals dynamically. In tests on playing Domination (DOM) games, the resulting system (CB-gda) outperformed a rule-based variant of GDA (GDA-HTNbots) and a pure replanning agent (HTNbots) against the most difficult manually-created DOM opponents and performed similarly versus the easier ones. In further testing, we found that CB-gda significantly outperformed each of these manually-created DOM opponents. Finally, in direct matches versus GDA-HTNbots and HTNbots, CB-gda outperformed both algorithms.

Acknowledgements

This work was sponsored by DARPA/IPTO and NSF (#0642882). Thanks to PM Michael Cox for providing motivation and technical direction. The views, opinions,

and findings contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of DARPA or the DoD.

References

1. von Moltke, H.K.B.G.: Militarische werke. In: Hughes, D.J. (ed.) Moltke on the art of war: Selected writings. Presidio Press, Novato (1993)
2. Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D., Washington, R.: Incremental contingency planning. In: Pistore, M., Geffner, H., Smith, D. (eds.) Planning under Uncertainty and Incomplete Information: Papers from the ICAPS Workshop, Trento, Italy (2003)
3. Goldman, R., Boddy, M.: Expressive planning and explicit knowledge. In: Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, pp. 110–117. AAAI Press, Edinburgh (1996)
4. Muñoz-Avila, H., Aha, D.W., Jaidee, U., Klenk, M., Molineaux, M.: Applying goal directed autonomy to a team shooter game. To appear in Proceedings of the Twenty-Third Florida Artificial Intelligence Research Society Conference. AAAI Press, Daytona Beach (2010)
5. Molineaux, M., Klenk, M., Aha, D.W.: Goal-driven autonomy in a Navy strategy simulation. To appear in Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI Press, Atlanta (2010)
6. López de Mantaras, R., McSherry, D., Bridge, D.G., Leake, D.B., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K.D., Keane, M., Aamodt, A., Watson, I.D.: Retrieval, reuse, revision and retention in case-based reasoning. Knowledge Engineering Review 20(3), 215–240 (2005)
7. Auslander, B., Lee-Urban, S., Hogg, C., Munoz-Avila, H.: Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 59–73. Springer, Heidelberg (2008)
8. Nau, D.S.: Current trends in automated planning. AI Magazine 28(4), 43–58 (2007)
9. Nau, D., Cao, Y., Lotem, A., Muñoz-Avila, H.: SHOP: Simple hierarchical ordered planner. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pp. 968–973. AAAI Press, Stockholm (1999)
10. Cox, M.T.: Perpetual self-aware cognitive agents. AI Magazine 28(1), 32–45 (2007)
11. Fox, M., Gerevini, A., Long, D., Serina, I.: Plan stability: Replanning versus plan repair. In: Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, pp. 212–221. AAAI Press, Cumbria (2006)
12. Warfield, I., Hogg, C., Lee-Urban, S., Munoz-Avila, H.: Adaptation of hierarchical task network plans. In: Proceedings of the Twentieth Flairs International Conference, pp. 429–434. AAAI Press, Key West (2007)
13. Hoang, H., Lee-Urban, S., Muñoz-Avila, H.: Hierarchical plan representations for encoding strategic game AI. In: Proceedings of the First Conference on Artificial Intelligence and Interactive Digital Entertainment, pp. 63–68. AAAI Press, Marina del Ray (2005)
14. Ayan, N.F., Kuter, U., Yaman, F., Hotride, G. R.: Hierarchical ordered task replanning in dynamic environments. In: Ingrand, F., Rajan, K. (eds.) Planning and Plan Execution for Real-World Systems – Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop, Providence, RI (2007)

15. Myers, K.L.: CPEF: A continuous planning and execution framework. AI Magazine 20(4), 63–69 (1999)
16. Ghallab, M., Nau, D.S., Traverso, P.: Automated planning: Theory and practice. Morgan Kaufmann, San Mateo (2004)
17. van den Briel, M., Sanchez Nigenda, R., Do, M.B., Kambhampati, S.: Effective approaches for partial satisfaction (over-subscription) planning. In: Proceedings of the Nineteenth National Conference on Artificial Intelligence, pp. 562–569. AAAI Press, San Jose (2004)
18. Coddington, A.M., Luck, M.: Towards motivation-based plan evaluation. In: Proceedings of the Sixteenth International FLAIRS Conference, pp. 298–302. AAAI Press, Miami Beach (2003)
19. Meneguzzi, F.R., Luck, M.: Motivations as an abstraction of meta-level reasoning. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS (LNAI), vol. 4696, pp. 204–214. Springer, Heidelberg (2007)
20. Bergmann, R.: Experience management: Foundations, development methodology, and internet-based applications. Springer, New York (2002)

Case Retrieval with Combined Adaptability and Similarity Criteria: Application to Case Retrieval Nets

Nabila Nouaouria and Mounir Boukadoum

Department of Computer Science,
University of Quebec at Montreal
CP. 8888, Succ. Centre-ville
Montréal, QC H3C3P8 Canada
phone. (514) 987-3000 p./ext. 4565#
nouaouria.nabila@gmail.com, boukadoum.mounir@uqam.ca
www.info2.uqam.ca/~boukadou

Abstract. In Case Based Reasoning (CBR), case retrieval is generally guided by similarity. However, the most similar case may not be the easiest one to adapt, and it may be helpful to also use an adaptability criterion to guide the retrieval process. The goal of this paper is twofold: First, it proposes a method of case retrieval that relies simultaneously on similarity and adaptation costs. Then, it illustrates its use by integrating adaptation cost into the Case Retrieval Net (CRN) memory model, a similarity-based case retrieval system. The described retrieval framework optimizes case reuse early in the inference cycle, without incurring the full cost of an adaptation step. Our results on a case study reveal that the proposed approach yields better recall accuracy than CRN's similarity only-based retrieval while having similar computational complexity.

Keywords: Case Based Reasoning, Case Retrieval, Adaptation-Guided Retrieval, Case Retrieval Nets.

1 Introduction

In CBR, the case recall or retrieval process is generally guided by the similarity between old cases stored in memory and the problem at hand. However, the most similar case may not be the easiest one to adapt [1,2], and while similarity and other factors do influence the retrieval performance of a CBR system, often, what matters most is the ease with which an old case can be reused to solve the target problem. Thus, effective retrieval is not simply a matter of finding cases that are similar, but cases that are usefully similar. In this perspective, similarity appears to have been used as a proxy for solution utility. This brings forth the question of whether similarity is always an adequate proxy for solution utility [3]. In situations where it is not, it becomes necessary to bring other forms of knowledge to bear on the retrieval task.

In [4], Smyth & Keane question the assumption that the most similar case is the one easiest to adapt. They argue that, sometimes, the most similar case is difficult or impossible to adapt. This can happen, for example, when the adaptation knowledge is incomplete as in the weak-theory domains commonly targeted by CBR. To address

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 242–256, 2010.

the issue, Smyth & Keane introduce Adaptation-Guided Retrieval (AGR), in which the case adaptation requirements are explicitly assessed during retrieval by means of domain-specific adaptation knowledge. In this work, we explore the AGR principle from a new standpoint with the aim to define a knowledge transfer cost that completely captures the reuse semantics in CBR. Fundamentally, we argue that in CBR, the best case to retrieve and reuse is the one most similar to the target in its problem description while having the lowest adaptation cost for the solution it offers.

As a second objective, we illustrate the proposed framework's use by integrating it in a non-classical case retrieval memory model, the Case Retrieval Net (CRN) associative memory [5]. CRN was conceived to be a high performance case retrieval system based on similarity. Our goal is to improve CRN performance in terms of retrieved case utility by including an adaptability criterion, in addition to similarity, in the retrieval process. Our approach has the following features: 1) Look-ahead evaluation of the adaptation stage without incurring the full cost of adaptation; 2) Rapid propagation of the ensuing knowledge throughout the network.

The paper thus presents AECRN (Adaptability Enhanced Case Retrieval Net). AECRN's quick-look vision of the adaptation stage focuses on transformational adaptation. It uses adaptability knowledge that is not in the case base but is assessed according to the reuse context. Also, only the relevant parts of the source case are considered for adaptation in order to satisfy the constraints of the target case.

In the balance of this paper, the next section provides a survey of related works. Section 3 skims over CBR and points out the limitations of using similarity alone during the recall process. It proposes a reuse-based retrieval framework. In Section 4, we describe our memory model and follow up with an illustration and experimentation in Section 5. A discussion ends the paper in Section 6.

2 Related Works

Smyth & Keane created "Déjà vu", the reference system for AGR [15]. Traditionally, adaptation knowledge in CBR takes the form of collections of solution transformation rules called action knowledge. "Déjà Vu" goes one step further by providing so called capability knowledge alongside the transformation rules. This capability knowledge characterizes the type and function of a particular set of transformation rules and allows the system to predict the potential of various forms of adaptation at an early stage during retrieval. The methodology used to find the recall cases does not exploit their resemblance to the target, and only the criterion of adaptability guides the recall process. In "Déjà vu", the recall process is carried out in 4 steps. 1) Feature Promotion (The target specification is analyzed to identify relevant features with respect to adaptation); 2) Candidate Selection (A case that shares no active specialists with the target cannot be properly adapted and therefore is eliminated from further consideration); 3) Local Adaptability Assessment (Fully adaptable cases are identified during this stage, and all other candidates are removed from further consideration); 4: Global Adaptability Ranking (Adaptation strategies are used to recognize adaptation conflicts in the locally adaptable cases. The global adaptation cost of each case can then be computed by combining the costs of all relevant adaptation specialists and strategies. Finally, the adaptable cases are rank-ordered according to increasing adaptation cost).

Another early work in the area was by Leake & al. [18]. It addresses the adaptation effort and the impact of traditional semantic similarity measures on adaptation. Leake & al. take into account the adaptation effort during the retrieval time in order to facilitate the adaptation step. This consideration is embodied by including an "adaptation cost" to extend similarity. Assessment of the similarity between the source cases and target problem is conducted in two stages: A first step of similarity guided retrieval is followed by scheduling the retrieved cases according to an adaptation cost.

In [21], Tonidandel & Rillo present an adaptation guided similarity metric based on estimating the number of actions between states, called ADG (Action Distance Guided). It is essentially applied to planning tasks and is determined by using a heuristic calculation that provides an estimate of the appropriate distance between states for similarity measures. The proposed similarity metric is suitable to be applied in conjunction with any method that reduces the search space of cases. The problem perception in [21] is in terms of task specific actions to be performed to transit between states.

In [22], Diaz-Agudo & al. propose another approach to AGR. They use Formal Concept Analysis (FCA) as an inductive technique to elicit embedded knowledge in a case library. The implicit dependency knowledge contained in the case base is captured during the FCA process in the form of dependency rules between the attributes describing the cases. A substitution-based adaptation process is proposed that profits from these dependency rules since substituting an attribute may require substituting dependent attributes. The dependent rules guide an interactive query formulation process that favors retrieving cases where successful adaptation can be accomplished. Dependencies are studied between attributes and a concept lattice structure is computed as a result. This structure groups together cases sharing the same attribute values.

In [23], Haouchine & al. consider a technical diagnostic study. A specific formalization of the diagnosis case, based on two descriptor types is proposed. Thereafter, a similarity measure is developed based on local similarities. The global similarity measure is obtained by aggregation of these functions from the whole set of descriptors. From this measure, a set of cases can be selected. Secondarily, an adaptation measure is computed. It takes into account the source cases descriptors which are different from the target case and will be linked to the class and to the functional mode compared to the solution descriptors. The adaptation step consists of determining the dependency relations between the problem and the solution. These dependency relations between qualitative data are not trivial to implement. Three relationships are defined (high, low and none) and exploited to adapt a retrieval case. The proposed approach is demonstrated on a pallets transfer system as an industrial application. Its feasibility is studied on 20 generic cases.

All of the above models accomplish case retrieval in at least two steps which can typified as a pre selection step based on similarity and a final selection based on AGR. Thus the search space is first reduced by a similarity criterion and then by an adaptability criterion. However, the first reduction may introduce a bias that will hinder the usefulness of the second, and using the second alone as in Déja Vu may not be sufficient for optimal recall. To prevent this, the similarity and adaptability criteria should be combined and applied *simultaneously*, within the same retrieval stage. We

propose a general framework that implements this idea, where the retrieval step exploits a composite reuse equation (defined in the next section) in a look-ahead manner. As a result, the adaptation knowledge is exploited very early in the CBR cycle.

3 Case Based Reasoning and the Recall Problem

It is useful to consider CBR as operating within two distinct symbol spaces, a specification or problem space, and a solution space [10, 11, 12, 13, 14]. Retrieval usually operates in the specification space, relying on pairings between the specification features of the target problem and the specification features of cases. On the other hand, adaptation operates in the solution space, using transformational pairings between necessary parts of the target solution and available parts of a retrieved case solution. From this perspective, retrieval is a search in the specification space for the right set of matches between target and case specification features, and adaptation is a search in the solution space for the right set of transformations from a case solution to the target solution.

The conventional approach to case reuse separates the retrieval and adaptation stages, with the assumption that specification space similarity can be used to predict the usefulness of a case. By doing so, it ignores the link(s) between the specification space and the solution space, with the consequence of disregarding the cohesion between retrieval and adaptation. The similarity assumption often relies on "similarity knowledge" that is encoded in a similarity measure, generally implemented with a geometric distance [16]. However, in many application domains, distances may not be sufficient to obtain reasonable results (with the possible exception of non classical measures such as fuzzy sets). In fact, a simple geometric distance between the problem descriptions can be quite a bad approximation of the solution quality for reuse. An illustration of this problem is provided in [16].

The problems of neglecting the link(s) between problem and solution spaces, and of ignoring the impact of classical similarity measures on solution quality, have a direct incidence on the cohesion between the retrieval and adaptation stages. To address this issue, our proposed method of case reuse uses a new mechanism of knowledge transfer from the source case to the target case. Figure 1 summarizes it. The inference consists of trying to match the target problem with the source problem and, based on similarities and dissimilarities issued from the match, deciding what to copy from the old solution (solid lines) and what to adapt to satisfy the new problem (dashed lines). For a source case s and a target case t, we define a cost for the knowledge transfer process as follows:

$$\begin{aligned} Transfer\_Cost(s, t) = \; & Similarity(s, t) * Copy\_cost(s, t) + \\ & Dissimilarity(s, t) * Adaptation\_cost(s, t) \end{aligned} \tag{1}$$

The similarity between the source and target problems is computed by a metric or a heuristic depending on the nature of the problem descriptors; dissimilarity is the dual notion. Copy_cost is the cost of copying the solution unchanged; it is always incurred since the source is always needed, whether to start adaptation or to be copied as such if no dissimilarities exist between the source and target cases (in other words, Adaptation_cost is actually the sum of a copy part and an adaptation part).

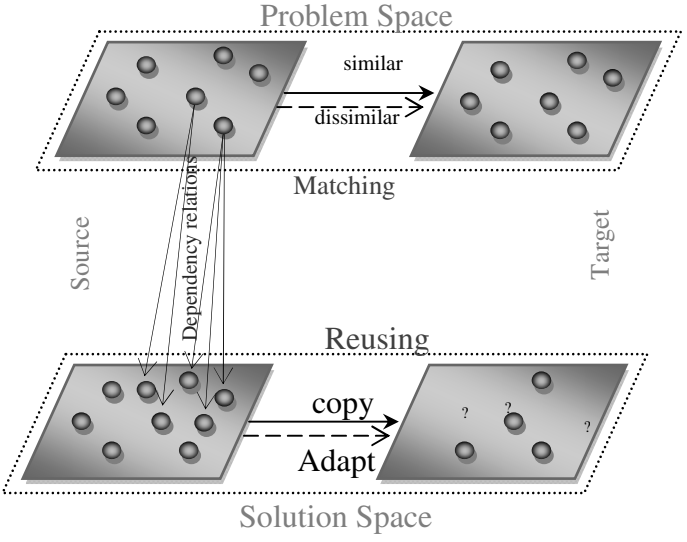


Fig. 1. Dependency relations situated in the analogy square

When the query (target problem) is identical to the source problem, there are no dissimilarities between the two problem descriptors and case reuse is a copying task. As a result, there is no adaptation and only the similarity term acts in equation (1). At the other extreme, when the source problem presents dissimilarities with the target problem on all descriptors, case reuse become a copying task followed by adaptation, and the adaptation cost must be computed for all the solution descriptors that have links to the problem descriptors (this is illustrated with vertical arrows in Figure 1); then, the adaptation effort will be maximal since only the dissimilarity term acts in equation (1). For cases that fall between the two extremes, the cost of adapting an old solution depends on the number of dependency links between dissimilar parts of the source and target problems.

If we set aside the copy cost when adapting, since it is also present for perfectly similar source and target cases, the adaptation cost for the found dissimilarities may be assessed by a function μ( ) that evaluates the effort needed to adapt a solution. It is computed by summing up the costs of adapting the solution parts that are linked to dissimilar problem parts via a dependency relation, and its value depends on the number of adaptation operators that are required for each part. For a case c we have:

$$\mu(c) = \sum_i \mu(p_i) \tag{2}$$

where p_i stands for solution part i and $\mu(p_i)$ is the number of operators needed to adapt the related dissimilar problem parts to it.

Illustration

The following example is extracted from the PC-CONFIG system to be presented in section 4. PC-CONFIG is an "intelligent" sales support tool for computers where a

problem description is expressed in terms of user preferences about word processing, multimedia, game, video-montage, image processing, programming, music, internet, brand and price. The user specifies the desired applications he/she wants to use via a rating system that indicates his/her preferences and PC-CONFIG processes the entered information to recommend a PC configuration.

From a cognitive analysis of the domain, an expert would use adaptation heuristics similar to the following two examples:

1. *IF (Target.Multimedia > 7) AND (Source.Multimedia < 3)*
   THEN modify (graphical card)
2. *IF(Target.WordProcessing = 0) AND (Source.WordProcessing > 7)*
   THEN suppress (printer), Modify (screen)

Thus, the dependency relationship between the problem part and the solution part is established by a heuristic approach based on adaptation rules of the form:

IF cond (DescPbm)
THEN {modify/add/suppress} (DescSol)

In this way, every solution descriptor (DescSol) is dependent on one or more problem descriptor (DescPbm). Table 1 lists some of the established dependencies for PC-CONFIG.

Table 1. Partial Problem/solution dependencies in PC-CONFIG

Problem / *Solution*	*Multimedia*	*Word proc*	*game*	*Program-ming*	*Image proc*	*music*	*Internet*	*Video*
graphic Card	√		√		√			√
CD/DVD	√					√		√
HD				√				
Printer		√		√				
Burner	√			√				
CPU brand			√	√				
CPU speed			√	√	√			
RAM size			√	√	√			
RAM type			√	√				
Scanner					√			
Modem							√	
Screen size		√						
Mouse		√						
Audio card						√		√
Price	√	√	√	√	√	√	√	√

To determine the cost of the adaptation step, we first need to quantify the dependencies: for example, to which degree the game constraint influences processor power; this information is easily derived from the pre-established dependency table (Table 1). The following illustrates two situations with different adaptation costs:

Situation 1

The source problem presents dissimilarities with the target problem on the Game and Programming descriptors. So, the partial adaptation costs are:

$\mu(Game) = 6$ (number of checked lines in column Game of table.1)

$\mu(Programming) = 8$ (number of checked lines in column Programming of table.1)

and the total cost of adaptation is: $\mu(source\ case) = 14$.

Situation 2:

The source problem presents dissimilarities with the target problem on Music and Programming descriptors. The partial adaptation costs are:

$\mu(Music) = 3$

$\mu(Programming) = 8$

and the total adaptation cost is: $\mu(source\ case) = 11$

Even though in both situations the difference between the source and target problem is about two descriptors (same similarity), the source case in situation 2 is easier to adapt than the one in situation 1, the adaptation effort for it (adaptation cost = 11) is less costly than for the first one (adaptation cost = 14).

Since the adaptation cost of the old solution parts relative to dissimilar problem parts is the total number of the adaptation operators to apply, it can be determined ahead of any adaptation, during the retrieval phase. Section 4 will describe how the adaptability cost is incorporated into an actual memory model.

4 Adaptability Enhanced Case Retrieval Nets

All memory models that use top-down search share two desirable features [5]: Data structuring by regrouping related objects and efficient retrieval by traditional tree search algorithms. Unfortunately they also have potential limitations, of which memory traversal by answering an ordered sequence of internal node questions (in the case of incomplete information, this could lead to erroneous paths) and difficult access to neighboring clusters with similar cases when reaching a cluster at a tree leaf.

CRN is a memory model that uses spreading activation and information completion; it was introduced by Lenz & al. [5] in the context of a virtual travel agency. The CRN formalism offers significant retrieval speedup in comparison to linear search, and Lenz & al. [5] have successfully deployed it over case bases containing as many as 200 000 records. CRNs have been used in various fields, of which spam filtering [6] and text retrieval in large volumes of directory records [7], and their computational power is well established. However, as mentioned in [8], CRNs have a weakness in that they lack direct support for integrating adaptation knowledge during the retrieval phase. As their expanded name indicates, CRNs serve to retrieve cases; what the cases are used for is left to another system, or the user, to manage.

The foundation of CRN is inspired from neural network techniques and associative memory models. It has remarkable features [8], of which a case recall process that does not browse a path in a tree. In CRN, retrieval is made in a reconstructive way by recovering information entities and gradually rebuilding the case. The most

fundamental items in the context of CRNs are information entities (IEs), which represent knowledge items with the lowest level of granularity [8]. A case is a set of IEs and a case memory is a net of nodes corresponding to the IEs observed in the domain and additional nodes denoting the particular case. The IE nodes are connected by similarity arcs weighted by a function σ, and a case node is reachable from its constituting IE nodes via relevance arcs enabled by a binary function ρ. Different degrees of similarity and relevance are expressed by varying the values of σ and ρ for the different arcs [8].

Given this structure, a case search is performed in three steps:

1. Activation of the IE nodes given by the query (the case to resolve).
2. Propagation of the activation according to similarity through the net of IEs, in order to explore the neighborhood of the query,
3. Collection of the achieved activation in the associated case nodes.

The graphical description is given by a graph composed of the union E∪C, where E is a set of IEs and C a set of case nodes, with a similarity arc between each pair of IEs e_i and e_j labelled $\sigma(e_i,e_j)$ and a relevance arc from each IE e to a case node c labelled $\rho(e,c)$.

Activation of a CRN N is accomplished with a function $\alpha : E \cup C \rightarrow \Re$ such that $\alpha(n)$ expresses the importance of IE n activation - or alternatively the relevance of case node n activation - for the actual problem (the query). The case retrieval result for a given query activation α_0 is the preference ordering of cases according to decreasing activations, $\alpha_2(c)$, of case nodes $c \in C$ (as stated by theorem 5.1 in [8]).

As discussed previously, while the CRN model presents definite advantages over hierarchical memory models, the retrieval it offers does not always lead to feasible or easy adaptation step, as it is only guided by similarity. AECRN is an extension of CRN that addresses this problem by adding the criterion of adaptability.

Formally, we start from the definition of CRN given in [8] and modify it to include a knowledge transfer cost function as shown below. Notice that Theorem 5.1 in [8] remains valid, but since we transport a cost and not only a similarity, the preference order is by increasing and not decreasing collected values:

Definition of an AECRN:

An Adaptation Enhanced Case Retrieval Net (AECRN) is a structure $N = [E, C, \sigma, \rho, \beta, \Pi]$ where E is a finite set of IE nodes, C is a finite set of case nodes, σ is a similarity function $\sigma : E \times E \rightarrow \Re$ that provides the similarity between pairs of IE nodes, ρ is a relevance function $\rho : E \times C \rightarrow \Re$ that describes the relevance of an IE node to a case node, β is a knowledge transfer cost function $\beta : \Re \times \Re \rightarrow \Re$ such that $\beta(\alpha_t(e),\mu(e))$ considers both similarity and adaptability for IE node e, as defined in equation (1), and Π is a set of propagation functions $\pi_n : \Re \times E \rightarrow \Re$ such that, for each node $n \in E \cup C$, it transports similarity between IE nodes and relevance between IE nodes and case nodes

In the preceding, E, C, σ, ρ are as in the definition of CRN, β is added and Π is modified accordingly.

For pragmatic reasons, $\beta(\alpha_t(e),\mu(e))$ uses a simplified expression of equation (1), where the copy cost is assigned a value of 1. It is thus expressed as:

$$\beta(\alpha_t(e),\mu(e))= \alpha_t(e) + (1- \alpha_t(e))(\mu(e)+1) \quad (3)$$

Where, as stated previously, $\alpha_t(e)$ provides the normalized similarity of IE e with the request, propagated via the net, 1- $\alpha_t(e)$ provides its dissimilarity, and $\mu(e)$ is the adaptation cost exclusive of the associated copy cost.

Activation of an AECRN

The activation of an AECRN N is done by a function $\alpha: E \cup C \rightarrow \Re$ that is similar to that of a CRN.

Propagation process in an AECRN

Consider an AECRN N with E = $\{e_1,\ldots,e_s\}$ and let α_t denotes the activation function at instant t. The activation of IE node $e \in$ E, through similarity arcs, at instant t+1, relatively to its neighborhood is equal to $\{e_1,\ldots,e_s\}$ -$\{e\}$. On the other hand, the activation of case node $c \in$ C, via relevance arcs, at time t+1, is given by:

$$\alpha_{t+1}(c) = \pi_c(\rho(e_1,c)\beta(\alpha_t(e_1) , \mu(e_1)) ,\ldots,\rho(e_s,c)\beta(\alpha_t(e_s), \mu(e_s))) \quad (4)$$

When posing a query, the initial activation of the IE nodes is set in the same way as for CRN:

$$\alpha_0(e) = \begin{cases} 1 : & \text{if IE node } e \text{ is in the query.} \\ 0 : & \text{otherwise} \end{cases} \quad (5)$$

It is seen from the preceding that the retrieval process by propagation of activations is similar to that of CRN, but the relevance propagation is accomplished differently in the third step. Algorithm 1 summarizes the new procedure.

Step 1 –Initial Activation:

α_0 *is set for all the IE nodes associated with the query;*

Step 2 – Similarity Propagation:

α_0 *is propagated to all IEs* $e_i \in E$ *in the net :*

$\alpha_1(e) = \pi_e(\sigma(e_1,e)\ \alpha_0(e_1) ,\ldots,\ \sigma(e_s,e)\ \alpha_0(e_s))$

Step 3 – Relevance Propagation:

The result of step 2 is propagated to all case nodes $c \in C$*:*

$\alpha_2(c) = \pi_c(\rho(e_1,c)\beta(\alpha_1(e_1),\mu(e_1)) ,\ldots,\ \rho(e_s,c)\beta(\alpha_1(e_s),\ \mu(e_s)))$

Algorithm 1. Retrieval process with AECRN

To guide the retrieval process in the network, $\mu(e)$ should not statically figure at the IE nodes of a case but must be locally appreciated at the IE nodes describing the problem being propagated. $\mu(e)$ is spread in parallel with similarity. It follows that a node c will collect not only a measure of similarity between a source case c and the query, but also an evaluation of the adaptability of the case with regard to the problem to resolve. Since the transportation uses the same mechanism as similarity, the computational complexity of the activation and propagation algorithm is not substantially affected and the performance of AECRN is similar to that of CRN.

5 Illustration, Experimental Results and Comments

In order to evaluate the performance of the new case retrieval approach and compare it to CRN, we applied it to a specific problem: A Personal Computer sales support system with the following features:

- Well structured cases with well defined dependency relations between problem descriptors and solution descriptors.
- Use of transformational adaptation.
- Ease of extracting adaptability knowledge.

These characteristics make the application adequate for verifying and validating our approach. PC-CONFIG was inspired from [17] and [19]. It is an "intelligent" sales support tool for computers. PC-CONFIG makes easier the task of choosing a personal computer for a range of users (novices, experts, special users, etc.) with different needs (see figure 2). In PC-CONFIG, the relationship between a user description and the recommended solution is assessed both in terms of similarity and adaptability measures. The similarity is computed as the dual value of distance between preferences. PC-CONFIG works on the basis of a knowledge base of components and a personalization of the suggested products.

As stated in the previous section, a problem description in PC-CONFIG is expressed in terms of user preferences about word processing, multimedia, game, video-montage, image processing, programming, music, internet, brand and price. The user specifies the applications he/she wants to use via a rating system that indicates his/her preferences. For example, a score of 10/10 for games means that the configuration should be oriented toward games; a score of 4/10 for programming signifies a relatively low interest for this activity, etc.

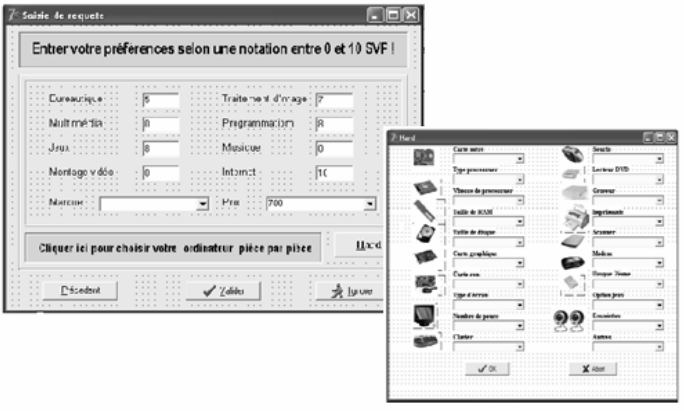


Fig. 2. Problem description in PC-CONFIG's user interface. An expert can also directly supply a direct description by clicking on button "Hard" (in this case PC-CONFIG is used just like any sales catalogue and no adaptation is performed except for price).

Table 1 synthesizes the dependencies existing between the problem and solution parts from a cognitive analysis of the domain. In the example provided in Section 3, we illustrated two cases with equal numbers of dissimilarities but different adaptation costs. The two partial adaptation costs in each situation should be transported via the relevance arcs ρ until the collector node *c* where their sum will be combined via the β function to the similarity value (step 3 of the AECRN algorithm). For situation 1 in the example, where the source problem presented dissimilarities with the target problem on the Game and Programming descriptors, with $\mu(Game) = 6$ and $\mu(Programming) = 8$, this leads to:

$$\alpha_2(c) = \pi_c(\rho(e_i,c)\ \beta(\alpha_1(e_i), 6),\ldots, \rho(e_j,c)\ \beta(\alpha_1(e_j), 8), \ldots)$$

and for Situation 2 where the dissimilarities were on the Music and Programming descriptors, with μ(Music) = 3 and μ(Programming) = 8, this lead:

$$\alpha_2(c) = \pi_c(\rho(e_i,c)\ \beta(\alpha_1(e_i), 3),\ldots, \rho(e_j,c)\ \beta(\alpha_1(e_j), 8), \ldots)$$

The final decision regarding the case to choose will rest on which of the two has the lower value for α_2.

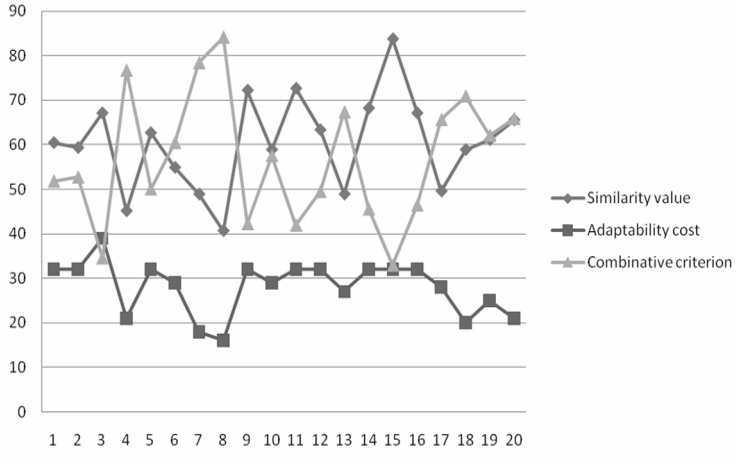


Fig. 3. Adaptability versus Similarity. (the y axis expresses a normalized similarity value, adaptability cost, or combination of both on a percentage scale).The x axis represents retrieved cases. Case #8 is less costly to adapt while not the most similar to the target problem. Case #15 is the most similar to the target but is costlier to adapt.

Figure 3 illustrates the difference between using similarity and adaptability. It shows how the most similar case is not necessarily the easiest one to adapt and vice-versa. The similarity curve computed in CRN style shows that case #15 is the most interesting to choose. But when considering the adaptability cost, it is case #8 that is the least costly to adapt, although not the most similar. The mixed criterion curve computed by AECRN allows selection of similar cases only when their adaptability cost is low; when it is high, they may not be ranked high even if their similarity rate is.

A formal evaluation of the retrieval effort's computational complexity is difficult to perform given the number of parameters to consider; and the choice of a particular similarity measure influences performance. Both in the case of our composite similarity-adaptability measure or in the case of only similarity measure, the factors influencing the effort required for the retrieval process are inherent to the net complexity and include the following [8]: 1) the size of the query: The more IEs have to be activated initially, the more has to be propagated through the net; 2) the degree of connectivity between IEs: The more non-zero similarity arcs exist, the more effort is required during similarity propagation; 3) the specificity of the IEs: The more cases are associated to the IEs, the more effort is required during relevance propagation.

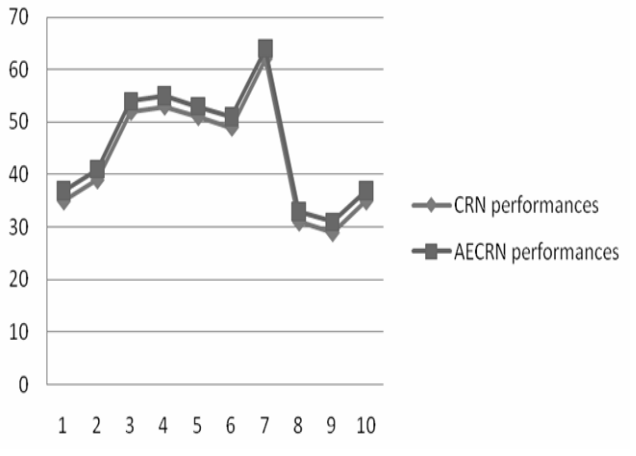


Fig. 4. Similar CRN and AECRN performances (expressed in terms of time on the y axis) for different requests (represented on x axis) in PC-CONFIG system

Still, to verify the claim that our approach does not add more complexity with the inclusion of the activation and propagation algorithm to CRN, we implemented two variants of PC-CONFIG. One based on CRN and the other on AECRN. As Figure 4 shows, our claim is corroborated; AECRN behaves like CRN with a little delay due to the adaptability factor. Since CRN performs 15% to 32% better than linear retrieval [8], for nearly the same time consuming effort, AECRN furnishes even better retrieval accuracy.

6 Discussion and Conclusion

The model presented in this paper is a general framework of application, even though we used transformational style adaptation and assumed the availability of knowledge about the dependency relations. Indeed, the two restrictions are linked. Transformational adaptation uses heuristics of adaptation describing how to modify the solution of the source case when source and target cases are different. By this,

heuristics describe the relation between the problem and the solution; in the case of derivational adaptation we did not have this knowledge.

When comparing our work to "Déjà vu" [15], we notice that both approaches evaluate the adaptability of a case via a quick estimation of the adaptation cost. However, in "Déjà vu", the cost is computed through analysis of the premises of applicable rules amongst adaptation heuristics, which can be a computation burden; in our approach, it is computed from readily available dependency relationships in the source description. In "Déjà vu", the methodology used to find the recall cases does not exploit their resemblance to the target, and only the criterion of adaptability guides the recall process; our approach uses both criteria. In "Déjà vu", the recall process is carried out in 4 steps, whereas in our approach, the two criteria are transported together, during the same time in the research phase.

The main difference between the approach proposed in [21] and our approach is that, the problem perception in [21] has to be in terms of task specific actions to be performed to transit between states. In our work, no such restriction is imposed. Also, the approach we propose guarantees the exploration and reduction of the search space with a double criterion of adaptability and similarity without using an auxiliary method in conjunction.

In [22], the approach to AGR relies on Formal Concepts Analysis (FCA). The major difference with our work is that [22] studies the dependencies between attributes to create a concept lattice structure that groups cases sharing the same attribute values together. In our work, dependencies are studied between problem descriptors and solution descriptors within the case to determine which part of the solution depends (so has to be adapted) on which part of the problem.

In summary, we described a novel methodology for CBR case retrieval that combines similarity with knowledge about the potential for adaptation of the retrieved case. This is implemented via a reuse equation (equation 1) that expresses the balance between the elements of reuse (copy and adapt) while considering the dual aspect of similarity and dissimilarity. To “mix” similarity and adaptability, in order to retrieve the most useful cases is inherent to equation 1, where we have the link between ‘copy’ and ‘adapt’, two cognitive tasks of CBR. Neglecting the similarity (as done in "Déjà vu") is neglecting the copy task (we ‘copy’ what is similar).

As seen in section 2 of the paper, considering both adaptability and similarity for case retrieval is well known. The novelty of our work is the traversal of search-space in one single step that considers the two criteria simultaneously, in contradistinction to other approaches that do it in two or more steps. The benefit of doing so is that no bias may be introduced as when having a pre-selection based on similarity (which induces a reduction of the search-space) and, then, a selection based on adaptability (performed on a part of the initial search-space).

Secondarily, the reuse equation was integrated in a memory model to implement the recall process. In doing so, we proposed an extension of CRN, a memory model for case retrieval based on similarity. The resulting AECRN model considers both the similarity and adaptability criteria, with a subsequent improvement in recall accuracy for nearly the same computational effort. The targeted extension lies not in CRN’s fundamental mechanisms but their utilization. In CRN we only transport and collect similarity knowledge. In AECRN we transport and collect both similarity and adaptability.

Multiple directions could be explored to extend this work; amongst them is the way to extract knowledge for use in computing the adaptability cost. At least three techniques can be exploited. The first one, presented in section 3, is the most direct since based on adaptation rules. It computes dependencies by scoring links between problem descriptors present in the premise and solution descriptors present in the conclusion of adaptation rules. Its advantages are that it doesn't need additional knowledge, it can be automated, and it is always applicable. On the other hand, it is highly dependent on adaptation heuristic completeness: We forget a rule, we forget a dependency. A second technique is related to data analysis tools and is essentially applicable for descriptors with numerical values. It has the advantage of being automatic and data-driven instead of rule-driven as the previous one. The third technique takes its roots from database analysis. The idea is that we can consider dependency relations as functional dependencies and use tools for constructing Functional Dependency Graph (FDG) like the CORDS tool [24].

Another possible extension of this work is to consider the recall problem from a cognitive point of view without biasing the search space by any structure. Indeed, in section 2, we exposed the problem of biasing the retrieval process by only considering the similarity criterion. But another bias, maybe more important, is introduced by the organization of the search itself (most of time hierarchically) as it imposes a specific perception of the description space. Suppose, then, that we have an associative memory that is addressable via a content-based indexing mechanism. The index computation could be ensured by an algorithm based on implementing equation (1), and no pre-established organization will be needed.

Acknowledgments. This work was possible thanks to the NSERC financial support.

References

1. Kolodner, J.L.: Judging which is the best case for a case-based reasoner. In: Proceedings of the Case-Based Reasoning Workshop, pp. 77–81. Morgan Kaufmann, San Mateo (1989)
2. Hammond, K.J.: Adaptation of Cases. In: Proceedings of DARPA Workshop on Case-Based Reasoning, Florida, pp. 88–89 (June 1989)
3. Lopez de Mantara, R., et al.: Retrieval, reuse, revision, and retention in case based Reasoning. The Knowledge Engineering Review 20(3), 215–240 (2006)
4. Smyth, B., Keane, M.T.: Retrieving Adaptable Cases. The role of adaptation knowledge in case retrieval. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) EWCBR 1993. LNCS (LNAI), vol. 837, pp. 209–220. Springer, Heidelberg (1994)
5. Lenz, M., Auriol, E., Manago, M.: Diagnosis and decision support. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 51–90. Springer, Heidelberg (1998)
6. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: A Case-based Technique for Tracking Concept Drift in Spam Filtering. In: Macintosh, A., Ellis, R., Allen, T. (eds.) Applications and Innovations in Intelligent Systems XII, Proceedings of AI, pp. 3–16. Springer, Heidelberg (2004)
7. Chakraborti, S., Lothian, R., Wiratunga, N., Orecchioni, A., Watt, S.: Fast Case Retrieval Nets for Textual Data. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 400–414. Springer, Heidelberg (2006)

8. Lenz, M.: Case Retrieval Nets as a Model for Building Flexible Information Systems. PhD Dissertation, Humboldt University, Berlin, Germany (1999)
9. Kolodner, J.L.: Case Based Reasoning. Morgan Kaufmann, San Francisco (1993)
10. Hammond, K.J.: Reasoning as Remembering; the Theory and Practice of CBR. In: Proceedings of AAAI Conference, San Jose, California, p. 865 (1992)
11. Bartsch-Spörl, B., Lenz, M., Hubner, A.: Case-Based Reasoning Surveys and Future Direction. In: Puppe, F. (ed.) XPS 1999. LNCS (LNAI), vol. 1570, pp. 67–89. Springer, Heidelberg (1999)
12. Kolodner, J.L.: Improving human decision through case based decision aiding. AI magazine 12(2), 52–68 (1991)
13. Watson, I.: Applying Case-Based Reasoning: techniques for enterprise systems. Morgan Kaufman, San Francisco (1997)
14. Pal, S.K., Shiu, S.C.: Foundations of Soft Case-Based Reasoning. John Wiley & Sons, Chichester (2004)
15. Smyth, B., Keane, M.T.: Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. Journal of Artificial Intelligence 102(2), 249–293 (1999)
16. Stahl, A., Schmitt, S.: Optimizing Retrieval in CBR by Introducing Solution Similarity. In: Proceedings of International Conference on Artificial Intelligence ICAI. CSREA Press, Las Vegas (2002)
17. Bergmann, R., Wilke, W.: Towards a new Formal Model of Transformational Adaptation in Case Based Reasoning. In: Proceedings of ECAI, pp. 53–57. John Wiley and Sons, Chichester (1998)
18. Leake, D.B., Kinley, A., Wilson, D.: Case-Based Similarity Assessment: Estimating Adaptability from Experience. In: Proceedings of the Fourteenth National Conference on AI, pp. 674–679. AAAI Press, Menlo Park (1997)
19. Fuchs, B., Lieber, J., Mille, A., Napoli, A.: An Algorithm for Adaptation in Case-based Reasoning. In: Proceedings of ECAI 2000, August 20-25, pp. 45–49. IOS Press, Amsterdam (2000)
20. Smyth, B., McCLave, P.: Similarity versus Diversity. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 347–361. Springer, Heidelberg (2001)
21. Tonidandel, F., Rillo, M.: An Accurate Adaptation-Guided Similarity Metric for Case-Based Planning. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 531–545. Springer, Heidelberg (2001)
22. Diaz-Agudo, B., Gervas, P., Gonzalez-Calero, P.A.: Adaptation Guided Retrieval Based on Formal Concept Analysis. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 131–145. Springer, Heidelberg (2003)
23. Haouchine, K., Chebel-Morello, B., Zerhouni, N.: Adaptation-Guided Retrieval for a Diagnostic and Repair Help System Dedicated to a Pallets Transfer. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 33–42. Springer, Heidelberg (2008)
24. Ilyas, I.F., Markl, V., Haas, P.J., Brown, P., Aboulnaga, A.: CORDS: Automatic discovery of correlations and soft functional dependencies. In: Proceedings of ACM SIGMOD, Paris, France, June 13-18, pp. 647–658 (2004)

Amalgams: A Formal Approach for Combining Multiple Case Solutions

Santiago Ontañón and Enric Plaza

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain),
{santi,enric}@iiia.csic.es

Abstract. How to reuse or adapt past solutions to new problems is one of the least understood problems in case-based reasoning. In this paper we will focus on the problem of how to combine solutions coming from multiple cases in search-based approaches to reuse. For that purpose, we introduce the notion of *amalgam*. Assuming the solution space can be characterized as a generalization space, an amalgam of two solutions is a third solution which combines as much as possible from the original two solutions. In the paper we define amalgam as a formal operation over terms in a generalization space, and we discuss how amalgams may be applied in search-based reuse techniques to combine case solutions.

1 Introduction

Case-based reasoning systems are based on the hypothesis that "similar problems have similar solutions", and thus new problems are solved by reusing or adapting solutions of past problems. However, how to reuse or adapt past solutions to new problems is one of the least understood problems in case-based reasoning. There are multiple open problems such as what knowledge is required for adaptation and how to acquire it [15], the relation between solution reuse and case retrieval [13], and solution revision [9]. In this paper we will focus on one of such problems, namely how to reuse solutions originating from multiple cases.

The three most common approaches to reuse are: substitutional adaptation, transformational adaptation, and generative adaptation [9]. In this paper we will focus on transformational adaptation techniques, and in particular in search-based approaches. In search-based approaches to reuse, the solution to a problem is seen as a description (or term) in a search space. The retrieved solution is seen as the starting point of a search process, which explores the search space by applying adaptation operators to the retrieved solution until a satisfactory solution to the problem at hand is found. These techniques are common, for instance, in CBR systems applied to planning domains [10]. Imagine a trip planning CBR system that has to generate a trip to go from city A to city C, and has been able to retrieve a case with a solution to go from A to B and another case that has a solution to go from B to C. If search-based techniques are used, the starting point of the search will be one of those two solutions, but the solution to

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 257–271, 2010.

the problem at hand could be generated by combining the solutions in the two retrieved cases. Therefore, a better starting point would be one which combines both (parts of) solutions.

This paper introduces the *amalgam* operation as a formal operation through which, given two terms, a third term which combines as much as possible from the two original terms is generated. We will also discuss how this operation can be used to define search-based approaches to reuse which can, in a natural way, reuse solutions from more than one case at a time. Our amalgam operation is applicable whenever the solutions to be generated by a CBR system can be represented as terms in a search space, and where a *more general than* (subsumption) relation exists in such space (which we wil call "generalization space").

The rest of the paper is organized as follows. Section 2 introduces and motivates, in an informal way, the notion of amalgam. Then, Section 3 presents the concepts of generalization space needed in Section 4, where the formal definition of amalgam is presented. Finally, Section 5 discusses how amalgams may be used in the CBR Reuse process. The paper closes with related work and conclusions.

2 The Notion of Amalgam

The notion of amalgam can be conceived of as a generalization of the notion of unification over terms. Unification of two terms (or descriptions) builds a new term, the unifier, by *adding* the content of these two terms. Thus, if a term ϕ is a unifier of two other terms ($\phi = \psi_a \sqcup \psi_b$), then all that is true for one of these terms is also true for ϕ. For instance, if ψ_a describes "a red vehicle" and ψ_b describes "a German minivan" then their unification ϕ is the description "a red German minivan." Two terms are not unifiable when they possess contradictory information; for instance "a red French vehicle" is not unifiable with "a red German minivan" since being French and German at the same time is not possible for vehicles.

The strict definition of unification means that any two descriptions with only one item with contradictory information cannot be unified. Now, imagine a scenario where two such descriptions have a large part of complementary information, which a CBR system would be interested in reusing: unification is not useful. Thus, what we would be interested in CBR is considering how to reuse their complementary information setting aside, at least momentarily, the contradictory aspects.

This is the idea that we intend to formalize with the notion of *amalgam* of two descriptions (or terms). Term is an amalgam of two terms whenever it contains parts from these two terms while forming a new coherent description. For instance, an amalgam of "a red French vehicle" and "a German minivan" is "a red German minivan"; clearly there are always multiple possibilities for amalgams, since "a red French minivan" is another example of amalgam.

Given to terms ψ_a and ψ_b, their possible amalgams constitutes a set of terms, and we propose that Case Reuse can exploit this set as a search space of possible solutions achieved by combining (amalgamating) descriptions ψ_a and ψ_b.

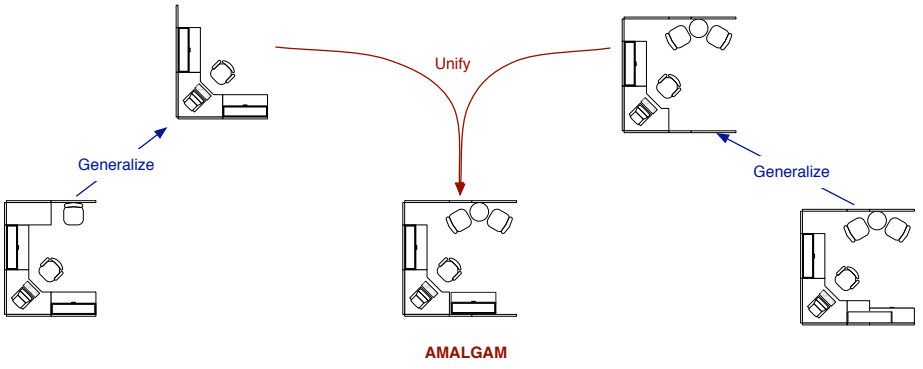


Fig. 1. An example of an amalgam of two cubicles

Intuitively, since formalization will be presented later, let us consider the content of ψ_a with respect to ψ_b divided in two parts: C_a, and I_a, where C_a is the information that is compatible with ψ_b, and I_a is that content in ψ_a that is not compatible with the content in ψ_b. If, for any two terms ψ_a and ψ_b, we are able to identify their parts ($\langle C_a, I_a \rangle$ and $\langle C_b, I_b \rangle$) there is a clear way to reuse content from both into an amalgam: $C_a \sqcup C_b$, i.e. unifying what is compatible of both terms. Nevertheless, there is part of the content in I_a and I_b that can be reused and added to an amalgam: it's just that some cannot be used together (like "German" and "French" in vehicles). Thus, reusing content in I_a and I_b to form different *coherent* combinations will constitute the space of possible amalgams that can be built from two (non-unifiable) descriptions.

Figure 1 shows an example of an amalgam of two descriptions of cubicles, where the signature "M" shape of the amalgam is made clear: given two cubicles, each one is generalized to one of their possible generalizations, and then they are unified, yielding new cubicle that amalgams aspects of both original cubicles.

3 Generalization Space

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

We say that a term ψ_1 subsumes another term ψ_2 ($\psi_1 \sqsubseteq \psi_2$) when ψ_1 is more general (or equal) than ψ_2[1]. Another interpretation of subsumption is that of an "information content" order: $\psi_1 \sqsubseteq \psi_2$ means that all the information in ψ_1 (all that is true for ψ_1) is also contained in ψ_2 (is also true for ψ_2).

The subsumption relation induces a partial order in the terms in a language $\mathcal{L}$, thus, the pair $\langle \mathcal{L}, \sqsubseteq \rangle$ is a *poset* (partially ordered set) for a given set of terms $\mathcal{L}$; additionally, we assume that $\mathcal{L}$ contains the infimum element $\bot$ (or "any"), and

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

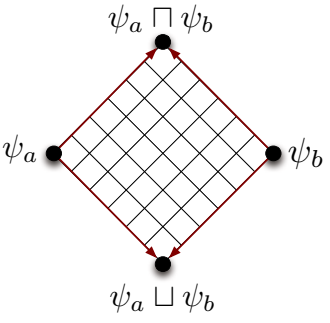


Fig. 2. A schema of the relationships between two terms ψ_a and ψ_b and their unification (below) and antiunification (above)

the supremum element $\top$ (or "none") with respect to the subsumption order. In the rest of this paper we will call a pair $\langle \mathcal{L}, \sqsubseteq \rangle$ a *generalization space*.

Given the subsumption relation, for any two terms ψ_1 and ψ_2 we can define the *anti-unification* of two terms $(\psi_1 \sqcap \psi_2)$ as their *least general generalization*, representing the most specific term that subsumes both. If two terms have nothing in common, then $\psi_1 \sqcap \psi_2 = \bot$. Thus, anti-unification encapsulates in a single description *all* that is shared by two given terms. Moreover, depending on the language $\mathcal{L}$, the anti-unification might be unique or not. Anti-unification is defined as follows:

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

The dual operation to the anti-unification is that of *unification* $(\psi_1 \sqcup \psi_2)$, which is the *most general specialization* of two given terms:

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

That is to say, the unifier's content is the addition of the content of the two original terms (all that is true in ψ_1 or ψ_2 is also true in $\psi_1 \sqcup \psi_2$). However, not every pair of terms may be unified: if two terms have contradictory information then they have no unifier —which is equivalent to say that their unifier is "none": $\psi_1 \sqcup \psi_2 = \top$. Moreover, depending on the language $\mathcal{L}$, the unification of two terms, when exists, may be unique or not.

Figure 2 shows both anti-unification (the most specific common generalization), and unification (the most general common specialization). Moreover, unification and anti-unification might be unique or not depending on the structure induced by subsumption over the terms in a language $\mathcal{L}$.

In our framework, a *refinement operator* $\rho : \mathcal{L} \rightarrow \wp(\mathcal{L})$ is a function that, given a term in $\mathcal{L}$, yields the set of direct (or minimal) generalizations or specializations of that term. In short, given a term ψ, a refinement yields a generalization (resp. specialization) φ such that there is no generalization (resp. specialization) between ψ and φ in the language $\mathcal{L}$. We will first define a refinement relation between terms:

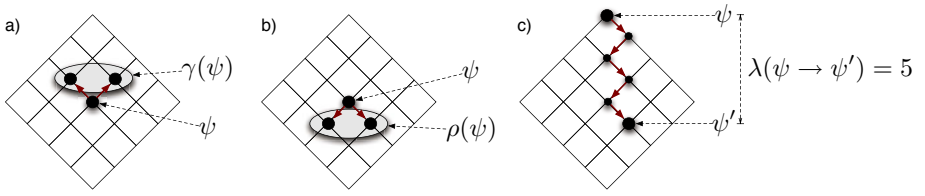


Fig. 3. Three schemas showing a generalization refinement operator γ, a specialization refinement operator ρ, and the distance between two terms

Definition 1. (Refinement Relation) *Two feature terms hold a refinement relation* $\psi_1 \prec \psi_2$ *iff*

$$\psi_1 \sqsubseteq \psi_2 \ \wedge \ \nexists \psi' \in \mathcal{L} : \psi_1 \sqsubset \psi' \sqsubset \psi_2$$

That is to say, the relation $\psi_1 \prec \psi_2$ holds when specializing from ψ_1 to ψ_2 is a minimal specialization step: there is no intermediate term ψ' in $\mathcal{L}$ between ψ_1 and ψ_2. Now a specialization refinement operator can be defined as follows:

$$\rho(\psi) = \{\psi' \in \mathcal{L} | \psi \prec \psi'\}$$

whereas a generalization refinement operator is defined as follows:

$$\gamma(\psi) = \{\psi' \in \mathcal{L} | \psi' \prec \psi\}$$

Figure 3 illustrates the idea of both generalization and specialization refinement operators. Refinement operators can be used to navigate the space of terms using search strategies, and are widely used in Inductive Logic Programming [8]. In previous work, we made use of refinement operators to define similarity measures for CBR systems [11]. For instance, in the vehicle example used above, if we have a term representing "a German minivan", a generalization refinement operator would return generalizations like "a European minivan", or "a German vehicle". If we apply the generalization operator again to "a European minivan", we can get terms like "a minivan", or "a European vehicle". A specialization refinement operator would perform the opposite task, and given a term like "a German minivan", would return more specific terms like "a Mercedes minivan", or "a red German minivan".

The length of the path between two terms ψ and ψ', noted as $\lambda(\psi \rightarrow \psi')$, is the number of times a refinement operator has to be used to reach ψ' from ψ. Moreover, the minimal path between two terms estimates their distance in the generalization space, as illustrated in Figure 3.c. Finally, the distance $\lambda(\bot \rightarrow \psi)$ measures the amount of information contained in one term (as discussed in [11]).

4 Amalgams on a Generalization Space

This section formally defines the notion of amalgam over a generalization space. Let us introduce some auxiliary definitions first. First, when two terms are not unifiable, some of their generalizations, however, may be unifiable:

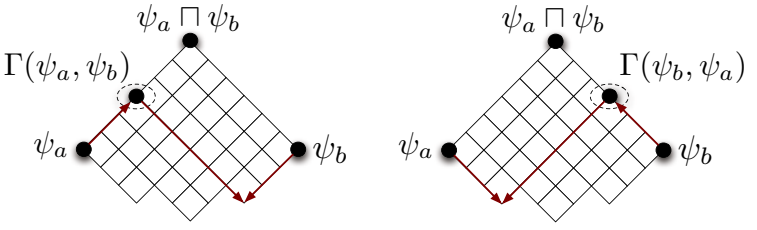


Fig. 4. A schema of the *most specific unifiable generalizations* $\Gamma(\psi_a, \psi_b)$ of term ψ_a with respect to ψ_b and $\Gamma(\psi_b, \psi_a)$ of term ψ_b with respect to ψ_a

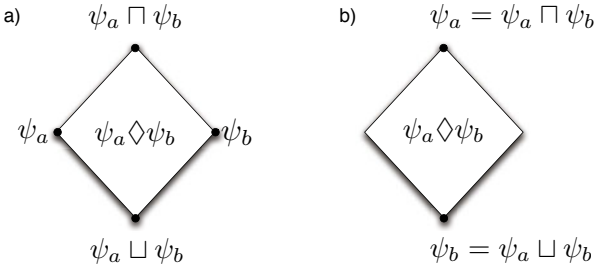


Fig. 5. Two schemas of the *interstice* space $\psi_a \Diamond \psi_b$ between two terms ψ_a and ψ_b: a) the general schema and b) the special case where $\psi_a \sqsubseteq \psi_b$

Definition 2. *The set* G *of* unifiable generalizations *of a term* ψ_a *with respect to another term* ψ_b *is:* $G(\psi_a, \psi_b) = \{\psi \in \mathcal{L} | \psi \sqsubseteq \psi_a \wedge \psi \sqcup \psi_b \neq \top\}$.

that is to say, the generalizations of ψ_a that are unifiable with ψ_b. Moreover, we are interested in the most specific unifiable generalizations in G.

Definition 3. (MUG) *The set* Γ *of* most specific unifiable generalizations *(mug) of term* ψ_a *with respect to* ψ_b *is:*

$$\Gamma(\psi_a, \psi_b) = \{\psi \in G(\psi_a, \psi_b) | \nexists \psi' \in G(\psi_a, \psi_b) : \psi' \sqsubset \psi\}$$

In other words, the most specific unifiable generalizations of a term ψ_a with respect to another term ψ_b (see Fig. 4) is the set of most specific generalizations of ψ_a which unify with ψ_b. Notice that if they are unifiable ($\psi_a \sqcup \psi_b \neq \top$) then $\Gamma(\psi_a, \psi_b) = \{\psi_a\}$.

Definition 4. (Interstice) *The* interstice *($\psi_a \Diamond \psi_b$) between two unifiable terms ψ_a and ψ_b, is the set of terms such that:*

$$\psi_a \Diamond \psi_b = \{\psi \in \mathcal{L} | (\psi_a \sqcap \psi_b) \sqsubseteq \psi \ \wedge \ \psi \sqsubseteq (\psi_a \sqcup \psi_b)\}$$

that is to say, the interstice (as shown in Fig. 5.a) is the set of terms in the $\Diamond$-shaped space defined by ψ_a and ψ_b (metaphorically "right and left"), and their antiunification $\psi_a \sqcap \psi_b$ and unification $\psi_a \sqcup \psi_b$ (metaphorically "up and down").

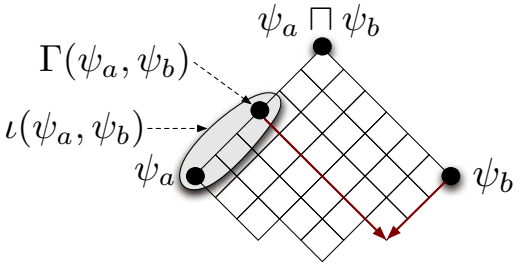


Fig. 6. Schema showing the sets of amalgamable generalizations $\iota(\psi_a, \psi_b)$ of a term ψ_a with respect to a term ψ_b

Notice that when ψ_a and ψ_b are not unifiable ($\psi_a \sqcup \psi_b = \top$) there is not strict limit below (the supremum $\top$ denotes failure to unify).

Particularly interesting is the interstice between two terms ψ_a and ψ_b when $\psi_a \sqsubseteq \psi_b$ (see Figure 5.b): this interstice contains all the terms ψ such that $\psi_a \sqsubseteq \psi \sqsubseteq \psi_b$ — i.e. all terms between ψ_a and ψ_b in the generalization space.

Out of all the terms in the interstice, we are interested in the subset of terms that contain the maximum amount of information from ψ_a and ψ_b, and which we will call the amalgam space.

Definition 5. *The set $\iota(\psi_a, \psi_b)$ of generalizations of a term ψ_a* amalgamable *with another term ψ_b is the following:*

$$\iota(\psi_a, \psi_b) = \bigcup_{\varphi_a \in \Gamma(\psi_a, \psi_b)} \varphi_a \Diamond \psi_a$$

that is to say, for each *mug* φ_a of ψ_a with respect to ψ_b there is an interstice between ψ_a and this *mug* φ_a; the terms in these interstices are amalgamable with ψ_b, and their union is the set of all amalgamable generalizations of a term ψ_a with another term ψ_b. Figure 6 illustrates this idea where the *mug* of ψ_a with respect to ψ_b is a set $\Gamma(\psi_a, \psi_b)$ with a single term, and where we can see that the terms in $\iota(\psi_a, \psi_b)$ correspond to the terms in the interstices between ψ_a and the *mug*, i.e. in the paths going from ψ_a to the *mug*.

Definition 6. (Amalgam) *The* amalgams *of two terms ψ_a and ψ_b is the set of terms such that:*

$$\psi_a \Upsilon \psi_b = \{\phi \in \mathcal{L} | \exists \varphi_a \in \iota(\psi_a, \psi_b) \wedge \exists \varphi_b \in \iota(\psi_b, \psi_a) : \phi = \varphi_a \sqcup \varphi_b\}$$

We call *amalgamation operation* the function $\Upsilon : \mathcal{L} \times \mathcal{L} \rightarrow \wp(\mathcal{L})$ that determines the set of amalgam terms for any pair of terms, and we call *amalgam space* of two terms the part of the generalization space that contains their amalgams. Thus, a term ϕ is an *amalgam* of ψ_a and ψ_b if ϕ is a unification of two terms φ_a and φ_b that are amalgamable with ψ_b and ψ_a respectively. Figure 7 illustrates this idea, showing as a grey area the space of amalgams, and showing one point in that

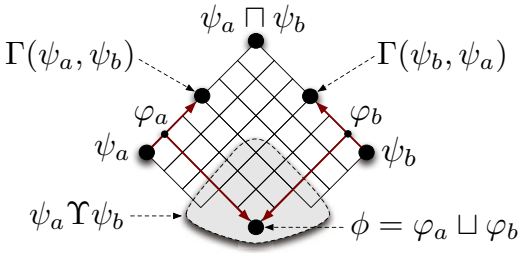


Fig. 7. Schema showing (in grey) the space of the *amalgams* $\psi_a \Upsilon \psi_b$ between two terms ψ_a and ψ_b

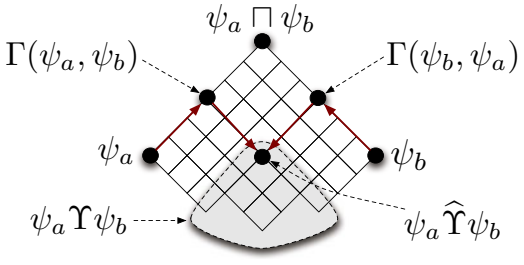


Fig. 8. Schema showing the set of upper bounds $\psi_a \widehat{\Upsilon} \psi_b$ of the space of *amalgams* $\psi_a \Upsilon \psi_b$ (in grey) between two terms ψ_a and ψ_b

space as the amalgam corresponding to $\phi = \varphi_a \sqcup \varphi_b$. In the special case where two terms are unifiable ($\psi_a \sqcup \psi_b \neq \top$), then unifiers and amalgams coincide: $\psi_a \Upsilon \psi_b = \psi_a \sqcup \psi_b$.

Next we will define the upper bounds of a space of amalgams:

Definition 7. *The set of* upper bounds $\psi_a \widehat{\Upsilon} \psi_b$ *of an amalgam space* $\psi_a \Upsilon \psi_b$ *is the minimum set such that* $\forall \phi \in \psi_a \Upsilon \psi_b,\ \exists \phi' \in \psi_a \widehat{\Upsilon} \psi_b : \phi' \sqsubseteq \phi$.

The set of upper bounds can be determined as follows:

$$\psi_a \widehat{\Upsilon} \psi_b = \{\phi \in \mathcal{L} | \exists \varphi_a \in \Gamma(\psi_a, \psi_b) \wedge \exists \varphi_b \in \Gamma(\psi_b, \psi_a) : \phi = \varphi_a \sqcup \varphi_b\}$$

That is to say, given two terms, ψ_a and ψ_b, the set of pair-wise unifications of terms in their mugs $\Gamma(\psi_a, \psi_b)$ and $\Gamma(\psi_a, \psi_b)$ produces the set of upper bounds of the space of amalgams. Figure 8 shows the upper bounds of two terms in the special case where their mugs are unique.

Figure 9 shows an example where two trains, ψ_a and ψ_b are represented using feature terms [2]. Each train is represented as a set of cars, where the first car is an engine. Each car has a type, and can be loaded with items, where each item is of a certain type (triangles or squares), and the number of items as an integer. The two trains in Fig. 9 do not unify, since the first car after the engine is an open hexagon for ψ_a, while it is a rectangle for ψ_b. Additionally, the number of items do not match. However, the description of the left train ψ_a can be generalized

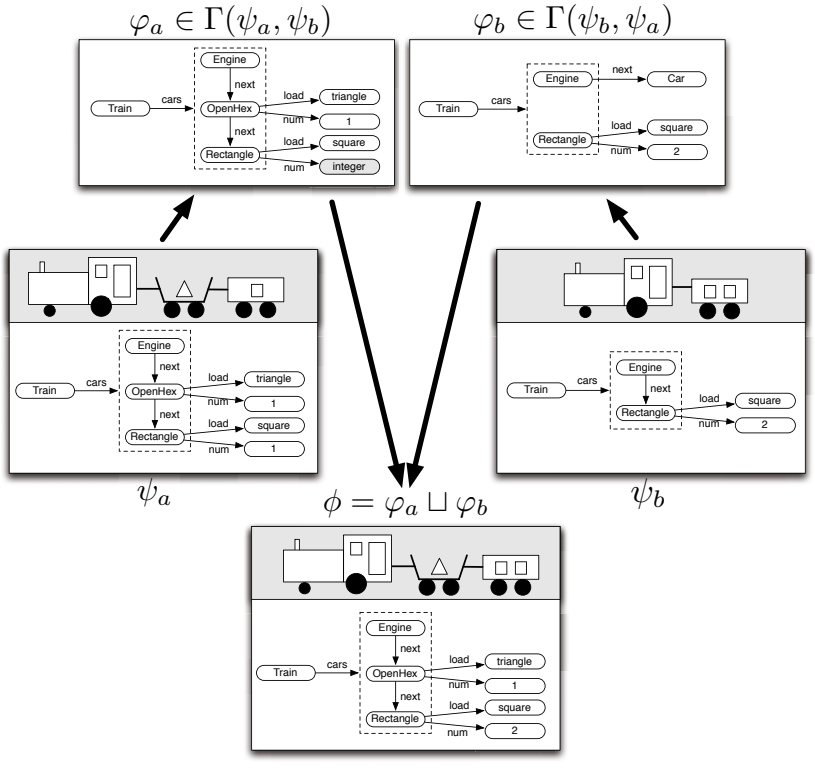


Fig. 9. Amalgam between two trains, represented using a feature terms

by removing the restriction that the number of items loaded in the second car is 1 (yielding the term φ_a), and the right train ψ_b can be generalized by removing the restriction that the car after the engine is a rectangle car (yielding the term φ_b). Noo, the two generalizations (φ_a and φ_b) unify in a term ϕ, which is an amalgam of both original trains. Notice that the resulting amalgam has features of both trains: ϕ has two cars after the engine, like the left train, but the load of the rectangle car are two squares, like in the right train.

The next section addresses the role of amalgams in the CBR Reuse process.

5 CBR Reuse through Amalgams

In this section we will discuss how the amalgam operation can be used for reuse. We'd like to emphasize that amalgams are not proposed as a complete reuse technique, but as a piece that can be used inside of CBR reuse techniques.

Search in the solution space is an approach to case reuse which is commonly used for tasks where the solution is a complex structure, such as in configuration [16] or planning tasks [1]. CBR systems which use search for reuse typically retrieve a single case, and use the solution in that case as the starting point of a

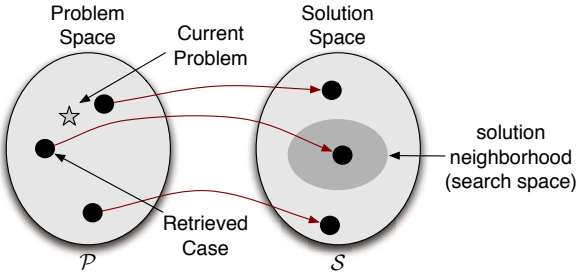


Fig. 10. Search-based case reuse: a single case is retrieved, and its solution neighborhood is explored using search

search process in the solution space. The main idea is that it is expected that the solution to the problem at hand will be close to the solution in the retrieved case. Figure 10 illustrates this idea, where given a target problem, the most similar case (according to a similarity in the problem space) is retrieved, and then the neighborhood of the solution of the retrieved case is explored using search.

When using a search-based reuse technique, we need both a) a strategy to (systematically) explore the solution space in a neighborhood of the retrieved solution and b) a criteria to stop the search. In our framework we model a) using refinement operators (introduced in Section 3) that define a systematic way to traverse the solution space, and concerning b) we will assume the existence of a predicate, $v(P, S)$, which given a problem and a candidate solution, determines whether a term is a valid solution for the problem at hand: $v : \mathcal{P} \times \mathcal{S} \rightarrow \{true, false\}$, where $\mathcal{P}$ is the problem space, $\mathcal{S}$ is the solution space, and $v(P, S) = true$ when S is a valid solution for P. For example, in a configuration task, such predicate would return true when a particular solution is a valid configuration and false when it is an invalid or incomplete configuration; in a planning domain, it would return true when the solution is a complete plan consistent with the problem specification. If no known restrictions about solutions exist in a domain, then v would just return $true$ for any solution.

Search-based approaches to reuse are, often, limited to reusing a solution from a single case. This is because search techniques typically require a starting point for the search process, which corresponds to the solution of the retrieved case. However, if we wanted to reuse solutions from more than one case, there is no clear starting point. Our proposal is to use an amalgam as the starting point to search for solutions to the problem at hand, as illustrated in Figure 11.

An underlying assumption of search-based reuse approaches is that it is preferable to preserve as much from the retrieved solution as possible [2]. If we use amalgams to generate a starting point for searching, we need a way to measure how much information of the original solutions an amalgam still preserves. For that purpose, we define the *preservation degree* of an amalgam.

[2] Although this assumption is only true for conservative adaptation techniques.

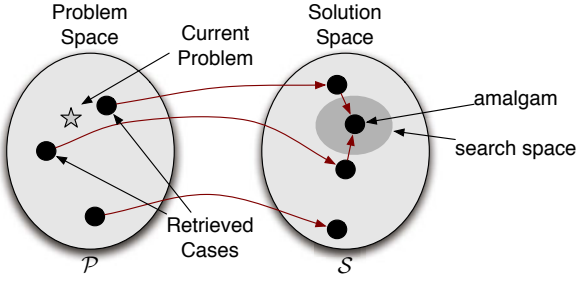


Fig. 11. Reusing the solutions of two cases by amalgamation plus search

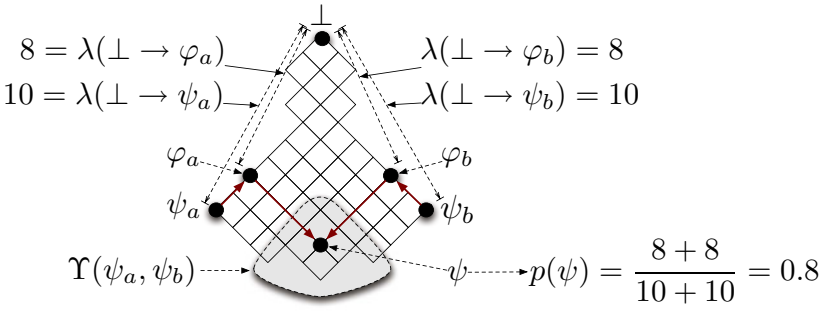


Fig. 12. A schema illustrating the preservation degree $p(\psi)$ of an amalgam ψ

Definition 8. (Preservation Degree) *Given an amalgam $\psi \in \psi_a \Upsilon \psi_b$ which is a unification $\psi = \varphi_a \sqcup \varphi_b$ of two terms such that $\varphi_a \in \iota(\psi_a, \psi_b)$ and $\varphi_b \in \iota(\psi_b, \psi_a)$, its* preservation degree $p(\psi)$ *is:*

$$p(\psi) = \frac{\lambda(\bot \rightarrow \varphi_a) + \lambda(\bot \rightarrow \varphi_b)}{\lambda(\bot \rightarrow \psi_a) + \lambda(\bot \rightarrow \psi_b)}$$

where $\lambda(\psi \rightarrow \psi')$ is the number of times a refinement operator has to be used to reach ψ' from ψ —i.e. the distance between ψ and ψ' in the generalization space.

The preservation degree (see Fig. 12) is the ratio of information preserved in the amalgam ψ with respect to the information present in the original terms ψ_a and ψ_b. The information preserved is measured by the addition of the information contained in the two amalgamable terms φ_a and φ_b yielding the amalgam ψ. When nothing is preserved $p(\psi) = 0$ since $\psi = \bot$, while $p(\psi) = 1$ when $\psi = \psi_a = \psi_b$.

As shown in Fig. 12, the preservation degree is high when ψ_a and ψ_b had to be generalized very little in order to obtain the amalgam ψ. In other words, if the λ-distances between ψ_a and φ_a and between ψ_b and φ_b are low, preservation is high. If ψ_a and ψ_b had to be greatly generalized before finding amalgamable

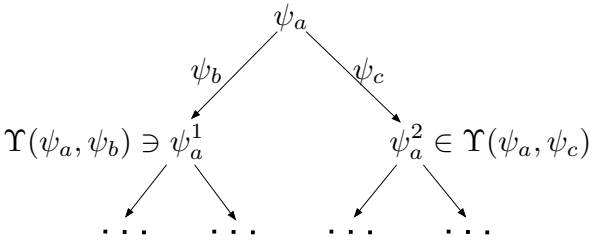


Fig. 13. The search space defined by the process of amalgamating a tentative solution ψ_a with other solutions from cases in the case base

generalizations, then the preservation degree of the resulting amalgam will be low. Thus, the higher the λ-distances between ψ_a and φ_a and between ψ_b and φ_b, the lower the preservation degree.

Using the previous notions, we can define a basic way of reusing the solutions ψ_a and ψ_b from two cases to solve a particular problem P in the following way:

1. Compute the amalgam $\psi \in \psi_a \Upsilon \psi_b$ with the highest preservation degree.
2. Search in the neighborhood of ψ (the neighborhood can be searched using refinement operators) until we find a solution ψ^* such that $v(P, \psi^*) = true$.

This section has presented amalgam as a solution to reusing solutions from multiple cases in the context of search-based reuse. We have focused our explanation on reusing solutions from two cases, but the techniques generalize easily to reusing three or more solutions. The amalgam operation can be extended to amalgamate more than two terms in an straightforward way.

Let us now consider a different scenario, where a CBR system initially finds a tentative solution ψ_a and, later, transforms this solution by adding elements from another solution ψ_b. For instance, let us consider the first cubicle of Fig. 1 as the initial solution, and let us imagine there are certain aspects of the second cubicle that we would like to add to the initial solution: the result is the "amalgam cubicle" in Fig. 1. If we iterate these amalgamation operations, as shown in Fig. 13, we have a search process over the solution space. Starting from ψ_a, the initial solution, we can amalgamate ψ_a with ψ_b (or with ψ_c) obtaining a new solution ψ_a^1 (resp. ψ_a^2). Next, we can iteratively produce new amalgams over ψ_a^1 and ψ_a^b, obtaining a search tree like that of Fig. 13.

Thus, the amalgam operation provides a theoretical framework for combining case solutions, which can be used inside of case reuse strategies in different ways, only a few of which we have discussed here. Finally, recall that the only assumption required for working with amalgams is that the solution space of a CBR system can be expressed as a generalization space. Tasks such as configuration or planning naturally suit this framework; but even classification tasks may fit in our framework in a straightforward way.

6 Related Work

The notion of amalgam is closely related to that of *merge* in the formalism of fluid construction grammar (FCG) [14] in that both relax or generalize the notion of unification. The merge operation is defined in FGC, a formalism in the family of unification-based grammars [6, 5] used in computational linguistics. Specifically, FGC has been proposed as a framework for language emergence and evolution, a process in which new structures are created or invented. This goal requires more flexibility than the operations used in unification-based grammars, which leaded to FGC defining operations like merge and an extended form of unification.

The *merge* operation is informally defined as follows: "[...] merging a source expression s and a pattern expression p means changing the source expression such that it unifies with the pattern expression" [14]. That is to say, $merge(p, s)$ is a an operation that finds a changed source expression s' and yields as result their unification $(p \sqcup s')$. Notice that merge is asymmetric, since the two terms being merged are distinguished: one is changed (because it is the source) while the other remains unchanged (because it is the pattern). Our notion of amalgam of two terms ψ_1 and ψ_2, however, is symmetric: both terms are in equal standing, both are "changed" by generalization, eliminating some parts, in order to obtain an amalgam by the unification of those generalizations.

Merging operators have also been studied in *belief merging* [7], where the goal is to merge two knowledge bases (beliefs plus integrity constraints) while maintaining consistency. This approach was applied to CBR [3] by viewing case combination in reuse as a belief merging problem. Specifically, each case is viewed as a knowledge base, and the merging is generating a new knowledge base that preserves the relevant integrity constraints. Integrity constraints play a similar role to our validity predicate. Moreover, CBR techniques for adapting solutions are also relevant to this paper, since the main research goal for introducing amalgams is to provide a formalization, as a search process, of reusing past solutions to build a new solution adapted to the current problem. Several reuse techniques can be understood as search processes over the solution space: local search, Abstract-Refine, compositional adaptation, and plan adaptation.

Compositional adaptation are reuse techniques that find new solutions from multiple cases, which were analyzed for configuration tasks in [16], where the approach *adaptation-as-configuration* is also presented. This approach has two specific operators (compose and decompose) that achieve compositional adaptation in "conceptual hierarchy oriented configuration." *Compose* merges multiple concept instances that have already been configured, while *Decompose* gives the subconcept instances of a given concept instance. These operations work upon *is-a* and *part-of* relations in an object-oriented hierarchy. Our notions of amalgam and interstice space are intended to formalize the process of combining parts of different case solutions in a more abstract, domain-independent way. We do not assume that the task is configuration or that an object-oriented hierarchy is present, only that solutions are in a generalization space. Moreover, amalgams allow us to characterize the solution space of CBR as a search space for the purposes of combining multiple solutions into a new solution.

The Abstract-Refine approach to case-based planning [1] performs reuse in two steps: first a description is abstracted (not generalized), and second the abstract description is refined in search of an adapted description. The main difference with our approach is that Abstract-Refine starts from one description in the solution space and then explores the neighborhood of that description, while the starting point using amalgams is a term which is a combination of the solutions in 2 or more cases.

Finally, local search uses a taxonomy of classes to perform a Generalize-Refine process in search for an adapted solution and, as before, the main difference from our approach is that it explores the neighborhood of one description. SPA (Systematic Plan Adapter) [4] is an approach for adapting plans as search in a refinement graph of plans. SPA is systematic and complete and, as local search, is based on adapting a single plan. MPA (Multi-Plan Adapter) [12] is an extension which allows for reusing multiple plans. MPA breaks the different plans into smaller pieces, which are then recombined together. The complexity breaking these plans into smaller pieces, however, is very high and MPA uses only a heuristic. Compared to MPA, our approach avoids the need of this previous process of breaking down while providing a systematic way to combine multiple solutions into a single one.

7 Conclusions

This paper has presented a new operation between terms called *amalgam*. This operation can be seen as a relaxation of unification. An amalgam of two terms is a new term which contains as much information from the two original terms as possible. This new term is constructed by generalizing the two original terms as little as possible, and then unifying them. Moreover, we have presented how can the amalgam between two solutions be used for multiple solution reuse in CBR systems. The framework presented in this paper is domain-independent, and does not assume anything about the task the CBR system has to solve other than being able to express the solution as a term in a generalization space.

CBR systems typically only exploit similarity in the problem space, but domains where solutions are complex structures could also benefit from analyzing and exploiting similarity in the solution space. The amalgam operation presented in this paper explores such idea, since the amalgam space defines a subset of terms in the solution space which are similar not to one but to two given solutions. We are already working on a technique implementing the amalgam operator for feature terms, but it's out of the scope of this paper.

As future work, we'd like to continue exploring the idea of using similarity relations in the solution space. More specifically, we plan to explore the applications of the amalgam operation to develop search-based techniques for CBR Reuse for specific tasks like configuration and hierarchical planning. Finally, amalgamating more than two solutions is also part of future work, since amalgamating more than two terms increases the number of constraints and, thus, enlarges the search space for the techniques implementing amalgamation.

Acknowledgements. This research was partially supported by project Next-CBR (TIN2009-13692-C03-01).

References

[1] Bergmann, R., Wilke, W.: Building and refining abstract planning cases by change of representation language. Journal of Artificial Intelligence Research (JAIR) 3, 53–118 (1995)
[2] Carpenter, B.: Typed feature structures: an extension of first-order terms. In: Saraswat, V., Ueda, K. (eds.) Proceedings of the International Symposium on Logic Programming, San Diego, pp. 187–201 (1991)
[3] Cojan, J., Lieber, J.: Belief merging-based case combination. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 105–119. Springer, Heidelberg (2009)
[4] Hanks, S., Weld, D.S.: A domain-independent algorithm for plan adaptation. J. Artificial Intelligence Research 2(1), 319–360 (1994)
[5] Jackendoff, R.: Foundations of Language: Brain, Meaning, Grammar, Evolution. Oxford University Press, Oxford (2002), http://www.isrl.uiuc.edu/~amag/langev/paper/jackendoff02.html
[6] Kay, M.: Functional unification grammar: a formalism for machine translation. In: Proc. 10th Int. Conf. on Computational Linguistics, pp. 75–78. Association for Computational Linguistics, Morristown (1984)
[7] Konieczny, S., Lang, J., Marquis, P.: Da2 merging operators. Artificial Intelligence 157(1-2), 49–79 (2004)
[8] Lavrač, N., Džeroski, S.: Inductive Logic Programming. In: Techniques and Applications. Ellis Horwood (1994)
[9] Mántaras, R.L.D., 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. Knowl. Eng. Rev. 20(3), 215–240 (2005)
[10] Muñoz-Ávila, H., Cox, M.: Case-based plan adaptation: An analysis and review. IEEE Intelligent Systems 23, 75–81 (2007)
[11] Ontañón, S., Plaza, E.: On similarity measures based on a refinement lattice. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 240–255. Springer, Heidelberg (2009)
[12] Ram, A., Francis, A.: Multi-plan retrieval and adaptation in an experience-based agent. In: Leake, D.B. (ed.) Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press, Menlo Park (1996)
[13] Smyth, B., Keane, M.T.: Adaptation-guided retrieval: questioning the similarity assumption in reasoning. Artificial Intelligence 102, 249–293 (1998)
[14] Steels, L., Beule, J.D.: Unify and merge in fluid construction grammar. In: Vogt, P., Sugita, Y., Tuci, E., Nehaniv, C.L. (eds.) EELC 2006. LNCS (LNAI), vol. 4211, pp. 197–223. Springer, Heidelberg (2006)
[15] Wilke, W., Bergmann, R.: Techniques and knowledge used for adaptation during case-based problem solving. In: 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA 1998, pp. 497–506. Springer, Heidelberg (1998)
[16] Wilke, W., Smyth, B., Cunningham, P.: Using configuration techniques for adaptation. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 139–168. Springer, Heidelberg (1998)

Recognition of Higher-Order Relations among Features in Textual Cases Using Random Indexing

Pinar Öztürk and Rajendra Prasath*

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

Abstract. We envisage retrieval in textual case-based reasoning (TCBR) as an instance of abductive reasoning. The two main subtasks underlying abductive reasoning are 'hypotheses generation' where plausible case hypotheses are generated, and 'hypothesis testing' where the best hypothesis is selected among these in sequel. The central idea behind the presented two-stage retrieval model for TCBR is that recall relies on lexical equality of features in the cases while recognition requires mining higher order semantic relations among features. The proposed account of recognition relies on a special representation called *random indexing*, and applies a method that simultaneously performs an *implicit dimension reduction* and discovers *higher order relations* among features based on their meanings that can be learned *incrementally*. Hence, similarity assessment in recall is computationally less expensive and is applied on the whole case base while in recognition a computationally more expensive method is employed but only on the case hypotheses pool generated by recall. It is shown that the two-stage model gives promising results.

Keywords: Textual case based reasoning, random indexing, dimension reduction, higher-level relations, distributed representations.

1 Introduction

Psychological plausibility of case-based reasoning (CBR) has long been proven. Nevertheless, CBR has not reached the popularity it deserves in industry. The reason lies most probably with the structured (i.e., typically attribute-value pairs) case representation which dominates classical CBR applications. Since the collections of experience in industry are predominantly documented and stored in text format, their use in classical CBR would require tremendous knowledge engineering work to convert the text reports into structured cases. This would

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

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 272–286, 2010.

be a daunting task and a rather costly one which the industry is not willing to pay. Textual CBR (TCBR) is a promising candidate to resolve this problem.

There are mainly two directions within the TCBR research each of which uses textual knowledge sources in radically different ways. The first one aims at extracting the structured content of a case from a textual report (i.e, explicit knowledge extraction) and automatically populate the case base with such structured cases. Then the classical CBR can be applied to retrieve the cases and reason with them. Brüninghaus and Ashley [5] investigated how to use natural language processing and information extraction methods to automatically extract relevant factual information. They have made significant contributions in automatically deriving abstract indexing concepts from legal case texts [6]. The second research line aims to develop methods for retrieval and reuse of *textual cases*, i.e., in free text format without converting into structured cases [29,8,2]. The work presented here falls into the second group of research.

We envisage retrieval of cases as an abductive inference process which was first studied by philosophers who maintained that it is reasoning from effects to causes and is commonly employed both in daily life and in science [22]. The two main subtasks underlying abductive reasoning are 'hypotheses generation' in which plausible hypotheses are generated, and 'hypothesis testing' in which the best hypothesis is selected among these in sequel.

This paper presents a two-stage textual case retrieval method. The first stage generates a set of candidate cases using a computationally less expensive process while the second manages a deeper assessment of the similarity between the limited number candidates.

MAC/FAC system [12], which makes use of the structural mapping account of analogical reasoning (AR) suggested by Gentner [11], implements a similar two stage approach to analogy which can discover the similarity between the solar system and a hydrogen atom (e.g., the planet and an atom revolve around the sun and the nucleus respectively). Although AR is different from CBR regarding the domains they operate with (i.e., AR is made across domains while CBR within the same domain), the analogy we draw between the underlying reasoning process of AR and CBR is sensible. In classical CBR, [1], [20], and [19] investigate a two-stage model of case retrieval. The retrieval approach presented here is similar to CREEK's [1] retrieval philosophy in classical CBR in that it also uses both shallow and deep knowledge. The approach also aligns with the idea of retrieval nets suggested in [19] since both envisage links between two cases through features common to both.

Different from all these works, our objective is to study how such a two-stage account of retrieval can be modeled in textual CBR which faces additional challenges. The two important differences are: *(i)* in classical CBR, selection of attributes/values does not have the same challenge because, by default, the designer selects a plausible set of attributes and values. In TCBR, the dimensionality of the attributes is a cardinal problem, *(ii)* structural and causal relations that are required for the deep similarity assessment are not as explicit as is in the classical models. Explicit representation of such relationship is also problem

in the classical CBR but can be solved in different ways than in TCBR. For example, in CREEK [1] and CoNSID-CREEK [21], domain specific knowledge was acquired using knowledge engineering. Discovery of higher-level relations among features in TCBR needs other solutions.

The information retrieval(IR) research community has, for decades now, addressed the issue of retrieving unstructured or semi-structured documents relevant to a given query. Textual case retrieval has substantial similarities to information retrieval, and as such it could borrow techniques from the IR field. In return, it inherits the problems with which IR is suffering from. Notably, the two problems formulated above are the long lasting research topics in the IR research agenda.

In the proposed two-stage retrieval model of textual cases (elaborated in Section 3), the first stage relies basically on associative memories which do not involve any deep knowledge, hence, it enables an efficient generation of case hypotheses (i.e. a set of cases possibly relevant to the new case). The second stage, on the contrary, requires elaboration of these hypotheses. Although scrutinizing cases in the second stage is a time consuming process, this would not lead to adverse consequences because it would not span the whole case collection but will be limited only to the cases that are in the *hypothesis pool* generated in the first stage. We elucidate these stages in section 3.1 and 3.2 respectively. The account takes into consideration some central problems peculiar to textual knowledge representation, such as high dimensionality and latent semantic relations, and suggests some solutions.

The organization of the paper is as follows: section 2 gives a brief description of recall and recognition studies in cognitive psychology. The proposed two-stage retrieval model of textual cases is given in section 3. Section 4 presents the evaluation method, data sets, results and a brief discussion. Finally, section 5 wraps up with conclusions and future directions.

2 Two Types of Memory Task: Recall and Recognition

Abductive inference was first identified by philosophers [22] as a distinct type of inference which can be traced back to Aristotle. Harman called it *inference to the best explanation* [14]. Abductive inference is typically relied upon in the face of incomplete or inconsistent information. As such, it is frequently used both in everyday life and in expert-level reasoning, and is consistently stated to be practiced by scientists, detectives, and physicians. Harman states that "when a detective puts evidence together and decides that it must be the butler, he is reasoning that no other explanation which accounts for all the facts is plausible enough or simple enough to be accepted" [15].

We maintain that the hypothesis generation process in TCBR retrieval dredges the memory for the set of features embraced in the new case text and activates a pool of memories each of which constituting a case hypothesis. This involves **recall** of memories, in cognitive psychology terminology. Hypotheses in the pool,

in turn, are scrutinized to unravel which of them 'explains' the current problem best, which resembles a **recognition** tasks.

Differences between two fundamental memory tasks, *cued recall* and *recognition*, have long been studied by cognitive psychologists [28] and neuroscientists [13]. Recall is defined as the act of retrieving a specific incident or any other item from the long term memory while recognition is termed as the process where new information is compared with information in memory. Findings in neuroscience indicate that different neural areas become activated during recall and recognition [13], i.e., possibly different types of memories and representations underly each of these processes. Besides researchers, commercial actors have also been interested in these memory tasks recently, with a rather different motivation, namely to find out how they can better manipulate/persuade customers. In particular, advertisement related research is exploring the perceptual and memory related requirements that advertisements should meet - there is even a dedicated journal, the Journal of Advertising Research.

Cognitive psychology research literature provides a plethora of exciting experiments designed to study the factors that lead to better recall and recognition performance. A main focus was how knowledge is encoded during recall and recognition. Baddeley [3] investigates this from the perspective of context and his findings indicate a distinction between what he called independent and interactive encoding. Briefly, in interactive coding, the relationship between the concepts are elucidated; features of the target concepts, that are triggered by the context word, becomes important. An example from Tulving and Osler's experiments [27] is when subjects are presented word 'city' with separate cues 'dirty' and 'village'. The concept 'city' is semantically very rich and aspects of it that are coded with the cues 'dirty' and 'village' are quite different, i.e., 'traffic fumes', 'garbage', and 'dust' versus 'quiet', 'clean, and 'friendly'. In independent encoding, entities (two words, one serving as context) are just stored along each other, e.g., 'white' and 'train' - unless the subject connects these through "the trains of wedding gowns are often white", as one of our reviewers did which we had not thought before.

Cognitive psychologists advocate a two stage model of remembering and learning where the first stage involves episodic memories while the second stage relies on semantic memories [4] where concepts are encoded interactively. Some researchers also call the two stages as 'access' and 'distinction' [18].

3 A Two-Stage Retrieval Model in TCBR

CBR systems have been classified along a knowledge-intensiveness dimension [1] where on the one end of the scale are the memory-based or instance-based methods which are considered as knowledge-lean while on the other end are the knowledge-intensive methods which incorporate cases with domain-specific knowledge (e.g., CREEK, [1]). In ConSID-CREEK retrieval was implemented as abductive inference and applied in medical diagnosis [21] where domain-specific knowledge was captured through structural and causal relationships.

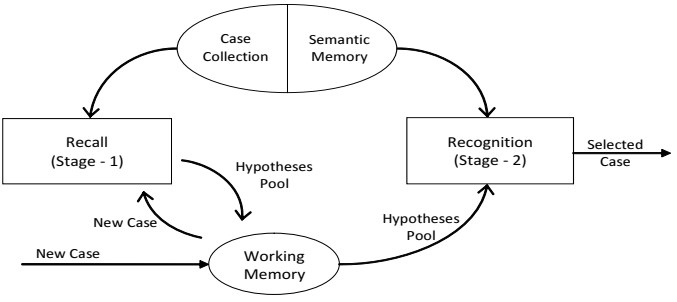


Fig. 1. Two-stage retrieval in TCBR

Such relation-chains of concepts correspond to what is called *higher order relations* in IR. Occurrence of two terms in the same context implies that they are 'similar'. The difference between the first-order and the higher-order relations between terms is that two terms, A and B, having a first-order relationship co-occur in the same context(s) while terms that have higher-order relations do not co-occur in the same context(s) but occur in similar contexts. An often used example in IR and natural language processing is the synonymous words; the second order relation between them is 'A is-synonym-of B'. For example, "high fever" and "infection" may occur in the same cases and therefore, they would have a first-order relationship. On the other hand, "mononucleosis", "glandular fever" and "kissing disease" would most possibly not occur in the same cases, since they are synonymous, but they would occur in similar cases.

In CBR, higher level relations involve also relations of causal or structural nature. In classical CBR applications, use of such higher-order knowledge was reported to contribute flexible and intelligent reasoning, especially in the assessment of similarity between the new problem and a past case [10,1]. That is, accuracy of similarity judgment increases when the higher-level relations are taken into consideration. Unfortunately, computational costs of performing such a knowledge-rich(er) similarity evaluation is high. This paper proposes a two-stage retrieval model where deeper knowledge is selectively used only in certain phases of the overall retrieval process, namely in the second stage of the two-stage model (see Figure 1). In this model, a set of relevant case hypotheses are generated in the recall stage and subsequently elaborated in the recognition stage.

Equation 1 captures the memory process that takes place in the first stage while equation 2 is the memory function used in stage 2.

$$\alpha : C_{new} \times C_C \mapsto C_H \tag{1}$$

$$\beta : C_{new} \times C_H \times C_C \mapsto C_s \tag{2}$$

where C_C is the entire set of cases in the case base, C_H is the generated set of case hypotheses ($C_H \in C_C$), C_{new} is the new case and C_s is the selected/best case ($C_s \in C_H$).

3.1 First Stage: Hypothesis Generation

Prior to any case retrieval, stop words are removed from the entire case repository, and from each remaining term/feature a link to each case including it is generated. The cases, in this way, become a part of a network where they are connected through features they share. In the first stage of retrieval, a set of case hypotheses are generated based on cued recall. The features in the new textual case (i.e., to be solved problem) act as cues and the cases activated by these cues constitute the case hypotheses. Only the features that occur in the new case are used in generation of case hypotheses, limiting the features to a subset of the entire feature set. The k most similar recalled past cases form the hypotheses pool (see Figure 1). Regarding the similarity computation, this may be done in various ways. In classical CBR, relevance of a feature to a case, specified as 'weight', is used for this purpose. This can also be done in TCBR by assuming relevance of a feature to a case, for example, as 1 (e.g., [8]). Although a finer-tuned relevance computation method in TCBR is strongly required, this is not in the scope of the current paper. Here we employ a traditional bag of words approach and construct a feature-to-case matrix in which the cells represent *tf-idf* values (i.e, the, 'term frequency' times 'inverse document frequency') and the rows correspond (only) to features that occur in the new case while columns represent the cases. Construction of this matrix in this focused way increases the efficiency of hypothesis generation. Similarity of each case in the matrix with the new case is computed using one of the standard similarity metrics in IR.

This kind of similarity judgment is considered in AI and classical CBR terminology a superficial similarity. It is superficial because it does not consider latent and relational similarities. Simply, lexically unlike terms (e.g., the shadowed upper part of Figure 2) do not contribute to the similarity.

This stage cannot discover a causal relationships, for example, between *dental surgery* which may be a feature in the new case and *unemployed* that may be a feature in a past case, when 'infectious endocarditis' is the correct diagnosis:

dental-surgery → *may-cause* → *infection* → *is-a-trigger-of* → *infectious-endocarditis* and

unemployed → *causes* → *bad-economy* → *causes* → *psychological-problem* → *may-cause* → *drug-abuse* → *uses* → *injection* → *may-cause* → *infection* → *is-a-trigger-of* → *infectious-endocardis.*

The similarity assessment method employed in the second stage of the proposed two-stage model is radically different and aims to discover the type of relations mentioned in the above example.

3.2 Second Stage: Hypothesis Evaluation

The issue of representation can never be overemphasized; it has a vital importance in computer science, and even more in AI. In TCBR, the expressive power

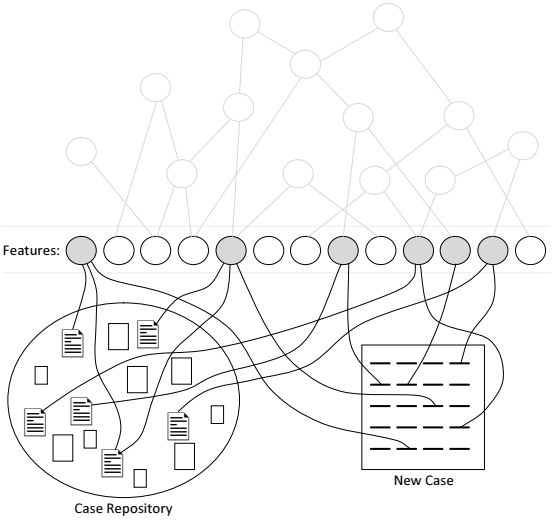


Fig. 2. The first stage of the two-stage retrieval model does not involve causal/structural relations depicted in the upper part of the figure with shadowed nodes and links

of a certain representation and efficiency of the reasoning it sanctions are determining factors in retrieval; quality of the retrieved cases is highly dependent on the information/knowledge that is taken into account in the similarity judgment. Therefore, the question of how time consuming it is to involve more than merely surface similarities is of vital importance.

There are two downsides of the 'bag of words' approach - which we used in the hypothesis generation stage. First, it considers only the first-order similarity between two terms which may not be sufficient for a thorough similarity assignment. A reason is that natural language is highly redundant and gives room for individual preferences with respect to word choices. For example, two words may be lexically different but capture the same meaning (i.e., synonymous) which may not be discovered using merely direct co-occurrence information because they hardly occur in the same incident report. For example, in one document terms A an B may co-occur, in a second document B may occur together with C and in a third document C and D may co-occur. Even though A and D do not occur together in any of the textual cases , their indirect relation through B and C (i.e., A-B-C-D) may unravel an important 'similarity' between A and D.

The second flaw of the "bag of words" approach relates to dimensionality of the search space. Features constitute a vector space in which the new case and the past cases are compared, and selection of these features happens through a number of preprocessing steps, e.g., stop word removal, stemming, etc. The main purpose of preprocessing is to identify as small as possible number of influential dimensions (i.e., features) to reduce the search cost. The dimensionality in this space may still be very high if the number of cases is large, which forms a threat

for the efficiency of the retrieval. In our account, the second stage of retrieval involves a special representation formalism, which will be described shortly, that both reduces dimensionality and discovers higher-order relational similarity at the same time.

'Meaning' of a case may be defined in terms of the meaning of the features that constitute it. In traditional IR approaches, meaning of a term is prevalently represented as a vector in the vector space. Each element of such a vector represents the frequency (in the simplest form) of occurrence of the respective term in a certain document. Our account is also grounded on a vector space model, however, we use a representation formalism called **Random Indexing** (RI) [17,24] where only the whole vector in its entirety has a meaning, not any single element of the vector alone. Hence, while the tf-idf representation is a local representation (each element/cell representing the frequency of a feature in a case), RI is a distributional representation.

RI encodes each feature initially as a vector of fixed length, consisting of zeros and a small number of randomly distributed 1 and -1s, called a **feature index vector**. RI is motivated by Johnson - Lindenstrauss Lemma [16] which asserts that a set of points in a high dimensional vector space can be mapped down into a reduced dimensional space such that the distance between any two points changes only insignificantly. Some examples of collections of high dimensional data that require compression can be found in the fields of audio, video, text documents and genome sequences. For an excellent introduction to the use of RI in text processing, the reader is referred to [24].

The meaning of a feature is represented as a **feature context vector** that accumulates information about the feature by training throughout the case collection. The amount of information that a context vector represents is proportional to its interactions with other features in the collection. At time $t = 0$, the context vector is equal to the *feature-index vector* of the feature that was randomly generated. When the whole corpus is traversed, the context vector of a feature will represent what is learned about the behavior of the feature. As such, the case collection constitutes the source of domain specific knowledge.

Superposition operation is used for updating the feature-context-vector upon its each occurrence in the case collection. Superposition of two vectors x and y gives the vector z where $z = x + y$. At time t, the context vector of each feature is computed by adding its context vector at time *t-1* with the index vectors of the features that it co-occurs with in the same "context". Context can variously be defined but in our work it is a text window of a certain length.

For each occurrence of the given feature in all cases, we focus on a fixed window of size $(2 * k) + 1$ centered at the target feature $feature_t$ (e.g., [25] suggests window size 5). Then context vector $C_{feature_t}$ for the target feature is computed using the following equation:

$$C_{feature_t} := C_{feature_t} + \sum_{j=-k; j\neq 0}^{+k} I_{feature_j \times \frac{1}{d^{|j|}}} \tag{3}$$

where $feature_j$ is the feature at the physical distance j from the target feature in the window and $I_{feature_j}$ is the index vector of $feature_j$. In the example, word *salmon* will have a larger effect on the meaning of *big* than *today* does, through the weighting term $\frac{1}{d^{|j|}}$ in the formula which captures the vicinity information. Hence, $\frac{1}{d^{|j|}}$ is the weight proportion with respect to the size j of the window (here we have taken $d = 2$).

Let us assume that k is equal to two, and target feature at the moment is **big**. We further assume that "The fisherman caught a big salmon today" which is the part of a past case we are currently using for training. Our windowed sentence for the feature *big* looks like this: *The*, [*fisherman, caught,* **big**, *salmon, today*]. Notice that 'a' is removed by stop word remover.

The feature-context-vector C_{big} for **big** becomes now:

$C_{big} := C_{big} + (0.25 \times I_{fisherman}) + (0.5 \times I_{caught}) + (0.5 \times I_{salmon}) + (0.25 \times I_{today})$

The **case-context-vector**, representing the meaning of *case* is computed based on the feature-context-vectors of the features that constitute it. It is simply:

$$C_{case} = \sum_{i=1} f_i \times C_{feature_i} \tag{4}$$

where f_i is the number of occurrences of $feature_i$ in *case*.

Algorithm 1. Stage - 2: Selecting the best case

INPUT: top k case hypotheses (output of stage - 1) and given new textual case

PROCEDURE:

Represent each case hypothesis as a *case_ context_ vector* using equation (4).
Represent the new case as a *case_ context_ vector* using equation (4)
for each case hypothesis **do**
  compute similarity between the new case and case hypothesis:
  sim(case_ context_ vector(new), case_ context_ vector(hypothesis)).
end for
Select the case hypothesis most similar to the new case

OUTPUT: Selected best case

The trained feature context vectors capture higher level feature associations similar to latent semantic indexing (LSI) [9]. However, RI has two important advantages over LSI. The first is that RI performs an 'eager' and 'implicit' dimension reduction in the sense that reduction is not performed as a separate an explicit operation. It is done in advance and 'hidden' in the specification of the term index vectors. Second, RI is an incremental algorithm. Addition of new past cases to the case collection amounts only to the update of the feature-context-vectors occurring in these case. It does not require to apply a separate reduction

Table 1. Results of tf-idf and RI methods for two new cases

New Case	TF-IDF		Random Indexing	
	Best hypotheses	Similarity	Best hypotheses	Similarity
c1	c1	1.0000	c1	1.0000
	c2	0.3696	c2	0.8698
	c3	0.0000	c3	0.5826
	c4	0.0000	c4	0.2865
	c5	0.0000	c5	0.0095
	c6	0.0000	c6	0.0055
c2	c2	1.0000	c2	1.0000
	c3	0.5000	c1	0.8698
	c1	0.3696	c3	0.7636
	c4	0.0000	c4	0.5145
	c5	0.0000	c5	0.2603
	c6	0.0000	c6	0.0142

operation similar to, for example, singular value decomposition in LSI. Upon receiving a new case it would not need to do dimension reduction all over again from the scratch.

For simplicity purpose, let us consider six cases each consisting of just two attributes: c_1: *machine, computer*, c_2: *computer, gold*, c_3: *gold, silver*, c_4:*silver, truck*, c_5:*truck, retrieval* and c_6:*retrieval, interface*, and the new case c_n is equal to the past case c_1 and to c_2 respectively in the simple experiment we provide. Table 1 shows that RI discovers more than first order relations between features. Case c_3 does not have a common feature with the new problem case c_1 and therefore bag of words account does not reckon any similarity between the two while RI discovers the indirect similarity between them. Although it is tiny the second example illustrates that ranking of the new case c_2's similarity with the 6 cases in the collection provided by the method that merely uses tf-idf weights is changed by the RI algorithm when there are several cases involving both first order and higher order relations between members of the feature set. For c_2 the second most similar case is found to be c_3 by tf-idf method and c_1 by RI.

4 Results and Discussion

In this section, we will compare the results of retrieval with and without the recognition stage, i.e., without employing a deep similarity involving a RI-based dimension reduction and higher-order relations. Our evaluation method is similar to the one given in [23]. To show the effectiveness of retrieved textual cases for the given problem description, we correlate the retrieved cases with classifier accuracy over two data sets. Throughout the experiments, we use cosine similarity to measure pairwise similarity between the given new case and the cases in the case base and then calculate classifier accuracy.

4.1 Dataset

In these experiments, we considered two different data sets: *Reuters 21578*: top 10 case bases formed from top 10 classes of Reuters 21578 newswire data set; and *TASA*: all 9 classes each with randomly selected 100 documents.

Reuters 21578 newswire data[1] is quite often used in the evaluation of text categorization. This collection contains 21,578 newswire articles (hence the name) in English. Each document, in this collection, was tagged by a human indexer with class label(s) that fell into five categories: TOPICS (119), PLACES (147), PEOPLE (114), ORGS (32), EXCHANGES (32)[the value inside brackets shows the number of subclasses]. We have omitted documents that contain empty body text (of course, we have omitted the body text having only "Blah Blah Blah" like sentences). From this collection, we have taken top 10 categories, namely *acq*, *uk*, *japan*, *canada*, *west-germany*, *grain*, *crude*, *earn*, *usa* and *money-fx*, of Reuters 21578 dataset, each having randomly chosen 100 cases and formed Reuters1000 dataset for our experiments.

TASA dataset is owned by Pearson Knowledge Technologies at the University of Colorado. TASA dataset consists of 10 million words of unmarked high school level English text. This collection consists of 37, 600 text documents arranged in 9 categories: *Business*, *Health*, *HomeEconomics*, *IndustrialArts*, *LanguageArts*, *Science*, *SocialStudies*, *Miscellaneous* and *Unspecified*. Due to memory limitations during the training phase of feature context vectors, we have considered 100 documents from each category and formed TASA900 dataset for our experiments. Stop words are removed using the SMART stop words list[2]. The features are not stemmed in our experiments.

Although cases in these data sets do not contain a separate solution part, they can sufficiently show the benefit of the proposed approach, in particular, for retrieval. Similar data sets have been used by other researchers in the TCBR community previously because large enough data sets with solution parts are still relatively scarce. We are currently working on an incident case set that would offer more natural testbed for TCBR.

4.2 Evaluation Methodology

In our experiments, we randomly split the Reuters-21578 dataset into two parts, one split with 60% for training and another with 40% for testing. Each document is considered as a case. For each feature, we obtained a feature context vector as described in section 3.2.

We use the following feature vector weighting schemes similar to [7,26]:

Case feature weights(W_{c_i}):

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

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[2] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

where w_{c_i} is the superposition of feature context vectors, each multiplied with its frequency, in the case c_i.

Query(new case) feature weights(W_{q_i}):

$$\frac{w_{q_i}}{\sqrt{\sum_{i=1}^{m} w_{q_i}^2}} \tag{5}$$

where w_{q_i} is superposition of feature context vectors, each multiplied with its frequency, in the query case q_i.

Similarity between the query (new case) and the case in the case base:

$$sim(q_i, c_i) := \sum_{\text{matching features}} W_{q_i} \times W_{c_i} \tag{6}$$

During the training of feature context vectors, the values of the vector increase with the number of occurences in the case collection. In such situations, we could apply vector length normalization. To length normalize the feature context vector $\overrightarrow{V}(c)$, for the case c having m features: $\{\overrightarrow{V_{t_1}}(c), \overrightarrow{V_{t_2}}(c), \cdots, \overrightarrow{V_{t_m}}(c)\}$, to unit vector, we do the following: $\overrightarrow{V_{t_i}}(c) := \overrightarrow{V_{t_i}}(c)/|\overrightarrow{V_{t_i}}(c)|$ where denominator denotes Euclidean length of the feature context vector of t_i in c. At the same time, any normalization factor has an effect of decreasing weight of the document features thereby reducing the chances of retrieval of the document. Therefore, the higher the normalization factor for a document, the lower the chances of retrieval of that document are [26]. Thus, depending upon the size of the data set, a suitable normalization factor can be chosen. In this work, we perform vector normalization using $\overrightarrow{V}_{norm(t_i)} = \overrightarrow{V_{t_i}}/\sum_j |\overrightarrow{V_{t_j}}(c)|$. Interesting issue here is when to normalize the feature context vector? Applying normalization at the end is not a fair idea where as progressive normalization suits as the better way.

In our experiments, given a new case, we have extracted k (= 30, here) top cases in the recall phase which is implemented as a bag of words method. We applied then random indexing for the selected top cases and the new case. Then

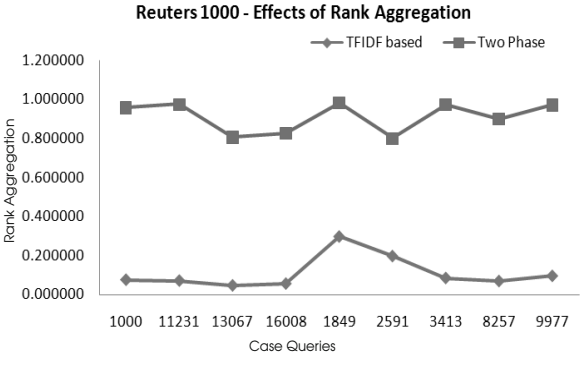


Fig. 3. Effects of rank aggregation of top k best matching cases of Reuters 1000

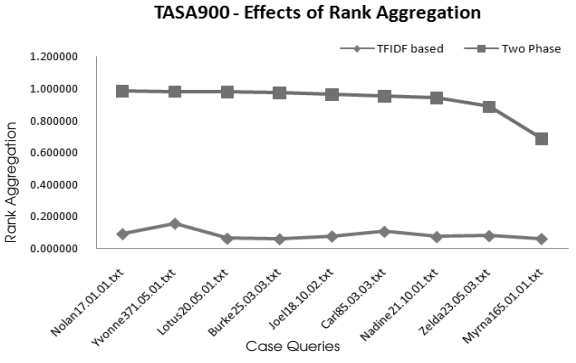


Fig. 4. Effects of similarity score aggregation of top k best matching cases of TASA 900

from the retrieved case list, similarity scores of all k top cases are aggregated and the results of the single stage and two-stage retrieval are compared.

First we have conducted the tests for Reuters1000 dataset. In this test we have taken 396 queries randomly selected from Reuters1000 dataset. The effects of similarity score aggregation of Reuters 1000 is given in Figure 3. Similarly, we repeated our experiments for 396 randomly selected Case Queries from TASA900 datasets. Figure 4 shows the similarity aggregation of a few selected queries with its similarity aggregation of top 30 best matching cases.

It is observed that the two-stage algorithm with the RI-based recognition outperforms the single stage bag of words approach in capturing the associated feature association in all 396 case queries. The poor performance of tf-idf based weighting is basically due to the fact that it operates only on presence or absence of the features in the new case with the features of past cases. Additionally we have not played much with the tuning of vector normalization (which has be to handled carefully to avoid orthogonality) so as to improve the results.

5 Conclusion

TCBR research needs to develop efficient retrieval mechanisms to cope with the combination of a large search space and the need for a good similarity assessment method. We envisage that retrieval is not a monolithic process and present a two-stage model that provides a mechanism for a focused, thorough similarity judgment between the new case and the past cases. Since the search space is pruned by the first stage, a deeper similarity assessment becomes affordable. Another source of efficiency relates to how features and cases are represented. RI is a compressed representation that allows learning the meaning of features and cases through experience, that is, adding new cases to the case base incrementally. Hence the method addresses several problems pertinent to retrieval in the conventional CBR and the TCBR.

We have applied the two-stage retrieval to two sets of data, Reuters and TASA text collections. The experimental results show that the two stage approach provides better retrieval of top cases employing feature associations at high orders than the bag of words type of textual case retrieval. In future, we plan to perform experiments to large text collections to analyze the effect of different similarity measures and feature vector normalization on effective textual case retrieval.

References

1. Aamodt, A.: Knowledge-intensive case-based reasoning in creek. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 1–15. Springer, Heidelberg (2004)
2. Adeyanju, I., Wiratunga, N., Lothian, R., Sripada, S., Lamontagne, L.: Case retrieval reuse net (cr2n): An architecture for reuse of textual solutions. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 14–28. Springer, Heidelberg (2009)
3. Baddeley, A.: Domains of recollection. Psychological Review 86(6), 709–729 (1982)
4. Baddeley, A.: Human memory. Lawrence Erlbaum, Mahwah (1990)
5. Brüninghaus, S., Ashley, K.D.: The role of information extraction for textual CBR. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 74–89. Springer, Heidelberg (2001)
6. Brüninghaus, S., Ashley, K.D.: Progress in textual case-based reasoning: predicting the outcome of legal cases from text. In: AAAI 2006: Proceedings of the 21st National Conference on Artificial Intelligence, pp. 1577–1580. AAAI Press, Menlo Park (2006)
7. Buckley, C., Salton, G., Allan, J., Singhal, A.: Automatic query expansion using smart: Trec 3. In: TREC (1994)
8. Chakraborti, S., Wiratunga, N., Lothian, R., Watt, S.: Acquiring word similarities with higher order association mining. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 61–76. Springer, Heidelberg (2007)
9. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407 (1990)
10. Díaz-Agudo, B., González-Calero, P.A.: Cbronto: A task/method ontology for CBR. In: Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference, pp. 101–105. AAAI Press, Menlo Park (2002)
11. Gentner, D.: Structure-mapping: A theoretical framework for analogy. Cognitive Science 7(2), 155–170 (1983)
12. Gentner, D., Forbus, K.D.: Mac/fac: A model of similarity-based retrieval. Cognitive Science 19, 141–205 (1991)
13. Habib, R., Nyberg, L.: Neural correlates of availability and accessibility in memory. Cerebral Cortex 18, 1720–1726 (2008)
14. Harman, G.H.: The inference to the best explanation. The Philosophical Review 74, 88–95 (1965)
15. Harman, G.H.: Enumerative induction as inference to the best explanation. Journal of Philosophy 65, 139–149 (1968)
16. Johnson, W., Lindenstrauss, L.: Extensions of Lipschitz maps into a Hilbert space. Contemporary Mathematics 26, 189–206 (1984)

17. Kanerva, P., Kristofersson, J., Holst, A.: Random indexing of text samples for latent semantic analysis. In: Proceedings of the 22nd Annual Conference of the Cognitive Science Society, pp. 103–106. Erlbaum, Mahwah (2000)
18. Kintsch, W., Miller, J.R., Polson, P.G.: Methods and Tactics in Cognitive Science. L. Erlbaum Associates Inc., Hillsdale (1984)
19. Lenz, M., Burkhard, H.D.: Case retrieval nets: Basic ideas and extensions. In: Görz, G., Hölldobler, S. (eds.) KI 1996. LNCS, vol. 1137, pp. 227–239. Springer, Heidelberg (1996)
20. McLaren, B.M., Ashley, K.D.: Case representation, acquisition, and retrieval in sirocco. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 248–262. Springer, Heidelberg (1999)
21. Öztürk, P., Aamodt, A.: A context model for knowledge-intensive case-based reasoning. Int. J. Hum.-Comput. Stud. 48(3), 331–355 (1998)
22. Peirce, C.S.: Collected Papers of Charles Sanders Peirce. In: Hartshorne, C., Weiss, P., Burks, A. (eds.), vol. 8. Harvard University Press, Cambridge (1958)
23. Raghunandan, M.A., Wiratunga, N., Chakraborti, S., Massie, S., Khemani, D.: Evaluation measures for TCBR systems. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 444–458. Springer, Heidelberg (2008)
24. Sahlgren, M.: An introduction to random indexing. In: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005 (2005)
25. Sahlgren, M.: Vector-based semantic analysis: Representing word meanings based on random labels. In: ESSLI Workshop on Semantic Knowledge Acquistion and Categorization. Kluwer Academic Publishers, Dordrecht (2001)
26. Singhal, A., Salton, G., Mitra, M., Buckley, C.: Document length normalization. Inf. Process. Manage. 32(5), 619–633 (1996)
27. Tulving, E., Osler, S.: Effectiveness of retrieval cues in memory for words. Journal of Experimental Psychology 77, 593–601 (1968)
28. Tulving, E., Pearlstone, Z.: Availability versus accessibility of information in memory for words. Journal of Verbal Learning and Verbal Behavior 5, 381–391 (1966)
29. Wiratunga, N., Lothian, R., Massie, S.: Unsupervised feature selection for text data. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 340–354. Springer, Heidelberg (2006)

Extending CBR with Multiple Knowledge Sources from Web*

Juan A. Recio-García, Miguel A. Casado-Hernández, and Belén Díaz-Agudo

Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid, Spain
jareciog@fdi.ucm.es, migacasa@pas.ucm.es, belend@sip.ucm.es

Abstract. There has been a recent interest in the Web as a very promising source of cases for CBR applications. This paper describes some alternatives that could be exploited to include these sources of cases in real CBR systems. A theoretical framework is presented that categorizes different approaches to exploit Web sources for populating the case base or obtaining the background knowledge required to perform the retrieval and adaptation stages. Finally, we introduce a real CBR system that uses Web knowledge to improve the reasoning cycle.

1 Introduction

Recent CBR conferences have pointed to the Web as a very interesting source of knowledge in general, and cases in particular, for CBR applications [11]. This paper focuses on some emerging aspects on the Web that are an opportunity and challenge for Case-based Reasoning or alternatively consistent use of CBR. The main goal of this research consists on studying different ways of improving current CBR techniques by exploiting the knowledge available in the Web.

Web pages, blogs, wikis, etc. represent a very important source of experiences that, if exploited, could greatly improve the performance of CBR applications as a huge experience base. It is very noticeable that the challenge of exploiting Web knowledge consists on developing Information Extraction techniques able to process and acquire the knowledge from textual sources [17,18,2]. There are different approaches using Language Processing and Information Extraction techniques to obtain structured representations from Web textual sources and to use these structures to define a local case base. Moreover, many companies have noticed the big importance of Web knowledge and they offer different tools to get these structured representations within Web knowledge. Google Squared, Yahoo! Query Language or Freebase are examples of these tools. This paper explores and summarizes different existing tools and how they could be integrated as software components into the jCOLIBRI2 framework to be used in the development of CBR systems.

Besides case base knowledge, general knowledge usually plays a part in this cycle, by supporting the CBR processes. This support may range from very

* Supported by the Spanish Ministry of Sci. and Ed. (TIN2009-13692-C03-03).

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 287–301, 2010.

weak (or none) to very strong, depending on the type of CBR system. By general knowledge we mean general domain-dependent knowledge, as opposed to specific knowledge embodied by cases. There is an other line of work related with CBR and Web that takes into account the semantic Web community and its strong efforts in the direction of establishing a solid theoretical base and expressive formal languages to represent, share and reuse complex knowledge bases (ontologies) [7]. Our previous research has explored the synergies between ontologies and semantic reasoning in CBR [12,6]. In this paper we take a different approach as we consider Web as a source of structured knowledge that is used to populate the local case base or to complement the system containers of knowledge. Our approach is based on how the CBR system obtains the knowledge to reason and it opens a wide range of new CBR system architectures and designs that can vary depending on the degree of knowledge already included in the system and the amount of knowledge obtained dynamically from Web sources. The goal of our current research is to study different ways to include Web Knowledge to the CBR cycle. Working with Web sources has obvious advantages as the Web is the biggest source of knowledge in the world and it allows the collaboration of thousands of users to generate knowledge. However, there are some aspects to consider when embedding this knowledge in CBR systems. For example, the requirement of a fast on-line connection during the system runtime, or the inherent problem of incompleteness of information in Web documents.

The paper runs as follows. In Section 2 we describe a knowledge based theoretical framework to include Web knowledge into CBR systems. Section 3 describes the current state of the art in structured Web knowledge bases and analyses their main features. Section 4 describes the architectural and functional implications of including Web knowledge into the jCOLIBRI framework and we present a new connector to load cases from the Web source Freebase. Section 5 describes the use of Freebase as a source of CBR knowledge and provides several examples from the Food & Drink domain. Section 6 concludes the paper.

2 Including Web Knowledge Sources in CBR Systems

It is common to describe CBR systems following the 4R's cycle (retrieve, reuse, revise and retain) introduced by [1]. This cycle uses two main sources of knowledge: the case base and the background knowledge. Both cases and background knowledge have been also described as the knowledge containers required by CBR applications [13], being the case base (kc_{cb}), the retrieval knowledge (kc_{ret}), and the reuse knowledge (also called adaptation, so we will name it as kc_{adapt}) the most important containers. These containers store the main knowledge required to perform the CBR cycle and they are the three main elements involved when including Web-mined resources into a CBR application.

Including knowledge from Web sources in the CBR cycle can be performed in different ways (see Figure 1). A simple approach consists on extracting the required resources from the Web before executing the application, fill the different knowledge containers with the obtained information, and run the system

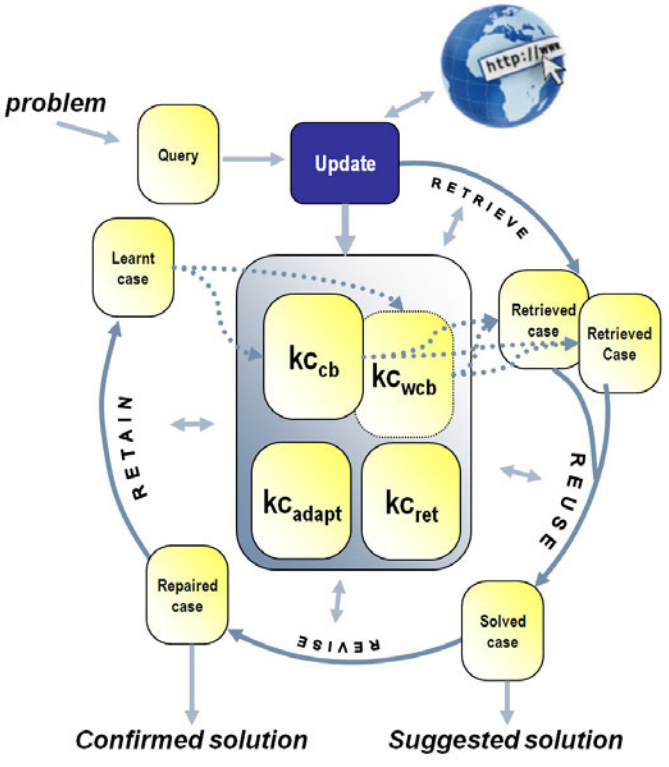


Fig. 1. Schema of a Web CBR cycle

using the typical CBR cycle. However, the inclusion of Web knowledge sources in CBR systems can be performed in a much more interesting approach: updating the knowledge containers on-demand depending on the users' requirements and their queries. For example, it is easy to imagine a CBR system that, depending on some features of the query, obtains cases from Web sources that can be useful to solve that problem plus the required retrieval or adaptation knowledge to perform the reasoning cycle. This way, our system obtains the reasoning knowledge at runtime and tailors the knowledge containers to the user requirements.

This new approach opens a wide range of new CBR system architectures and designs that can vary depending on the degree of knowledge already included in the system and the amount knowledge obtained dynamically from Web sources. We will refer to the knowledge already included in the system as the local knowledge. Therefore, in our classification, typical CBR systems only have local knowledge, whilst Web-boosted CBR applications will complement this knowledge with Web sources. It is possible to find different levels of knowledge complementation for Web-boosted CBR systems as there are many approaches to generate the knowledge containers from Web sources [14]. This paper describes how to obtain the case base (kc_{cb}) from Web sources. However,

our on-going work also explores the possibilities of creating kc_{ret} and kc_{adapt} by means of dynamically generated taxonomies obtained through data-mining techniques according to the concrete domain and features of the query.

The addition of Web knowledge in CBR applications at runtime implies several modifications to the classical CBR cycle and the way of managing the associated knowledge containers. Regarding kc_{cb} we can have a pure local approach (this is, a local case base), a dynamic case base updated every cycle depending on the query, or a mixed approach with local and Web cases. A similar scenario is found for kc_{ret} and kc_{adapt} as this knowledge can be embedded in the system during the implementation stage (as it is done in typical approaches), obtained dynamically from the Web every cycle, or -again- it is possible to mix-up local and Web-generated knowledge. Although a CBR system could use any of these approaches, we think the option that could lead to the best performance results is the hybrid approach, this is, mixing-up Web knowledge on every knowledge container. Of course, there are many issues associated to each design option to be discussed. Although the inclusion of Web sources implies some benefits as an expected improvement of the system performance, there are also some drawbacks -like the response time- that must be taken into account.

Previously described modifications in the design of CBR systems only relate to the knowledge containers, however, there are also some modifications from the point of view of the CBR cycle. A CBR system using Web-knowledge must update the involved knowledge containers *before* solving a given query. New cases will be included in kc_{cb} and additional knowledge will be embedded in kc_{ret} and kc_{adapt}. Another important modification implies the representation of cases because, quite possibly, the attributes describing the cases may change depending on the source they are obtained from. Therefore, flexible representations are required.

In this paper we focus on the inclusion of Web knowledge in the kc_{cb} container. As we will present in following sections, we are working on different approaches to dynamically include new cases depending on the user query. The basic idea consists of connecting to Web providers and getting cases that can be useful to solve the query issued by the user. This new approach raises some interesting research questions. The first one is the policy for case acquisition. For example, it is possible to obtain new cases on every execution, or measure the quality of the existing local cases with respect to the query and then decide if new cases are required. This quality measure could be just the similarity degree or any other more elaborated technique like competence models [15], or case coverage measures [10]. Then, a second design decision implies the retention policy. Cases obtained from Web sources can be discarded after every execution or included in the local case base together with the solved query. Furthermore, different policies could decide which cases should be included definitively in kc_{cb} depending again on any kind of quality measure.

From the architectural point of view, there is also an open issue depending on the way of storing these cases. The inclusion of many cases on every execution of the CBR cycle could lead to very large and noisy case bases (kc_{cb}). Therefore we could create a secondary knowledge container for cases obtained from Web

(kc_{wcb}). This new container could act simply as a kind of cache memory or to have a more important role in the CBR cycle. For example, it could be possible to modify the retrieval stage to compare the query with kc_{cb}, check the quality of the results, and if they were not good enough, compare again the query against kc_{wcb}. However, the implementation of multi-case-base-reasoners implies further implications and modifications that must be taken into account as detailed in [9]. In this paper, we do not address this problem but we present the possibilities of including an extra knowledge container with cases obtained from web sources.

Next Section presents a review of some of the most promising Web sources of structured knowledge that we have evaluated after an exhaustive review.

3 Comparing Web Structured Knowledge Sources

There is a lot of information in the Web representing experiences that can be exploited through CBR techniques. However, this knowledge is usually represented in natural language, and it really complicates its exploitation. Although it is possible to apply Information Extraction (IE) techniques to obtain cases from free text, it is a difficult task due to the nature of Web: users tend to misspell, use idioms or make grammatical mistakes that lead to poor results when applying IE techniques. Fortunately, there are some Web content providers that have realized about the importance of structured data for many automatic processing tasks like Web search, knowledge discovery, etc. Therefore they have started to offer this kind of services. Although the provided knowledge is not primarily conceived as a source of experiences, it is easy to adapt these structured representations to be included in CBR applications.

To achieve this goal, this Section presents a review of the main structured knowledge sources available in the Web. This study describes their main features and applicability, as we propose to use these sources to populate the case base of CBR systems. These structured knowledge sources are:

1. Google Squared[1]. It is a very intuitive search tool to build a collection of facts for any specified topic. Facts are organized into a table of items and attributes (called "Squares"). These squares can be customized to filter some items or attributes and can be exported to CSV files. From the CBR point of view, we can see the rows of the square as cases, whilst columns are the attributes of every case.
   The major drawback of Google Squared is its lack of API. To use this approach as a source of cases, it is required to parse directly the html code. Moreover, using Google Squared as a source of cases also requires an extra layer to check if the returned attribute values are coherent because it sometimes returns irrelevant values.
2. The Yahoo! Query Language[2] is an expressive SQL-like language that allows to query, filter, and join data across Web services. It only offers information

[1] http://www.google.com/squared
[2] http://developer.yahoo.com/yql/

from Web sites belonging to Yahoo! (like flickr). Although it offers an API, it is not general enough to be used for obtaining cases because it is limited to the information provided by some few Web sites.

3. Search Monkey[3] is a Yahoo! initiative to provide structured search results. They propose Web developers to include structured information of their Web content using xml annotations. Then, Yahoo!'s Web spider will collect, process and store this data into a database that will be later used by Yahoo! Search to offer structured results. The main drawback is that the database is only accessible through an on-line tool that provides a very limited functionality because there is not an API that could be used to issue general queries.
4. Finally, Freebase [4] is an on-line collection of structured data harvested from many sources, including wikipedia and individual contributions. Freebase aims to create a global resource which allows people (and machines) to access common information more effectively. Currently it has more than 3000 tables, and more than 30,000 columns. To access the data in Freebase, there is a language called MQL (MetaWeb Query Language) similar to SQL. MQL can be run through an API that is very simple to use.

After this revision we can discard Yahoo Query Language because it is very limited and SearchMonkey because it does not provide any public API. Google Squared could be a very good option but it cannot be used (at least for now) because it requires the parsing of the html code. However, Freebase presents appropriate features to be exploited as a source of cases: it has a clear API with a query language, and it contains general domain knowledge.

As Freebase provides the capabilities required to include cases in a CBR application, we have developed a software component able to obtain cases from this Web source and integrate them in the CBR cycle. This component has been included in the jCOLIBRI framework. To explain the process of integrating Web knowledge in CBR systems, Section 4 describes some architectural details of jCOLIBRI and the Freebase connector that we have included in the framework. Afterwards Section 5 explains how to use this connector as a source of Web knowledge within CBR applications.

4 Including Cases Obtained from Web Sources in jCOLIBRI

jCOLIBRI2 is an object-oriented framework in Java for building CBR systems that is aimed at CBR system designers. In this paper any reference to jCOLIBRI refers to the jCOLIBRI2 version. Although there are other frameworks for building CBR systems like myCBR[16] or IUCBRF[3], jCOLIBRI is currently a reference platform in the CBR community, being used in many international universities for teaching. It also includes many contributions from several international research groups [5].

[3] http://developer.yahoo.com/searchmonkey/
[4] http://www.freebase.com/
[5] More information and download at: http://jcolibri.net

The inclusion of cases obtained from Web sources in jCOLIBRI2 implies modifications from two different points of view. The first modification is architectural, meaning that a new component is required in order to obtain cases for the application. The second modification is functional, in the sense that the typical CBR cycle must be modified to support the retrieval from this new memory of cases. Although we illustrate how to perform this modifications in the jCOLIBRI2 framework, the following indications could be taken into account for modifying or designing any other CBR application implemented without the framework.

4.1 Architectural Modifications

The architecture of jCOLIBRI divides the management of cases into two different layers. The *persistence layer* represents any kind of persistence media that could contain cases. Persistence is built around connectors. Connectors are objects that know how to access and retrieve cases from the medium and return those cases to the CBR system in a uniform way. The use of connectors gives jCOLIBRI flexibility against the physical storage so the system designer can choose the most appropriate one for the system at hand. Currently there are different connectors available to manage cases from different medias: databases, ontologies, textual files and ARFF files.

On top of the persistence layer we find the *in-memory* organization of cases. This layer stores the cases using different organizations (for example, linear or indexed cases bases). This in-memory organization of cases can be fed by any connector. Therefore, the inclusion of Web sources of cases in the jCOLIBRI2 architecture just implies the development of the corresponding connector. This connector will access the Web source, obtain the cases and transfer them to the in-memory organization. Then, methods will be able to manage the cases independently of their source. As we have previously mentioned, we have developed a connector to use Freebase as a Web source of cases. The features of this Web connector are detailed in Section 4.3.

There is also another optional variation depending on the retention strategy. As we stated in Section 2 we could have a secondary knowledge container kc_{wcb} to store the cases obtained from Web. This requirement implies the internal division of the in-memory organization into two different structures: one containing local cases, and another with Web cases. Then, the retention policy will choose which cases should be stored in the persistence media. This feature is a new functional requirement that is described next.

4.2 Functional Modifications

The second set of modifications required to include Web sources involve the functional behaviour of the application. As we described in Section 2 there are some few changes in the CBR cycle to be made: the update of the knowledge containers when the query is issued to the system, and the retention policy.

Before addressing these issues, let's review the organization of the CBR cycle in jCOLIBRI2. The framework structures a CBR application's behaviour into 3 steps:

- Precycle: Initializes the CBR application, usually loading the case base and precomputing expensive algorithms (really useful when working with texts). It is executed only once.
- Cycle: Executes the CBR cycle. It is executed many times each time a query arrives.
- Postcycle: Post-execution or maintenance code.

The main change to be performed when working with Web sources is the functionality of the precycle. Precycle could be useless in this new scenario because the case base is not pre-loaded into memory but updated according to the query provided by the user at the beginning of the cycle [1]. However, this stage continues being appropriate if we opt for a mixed approach that combines Web and local cases.

Consequently, it is the cycle stage which requires the most important modifications. To support the dynamic acquisition of cases we must introduce a new stage before the retrieval. This stage uses one or many Web connectors to update the case base. These new cases will be stored in kc_{cb} or kc_{wcb} depending on the architectural design chosen by the developer. Finally the new stage may also update the other knowledge containers kc_{ret} and kc_{adapt}.

Although the adaptation knowledge kc_{adapt} is typically independent of the case base, there is a degenerated scenario where both containers can be seen like the same entity. This scenario involves the transformational and constructive strategies for case reuse. These techniques -detailed in [5]- generate new solutions by replacing the attributes in the best nearest neighbour that do not match the query. Replacements are performed by looking for suitable values in the remaining cases[6]. Therefore, these techniques use the case base kc_{cb} as the source of adaptation knowledge kc_{adapt}. Taking advantage of our new capabilities for obtaining specific cases from Web sources given a query, it is quite simple to adapt previous techniques to get from the Web connector cases containing the values required to perform the substitution process. This way, the kc_{wcb} container becomes a source of adaptation knowledge (this is, $kc_{adapt} \approx kc_{wcb}$).

A final modification involves the retrieval stage. As we could have two knowledge containers for cases, retrieval must be performed on every one, and results must be combined. Finally, the retain stage is also modified to include a concrete retention policy for Web cases. A good strategy is to store in the local case base kc_{cb} those cases that were retrieved from kc_{wcb} as the k nearest neighbours of the query.

4.3 Freebase Connector

As we have described in the previous Section, we have created a new connector for jCOLIBRI2 to load cases from the Web source Freebase. This connector is organized as follows. A lot of data in Freebase corresponds to information you might

[6] Note that the constructive approach uses an empty case instead of the 1-NN

find on Wikipedia. Corresponding to a Wikipedia article is a Freebase *topic* (as in "topic of discourse"). The term topic is chosen for its vagueness because there are all kinds of topic. Topics can range from physical entities, artistic/media creations, classifications, abstract concepts, etc. There can be many aspects to the same topic. For example, "Jaguar" could mean an animal, car or operative system. In order to capture this multi-faceted nature of many topics, there is the concept of *types* in Freebase. The topic about "Jaguar" is assigned several types for every meaning. Each type carries a different set of *properties* representative of that type. For example, the car type could have properties describing the brand, engine, etc. Finally, just as properties are grouped into types, types themselves are grouped into domains.

Our connector uses the HTTP API provided by Freebase to obtain topics for a given type related to the query[7]. These queries are issued using the MQL query language. Retrieved topics conform the cases and their properties are taken as the attributes of the obtained cases. The connector also allows to query Freebase about existing types and their associated properties.

5 Freebase as a Source of CBR Knowledge

This Section describes how we use Freebase to feed the knowledge containers in a CBR system. We illustrate the algorithms in a case study that uses the Freebase connector. The domain selected to perform this case study is the Food & Drink domain. We have chosen this concrete domain because there are many related applications available due to the Cooking Contest held at every ICCBR conference. The case study measures the repercussions of integrating the Web connector in CBR applications to obtain useful cases. We also present the required functional modifications to the CBR cycle. Firstly, we describe a Web only retrieval approach. Then we compare this retrieval process with the typical local approach, and a mixed retrieval process is described. Finally, we also introduce how to reuse these Web cases to perform a transformational adaptation.

The "Food & Drink" domain in Freebase covers a big amount of information about practically all kinds of food and drinks. Like other domains in Freebase, this one is in constant growth, increasing some weeks in about 300 *topics*. Looking inside this category, we can find 58 kind of different *types*, from Chefs, diets, ingredients to such specific things like *cheese textures*. For example, if we look into the *type* Chef, we can find out properties like *influences*, kinds of *cuisine* or the *restaurants* where they work.

5.1 Web Retrieval

First lets consider a scenario where we only obtain knowledge from the Web source. It implies that we are not working with a local case base. We have to

[7] For example, we could access the information about the *casear salad* dish through the following url: http://www.freebase.com/api/service/mqlread?query={"query":{"type":"/food/dish","name":"caesar salad","ingredients":[]}}

build a query, send that query to Freebase and then, receive and interpret the given results.

As we described in the previous section, we consider Freebase *topics* as the cases of the CBR system. Then, to retrieve cases we should ask for the topics belonging to a concrete type. This way, every type would be a kind of case base that is filled with different topics (cases). Then every property of the type would be the attributes of the case. Of course, this design option implies that we are limiting our systems to work with the types in Freebase. However, the 3000 types available can be considered as significant.

In our algorithm we try to avoid small retrieval sets. Therefore it partially matches the query against Freebase to maximize the number of results. This is performed by issuing several independent Web queries to Freebase, each one containing only a subset of the attributes specified by the user. The algorithm runs as follows:

Algorithm **WebRetrieval(Q)**:

1. Given a input query $Q = \{\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, \ldots, \langle a_n, v_n\rangle\}$ composed of ⟨attribute, value⟩ pairs.
2. Let FB_{types} be the set of types available in Freebase, and $Atr(type)$ the set of attributes (or Freebase properties) associated with a Freebase type t_i.
3. Let $Atr(Q) = \{a_1, a_2, \ldots, a_n\}$ be the set of attributes of the user query Q.
4. Find the type in Freebase containing the maximum number of attributes in the query:

$$Type_{max} = \{t_i \in FB_{types} \mid |Atr(type_i) \cap Atr(Q)| > |Atr(type_j) \cap Atr(Q)|\,, \forall t_j \in FB_{types} \wedge t_i \neq t_j\}$$

5. Let M be the set of attributes (or properties) that $Type_{max}$ and Q have in common, with $|M| = m$.

$$M = |Atr(type_{max}) \cap Atr(Q)|$$

6. Incrementally build Web queries for Freebase using the attributes in M: Let B_j be the set of cases retrieved from Freebase when querying using only j of the m attributes in M. For example, B_3 is the set of cases retrieved when querying using only attributes a_1, a_2 and a_3.
7. Finally, the retrieval set RS_{Web} is:

$$RS_{Web} = \bigcup_{j=1}^{m} B_j$$

8. Try to fill the values of the remaining attributes $\{Atr(Q) - Atr(M)\}$ for every case in RS_{Web}.

This algorithm implies that some attributes of the case will be empty if they have not any correspondence with a property of the type, thus the last step of

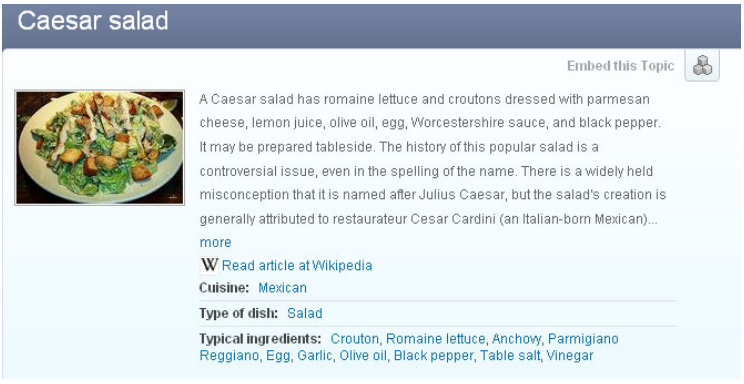


Fig. 2. Example of a dish in Freebase

the algorithm tries to fill these values. Although we have not faced this problem, a possible solution could be the use of custom IE techniques that may crawl other Web sources and fill up the empty values.

In our particular case of study, we work with the type *dish*. Dishes have several properties of which we will concentrate on two of them, *cuisine* and *ingredients*. An example of a concrete dish is shown in Figure 2. The Caesar salad belongs to Mexican cuisine (*cuisine* property) and the *type of dish* property is *salad*. We can also find the list of ingredients (*ingredients* property) to prepare the dish.

Now, lets imagine a query for some kind of Spanish and healthy recipe: Q={cuisine = Spanish cuisine, ingredient = olive oil}. Supposing that both attributes are in Freebase ($m = 2$), we could build two partial sets of cases. First, we retrieve dishes from Spanish cuisine (B_1) and later dishes with olive oil as ingredient (B_2). The last step of the algorithm combines these results and tries to fill the remaining attribute values if possible.

The division into retrieval subsets is because the connector performs exact retrieval. If we ask for all the Spanish dishes with olive oil, we would get a very limited retrieval set. Because of that, we work locally with the union of both sets: dishes of a specific cuisine plus dishes with a concrete ingredient.

5.2 Web Retrieval vs. Local Retrieval

Once the Web retrieval algorithm is defined, it is possible to compare the cases obtained from the Web with cases obtained using a typical local case base. To perform this comparison we can check how many of the recipes given by an *offline* application are available at the Web source, focusing on the percentage of repeated information.

The application with a local case base to be compared is ColibriCook [4]. ColibriCook is a system that participated in the Computer Cooking Contest 2008 and works with a local case base of approximately 1000 recipes. In this section we compare the cases retrieved by the ColibriCook system with the dishes recovered from Freebase.

Firstly, we generated random queries for different types of cuisine (Mexican, Indian, Mediterranean, Oriental and unknown cuisine). For all the cases retrieved by ColibriCook we observed that only 31% were included in Freebase. Regarding the *type of dish* attribute, we generated random queries like salads, soups, cakes, etc. This time, only 14% of cases were found on-line. This result led us to skip this attribute in our *WebRetrieval* algorithm. Finally, we obtained how many recipes from ColibriCook were included in Freebase when asking for one ingredient ($j = 1$). We queried dishes with several ingredients without taking into account if they were the main ingredient or not. We measured how many dishes out of the 34 provided by ColibriCook were found in the on-line source. The result was 10 (29%). The explanation for this result is that some of the dishes were impossible to obtain using the ingredient we were trying to query about. It is a real example of the main disadvantage of Web sources: incompleteness of the information.

Summarizing, we found about 30% of the local dishes. Obviously, it is clear that using only Web sources in CBR applications will not be enough to achieve an acceptable performance. Therefore, these results illustrate the importance of a good balance between the local case base and the on-line source in this kind of systems.

5.3 Local Retrieval Combined with Web Retrieval

In this section we explain an algorithm to combine local searches with on-line information. The general idea consists on retrieving cases from the local case base (kc_{cb}), but if they are not *similar enough*, the system will try to extend the retrieval set with cases from Web (kc_{wcb}).

Formally, the **CombinedRetrieval(Q)** algorithm follows these steps:

1. Let $Q = \{\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, \ldots, \langle a_n, v_n\rangle\}$ be the user query.
2. Let $RS_{local} = Retrieve(Q, kc_{cb}, k)$ be the result of retrieving k cases from the local case base.
3. if $similarity(RS(i)) < \delta, \forall i : 1 \leq i \leq k$ then
   (a) $RS_{Web} = WebRetrieval(Q)$ (algorithm detailed in Section 5.1)
   (b) $RS = RS_{local} \bigcup RS_{Web}$
   else $RS = RS_{local}$

Lets illustrate the algorithm with a real example. Imagine we want *sushi* for lunch and we define the query $Q =$ { *ingredient=fish, cuisine=Japanese* }. None of the retrieved cases is sushi because this dish is not stored in our local case base. Searching only by oriental dishes we could find recipes like *Chinese barbecue* or *Oriental Ginger chicken* which do not even look like sushi. Adding fish to the query, we find more similar dishes like *Chinese Cashew Nut Prawns* or *Chinese Spiced Shrimp* as the most similar recipes to the query. However these dishes are still not similar enough to the query, so we try the on-line source.

A new query is built to access to the Freebase database. This second query will be divided into two parts. On one hand we ask for dishes with the ingredient fish, and on the other hand, we ask for all the dishes included in Japanese

cuisine. The results obtained from Freebase are: $Web_{results}$ ={Sushi, Fish and chips, Sashimi, Gefilte fish, Kedgeree, Nori, Tonkatsu, Soy sauce}. Finally, we join both sets, $Local_{results}$ and $Web_{results}$. This new retrieval set -that does include sushi recipes- is used to select the k-NN (through any of the similarity measures provided by the framework) and perform the adaptation. This way, we have complemented the local case base with new information from the Web. Depending on the design of the application, the new recipes would be also stored in the local case base for future reuse.

5.4 Using Web as Adaptation Knowledge

This section describes the use of Web knowledge in the adaptation process. We follow the transformational approach introduced in [5] where unsuitable attributes are replaced by looking in the case base for cases containing better values according to the query. Our novelty consists on reusing the cases obtained from the Web to confirm correct substitutions. This way, we do not find in the Web new values for incorrect attributes as in [8], where the system searches the Web to find new suitable values. Instead, we do confirm a given substitution using only the local knowledge. We propose the simple algorithm that follows.

Algorithm **WebAdaptation(Q)**:

1. Let $Q = \{\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, \ldots, \langle a_n, v_n\rangle\}$ be the user query.
2. Let $RS_{local} = Retrieve(Q, kc_{cb}, k)$ be the result of retrieving k cases from the local case base.
3. $C = \{\langle a_1, v'_1\rangle, \ldots, \langle a_j, v'_j\rangle, \ldots, \langle a_n, v'_n\rangle\}$ is the 1-NN from RS_{local}.
4. Let $\langle a_j, v'_j\rangle$ be an attribute to be substituted in C, and v''_j the new value for a_j.
5. Let $M = \{Atr(Q) \cap Atr(C) + \langle a_1, v''_1\rangle\}$ be the set of common attribute values of Q and C plus the attribute to be substituted.
6. Lets build a query for Freebase Q_{FB} using the attributes in M.
7. If exists a Freebase topic matching Q_{FB} it means the attribute can be substituted.

To understand the algorithm we can use a simple example. Let us make a query Q to get a dish with *potato*. We assume that we are working with a local data base of recipes. In our particular example $Q = \{ingredient_1 = potato, ingredient_2 = salt\}$. Executing this request with the recipes base of ColibriCook we can get a solution like:
$C = \{ingredients = \{chicken, raisin, macaroni, yam, scallion, mayonnaise,$ $sour\ cream, red\ chilli, salt\}, type = salad\}$. Here potato has been replaced by *yam*. Our algorithm will check if this substitution is correct. Therefore, we construct a Freebase query to ask if there is any other salad (because the *type* of dish is *salad*) with the ingredients of the query Q that are not present in C plus the substitution candidate. This way, we will look for any dish in Freebase with $type\ of\ dish = salad$ and $ingredients = yam$. There is one, "Gado-gado". It means that *yam* is a suitable substitute for *potato* because there is another salad with *yam* and *salt*.

6 Conclusions

This paper analyses several approaches to include Web knowledge in CBR applications. We have generated a theoretical framework to characterize the different architectural and functional modifications required to extend the local knowledge of CBR systems with knowledge acquired from Web sources. This theoretical framework is complemented with a comparative study of available Web services that could be exploited to obtain cases and background knowledge for CBR developments. Then, one of these sources -Freebase- is selected to implement a real case of study by means of the jCOLIBRI2 platform. This case of study uses the cooking domain to exemplify the possibilities of obtaining cases from the Web.

Firstly, we develop an algorithm to populate the case base with Web experiences. Results show that using only Web sources may lead to poor results due to the incompleteness of on-line information. Therefore, we conclude that the best design option is the combination of both local and Web knowledge. Consequently we present a second algorithm that performs this combination. Our approach is to create an initial retrieval set using the local case base, and complement it if cases are not similar enough to the given query. Finally, an algorithm to perform a simple adaptation process is also described.

Web sources are a very promising research line within CBR, and this paper points out several approaches to exploit this kind of knowledge. Our immediate future work consists of exploiting the knowledge acquired through our Web connector for other tasks like more elaborated adaptation techniques. We also plan to perform a comprehensive study on the quality (not only the quantity) of the retrieved sets, together with the effectiveness of the proposed algorithms for other domains.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Commun. 7(1), 39–59 (1994)
2. Asiimwe, S., Craw, S., Taylor, B., Wiratunga, N.: Case authoring: From textual reports to knowledge-rich cases. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 179–193. Springer, Heidelberg (2007)
3. Bogaerts, S., Leake, D.B.: Increasing ai project effectiveness with reusable code frameworks: A case study using iucbrf. In: Procs. of the FLAIRS Conference, pp. 2–7. AAAI Press, Menlo Park (2005)
4. DeMiguel, J., Plaza, L., Díaz-Agudo, B.: Colibricook: A CBR system for ontology-based recipe retrieval and adaptation. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 199–208. Springer, Heidelberg (2008)
5. Díaz-Agudo, B., Plaza, E., Recio-García, J.A., Arcos, J.L.: Noticeably new: Case reuse in originality-driven tasks. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 165–179. Springer, Heidelberg (2008)

6. Díaz-Agudo, B., González-Calero, P.: An Ontological Approach to Develop Knowledge Intensive CBR Systems. In: Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems, December 2006, pp. 173–214. Springer, Heidelberg (2006)
7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRC Press, Boca Raton (2009)
8. Leake, D.B., Powell, J.H.: Knowledge planning and learned personalization for web-based case adaptation. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 284–298. Springer, Heidelberg (2008)
9. Leake, D.B., Sooriamurthi, R.: Automatically selecting strategies for multi-case-base reasoning. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 204–233. Springer, Heidelberg (2002)
10. McSherry, D.: Automating case selection in the construction of a case library. Knowl.-Based Syst. 13(2-3), 133–140 (2000)
11. Plaza, E.: Semantics and experience in the future web. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 44–58. Springer, Heidelberg (2008)
12. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A., Sánchez, A.: Ontology based CBR with jCOLIBRI. In: Procs. of the 26th SGAI Int. Conference AI-2006, Springer, Heidelberg (2006)
13. Richter, M.: The knowledge contained in similarity measures. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, Springer, Heidelberg (1995)
14. Scime, A. (ed.): Web Mining: Applications and Techniques. Idea Group, USA (2004)
15. Smyth, B., McKenna, E.: Competence models and the maintenance problem. Computational Intelligence 17(2), 235–249 (2001)
16. Stahl, A., Roth-Berghofer, T.R.: Rapid prototyping of CBR applications with the open source tool myCBR. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 615–629. Springer, Heidelberg (2008)
17. Wiratunga, N., Koychev, I., Massie, S.: Feature selection and generalisation for retrieval of textual cases. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 806–820. Springer, Heidelberg (2004)
18. Wiratunga, N., Lothian, R., Massie, S.: Unsupervised feature selection for text data. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 340–354. Springer, Heidelberg (2006)

Taxonomic Semantic Indexing for Textual Case-Based Reasoning*

Juan A. Recio-Garcia[1] and Nirmalie Wiratunga[2]

[1] Universidad Complutense de Madrid, Spain
jareciog@fdi.ucm.es
[2] Robert Gordon University, Aberdeen, United Kingdom
n.wiratunga@rgu.ac.uk

Abstract. Case-Based Reasoning (CBR) solves problems by reusing past problem-solving experiences maintained in a casebase. The key CBR knowledge container therefore is its casebase. However there are further containers such as similarity, reuse and revision knowledge that are also crucial. Automated acquisition approaches are particularly attractive to discover knowledge for such containers. Majority of research in this area is focused on introspective algorithms to extract knowledge from within the casebase. However the rapid increase in Web applications has resulted in large volumes of user generated experiential content. This forms a valuable source of background knowledge for CBR system development. In this paper we present a novel approach to acquiring knowledge from Web pages. The primary knowledge structure is a dynamically generated taxonomy which once created can be used during the retrieve and reuse stages of the CBR cycle. Importantly this taxonomy is pruned according to a clustering-based sense disambiguation heuristic that uses similarity over the solution vocabulary of cases. Algorithms presented in the paper are applied to several online FAQ systems consisting of textual problem-solving cases. The goodness of generated taxonomies is evidenced by improved semantic comparison of text due to successful sense disambiguation resulting in higher retrieval accuracy. Our results show significant improvements over standard text comparison alternatives.

1 Introduction

Text comparison is important in many research areas including IR, NLP, semantic web, text mining and textual case-based reasoning (TCBR). In TCBR as with traditional CBR the aim is to compare a problem description with a set of past cases maintained in a casebase with the exception that descriptions are predominantly textual. The final aim is to reuse solutions of similar cases to solve the problem at hand [22]. Clearly ability to compare text content is vital in order to identify the set of relevant cases for solution reuse. However a key challenge with text is variability in vocabulary which manifests as lexical ambiguities such as the polysemy and synonymy problems [19]. Usually the similarity metric used to compare cases rely on both general purpose lexical resources (such as Thesauri and dictionaries) and hand-built domain-specific knowledge structures (such as ontologies or taxonomies). Naturally domain-specific resources are more

* Supported by the Spanish Ministry of Science and Education (TIN2009-13692-C03-03) and the British Council grant (UKIERI RGU-IITM-0607-168E).

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 302–316, 2010.

attractive since general purpose resources lack coverage. However coding extensive knowledge for each CBR application is costly and make it appealing to have tools to learn this knowledge with minimum human intervention [10].

The Web contains a rich source of experiential knowledge and for CBR the challenge is to develop tools to filter and distill useful information to augment the CBR cycle [15]. Clearly the coverage of knowledge and domain-independence is a strength but the risk of irrelevant content extraction is a threat. Still the Web has successfully been used as a resource for similarity knowledge for NLP applications [20], Ontology matching [12] and word sense disambiguation [3]. Generally co-occurrence statistics from retrieved documents are used to quantify inter-relatedness of sets of keywords. In this paper we are interested in gathering a taxonomy to capture the semantic knowledge in textual cases that cannot be obtained through statistical methods alone. Our contribution is two folds: firstly we propose to guide the taxonomy generation process using a novel CBR-specific disambiguation algorithm and secondly case comparison is improved by means of Taxonomic Semantic Indexing, a novel indexing algorithm that utilises the pruned taxonomy. While this approach has the advantages from using the Web as background knowledge, it provides an elegant solution to address the tradeoff between the use of multiple web resources with greater coverage and irrelevant content extraction.

Related work in text representation appears in Section 2. Our approach to case indexing is discussed in Section 3 followed by the taxonomy generation process in Section 4. The case-based disambiguation algorithm which we use to prune the taxonomy is presented in Section 5. Next Section 6 presents experimental results followed by conclusions in Section 7.

2 Related Work

Variability in vocabulary and the related sparse representation problem is common to many research areas involving textual content. Solving this requires that generalised concepts are discovered to help bridge the vocabulary gap that exists between different expressions of similar meaning terms. Research in Latent analysis such as Latent Semantic Indexing [5] and latent Dirichlet Allocation [2] do exactly this by creating representations of original text in a generalised concept space. A difficulty with these approaches is that they generate non-intuitive concepts that lack transparency. These concerns have to some extent been addressed by word co-occurrence techniques [23] and in related work where taxonomic structures are generated from text collections [4]. The latter embody richer semantics and is particularly well suited for textual case comparison. Still since co-occurrence statistics are often poor (also due to sparsity), distributional distance measures are needed instead to capture higher-order co-occurrence statistics [24]. However all these techniques tend to be computationally expensive with repetitive calculations and lack scalability.

Work in word sense disambiguation (WSD) pioneered by the classical Lesk algorithm establishes the meaning of a piece of text by examining adjacent words [11]. Essentially a dictionary definition that maximises the overlap with adjacent terms is used to disambiguate the original text. This key idea has since been applied in conjunction with the popular human-edited Thesauri Wordnet, whereby the correct sense of a given

word is determined based on the overlap with example sentences called glosses associated with each candidate sense [1]. However Wordnet is general-purpose and lacks coverage essential for domain-specific tasks and so resources with greater coverage such as Wikipedia are more attractive. The popular bag-of-concept (BOC) representation introduced in Explicit Semantic Analysis (ESA) treats each Wikipedia page as a concept and each word in a case is represented by a Wikipedia concept vector [7]. Essentially the vector captures the importance of a given domain-specific word within Wikipedia pages. Since individual words are mapped onto a common concept space a granular yet generalised representation is obtained to help resolve typical ambiguities in meaning. Although ESA's BOC approach is particularly appealing it is arguable that techniques need not to be restricted to a single web resource and the instantiation of the concept vector need not be restricted to semantics inferred from term frequency alone. Therefore, the contribution of our work to this line of research is to explore how the BOC approach can be applied with semantically richer taxonomic structures derived from multiple heterogeneous Web resources.

Using web search to resolve the sparseness problem is not new [9]. The general idea is to retrieve documents by formulating web search queries that explicitly capture semantic relationships (such as part-of and is-a relationships) using linguistic patterns proposed by Hearst [8]. These relationships are important because case reuse and revision stages in CBR rely heavily on substitutional and compositional knowledge respectively. Typically the strength of the relationship is implicitly captured by the frequency of the recurring pattern in ranked documents. In the PANKOW system such relationships are used to annotate web documents [14] and in more recent work these relationships are combined to generate taxonomic structures [18]. A common advantage with all Web-related approaches is greater knowledge coverage however in reality relationships mined from the web alone can be very noisy. This coverage-noise trade-off needs to be addressed and we propose to do so by guiding the discovery process using word distributional evidence obtained from the case solution vocabulary.

3 Taxonomic Semantic Indexing

The *Bag of Words* (BOW) representation for text uses a word vector and the cosine angle between two vectors to quantify similarity between any two cases, i.e. the smaller the angle the greater the similarity [17]. However, this approach to case comparison fails to capture any semantic relationships between terms. For example, terms *apple* and *banana*, although similar in that they are both fruits will not be captured by a metric that is simply focused on word-word comparisons and will incorrectly result in minimum similarity. If however we were to use a taxonomy where *apple* and *banana* are sub-concepts of *fruit*, then the presence of a common concept refines the distance computation to reflect a greater degree of similarity [13]. This is explicitly achieved by extending the vector representation of each case to include new concepts (such as fruit) with suitable weights. Indexing cases in this manner using "Bag of Concepts" (BOC) is referred to as Explicit Semantic Indexing and in [7] BOC extracted from Wikipedia alone significantly outperformed BOW.

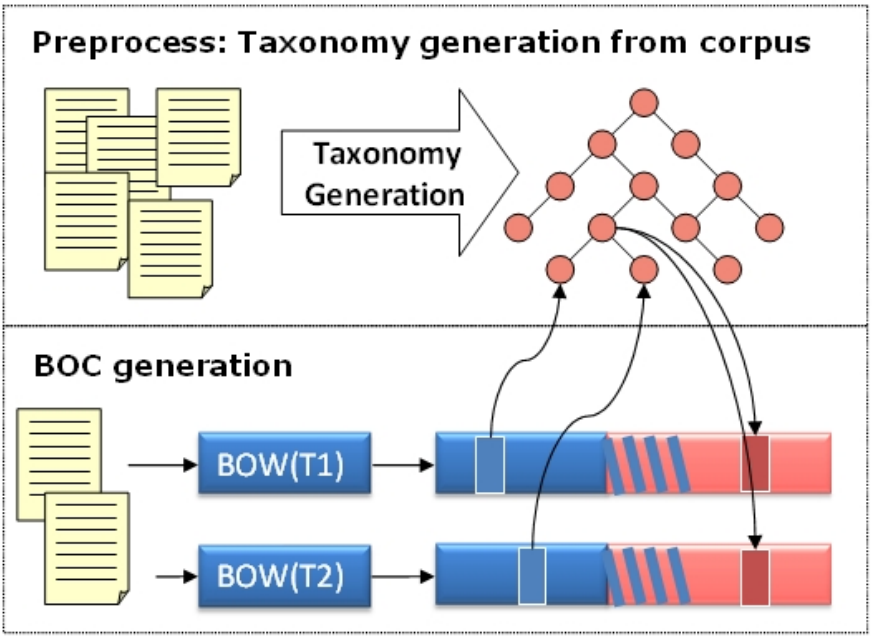


Fig. 1. Taxonomic Semantic Indexing

Figure 1 illustrates our Taxonomic Semantic Indexing (TSI) approach where the BOC obtained from a taxonomy extends the BOW representation. Here the semantic knowledge is encapsulated in a taxonomy, $\mathcal{T}_i = \cup\langle h_-, h^-\rangle$, where $\langle h_-, h^-\rangle$ is a hypernym-hyponym relationship pair. A hypernym, h^-, is a term whose semantic range includes the semantic range of another word called hyponym, h_-. In our example *fruit* is the hypernym of *apple*, whilst *apple* is the hyponym in this relationship. $\mathcal{T}$ is recursively extracted from the Web and is detailed in Section 4.

We formalise TSI as follows. Given a textual casebase CB, represented using a vocabulary $\mathcal{V}$, and composed of a set of textual cases $C = \{C_1, \ldots, C_i, \ldots, C_{|CB|}\}$, each case represented using $BOW(C_i) = \{w_1, \ldots, w_j, \ldots, w_{|\mathcal{V}|}\}$ is extended with hypernyms obtained from $\mathcal{T}$ to form the extended BOC representation:

$$BOC(C_i) = \{w_1, \ldots, w_{|\mathcal{V}|}, h_1^-, \ldots, h_{|\mathcal{T}|}^-\} | \exists \langle w_i, h_j^-\rangle \in \mathcal{T}$$

This can be summarised as follows:

$$BOC(C_i) = \{BOW(C_i), h_1^-, \ldots, h_{|\mathcal{T}|}^-\} | \{\exists \langle w_i, h_j^-\rangle \in \mathcal{T}\} \wedge \{w_i \in BOW(C_i)\}$$

Note that every leaf of $\mathcal{T}$ corresponds to a w_i whilst internal nodes may or may not. This is because the hypernyms in the taxonomy are usually extracted from the Web and are not likely to be in $\mathcal{V}$. These new hypernyms forming vocabulary $\mathcal{V}'$, extend the original vocabulary, $\mathcal{V}$, to create the extended vocabulary $\mathcal{E} = \mathcal{V} \cup \mathcal{V}'$. Accordingly internal nodes in our taxonomy can consists of both original terms or concepts discovered from the Web.

4 Taxonomy Generation

Hyper-hyponym relationships are ideal for taxonomy creation because they capture the *is-a* relation that is typically used when building ontologies. The basic Hearst extraction patterns summarize the most common expressions in English for $\langle h_{\_}, h^{-} \rangle$ relationship discovery. Expressions like "X such as Y" (also "X including Y", and "X especially Y") are used to extract the relationship "Y is a hyponym of X". For example, given the term "food" a search for "food such as" in the text "food such as grapes and cereal" will discover hyponyms "grapes" and "cereal". In reality the set of candidate hyponyms needs filtered so that irrelevant relationships are removed. Therefore taxonomy generation can be viewed as a 2-staged search-prune process which when repeated on newly discovered terms generates the taxonomy in a top-down manner [18]. Note that the search step can be performed using web search engines (or search restricted to a local document corpus).

TSI presented in the previous section, calls for a bottom-up taxonomy discovery approach, because the BOC representation is based on finding h^{-} from $h_{\_}$ in $\mathcal{V}$ (and not $h_{\_}$ from h^{-}). Therefore we need to start with leaf nodes corresponding to terms in $BOW(C_i)$ and progressively extract higher-level concepts from the Web. Hearst's patterns can still be used albeit in an inverse manner. For example to extract h^{-} for term "fish" we can use the pattern "X such as fish", where *X* is our h^{-}. We have also had to refine these inverse patterns in order to remove false positives that are common due to problems with compound nouns and other similar grammatical structures. Figure 2 summarises the original Hearst's and our modified inverse patterns. Here NP refers to the Noun Part-Of-Speech tag and the extracted terms are in bold font. With the inverse patterns (unlike with the original patterns), *queryTerm* is the hyponym and tagged as NP. The negation is included to avoid compound nouns, which have the tendency to extract noisy content. For example, the query "such as car" returns the following snippet: *"Help and advice is always available via the pedal cars forum on topics such as car design, component sources, ..."* where topic could be a valid hyper-term for design but not car.

In order to generate the bottom-up taxonomy a search query is formulated for every term in $\mathcal{V}$ based on the inverse Hearst's pattern [1]. We found that both the singular and

Original Hearst Patterns
queryTerm {,} including {**NP**,}* {or—and} **NP**
queryTerm {,} such as {**NP**,}* {or—and} **NP**
queryTerm {,} especially {**NP**,}* {or—and} **NP**
Inverted Hearst Patterns
¬NP **NP** including *queryTerm* ¬NP
¬NP **NP** such as *queryTerm* ¬NP
¬NP **NP** especially *queryTerm* ¬NP

Fig. 2. Original Hearst patterns and adapted inverse patterns for hypernym extraction

[1] Search engines such as Yahoo!, Google or Bing can be queried. Bing was best whilst Yahoo had long response times and Google's API returns just 10 results per query.

Term: insomnia. Pattern/Query: “such as insomnia”
... Another effect is said to be, having sleeping **problems** *such as insomnia* or having nightmares. Not wanting to go to school is suggested to be an effect of bullying for many ... ... Stress is the main cause of **illness** *such as insomnia*,bad memory, bad circulation and many more. The need for healing is ... ... Acupuncture and TCM is also a very powerful means of treating emotional problems and the physical manifestations that can arise as a ~~result~~ *such as insomnia*, headache, listlessness ...
Term: acne. Pattern/Query: “such as acne”
... is a leader in all-natural skin and body care products for **problems** *such as: acne*, cold sores, menstrual cramps ... ... Do you suffer from troublesome skin or problem ~~skin~~ *such as acne* or rosacea? An American pharmacist ...
Term: breakout. Pattern/Query: “such as breakout”
... Casting growth instability leads to a variety of process and product **problems** *such as breakout*, undesirable metallurgical structures, surface and subsurface cracking, and ... ... Like the original, it is patterned after classic ball-and-paddle arcade **games** *such as Breakout* and Arkanoid ... ... Check back for more material and ~~information~~, *such as: Breakout* Session Summaries; Misc. Forms/Press Releases; ...

Fig. 3. Examples of web search results in response to queries formed using inverse patterns

plural forms of terms need to be encoded in these patterns[2]. In the interest of efficiency only hyponyms contained in summary snippets generated by the search engine were extracted. Finally the most frequent hyponyms in snippet text are considered for the taxonomy.

Figure 3 presents some real examples of inverse patterns an their corresponding extraction results from a web search engine. This table shows some noisy results (strike-out font) to illustrate the difficulties of the method. Here a common hypernym such as “problems” is linked with three different terms: “insomnia”,“acne”, and “breakout”. Clearly the first and second terms have the semantic sense of health problems, whilst the last one is completely different and should ideally be pruned. We next present a disambiguation algorithm to prune TSI’s taxonomies.

5 Pruning as Disambiguation

Knowledge extracted from the Web may contain relationships that are contextually irrelevant to the TCBR system. For example in a cooking domain whilst *fruit* is a sensible hypernym extraction for *apple*; *computer* is not! The question is how can we detect these noisy relationships in order to prune our taxonomy?

[2] For example, the term “biscuit” does not return any h^-s with query pattern “such as biscuit”. However its plural form does.

Verification patterns with a conjunction is commonly used for this purpose: "h^- such as h_- and * "; checks if an extracted hypernym (h^-) is also a common parent to a known hyponym (h_-) and other candidate hyponyms($*$) [25]. Probabilistic alternatives include the test of independence using search engine hits:

$$hyponymProbability(h^-, h_-) = \frac{hits(h^- \; AND \; h_-)}{hits(h^-)hits(h_-)}$$

However in reality all such verifications require many queries to Web search engines slowing system performance, and crucially fail to incorporate contextual information implicit in the casebase. For example candidate hyponyms *apple* and *banana* for hypernym *fruit* are likely to be used in a similar context within a cooking casebase. In contrast, the candidate hyponym *oil* incorrectly related[3] to *fruit* will have a different context to that of *apple* or *banana*.

5.1 Creating a Hyponym Context

Context of a term is captured by its co-occurrence pattern. Often, related words do not co-occur, due to sparsity and synonymy. Therefore co-occurrence with a separate disjoint target set such as the solution vocabulary is used instead [24]. Essentially for a given hypernym its candidate hyponyms can be pruned by comparing their distributions conditioned over the set of solution words. The intuition is that hyponyms having the same hypernym should also be similarly distributed over the target vocabulary. Therefore the more similar the conditional probability distributions of candidate hyponym terms the more likely that the extracted hypernym is correct.

5.2 Disambiguation Algorithm

In this section we formalise our disambiguation algorithm with which we prune the taxonomy.

1. A case base CB contains a set of cases, where each case C, is a problem-solution pair $\langle p, s \rangle$. Accordingly the case base CB can further be viewed as consisting of instances of problems and solutions $CB = \langle P, S \rangle$ and related vocabularies. We define problem $\mathcal{V}_p$ and solution $\mathcal{V}_s$ vocabularies as:

$$\mathcal{V}_p = \{\cup \, w \in P \mid relevance(w) > \alpha\}$$

$$\mathcal{V}_s = \{\cup \, w \in S \mid relevance(w) > \alpha\}$$

   where $relevance(w)$ measures the relevance of the term w in the corpus (usually based on TFIDF filtering). Here α is a term selection parameter.
2. The context $\varphi_{w,C} \in \mathbb{R}^{|\mathcal{V}_s|}$ of a term w in a case C as the frequency vector of the terms in $\mathcal{V}_s$. Note that $\varphi_{w,C}$ is only computed if w appears in the problem space ($w \in \mathcal{V}_p$).

[3] From Web snippet "brines contain components from fruit (**such as oil**, sugars, polyols and phenolic compounds)

Any term in a problem description should have a unique context. However we have used just the frequency vector to simplify the computation of the context. Future implementations could compute $\varphi_{w,C}$ as any other function that reflects the relationship of a term w with other terms in C. One possibility is the conditional probability.

3. Given two terms $x, y \in \mathcal{V}_p$ with a common hypernym candidate h^-, we obtain the relevant sets of cases RS_x and RS_y containing x or y in the problem description:

$$RS_x = \{\, C \mid \{x \in P\} \wedge \{C = \langle P, S\rangle \in CB\}\,\}$$

$$RS_y = \{\, C \mid \{y \in P\} \wedge \{C = \langle P, S\rangle \in CB\}\,\}$$

4. We generate the similarity matrix $X_{|RS_x|\times|RS_x|}$ by computing all pair-wise similarities between members in RS_x (solution parts of cases containing x in their description). This similarity matrix represents the distance between the contexts φ_{x,C_i}, reflecting the possible senses of x.

$$Xij = distance(\varphi_{x,C_i}\ ,\ \varphi_{x,C_j})\ where\ \forall C_j, C_i \in RS_x$$

A suitable distance metric such as Euclidean, Cosine or KL-Divergence is used. Analogously, we compute the matrix $Y_{|RS_y|\times|RS_y|}$:

$$Yij = distance(\varphi_{y,C_i}\ ,\ \varphi_{y,C_j})\ where\ \forall C_j, C_i \in RS_y$$

Analyzing the similarity matrix X or Y we can infer different senses of a term. Terms with similar senses will conform groups that can be obtained by a clustering algorithm. If the clustering results in one cluster, it indicates only one sense, whilst multiple clusters suggest different senses of that term.

5. For candidate hyponym x we generate a clustering G^x consisting of independent groups $\{g_1^x, \ldots, g_n^x \mid \cap g_i^x = \emptyset\}$. A group g^x consists of a set of similar contexts for the term $x : \{\varphi_{x,1}, \ldots, \varphi_{x,m}\}$. We also compute the analogous clustering for term candidate hyponym y: $G^y = \{g_1^y, \ldots, g_m^y \mid \cap g_i^y = \emptyset\}$.
6. To determine the validity of a hypernym h^- we compute distance between every pair of groups $\langle g^x, g^y\rangle$. Groups with similar senses are expected to be more similar. We use a distance based heuristic to determine the validity of the candidate hypernym h^-. Distances between group centroids are computed as follows:

$$Distance(g^x, g^y) =$$

$$distance(\overline{\varphi_{x,i}}, \overline{\varphi_{y,j}})\ where\ \varphi_{x,i} \in g^x \wedge \varphi_{y,j} \in g^y$$

7. If $Distance$ is below some predefined threshold ($< \beta$) we can infer that both terms x and y are used in a similar sense and, therefore, the candidate h^- is valid, otherwise $\langle x, h^-\rangle$ and $\langle y, h^-\rangle$ relationships are deleted. When hypernyms are deemed valid, we annotate documents in g^x and g^y with the new concept h^-. Moreover we could also include the $Distance$ value with the BOC representation.

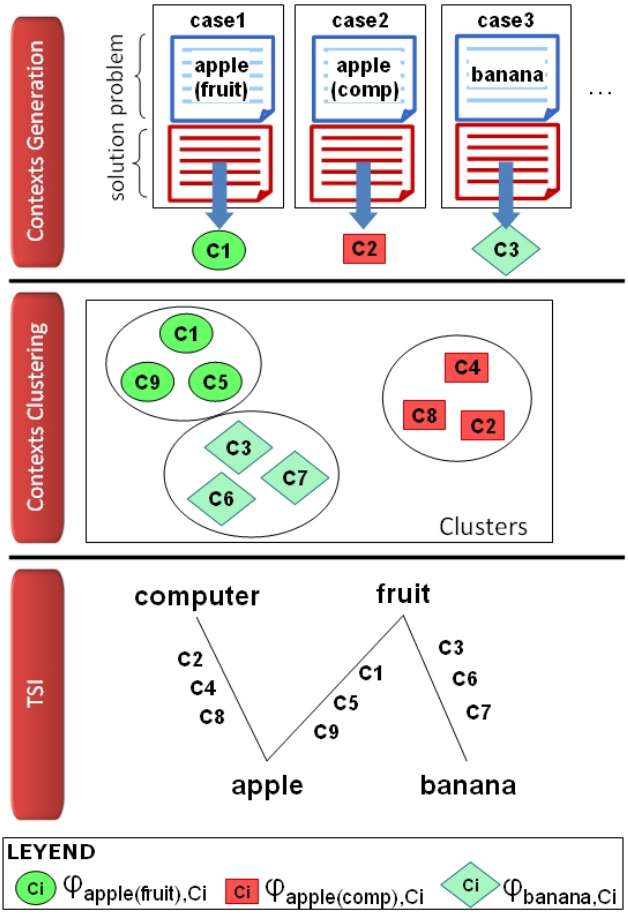


Fig. 4. Disambiguation process using the solution space

The end product is a *pruned taxonomy* where noisy relationships are deleted and every hypernym-hyponym relationship is associated with the list of documents where this relationship holds. Figure 4 summarizes the process using the example discussed in this Section. Contexts representing the terms "apple" -meaning fruit- and "banana" have a short distance, however the group of contexts capturing the sense of "apple" but meaning computer are in a distant cluster.

The list of relationships listed in Figure 5 represent actual examples of sense disambiguation when generated taxonomies were pruned in our experimental evaluation. Here strikeout relationships denote noisy hyponyms that were removed by the disambiguation algorithm for the sample domains.

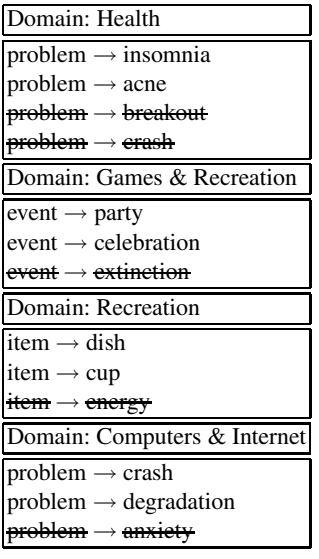


Fig. 5. Disambiguation examples

6 Experimental Evaluation

The aim of our evaluation is to establish the utility of TSI with and without disambiguation when compared to standard case indexing with BOW representations. Therefore a comparative study is designed to compare the following case indexing algorithms:

- *BOW*, Bag-Of-Words representation;
- *BOC*, Bag-of-Concepts with TSI where the pruned taxonomy is obtained with the disambiguation algorithm; and
- *BOC with no disambiguation*, using TSI with an unpruned taxonomy.

These algorithms are compared on FAQ recommendation tasks. Like help-desk systems a FAQ recommender system retrieves past FAQs in response to a new query. In the absence of user relevance judgments system effectiveness is measured on the basis of FAQ classification accuracy using a standard k-NN algorithm (with k=*3*). Since each case belongs to a predefined category we can establish classifiers accuracy for each indexing algorithm. Significant differences are reported using the Wilcoxen signed rank test with 99% confidence. Individual recommender systems are built using jCOLIBRI, a popular CBR reference platform [6], with the Taxonomic knowledge structure integrated with CBR's indexing knowledge container [16]. Our implementation uses (besides the standard Textual-CBR capabilities of jCOLIBRI) the Lucene toolkit[4] to organize and filter the texts, the snowball[5] stemmer, the OpenNLP[6] part-of-speech tagger and the

[4] http://lucene.apache.org
[5] http://snowball.tartarus.org/
[6] http://opennlp.sourceforge.net/

Morphadorner lemmatizer[7] to obtain the singular and plural forms of web query terms. We have also developed several components to automatically connect to the search engine APIs, submit search queries and process retrieved results.

6.1 Datasets

Several Web-based FAQ casebases were extracted with the online Yahoo!Answers[8] Web site. Here thousands of cases in the form of question-answer pairs are organized into topic categories. Importantly textual content from these FAQs can be extracted dynamically through a public Web API. Therefore, for a given FAQ category (e.g. health, sports etc.) a set of textual cases can be extracted dynamically from Yahoo!Answers. This flexibility enabled us to extract casebases for six textual CBR systems from the Web. Essentially FAQ's corresponding to 12 different categories were extracted and later grouped together to form 6 casebases with each containing cases from no more than 3 categories. These groupings were made with the intention to create casebases with similar, whilst others with quite distinct categories. For example, corpus A contains quite similar texts as its categories are: "Consumer Electronics", "Computers & Internet" and "Games & Recreation". However, corpus C is composed of heterogeneous texts: "Science & Mathematics", "Politics & Government", "Pregnancy & Parenting". Every casebase contained 300 cases (question-answer pairs), with equal distribution of cases i.e. 100 cases per category. The size of the FAQ question description vocabulary $|\mathcal{V}_p| \backsimeq 2000$, whereas solution space $|\mathcal{V}_p| \backsimeq 3500$. In general Taxonomies extracted from the Web contained 550 concepts on average of which 25% were new concepts i.e. the $\mathcal{V}'$ vocabulary.

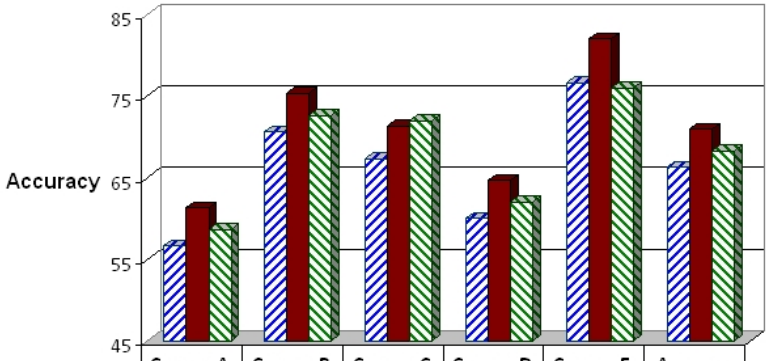
6.2 Experimental Setup

The $relevance(w)$ function in relation to the disambiguation algorithm mentioned in Section 5 is implemented by way of a simple frequency heuristic, where words occurring in more than 50% of the cases are deleted. Removal of very rare words (i.e.: frequency $< 1\%$ of cases) was also considered, however this had a negative impact on overall performance. We found that the FAQ vocabularies were large due to different users generating semantically similar yet lexically different content. Therefore, unlike with standard text classification here rare words were important.

The hypernym candidates obtained from search engines are filtered according to their frequency in Web snippet text. This filtering parameter -named *minimum pattern frequency* (mpf)- was configured experimentally, with best results achieved when $mpf = 6$. The clustering algorithm for the taxonomy relationship disambiguation process used a two threshold sequential algorithmic scheme (TTSAS) [21] with the cluster merge parameter β obtained experimentally ($\beta = 0.1$).

A BOW vector is implemented using standard TFIDF weights [17]. The BOC vector extends the BOW by explicitly inserting new concept values. This value is the concept

[7] http://morphadorner.northwestern.edu/
[8] http://answers.yahoo.com



	Corpus A	Corpus B	Corpus C	Corpus D	Corpus E	Average
Bag of Words	56.66	70.66	67.33	60.00	76.66	66.26
Bag of Concepts	61.33	75.33	71.33	64.67	82.00	70.93
BOC no disambiguation	58.66	72.66	72.00	62.00	76.00	68.26

FAQ domains:

Corpus A	Consumer Electronics, Computers & Internet, Games & Recreation
Corpus B	Health, Pregnancy & Parenting, Beauty & Style
Corpus C	Science & Mathematics, Politics & Government, Pregnancy & Parenting
Corpus D	Cars & Transportation, Games & Recreation, Business & Finance
Corpus E	Sports, Pets, Beauty & Style

Fig. 6. Experimental results

weight, and should ideally be a function of the confidence of the learnt taxonomy relationship and the importance of the hyponym involved in the relationship. Our initial exploration with the concept weight parameter -named *concept activation factor* (caf)-, suggest that $caf = 1.0$ leads to surprisingly good results. Basically copying the same TFIDF value of the hyponym involved in the hypernym for the BOC concept weight. Although multiple levels of hypernym relationships were tested to disambiguate terms, we did not find significant differences with more than one level. This could simply be unique to our sparse corpora where it was unlikely to find a common grandparent for two given terms. Therefore our experiments only use taxonomies generated with one level.

The configuration of the parameters was experimentally obtained for every corpus as our implementation provides a toolkit to obtain it automatically. However, it is important to note that usually optimal results for each corpus shared similar configurations. For example caf and mpf parameters had always the same values for all corpora, existing only small differences in the configuration of the clustering algorithm.

6.3 Evaluation Results

Average accuracy results for each dataset using a 10-fold cross validation experiment is presented in Figure 6. *BOC*'s TSI with pruned taxonomies has resulted in best

performance, whilst *BOC no disambiguation* has also improved upon *BOW*. *BOC* is significantly better than *BOW* with almost a 5% increase across all FAQ domains. As expected *BOC*'s performance is also significantly better than *BOC no disambiguation* in all but Corpus C. Here the disambiguation algorithm has incorrectly pruned some valid taxonomic concepts. Close examination of trials suggest that this domain is very sensitive to the clustering parameter (mainly the β parameter), and tweaking this leads to improvements with taxonomy pruning. However these initial observations call for further study into the relationship between dataset characteristics and parameter setting in the future.

7 Conclusions

The idea of Taxonomic Semantic Indexing (TSI) using multiple heterogeneous Web pages is a novel contribution of this paper. Use of contextual knowledge to disambiguate and prune the extracted taxonomy presents an elegant solution to the trade-off between knowledge coverage and the risk of irrelevant content extraction from the Web. We achieve this by way of a taxonomy relationship disambiguation algorithm that exploits contextual knowledge that is implicit in the Case-Based Reasoning (CBR) system's case solution vocabulary.

TSI can be viewed as an unsupervised approach to taxonomy generation from the Web and is relevant not only to CBR but also to semantic web, NLP and other related research areas. For CBR, taxonomic knowledge can be utilised for case indexing, retrieval, reuse or even revision. In this paper we evaluate the quality of extracted taxonomic knowledge for textual case indexing using several online Textual CBR domains. We employ the bag-of-concept (BOC) approach to extend case representations by utilising *is-a* relationships captured in the taxonomy. Results suggests significant performance improvements with the BOC representation and best results were obtained when taxonomies are pruned using our disambiguation algorithm.

An interesting observation is that there is no obvious manner in which CBR systems using TSI like approaches can be selective about Web resource choices beyond page-rank type search-engine specific rankings. In future work we plan to explore how a feedback mechanism might be built in so as to enable CBR systems to leave feedback annotations on Web resources.

References

1. Banerjee, S., Pedersen, T.: An adapted lesk algorithm for word sense disambiguation using word-net. In: Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, pp. 136–145 (2002)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
3. Cilibrasi, R.L., Vitanyi, P.M.B.: The google similarity distance. IEEE Trans. on Knowl. and Data Eng. 19(3), 370–383 (2007)

4. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. Journal of AI Research 24, 305–339 (2005)
5. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990)
6. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J.A., Sánchez-Ruiz-Granados, A.A.: Building CBR systems with jcolibri. Sci. Comput. Program. 69(1-3), 68–75 (2007)
7. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: Proceedings of The 20th International Joint Conference for Artificial Intelligence, Hyderabad, India (2007)
8. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th Conference on Computational Linguistics, pp. 539–545. Association for Computational Linguistics, Morristown (1992)
9. Keller, F., Lapata, M., Ourioupina, O.: Using the web to overcome data sparseness. In: EMNLP 2002: Proceedings of the ACL 2002 conference on Empirical Methods in Natural Language Processing, pp. 230–237. Association for Computational Linguistics, Morristown (2002)
10. Leake, D., Powell, J.: Knowledge planning and learned personalization for web-based case adaptation. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 284–298. Springer, Heidelberg (2008)
11. Lesk, M.: Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In: Proc. of SIGDOC 1986: 5th International Conference on Systems Documentation, pp. 24–26 (1986)
12. Marta Sabou, M.D., Motta, E.: Exploring the semantic web as background knowledge for ontology matching, 156–190 (2008)
13. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet: similarity - measuring the relatedness of concepts. In: Proceedings of the Nineteenth National Conference on Artificial Intelligence, AAAI 2004 (2004)
14. Philipp Cimiano, S.H., Staab, S.: Towards the self-annotating web. In: WWW 2004: Proceedings of the 13th International Conference on World Wide Web, pp. 462–471. ACM, New York (2004)
15. Plaza, E.: Semantics and experience in the future web. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 44–58. Springer, Heidelberg (2008)
16. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A., Sánchez-Ruiz-Granados, A.: Ontology based CBR with jcolibri. In: Applications and Innovations in Intelligent Systems XIV. SGAI 2006, pp. 149–162. Springer, Heidelberg (2006)
17. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
18. Sanchez, D., Moreno, A.: Bringing taxonomic structure to large digital libraries. Int. J. Metadata Semant. Ontologies 2(2), 112–122 (2007)
19. Simpson, G.B.: Lexical ambiguity and its role in models of word recognition. Psychological Bulletin 92(2), 316–340 (1984)
20. Strube, M., Ponzetto, S.P.: Wikirelate! computing semantic relatedness using wikipedia. In: AAAI 2006: Proceedings of the 21st National Conference on Artificial Intelligence, pp. 1419–1424. AAAI Press, Menlo Park (2006)
21. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 3rd edn. Academic Press, London (2006)

22. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual case-based reasoning. The Knowledge Engineering Review 20(03), 255–260 (2006)
23. Wiratunga, N., Lothian, R., Chakraborty, S., Koychev, I.: Propositional approach to textual case indexing. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 380–391. Springer, Heidelberg (2005)
24. Wiratunga, N., Lothian, R., Massie, S.: Unsupervised feature selection for text data. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 340–354. Springer, Heidelberg (2006)
25. Zornitsa Kozareva, E.R., Hovy, E.: Semantic class learning from the web with hyponym pattern linkage graphs. In: Proceedings of ACL 2008: HLT, pp. 1048–1056. Association for Computational Linguistics, Columbus (2008)

A Case for Folk Arguments in Case-Based Reasoning

Luís A.L. Silva[1,*], John A. Campbell[1], Nicholas Eastaugh[2], and Bernard F. Buxton[1]

[1] Department of Computer Science, University College London
Malet Place, London – UK, WC1E 6BT
{l.silva,j.campbell,b.buxton}@cs.ucl.ac.uk
[2] The Pigmentum Project, Research Laboratory for Archaeology and the History of Art, University of Oxford
Dyson Perrins Building, South Parks Road, Oxford – UK, OX1 3QY
nicholas.eastaugh@pigmentum.org

Abstract. An approach to enhancement of case-based reasoning in situations where substantial amounts of the knowledge are expressed as informal or "folk" arguments is applied to the authentication (dating) of paintings. It emphasizes knowledge acquisition templates, indexing through numerical taxonomy and close attention to typing of the arguments. This work has shown that even simple types can be regarded as attributes of the arguments, hence attributes of their cases. The cases are then organized and retrieved through structuring of the case bases by methods of numerical taxonomy. Expertise expressed as texts of detailed reports on dating of paintings from historical and chemical knowledge is the source material from which the cases are constructed. Case bases with and without folk-argument knowledge, for the same paintings, are compared for their ability to assign correct date ranges. In this test, the performance of the case base containing argument knowledge is consistently superior.

1 Introduction

Field experts explain their knowledge most naturally in informal narrations of problem solving that they call "cases". In many applications, it is not always straightforward to translate such narrations into the general format of case-based reasoning (CBR). It is reasonable to ask what can be done to bridge the consequent gap. The point is to capture as much knowledge as possible from the narrations so that it can be indexed and used in CBR systems.

In previous work [1] we have found two techniques to be particularly helpful in collecting and organizing such case-like knowledge: construction and use of templates (e.g. [2]) to represent the behavior of experts in their field of expertise, and numerical taxonomy [3] to support indexing tasks. These receive some attention in this paper. Most importantly, we have observed repeatedly that experts reveal much knowledge in the form of arguments – in effect, running informal arguments with themselves for and against various hypotheses and features of their case studies. Many parts of these "folk arguments" would find no place in an ordinary CBR format and would probably therefore be overlooked. The method we have developed for

* This work is supported by CAPES/Brazil (grant number 2224/03-8).

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 317–331, 2010.

representing their essential features and using them in CBR, as discussed in [1], evolved first for the application of frequency allocation in shortwave broadcasting.

The present paper treats a second expert activity: authentication of paintings [4]. Here, the folk arguments look qualitatively different in form from those in the shortwave area, and make up the largest parts of art experts' case studies. This is evidence that the successful use of the three concepts mentioned above, and the treatment of folk arguments in particular, is not a byproduct of their adjustment to the initial (shortwave radio broadcasting) application but occurs also in a very different field. Moreover, the application of CBR itself to expertise in art (specifically, the history and physical chemistry of art) appears novel and worth reporting. In particular, we examine how authentication knowledge previously hidden in past painting authentication reports can now be exploited by domain users in proposing case-based assignments of dates for paintings requiring authentication. Validation of the results of the CBR system is made against expert answers for previously authenticated paintings.

2 Background to the Work

Painting authentication can be approached in a variety of ways. For example, it has been connected recently to techniques of machine vision (e.g. [5]). This and other approaches such as machine learning need quite large numbers of paintings and/or significant human effort in collecting and representing appropriate data. While both can contribute useful insights, by far the most thorough treatment is still interpretation by a human expert, expressed in the text of a painting authentication report which states reasons for the eventual (often partly aesthetic and historical) judgment. In the reports we consider as cases in our project, the knowledge is not purely subjective – as it might have been among art experts some time ago – but relies also on observations on pigments through methods of physical chemistry. In practice, such reports are the most accessible source of what one can expect to find in a painting of such and such a period of time and, in some instances, by particular artists.

Rather than presenting a chain of logical reasoning, a report builds up a conclusion by accumulating a weight of evidence. Each piece of evidence is, broadly speaking, a single folk argument, as in items A1 and A2 in Fig. 3. In art authentication, such arguments make up the bulk of a case and contain almost all of the relevant expert knowledge. However, they are very different from what argumentation most often means in CBR, which is influenced by the apparent nature of argumentation in legal applications of CBR [6] and is something formalized via some version of logic [7].

Our framework for capture and use of the consequent rather "knowledge-light" information given by experts emphasizes two ideas which have emerged from our original study of the shortwave broadcasting application. First, in order to organize knowledge reliably, it is necessary to understand the basics of what the expert in a field actually does. We express that in a template (e.g. see Fig. 2), which not only serves as a means of organization but can also act as a strong cue for the expansion or clarification of knowledge when we are questioning or otherwise working with the expert on the development of a CBR system for an application. This idea is well understood in knowledge acquisition in general [2] and evolved first in connection with expert systems projects, but has not been particularly strongly promoted where the knowledge occurs in the form of cases. Second, once the template is in place, it shows

where each individual folk argument belongs in the expert's reasoning and therefore locates those arguments correctly in an overall case's structure.

Folk arguments such as those in Fig. 3 may not be simple to process in computation, but we have found that a reasonable amount of the knowledge they contain, in the context of a case, can be represented by its "argument type" (AT). Such types give folk arguments a practical role in CBR. What types exist in an application is ultimately for the domain expert to decide, but a knowledge engineer rather than an expert can make a first draft of a set of types. The expert is then involved mostly in checking whether and/or where this draft needs amendment. Details of the painting authentication application and its case representation results now follow.

3 The Organization of Painting Authentication Case-Bases

Painting authentication cases are represented by factual and argumentation information. Elemental compositions (i.e. "EDX" compositions from X-ray spectrometry) and paint pigments [8] are the two kinds of factual items in this case representation. Cases were then organized within two distinct case bases: one with elemental compositions ("Element") and one with pigments ("Pigment"). In [9], we have discussed their organization and use in CBR to make dating judgments in painting authentication problems. At present, 48 cases are available as collections of both facts and expert arguments, out of a total of 77 painting authentication reports currently available for study. These 48 "extended cases" form a new kind of case base for the CBR system: the argumentation-based case base ("Argumentation").

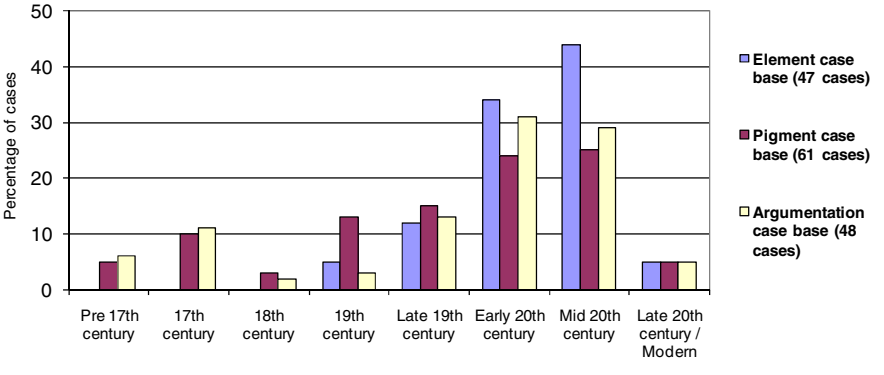


Fig. 1. Cases in different periods of time in the painting authentication case bases

These three case bases do not all contain the same cases: only 33 cases are common. Differences are due to the availability of source case material. As its conclusion, each case contains a date assignment (a range of time). Fig. 1 presents a summary of the time intervals in our set of cases. CBR queries can be submitted to any case base independently, given only that queries submitted to a certain case base should contain the features used in the indexing of that case base. In practice, the three case bases capture alternative perspectives on the authentication problem. Apart from using them

to examine how well the Argumentation case base performs against the others, we foresee that they can be helpful for art specialists and learners to explore the relative strengths and weaknesses of Element and Pigment information for their work.

4 The Organisation of Expert Arguments in Painting Authentication Cases through Knowledge Acquisition Templates

We ensure that we collect expert knowledge systematically by first obtaining an appropriate knowledge acquisition template and then using it to guide the activity of collection. We have followed standard practice for this process [2, 10]. In the present application, we had to construct a template because nothing in the existing template libraries described exactly the expert activity of Painting Authentication. (For our application, this was inference of dating; inference of attribution has further complications. However, the overall expectation is that if the "date" is correct for a painting, there is then a much higher likelihood that the authorship is also correct.) An immediate advantage of a template is, as we indicate below, that it gives strong cues for where to put folk (or other) arguments collected during subsequent knowledge acquisition. Fig. 2 presents our Painting Authentication template, in which "arguments" marks each place where a repository of the relevant arguments can be set up.

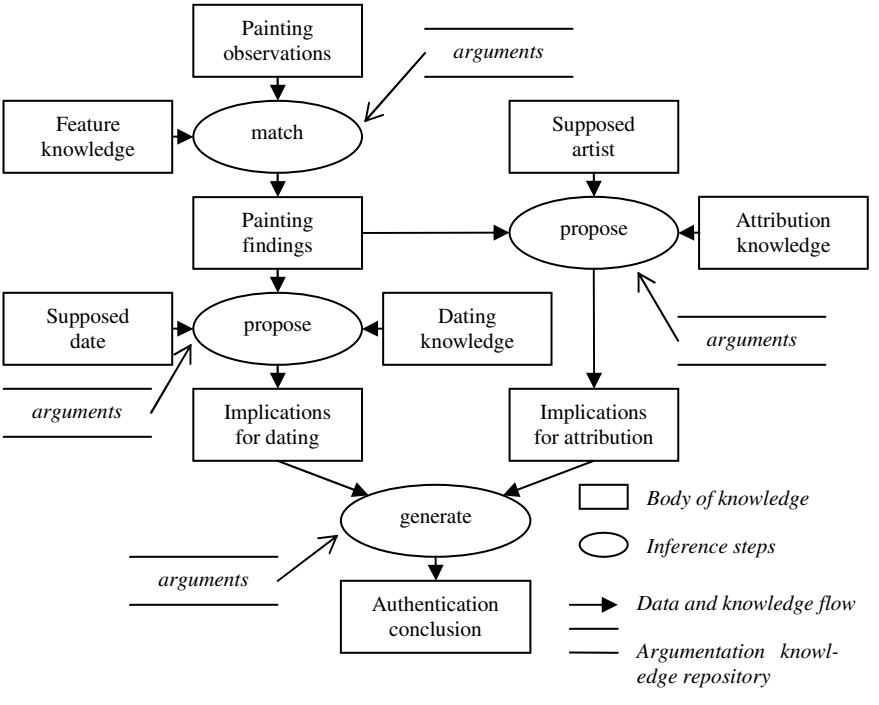
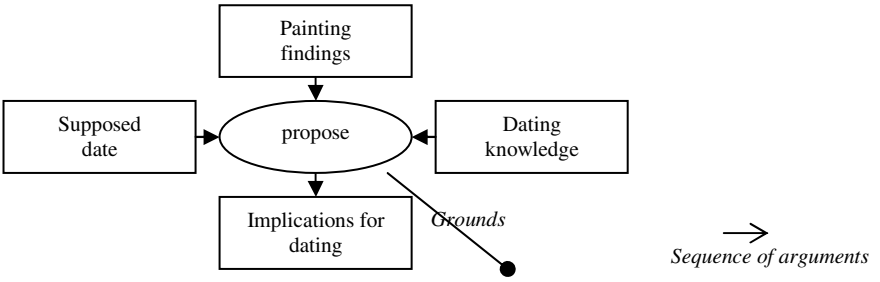


Fig. 2. The Painting Authentication template

Simple painting information can be used to illustrate how this template works in the collection and representation of case knowledge. Consider a target painting which is claimed to be a work dating from 1915 by Natalia Goncharova, a well-known Russian artist. If we look at the bodies of knowledge in the template structure, we see that the "supposed date" is related to 1915 and the "supposed artist" is connected to "Natalia Goncharova". In the analysis of this target painting, elemental composition results from yellow paint samples show that lead and chromium are present. These are the "painting observations" in the target authentication problem. Background knowledge regarding pigment composition indicates that lead and chromium in yellow coloring is probably a lead chromate pigment. Therefore, the match of "painting observations" and "feature knowledge" implies that a lead chromate pigment can be considered as a "painting finding". Following the structure of inference, the next problem-solving steps are the "proposal of implications for attribution" and "proposal of implications for dating". On one hand, attribution knowledge indicates that Goncharova has used lead chromates in other paintings. Therefore, the overall "implication for attribution" is that the painting could be by Goncharova. On the other hand, dating knowledge shows that lead chromate has been available since the 19th century. Such "dating knowledge" is significant to advance the idea that lead chromate is a useful finding for a 1915 painting. This is a valuable "implication for dating" in this target problem. In the end, such implications are used in the generation of authentication conclusions. A typical conclusion in this example is that there is no reason to not believe that the painting is by Goncharova in 1915. This example shows how that information in the problem situation can be associated with steps indicated in the bodies of knowledge represented in the template structure of a case. The structure and labeling of the knowledge acquisition template are also able to orient the identification and recording of expert arguments. In doing so, collections of folk arguments are recorded in the "argumentation knowledge repositories" (denoted by the word "arguments" in Fig. 2). Consequently, the template acts as a framework for the representation of such argumentation information in cases. To build this representation, the major inferences in the template are first selected by the knowledge engineer. Then, folk arguments are collected from the expert's authentication reports where it is clear that they belong in one of the "argument" locations shown in Fig. 2.

Fig. 3 displays a portion of Fig. 2 when the template itself is used as an organizing device for representing argument knowledge. The dating inference presented along with folk expert arguments there refer to one of the cases: the analysis of a painting that is thought to be by Mikhail Larionov, another significant figure in Russian art history. In our project, once collections of folk arguments were recorded in the case structure as shown in Fig. 3, the next task was the annotation of the nature of the information presented in these informal arguments. In that task, a list of different argument types was constructed from the inspection of the painting authentication material. This process involved the construction and refinement of a preliminary list of types, and then the validation/testing of these argument types in the characterization of the folk arguments. In practice, a folk expert argument identified using the template structure is annotated with one or multiple argument type labels. Instead of considering that an expert argument should be summarized by a single argument scheme [11], the use of multiple types of arguments in the annotation of expert arguments was found to be more natural in the treatment of paintings, e.g. because they



(A1) "No pigments inappropriate for a date of 1910 were found. For example, numerous false paintings in the style of the Russian Avant Garde movement contain titanium dioxide white pigments, for which current evidence suggests there was little general use before the 1950s. This pigment was not detected, nor others like it"; *ABSENT painting features; AT01: Correlation of pigments with supposed date and AT07: False paintings and AT04: Style of paintings and AT14: Post-dating painting features*

⋁

(A2) "Several pigments, though still used today, have largely fallen from use. Of these, the presence of the 'dry' process mercury(II) sulfide is of some interest. There has been a shift in production of mercury(II) sulfide as a pigment during the nineteenth and twentieth centuries. … the use of dry-process mercury(II) sulfide in the painting is broadly supportive of an earlier dating"; *PRESENT painting features; AT12: Decline of use of painting features and AT21: Use of material or painting features*

⋁

…

Fig. 3. Some discussion of implications for dating in a case of painting authentication

often contained informal observations that we could not reduce to the simple logical combinations of domain concepts that an effective use of argument schemes requires. This can be expected to happen in other application areas also.

The Painting Authentication template of Fig. 2 along with the examples presented in Fig. 3 show that the assignment of a date to a painting has a more exploratory and open-ended nature than assignment tasks in general [2]. The analysis of date authenticity in paintings thus presents more complexities than the assignment of known objects to determined resources, which is what "assignment" usually means. Any such Painting Authentication template also has some of the basic characteristics of "diagnosis" [12]. Similar to diagnosis, the reasoning relies on findings that are investigated according to some dating hypothesis stated previously (e.g. a range of dates suggested on other grounds). In summary, the Painting Authentication template can be regarded as an extension to template libraries, as it does not occur at present in any existing template catalogue [2, 10]. Moreover, the exploitation of such templates enhances existing CBR, since the structure of the reasoning task can now be utilized to partition and focus the collection of folk arguments from narratives of experts in the field.

5 The Indexing of Folk Argumentation Features through Numerical Taxonomy Techniques

Merely representing facts along with collections of folk arguments in the structure of a case may still not be enough for making key similarity computations in CBR. As

proposed in [1, 9], "numerical taxonomy" [3] can be exploited in the indexing of not only factual attributes, but also collections of folk arguments recorded in these cases. In general, numerical taxonomy is a tool for the study of classification problems. This study involves the numerical evaluation of the similarities between entities (here, cases) and their consequent organization in groups (here, taxonomies) on the basis of such evaluation. Similarities in numerical taxonomy are analyzed by using features (which are not necessarily numerical) of these entities as coordinates and by exploiting metric distance functions (the Euclidean distance form is the most common choice) based on these coordinates. Such indexing functions may use the coordinates with unequal weights. Although an equal weighting scheme is a default choice in numerical taxonomy studies, a trial and error process involving variation of weight values, with feedback from experts, is likely to show the relative importance of different coordinates for the construction of the best taxonomy for an application when there are clear variations in importance. In practice, the way such metric functions are constructed in numerical taxonomy enables the capture of the same kind of information that one expects or hopes to capture in similarity functions in CBR. The outcome of a numerical taxonomy study in such applications is thus a taxonomy whose underlying similarity function is central to the retrieval of cases from case bases in CBR systems. Moreover, such an approach supports the study of applications where there is no theory or other reason to say in advance how to calculate similarities between cases (e.g. in the shortwave radio and painting authentication applications).

In the numerical taxonomy study of cases, factual and folk argumentation attributes are selected and encoded first. Then, estimates of similarity are expressed in separate fact-based and argumentation-based similarity functions, each constructed in the same way. Similarities among related attributes (i.e. features from either facts or collections of folk arguments, or both) in cases being compared are computed and stored in a similarity matrix. Using the similarity matrix, a hierarchical method of clustering [3] leads to a "dendrogram". This tree structure is simple enough for the expert in the application field to give feedback about the organization and relevance of the groups formed (i.e. the taxonomies). In the painting authentication problem, for instance, expert date assignments in past authentications of paintings and statements that certain paintings are forgeries have been used in the evaluation of such taxonomies. The exploitation of numerical taxonomy techniques in the indexing of cases from the Element case base and the Pigment case base, which resulted in Element and Pigment taxonomies, has been discussed in [9]. The present paper is concerned with taxonomies from folk argumentation characteristics. By means of experiments with numerical taxonomy and expert feedback obtained from the grouping results, two elements of the expert's folk argumentation were found to be the best means of indexing the knowledge contained in such arguments: argument types (AT), and presence and absence of painting features (Present/Absent).

First, the knowledge engineer's analysis of case material allowed the determination of 26 different argument types for the characterization of folk expert arguments. Once the nature of the information used in advancing these arguments was annotated through the use of argument types, the types were treated as argumentation dimensions for the retrieval of cases recorded in the Argumentation case base. In order to do so, the list of argument types was examined in combination with information as to whether typical painting features were Present/Absent. The symbolic format of this combination can be written as Present-ATXX(feature) and Absent-ATXX(feature),

where "ATXX" stands for any particular type number in the list of 26 argument types and "feature" by any piece of factual information involved in a folk argument (e.g. a pigment, a synthetic fibre, an under-drawing, etc). For example, the knowledge engineer would regard the combination Present-AT07(feature) as representing a folk expert argument discussing "the presence of certain pigments in false paintings". If we take an azo-type pigment as the present feature, an instance of such a folk argument is: "The presence of an azo-type pigment also argues for a relatively recent date for the current painting. Yellow monoazo pigments were discovered by … No comprehensive data is available on the introduction of these pigments into artists' colours, but from the incidences noted in paintings the likelihood is again that there was little market penetration until much later in the twentieth century than the date of first discovery might seem to imply". In contrast, the combination Absent-AT07(feature) would indicate a folk expert argument discussing "the absence of certain pigments in false paintings". A1 in Fig. 3 is an example of this kind of argument. Table 1 shows the actual encoding of these folk argumentation dimensions (where AT03 means "cheaper painting features/methods" and AT04 means "style of painting"). The 1 and 0 values in the table cells indicate that such kinds of arguments are present or absent in the original template structure of a case (e.g. the Present-AT04(feature) is absent in the case p01 but present in p02).

Table 1. Argument types and Present/Absent features as taxonomic dimensions

Case name (including the dating conclusion)	Present-AT03 (feature)	Present-AT04 (feature)	…	Absent-AT04 (feature)	…
p01Maybe1914LikelyMid20th	0	0	…	1	…
p02Not1915ButPost1920to1930	0	1	…	0	…
…	…	…	…	…	…

Second, when the argument types and the presence and absence of painting features were taken as a single folk argumentation element, the resulting list of Present/Absent-ATXX folk argumentation dimensions (e.g. the number of columns in Table 1) became rather large (i.e. 2 x 26 argument types). However, knowledge engineer inspection of the painting authentication cases showed that the domain expert did not use all 26 types of arguments in the discussion of painting features that were absent in a problem situation (i.e. Absent-ATXX). The discussion of Absent aspects, which can refer to paint pigments but are not limited to them, is focused on such things as "the absence of painting features introduced into artistic practice at certain periods of time" (i.e. Absent-AT10) and "the absence of painting features in correlations of paintings" (i.e. Absent-AT06). For example, there would be no sense in discussing the progression of use (AT09) of absent pigments in paintings from different periods of time, where such argument would be characterized by Absent-AT09. Such considerations on the available painting authentication reports led to further reduction of the list of Present/Absent-argument type dimensions (i.e. Present/Absent-ATXX in Table 1). This situation can be expected to recur in other areas where expert arguments associated with case-like knowledge exist, and one can treat it in the same way.

Third, having produced a reduced list of Present/Absent-ATXX features, our next step was to conduct experiments of numerical taxonomy using the cases in the

Argumentation case base. Alternative similarity functions were tested in the assessment of similarity in the Present/Absent-ATXX dimensions. Eventually, the Sørensen-Dice coefficient of similarity [3] – number of attributes present in both of two items being compared, divided by the sum of the numbers present in each (i.e. a measure that ignores absences and slightly emphasizes co-occurrences) – produced the best intuitive match to what an expert said or felt about the similarity assessment of Argumentation cases. This can be understood from the fact that similarities coming from arguments advanced by painting analysts (i.e. statements characterized as Present/Absent-ATXX = 1) in cases being compared should naturally have a high significance in these similarity judgments. Taking no account of absence of argument types simply acknowledges that in this application there is no point in trying to analyze similarities of subjects that do not figure in discussion about authentication of a painting.

Fourth, we further reduced the number of argumentation dimensions by exploiting an "extreme heuristic" approach in the construction of clustering results for alternative highly weighted Present/Absent-ATXX dimensions (i.e. by trying out high weights in these dimensions of the overall similarity function; additional details of such kind of process can be found in [9]). Dendrograms generated when these highly weighted argumentation features were used as input in the computation of similarity were examined by both knowledge engineer and domain expert. In the analysis of Present/Absent-ATXX-based dendrograms for the painting authentication cases, the lack of agreement of date assignments in subgroups of cases was used as a recommendation for the complete removal of that Present/Absent-ATXX dimension from the overall similarity function. In particular, such lack of agreement appeared in subgroups of cases that were selected because a high weight had been used in a Present/Absent-ATXX dimension of the similarity function. As the outcome of this gradual knowledge acquisition activity supported by the analysis of intermediate dendrograms, the final list of Present/Absent-ATXX features resulted in 38 folk argument dimensions. These 38 dimensions were then exploited in the construction of Argumentation taxonomies. As a result of expert comments obtained when intermediate dendrogram results were inspected, no particular reason was found for having any adjusted weighting in the Argumentation similarity function. The Present/Absent-ATXX dimensions were therefore weighted equally.

Finally, Fig. 4 presents the groups formed when Present/Absent-ATXX features were used in similarity computations. Three major taxonomies are visible in the dendrogram of Fig. 4: a) {p17, p16, …, p23}, b) {p75, p47, …, p64} and c) {p31, p33, …, p71}. The first group {p17, p16, …, p23} includes mostly paintings from the mid 20th century, but also paintings from the early 20th century. According to the domain expert, significant numbers of cases involving forgeries are located there. This may be taken as a clue that this Present/Absent-ATXX encoding method is more effective in the identification of discussions involving forged paintings. The second group {p75, p47, …, p64} contains mostly paintings from the early 20th century, although a small number of paintings from the mid 20th century is also placed in this second cluster. A large number of authentic paintings is clustered there. The third group {p31, p33, …, p71} contains cases whose paintings are no more recent than the early 20th century. This third subgroup discriminates most of the cases involving the authentication of the oldest paintings recorded in the Argumentation case base. The cases p19 and p20 are outliers with respect to the three main groupings – they are modern forgeries.

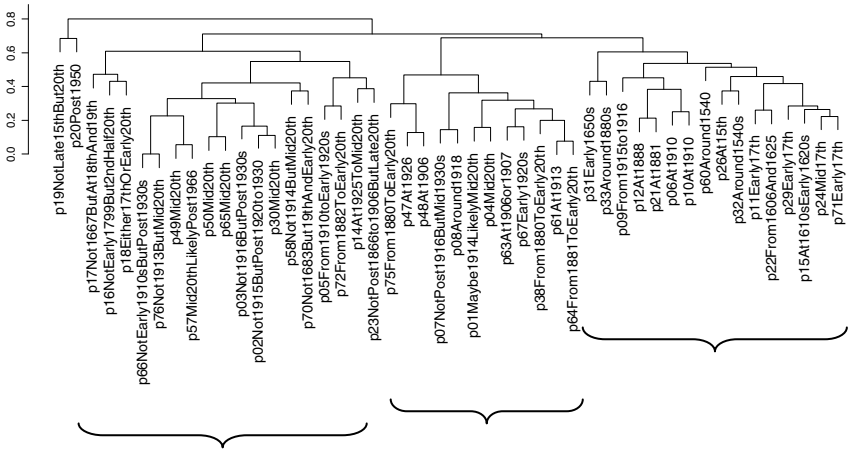


Fig. 4. Argumentation-based dendrogram formed when argument types and the presence and absence of painting features are used in the computation of similarity

Additional evidence regarding the quality of the Argumentation taxonomies of Fig. 4 has been collected with the help of the domain expert. According to him, the {p16, p17, p18} subgroup of cases in the first {p17, p16, …, p23} taxonomy is instructive here. This subgroup involves artistic forgeries that were thought to be much older than they were actually found to be. In contrast, the p07 and p08 pair in the second {p75, p47, …, p64} taxonomy may not be very informative. Case p07 involves a forgery that was supposed to be painted in 1916. In case p08 a similar date assignment is involved. Although the case p07 is placed correctly in this second subgroup, a better arrangement would place it together with the cases p01 and p04, forming a small group of forgeries from the mid 20th century. The {p09, p12, p21, p06, p10} subgroup of cases in the third {p31, p33, …, p71} taxonomy is undoubtedly useful according to the domain expert. The cases in this taxonomy involve paintings from French and Russian artists who were working in Paris in the same period of time. Case p09 refers to an authentication report constructed for resolution of a legal dispute, while cases p12 and p21 refer to reports involving the understanding of artists' methods and techniques. These three authentication reports not only examine dating aspects, but also involve related kinds of painting authentication explanations and justifications. From the inspection of this third taxonomy, the domain expert has also remarked that Dutch paintings from the first half of the 17th century are grouped together within it. This was considered to be a helpful arrangement of cases, since geographical considerations are often important in the construction of authenticity judgments for paintings from this period of time. To sum up the examples, the discussion above illustrates the kind of expert analysis that one can expect to obtain when such a dendrogram is evaluated in any application.

In conclusion, Fig. 4 shows all the groupings, including the major ones discussed above, that occur in a numerical taxonomy treatment applied to folk arguments in the case base. The last step of such a process is the generation of the taxonomies for the cases. In the same way as when fact-based painting authentication features are used in the similarity computations (see [9]), the three major Argumentation groupings in

Fig. 4 are then relevant for supporting any suggestion of date assignments in new painting authentication exercises. In order to test such relevance by comparison with the fact-based taxonomies, the domain expert was asked to look at results obtained when these taxonomies and their underlying similarity functions were used to find answers for CBR queries and to rate the quality of the results. A discussion of these tests follows.

6 Experiments in the Solution of Painting Authentication Problems

A simple way of inspecting the usefulness of the Argumentation taxonomy (Fig. 4) in relation to the Element and Pigment taxonomies is to ask the expert to rate the retrieval results obtained when selected cases in those taxonomies are used as queries (i.e. a "leave one out and test" exercise). In order to do so, we organized the 33 cases that are common to all case bases in 4 different subgroups reflecting their ranges of date. Cases involving paintings older than the 19th century were not used, because they were not available in all case bases. Then, we sorted cases in each subgroup so that each date range would have the same percentage of test cases. As a result, 12 cases were selected and tried out as queries in the CBR system. For each query, the four most similar cases in each case base were retrieved and presented to the expert. Then, he was asked to evaluate the CBR results for dating the painting in each test. This evaluation was performed with respect to the spread of dates presented in the most similar cases retrieved. Thus, for example, retrieval results for a query involving a painting from the early 20th century would be rated as "very useful" if all the four most similar cases retrieved were from this period of time.

Fig. 5 presents a summary of such tests. It shows that the performance of the CBR system is positive when these case bases are exploited individually in the answering of queries. For the queries where the expert's assessment was graded "useful" and "very useful", the score of Element retrieval results is 58%, the score of Pigment results is 67% and the score of Argumentation results is 75%. Over the three situations, one can make a qualitative average of this information to say that the performance is slightly to the "very useful" side of "useful" on the horizontal scale in Fig. 5. Further, the expert's explanation regarding retrieval results graded as "not helpful" is the same in all case bases, namely that "the date spread in the most similar cases retrieved is too great". In particular, this spread of date has occurred when the query involved a painting that was supposed to date from the early 20th century but which the original authentication analysis showed was a forgery from the mid 20th century (i.e. in case p76, which an expert has viewed as not 1913 but mid 20th century). This indicates an analysis that is sometimes not limited to the painting authentication aspects used by the CBR programs – an aspect that also appeared in the tests that were graded as "not helpful (forgery-related complications)". In such a situation, the most similar cases retrieved in these tests presented a mixture of cases where the paintings involved were from the early 20th century and the mid 20th century. Although some of these date determinations obtained from retrieved cases were not in a range narrow enough for the expert, the retrieval of paintings from both early and late periods of time may be taken to say that the target painting for the query cannot be from an

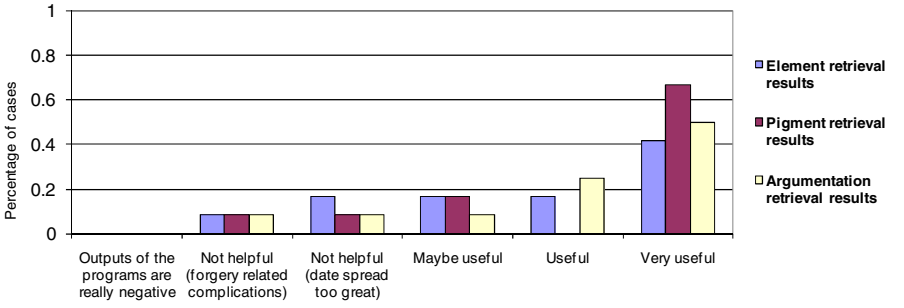


Fig. 5. Expert evaluation of retrieval results

Table 2. Computed and expert date assignments for test paintings

Test paintings	Date assignments according to Element retrieval results	Date assignments according to Pigment retrieval results	Date assignments according to Argumentation retrieval results	Date assignments in the original painting authentication reports
Painting E	1940s/1950s or later on the basis of Titanium dioxide white pigment (TiO_2 – synthetic)	Post 1950s on the basis of Titanium dioxide white pigment	Post 1950s	Not from 1863 to 1944 but from post 1944
Painting F	Early 20th century, in the first or the second decade	From the late 19th century to the earlier 20th century	Early 20th century	1913
Painting G	From the mid 19th century to the late 19th century	Difficult to assign a date	17th century	Around 1620s
Painting H	Around 1900-1925	Around 1920s	Early 20th century	1915/1916
Painting I	Difficult to assign a date (but it may be from the 19th century with later restoration in the 20th century)	16th century	Either from the 17th century or from the 19th century with a limited palette of pigments	Maybe 17th century but likely to be from the mid 19th century to the early 20th century

earlier time. This kind of interpretation would follow if there were an unconditional expert rule stating that: if a material exists only from date range X, and there is no evidence of restoration, then the painting that contains it cannot be from earlier than date range X. However, in this experiment, the domain expert has made no use of such an interpretation in the analysis of the test results (neither has the CBR system any such knowledge built into its problem-solving methods). Accuracy in the determination of date was the single condition used by the domain expert in the evaluation.

A natural use of the CBR resources discussed here is to support painting analysts in the assignment of dates and consequent discussion for previously unexamined paintings. In order to test this, an additional set of validation experiments was conducted. In it, 5 paintings (E through I) were submitted to the CBR system. Elemental compositions, pigments and collections of expert arguments were all available for these 5

paintings. These cases could therefore be expected to show up any contrasts among the CBR suggestions obtained when the three different case bases were used. Second, the test paintings were selected because they involved both authentic paintings (F, G and H) and forgeries (E and I). Third, they were painted by 5 different artists. Finally, the cases involved paintings in distinct periods of time: pre 19th century, 19th century, early 20th century and mid 20th century. Importantly, we have used no information from the authentication reports for the 5 test paintings in any previous experiment. Moreover, every effort was made to hide the identity of these paintings from the domain expert. The tests were planned as blind experiments so that the expert could offer objective comments regarding the quality of the CBR results. Paintings E-I were completely new cases for the test of the CBR system.

With each of the three basic types of information recorded in the test items – elemental compositions, pigments and collections of folk arguments – the four best cases and a similarity threshold of 0.4 (i.e. "rather low" but adequate for small case bases) were used in the retrieval computations. In this process, Present/Absent-ATXX dimensions were exploited in the computation of retrieval results for each test painting. The expert was then asked to assign a date for each test painting by considering the cases in the retrieval results only, as presented in Table 2.

Along with the date assignments in Table 2, the domain expert stated that both the Pigment and the Argumentation retrieval results were showing a very clear range of dates for painting E. The assignment of the date for painting E was clearer than the equivalent assignment from the consideration of the Element retrieval. In contrast, the corresponding Pigment results indicated a broader range of dates for painting F than the equivalent results obtained from Element retrieval results. In the Argumentation retrieval results of painting F, the range of dates was also narrower than the ranges obtained when the Pigment retrieval results were examined. A precise range of dates was difficult to assign for painting G when the Pigment retrieval results were inspected: the range of dates was too broad. However, the painting specialist also stated that the Argumentation retrieval results implied that painting G was definitely from the 17th century. No particular comments were offered after the analysis of the retrieval results for painting H. Finally, the date of painting I was difficult to determine through the inspection of Element retrieval results. However, when pushed to offer a date decision for painting I, the expert stated that this painting could be from the 19th century with later restoration in the 20th century. In contrast, the Pigment retrieval results indicated that painting I was from the 16th century. Assignment of date for painting I according to Argumentation retrieval results was very clearly the best according to the domain expert.

After collection of these verdicts which involved performance information beyond single date assignments, the identity of the paintings E – I was revealed to the domain expert who was then asked to re-examine the CBR results in the context of previous experience with authentication of the paintings. His main assertion was that the Argumentation results were "performing much better" than the other case-based results in all the tests. Basically, the cases that were being retrieved from the Argumentation case base involved narrower ranges of dates than the ranges obtained from the others. Of all the results, those from the Argumentation case base had the best combination of accuracy and expert-like caution, especially for painting H. A subjective further reason for the expert's positive evaluation of that case base was an apparent growth of confidence as the expert observed that its assignments were always clear

and convincing as far as they went (see Table 2), while the other case bases delivered assignments about which the expert usually had some reservations.

Furthermore, the Element assignment of date for painting G was instructive despite the fact that its suggestion was not accurate. It was informative because the Element case base did not have paintings older than the 19th century, i.e. nothing resembling G, which was from the 17th century. Nevertheless, all the most similar cases retrieved for G involved the oldest paintings in the Element case base. As the domain expert stated: this is what the expert would do if asked to analyze the correlation of G and the other paintings in the Element case base. Only two cases were retrieved from the Pigment case base when painting G was used as a query: a painting from before the 17th century and a painting from the 17th century. By contrast, such problems disappeared when Argumentation retrieval results were used. All the most similar cases retrieved were then from the 17th century, which is consistent with the supposed date of G. Additionally, the domain expert threw extra light on the understanding of why the retrieval results were not offering a precise date determination for G by saying that concerns regarding the authenticity of G still remained among people who had studied it. The authenticity of G continues to be open to investigation.

Finally, the Element case base did not have enough cases involving paintings from before the 19th century to allow a date for painting I to be determined accurately. Nevertheless, a date assignment from the mid 19th century to the early 20th century was found by looking only at the Element retrieval results. Interestingly, a similar kind of assignment was presented in the original painting authentication report. As for painting G, the assignment of date for painting I was not correct when the only two cases from the 16th century retrieved from the Pigment case base were used. However, an improvement in performance was observed when Argumentation retrieval results for painting I were inspected, as shown in Table 2. In particular, the use of a limited palette of pigments in this painting was also noted when the Argumentation retrieval results were examined. By raising this issue via Table 2, the CBR system has made a constructive suggestion about developing the original authentication report in an additional direction.

In summary, these experiments are evidence that the recording of facts along with folk expert arguments regarding other authentication aspects leads to the retrieval and inspection of useful painting authentication knowledge from cases. Consequently, the use of such a CBR approach for painting authentication has the potential to: a) aid in those cases where a decision cannot otherwise be reached because of lack of relevant data, b) complement other methods where parts of the evidence are lacking thereby potentially improving the performance and, c) pass on expert knowledge educationally to less experienced painting analysts. These are relevant contributions to the field of painting authentication, where interpretation has so far been limited to small numbers of specialists and where establishing systematic and inspectable results and knowledge bases is therefore desirable.

7 Concluding Remarks

Folk arguments are likely to be common in CBR applications where expert consultancy is hard to obtain except in small installments and where case bases are (because of their subject matter) necessarily slow to build, as in the painting authentication application. In this paper, we present an enhanced CBR framework which recognizes

the value and knowledge content of collections of folk arguments in cases and exploits that content to obtain practical CBR results. We also show that these results are better than comparable results from case bases that omit such folk arguments. The overall CBR framework involving folk arguments, knowledge acquisition templates and numerical taxonomy gets additional support because it has now (because of this painting authentication application) been shown to work well both here and in short-wave radio broadcasting [1]: two very different applications.

Our first target in future work is the expansion and testing of the case bases to cover at least the 77 paintings for which current authentication reports exist, and to add new examples. The testing should include investigation of what can be gained (e.g. as cues for acquisition of new argument knowledge that was previously implicit, or for training of analysts of paintings) by using the outputs from the Element or Pigment case bases as potential critics of an Argumentation output for the same queries, when those outputs are different. At the same time, given that we have so far needed to use only a limited amount (the types) of the knowledge contained in the folk arguments, we are intend to exploit more of it, such as *balance* details [1] in sequences of pro and con arguments in a case, to obtain further improvements in performance and to seek links with the various more formal approaches to argumentation that exist elsewhere.

References

1. Silva, L.A.L., Campbell, J.A., Buxton, B.F.: Folk Arguments, Numerical Taxonomy and Case-Based Reasoning. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 518–532. Springer, Heidelberg (2008)
2. Schreiber, A.T.G., Akkermans, H., Anjewierden, A., Hoog, R.D., Shadbolt, N., Velde, W.v.d., Wielinga, B.: Knowledge Engineering and Management - The CommonKADS Methodology. The MIT Press, Cambridge (2000)
3. Sneath, P.H., Sokal, R.R.: Numerical Taxonomy - The Principles and Practice of Numerical Classification. W. H. Freeman and Company, San Francisco (1973)
4. Eastaugh, N.: Scientific Dating of Paintings. Infocus Magazine, 30–39 (2006)
5. Berezhnoy, I., Postma, E., van den Herik, J.: Computer Analysis of Van Gogh's Complementary Colours. Pattern Recognition Letters 28, 703–709 (2007)
6. Ashley, K.D., Rissland, E.L.: Law, Learning and Representation. Artificial Intelligence 150, 17–58 (2003)
7. Chesñevar, C., Maguitman, A., Loui, R.P.: Logical Models of Argument. ACM Computing Surveys 32, 337–383 (2000)
8. Eastaugh, N., Walsh, V., Siddall, R., Chaplin, T.: Pigment Compendium: A Dictionary and Optical Microscopy of Historic Pigments. Butterworth-Heinemann, Butterworths (2008)
9. Silva, L.A.L., Campbell, J.A., Eastaugh, N., Buxton, B.F.: A Case for Numerical Taxonomy in Case-Based Reasoning. In: Zaverucha, G., da Costa, A.L. (eds.) SBIA 2008. LNCS (LNAI), vol. 5249, pp. 177–186. Springer, Heidelberg (2008)
10. Gardner, K.M., Rush, A.R., Crist, M., Konitzer, R., Teegarden, B.: Cognitive Patterns: Problem-Solving Frameworks for Object Technology. Cambridge Univ. Press, Cambridge (1998)
11. Walton, D., Reed, C., Macagno, F.: Argumentation Schemes. Cambridge Univ. Press, Cambridge (2008)
12. Benjamins, R.: Problem-Solving Methods for Diagnosis and Their Role in Knowledge Acquisition. International Journal of Expert Systems 8, 93–120 (1995)

Reexamination of CBR Hypothesis

Xi-feng Zhou[1,2,3,4], Ze-lin Shi[1,2,3], and Huai-ci Zhao[1,2,3]

[1] Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
[2] Key Laboratory of Optical-Electronics Information Processing,
Chinese Academy of Sciences, Shenyang 110016, China
[3] Key Laboratory of Image Understanding and Computer Vision, Liaoning Province,
Shenyang 110016, China
[4] Graduate School of the Chinese Academy of Sciences, Beijing 100049, China
xifeng_2002@163.com, {zlshi,hczhao}@sia.cn

Abstract. Most of the recent literature on complexity measures in textual case-based reasoning examined alignment between problem space and solution space, which used to be an issue of formulating CBR hypothesis. However, none of existing complexity measures could dispel the specter of predefined class label that does not appear in public textual datasets available, or clarify the correctness of the proposed solutions in the retrieved cases most similar to a target problem. This paper presented a novel alignment measure to circumvent these difficulties by calculating rank correlation between most similar case rankings in problem space and most similar case rankings in solution space. We also examined how to utilize existing alignment measures for textual case retrieval and textual case base maintenance. Empirical evaluation on Aviation Investigation Reports from Transportation Safety Board of Canada showed that rank correlation alignment measure might become a promising method for case-based non-classification systems.

1 Introduction

In the classic paper of D. B. Leake in 1996, he proposed two assumptions on that the case-based reasoning (CBR) approach is based, one of which is similar problems have similar solutions [1] that is usually called similarity assumption or CBR hypothesis. Afterwards in 1999, D. B. Leake and D. C. Wilson presented problem-solution regularity [2], which describes the relationship between problem descriptions and solutions, as a formulation of the aforementioned CBR hypothesis. Although many of the subsequent contributions have focused on the alignment between problem space and solution space, there have been few attempts to calculate alignment between most similar case rankings in problem side and most similar case rankings in solution side, instead of alignment between problem side similarities and solution side similarities.

Recently, the concept of complexity, which is equivalent to alignment, has been introduced in order to measure the extent to which CBR hypothesis hold true in TCBR [3, 4, 5, 6, 7, 8, 9]. Unfortunately, there are some limitations that are needed to be overcome in existing complexity measures for TCBR. In addition, few of them can clarify the correctness of proposed solutions in the retrieved cases most similar to a

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 332–345, 2010.

target problem (query) from different system designs. It is necessary to reexamine the problem of formalizing the CBR hypothesis in case-based non-classification systems in order to make these issues clear and to be solved.

In particular, three limitations of the existing alignment measures for case-based non-classification systems that are in the majority of current CBR systems, such as textual CBR (TCBR), may be noted:

- The class label that to a certain extent represent/determine the correctness of a textual solution, on which evaluation of effectiveness of existing complexity measures also rely, does actually not appear in public textual datasets available, like Aviation Investigation Reports (AIR) from Transportation Safety Board of Canada.
- Performance indicators don't have a standard definition because the goal of developing TCBR systems is different from diverse application domains, and at present there is not consensus about TCBR system framework.
- Although some of these alignment measures can provide case base profile for case authoring, they have not shown any further improvement in non-classification case retrieval and non-classification case base maintenance.

The computation of alignment measure or complexity measures for TCBR systems can be categorized into three classes in terms of different variables used, such as similarity scores, retrieval sets, case features. In essence, alignment measure or complexity measure is an informal definition of CBR hypothesis that describe the extent to which similar problems have similar solutions hold true in case-based reasoning systems. However, to our knowledge, there are few contributions that calculate alignment with case rankings.

In this paper, our primary motivation is to examine how to utilize alignment score to provide support for textual case retrieval and textual case base maintenance. A novel alignment measure is presented in order to formalize CBR hypothesis with partial rankings correlation between most similar case rankings in problem side and most similar case rankings in solution side. Inspired by the concept of correlation, we redefined a few performance indicators for evaluating the output of a non-classification CBR system, like TCBR.

This paper is organized as follows. Section 2 reviews related work about formulation of CBR hypothesis and positions our work in context. Section 3 argues that it is difficult to define the correctness of a predicted textual solution, and we circumvents this issue by evaluating partial rankings correlation between most similar case rankings in problem side and most similar case rankings in solution side. We provide empirical results to substantiate this idea in the following experimental section. Section 4 proposes a novel technique for textual case retrieval and textual case base maintenance based on correlation. Section 5 discusses our evaluation method and presents empirical findings on how well the alignment measure proposed by us can clarify the importance of each nearest neighbor. We highlight our main contributions and conclude in Section 6.

2 Related Work

Although many researchers considered CBR hypothesis as the main assumption that underpin case-based reasoning (CBR) as a suitable problem-solving methodology for

a particular domain, there are still not standard explanation or formulation about it. To date there appeared many explanations about CBR hypothesis suggested by different researchers that can be classified two categories, qualitative description and quantitative formulation. To our knowledge, the earliest explanation given by J. L. Kolodner in 1993 was that the intuition of case-based reasoning was that situations recur with regularity [10]. The regularity was further enlarged by D. B. Leake in 1996 who claimed that the CBR approach was based on two tenets about the nature of the world, which are similar problems have similar solutions and the types of problems one encounters tend to recur [1]. In 1997, B. Faltings made a possible explanation that presumed a problem with similar features as an earlier one was likely to have the same solution [11]. These studies have emphasized qualitative description and theoretical analysis opposed to quantitative formulation of CBR hypothesis. Although B. Faltings used probability theory to assure CBR hypothesis to be correct on the average, his explanation was still a qualitative description.

In 1999 D. B. Leake revisited CBR hypothesis and proposed problem-solution regularity and problem-distribution regularity in explanation of his earlier two tenets, which was the first time to formulate CBR hypothesis in a quantitative way as far as I know. He made a clear explanation on CBR hypothesis with mathematical equation for problem-solution regularity. Problem-solution regularity has become an oracle of subsequent definition of alignment measures, which captures how well problem similarities approximate solution similarities in practice [2]. For example, Case Alignment [4], Similarity Profile [6], MST and Weighted Correlation [9], etc. belongs to this category. The research has tended to focus on alignment between problem side similarities and solution side similarities (Fig. 1).

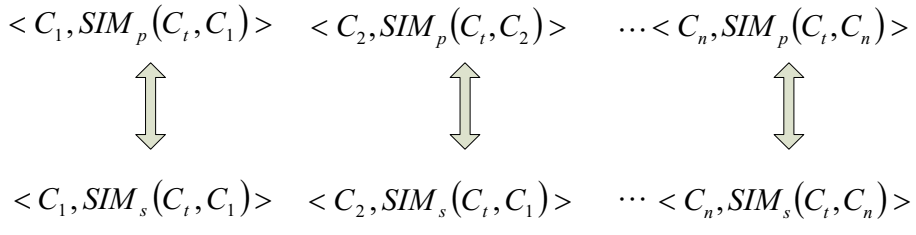


Fig. 1. Alignment between similar problems sequence and similar solutions sequence

The other two categories of method for calculating alignment appeared in recent years. GAME [5, 8] adopted case base image metaphor as a case clustering method with textual case features, which could be used to compare clusters alignment in both problem and solution space. Although considerable research on similarity value of retrieval result <CaseID, SimilarityValue> has been devoted to formalize CBR hypothesis for case-based classification or non-classification systems, rather less attention has been paid on the third category of methods for a long time, which utilized case ID of retrieval result to measure alignment (Fig. 1). This situation held in line until L. Lamontagne proposed Case Cohesion (alignCohesion) alignment measure [3] in 2006, which measures level of overlap in retrieval sets, in order to evaluate the competence of different system designs. However, threshold must be set using trial

and error to help to select the number of nearest neighbors in problem side and solution side in this method. Therefore, it would be a matter of interest to learn how to formulate CBR hypothesis with alignment measure based on case ID rankings between the retrieval sets in this paper.

$$alignCohesion(t) = \frac{|RS_p(t) \cap RS_s(t)|}{|RS_p(t) \cup RS_s(t)|} \tag{1}$$

After analyzing its equations for calculating alignment scores, we found that Case Alignment [8] (alignMassie) could not get to the maximum alignment value when problem space is completely in alignment with solution space, which is shown in our following experiments. Although ref. [9] presented two parameters-free complexity measures, the effectiveness of these measures need to correlate with human judgments like precision or classification accuracy, just like ref. [3] and ref. [8] did, that does not actually exist in public textual datasets available.

$$Align(t, c_i) = 1 - \frac{D_s(t, c_i) - D_{s\min}}{D_{s\max} - D_{s\min}} \tag{2}$$

$$alignMassie(t) = \frac{\sum_{i=1}^{k} (1 - D_p(t, c_i)) * Align(t, c_i)}{\sum_{i=1}^{k} (1 - D_p(t, c_i))} \tag{3}$$

Although there exist many explanations about CBR hypothesis, few of them is designed to formalizing it for case retrieval and case base maintenance in non-classification case-based systems. Our aim is to formulate CBR hypothesis for non-classification case-based systems by emphasizing the importance of case rankings, and utilize it to guide non-classification case authoring, case retrieval and case base maintenance.

3 Definition of the Correctness for Non-classification Solutions

Unlike classification domains where a solution is a class label associated with a group of cases, textual problem descriptions map into unique solutions. Especially during textual case retrieval, we do not know whether a retrieved textual solution to a query is right or not because it is usually a posterior knowledge available only after the retrieved textual solution is applied to solve the query. Before that, we can not say the textual solution is correct to any other textual problems. This is the property of uniqueness of textual solutions.

However, each case in the neighborhood of the query has a right solution. For example, in TCBR, one can say a textual solution is only correct to its corresponding textual problem description because it is the known case already existed in case base. Therefore, we can get to know whether the nearest neighbors (cases) of the query in

problem side can be solved or how difficult it is to solve those nearest neighbors (cases) using CBR methods with other cases in case base. It is not easy to acquire knowledge of whether a textual case (problem) is successfully solved or how difficult it is to solve the textual case (problem).

Because CBR is the approach we adopted to solve problems, the difficulty to solve a problem should be measured according to whether the basic tenet of this approach, similar problems have similar solution, hold true. In ICCBR 2008, Raghunandan M. A. etc. has pointed out that complexity reflects the degree to which we can expect a CBR system to be competent in answering a new problem by retrieving solutions to similar problem in the repository and that the complexity of a domain in CBR is a measure of the difficulty of the problem being faced. In ICCBR 2009, they presented a weighted correlation method (alignCorr) for calculating alignment measures that were designed to formulate similar problems have dissimilar solutions. The following are the equations to calculate alignCorr [9].

$$alignCorr(t) = wtCorr(PS(t), SS(t), PS(t)) \tag{4}$$

$$wtCorr(x, y, w) = \frac{wtCov(x, y, w)}{\sqrt{wtCov(x, x, w) * wtCov(y, y, w)}}^{1} \tag{5}$$

$$wtCov(x, y, w) = \frac{\sum_i (w_i (x_i - m(x, w))(y_i - m(y, w)))}{\sum_i w_i} \tag{6}$$

$$m(x, w) = \frac{\sum_i w_i x_i}{\sum_i w_i} \tag{7}$$

Inspired by aforementioned idea, we suggested employing complexity to measure the difficulty to solve a case in case base as a nearest neighbor of a query. Now, the problem is how we take advantage of this information to define the correctness of the prediction for the query.

In Leave-One-Out evaluation for a textual case base, each case in case base can be a target case that is composed of target problem and target solution. All the nearest neighbors of the query (target problem) in descending problem similarity scores in problem side are called Similar Problems Sequence (SPS). Similarly, all the nearest neighbors of the target solution in descending solution similarity scores in solution side are called Similar Solutions Sequence (SSS). If the *ith* most similar solution in SSS can solve the *ith* similar problem in SPS, we can say the *ith* nearest neighbor of the query are aligned. Assuming C1 is one most similar case in top k nearest neighbors to the query, its position in SPS is denoted as Position(C1, SPS), and its

[1] After communicating with Raghunandan M.A., we rectified the weighted correlation function in his original paper.

position in SSS is denoted as Position(C1, SSS). Check if the two positions are equal. If so, C1 is aligned for the query. If not, C1 is not aligned for the query. If each of the top k nearest neighbors is aligned for the query, we can say the target case completely respects CBR hypothesis. That means when the solution of the most similar case to the query in problem side is adopted to solve the query, it would be guaranteed that the best solution with max utility is selected. However, this is not the case. In addition, this kind of description is too informal to be an alignment measure. There must be some measures defined that can address this issue.

Our primary aim is to make sure every case in case base respects alignment between problem space and solution space. In other words, the position of a nearest neighbor of a target case in SPS approximates the position of the nearest neighbor of the target case in SSS. If not, we would penalize them with a penalty parameter. The measure that implemented this function is rank correlation.

4 Comparing Partial Rankings

In this paper, we introduced Kendall profile metric [12] as a new alignment measure, which is a generalization of the Kendall's τ rank correlation coefficient for comparing partial rankings. In statistics, a ranking can be seen as a permutation of a set of objects. Rank correlation is the study of relationships between different rankings on the same set of objects. Rank correlation coefficient measures the correspondence between two rankings and assesses its significance. Kendall's tau rank correlation coefficient is a measure of the similarity of the orderings of the data when ranked by each of two quantities. However, Kendall's τ rank correlation coefficient measures only similarity between full lists, not that between top k lists. In 2006, R. Fagin etc. generalized it for calculating distance between top k lists, namely Kendall profile metric [12].

In this paper, the top k nearest neighbors of a case (the target problem) in problem space and the top k nearest neighbors of the case (the target solution) in solution space are our concerns. We consider SPS and SSS of the case as two partial rankings with an emphasis on their top k elements and employ Kendall profile metric to calculate alignment scores of the case, as shown in equation (8).

$$K_{prof}(\delta_1, \delta_2) = \sum_{\{i,j\} \in P} \bar{K}^{(1/2)}_{i,j}(\delta_1, \delta_2) \tag{8}$$

In the equation (8), δ_1, δ_2 are two partial rankings with domain D. In addition, $P = \{\{i, j\} \mid i \neq j \text{ and } i, j \in D\}$. And $\bar{K}^{(1/2)}_{i,j}(\delta_1, \delta_2)$ is a penalty for δ_1, δ_2 for $(i, j) \in P$, whose calculation method is detailed in ref. [12] on page 633.

Going through the equation (8), it would be noticed the value obtained is a number greater than 0. When SPS and SSS are reverse orders to each other, equation (8) reaches its maximum value. Therefore, the equation calculating alignment scores with rank correlation, namely alignRank, is normalized to [-1,1] as an correlation measure in the following equation.

$$alignRank(t) = 1 - \frac{K_{prof}(\delta_1, \delta_2)}{\max(K_{prof}(\delta_1, \delta_2))} \tag{9}$$

The value of alignRank greater than 0 would indicate that the top k elements in SPS and the top k elements in SPS are well aligned around the target case, whereas a value equal or less than 0 would indicate poor alignment. Specially, we consider well aligned cases respect CBR hypothesis, and poor aligned cases don't respect CBR hypothesis. The bigger the value of alignRank is, the higher the degree of CBR hypothesis hold true. Due to the needs of computing similarity with other cases in case base, rank correlation becomes a global alignment measure.

Given k value, alignment score between the top k elements in SPS and the top k elements in SSS of a case could be regarded as a property of each case in case base. But it is a derived one, instead of a component property like problem description, solution, justification of solution, or result. As a result, we had to acquire this property before retrieval or maintenance. We had better complete it after the case base is stable or the case base is not changing within a relatively long time, at least not during retrieval or maintenance. However, it is impossible to directly compute rank correlation of a target problem that does not appear in case base, not to mention using it for textual case retrieval or textual case base maintenance.

Before retrieval or maintenance, it is necessary to compute nearest neighbors of each case in case base in problem and solution side, respectively. After determining k, the number of nearest neighbors that would be used to vote during retrieval, we just regard k as a parameter and pass it with two rankings of each case to the procedure for calculating alignRank with normalized Kendall profile metric.

The rank correlation of each of the k nearest neighbors of the target problem is the only information available when retrieving a relevant solution to the target problem. This information could be used to predict the possible rank correlation of case rankings of the target problem. Now, the problem is how we can utilize the rank correlation of k nearest neighbors to vote in retrieval or maintenance?

We can check whether the output of rank correlation procedure for each case is positive. If the majority of the votes are positive, then the predicted correlation of the target case (query) is positive. Otherwise, the correctness is low. Just liked we defined for alignRank, if the prediction value for rank correlation of the target problem is positive, we could consider the nearest neighbors has the most similar and effective solution to the target problem with higher guarantee or confidence because relative majority of its top k nearest neighbors also respect CBR hypothesis. Similarly, this method can be applicable to maintain a textual case base.

Aforementioned process is just how we take advantage of the value of alignRank for assuring the correctness of a predicted textual solution through the rank correlation of the nearest neighbors of a query. At this very moment, we circumvent the problem of evaluating a predicted textual solution.

In a word, although it is difficult to define the correctness of a predicted textual solution for a query in non-classification domains, we really know whether the top k nearest neighbors of the query can be solved or not. As long as the case base respects CBR hypothesis, we can predict the query with most similar case rankings in a simple way and guarantee the prediction is the best choice for the query.

5 Empirical Evaluations

As we can see, alignment score can be regarded as a measure of the correctness of a predicted solution if the case base respects CBR hypothesis. Similar problems having similar rank correlation could guarantee the whole case base has well alignment. The bigger the rank correlation of the case base is, the more accurate the prediction is. Then next issue is how to testify the usefulness of rank correlation alignment measure proposed by us.

5.1 Datasets Preparation

TCBR 2007 workshop proposed a few challenges to create a TCBR system that might support investigators in tasks such as the following:

a. Authoring a new investigation report.
b. Proposing safety actions in response to a new incident.
c. Discovering recurring unsolved problems.

However, it is difficult to achieve any one of them. There are many sections in investigation reports that we can not determine which should be authored in a new one. Safety actions do not appear in each of all the investigation reports, which can be seen in part of 576 cases in Challenger 1.0 [14]. It is not easy for knowledge engineers to decide which section could be a problem and which section could be a solution, not mention to discovering recurring unsolved one.

Therefore, in this paper we try to take advantage of the sections appeared in the majority of cases in case base, and regard one of them as a problem description, another one as a solution to constitute a <problem, solution> case pair. These sections are Summary, Other Factual Information, and Analysis appeared in all 568 cases from the case base used in Challenger 1.0. There are 118 cases that own all six subtitles in the 568 cases if you also want to take other sections into account, such as Other Findings about Causes and Contributing Factors, Other Findings about Risks, and Safety Action Taken.

There are so many similarity measures available for text up to now, such as Jaccard Coefficient, Dice Coefficient, Cosine Coefficient, and Overlap Coefficient, etc. Due to variability in vocabulary usage and uniqueness of solutions, we choose to jump over feature-based case representation into featureless similarity measures, such as compression-based similarity for text, which has proved to be effective in spam filtering [15]. Two compression-based similarity measures for text that have been implemented in JCOLIBRI CBR framework was used to estimate problem similarities and solution similarities and to obtain the most similar case rankings in problem side and solution side. Hence, in our experiments, there is no feature extraction and selection.

In the following experimental evaluation, our main task is to investigate the alignment relationship between two of these sections according to aforementioned four alignment measures and check which one of them can demonstrate the case base profile in a reasonable and clear way. The outcome can be used as an input for the following textual case authoring, textual case retrieval, and textual case base maintenance.

5.2 Performance Indicators for TCBR Based on Correlation

Before evaluation, some performance indicators for TCBR should be first defined according to whether predicted correlation is positive. It is generally accepted that evaluation is a challenge for TCBR systems. However, we can predict the correlation between SPS and SSS of a target case (query) according to the correlation between SPS and SSS of each of its k nearest neighbors. Now we can redefine precision, recall, and accuracy according to whether the prediction about correlation of a case is right or not, just like we did in information retrieval (IR) before. The goal of IR is to retrieve relevant documents. The performance indicators of precision and recall are defined according to whether the retrieved document is relevant that depends on human judgments. However, in public TCBR datasets at present, these kinds of human judgments are unknown. We redefined the relevance according to whether the predicted solution (case) respects CBR hypothesis or not that is determined by the voting of correlation of a few nearest neighbors for a query.

We can define three performance indicators without the need of human judgments about the correctness of a prediction that is not a class label. In the definition of the various indicators below, we denote a as the total number of the retrieved cases whose correlation between SPS and SSS is positive which is identical with the positive correlation of the target case (query). We denote b as the number of the retrieved cases whose correlation between SPS and SSS is negative which is not identical with the positive correlation of the target case (query). We denote c as the number of the retrieved cases whose correlation is positive which is not identical with the negative correlation of the target case (query), and denote d as the number of cases whose correlation is negative which is identical with the negative correlation of the target case (query), then we can have following equations of performance indicators.

$$precision: p = \frac{a}{a+b} \tag{10}$$

$$recall: r = \frac{a}{a+c} \tag{11}$$

$$accuracy: acc = \frac{a+d}{a+b+c+d} \tag{12}$$

Once defined these three performance indicators, we can use the traditional CBR methods for non-classification case retrieval and non-classification case-based case base maintenance.

5.3 An Illustration Using Artificial Data

In order to testify the feasibility of our rank correlation alignment method, for a first evaluation we have used a very simple artificial test domain, due to the huge complexity of the AIR application domain.

Assuming there are 7 cases in a case base, C1, C2, C3, C4, C5, C6, C7, the SPS and SSS of each case are computed according to 3-NN rank correlation alignment. The correlation of these cases are +, +, -, +, -, -, +, respectively. The SSS of C7 is C1, C2, C3, C4, C5, and C6.

Considering C7 as a target case, we execute 3-NN retrieval. If the SPS is C1, C2, C5, C6, C4, C3, then the predicted correlation of C7 is + according to the majority votes from top 3 nearest neighbors C1(+), C2(+), and C5(-). The prediction is correct. Comparing the position of the most similar case C1 in SPS and SSS, we could find they are the opposite ones.

Similarly, if the SPS of C7 is C6, C5,C1, C2, C3, C4, then the predicted correlation of C7 in 3-NN retrieval is -, according to the majority votes from C6(-), C5(-), and C1(+). The prediction is wrong. At the moment, if we take the most similar case C6 as the candidate solution for C7, we will make a wrong decision. The negative correlation of C6 just makes us avoid this kind of mistake.

Of course, there are still many conditions of SPS needed to be considered that we would not further explain due to the limitation of the length of this paper. But the contribution to retrieval made by correlation has already been clearly clarified.

5.4 Textual Case Authoring Based on Correlation

Case selection is one of the tasks of case authoring [3]. Decisions about what textual descriptions should be included in the case base must be made, especially when there are no explicit instructions about which section should be considered as the problem or the solution among many sections in an investigation report. This task is not like other tasks, such as vocabulary selection, case structuring, similarity metrics and retrieval strategy selection. If there were a domain expert, this would not be a problem. However, knowledge engineers have no idea about the relationship among different sections. Alignment measures could help to find the hidden relationship between two of these sections. This is just what we want alignment measures to do.

In order to prove our method to be effective for alignment computation, we select the same textual content (Analysis) as problem and solution. Now, if average alignment score of the case base is 1, then our method is right. Otherwise, if the alignment score can't get to 1, then our method is not so perfect that it would need to be improved. This test method can also be applied to other alignment measures so that we can check if the alignment can take into the situation account when the problems and the solutions are completely paralleled.

Considering Summary, Other Factual Information, Analysis as problem description respectively, and Analysis as solution, we could organize every Air Investigation Report from Canadian Transportation Safety Board into different <problem, solution> case pairs. Of course, we could use all kinds of alignment measures. If the alignment score between two sections are greater than that between the other two sections for the whole case base through most of alignment measures, we would consider the first two sections as case pair for using in later processing, rather than the latter two ones.

We will compute and compare average alignment score of the case base with three different alignment measures available, Case Alignment (alignMassie), Case Cohesion (alignCohesion), and our Rank Correlation (alignRank). We choose 67 cases in 2000 as our experimental data. The results are as follows:

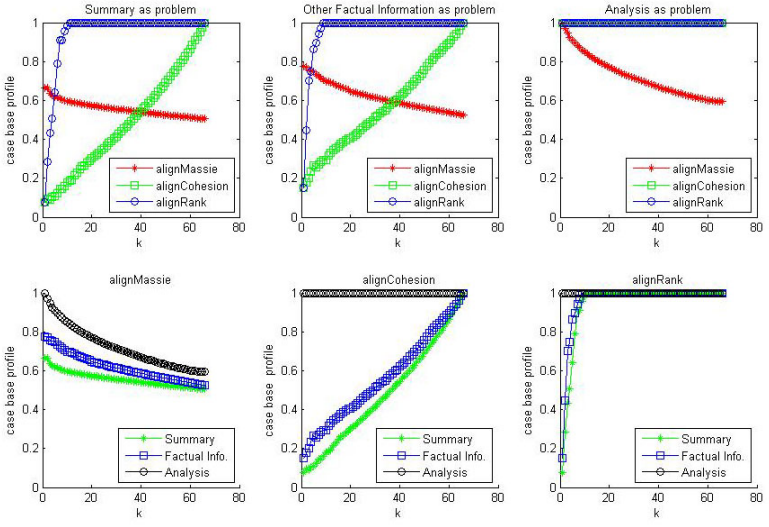


Fig. 2. Case base profile with three different alignment measures for three different subtitles as problem and Analysis as solution

As we can see in Fig. 2, the average alignment score of case base obtained from alignMassie is descending with the number of nearest neighbors that contribute to vote. On the contrary, the average case base alignment score of alignCohesion and alignRank is ascending. Specially, alignRank converge quickly to the max value 1. That is a outstanding outcome. The explanation we can present is that the formula of alignMassie descends with k. But the other two measures ascend with k. In addition, that our alignRank can converge to max alignment score means alignment score don't change with k when it has enough nearest neighbors to vote for his prediction of correlation. We think it is a virtue for selecting which section to constitute a case pair. As the case stands, when the problems and the solutions are completely paralleled alignRank and alignCohesion can demonstrate this situation, but alignMassie can not.

5.5 Textual Case Retrieval Based on Correlation

Rank correlation alignment could be used for textual case retrieval, which assumes the retrieved case have the same correlation with a query. We obtained our experimental results in Fig. 3 using 55 cases in 2000 and 2001 taken from 118 cases of Challenger 1.0 developed by J. A. Recio-Garcia etc. in TCBR'07 workshop [14].

We can see from Fig. 3 our rank correlation alignment measure, alignRank, acquire higher precision and recall than other three alignment measures after 5-NN and 3-NN respectively. alignCorr have the same precision with alignRank in 3-NN. However, alignRank outweighs alignCorr in the precision of 1-NN, 5-NN, 7-NN, and 9-NN. In addition, alignRank overruns alignCorr in recall and accuracy for all k-NN. alignMassie is an exception because the trend in direction of precision and accuracy is first up then down but the range of change is pretty small so that we can think alignMassie

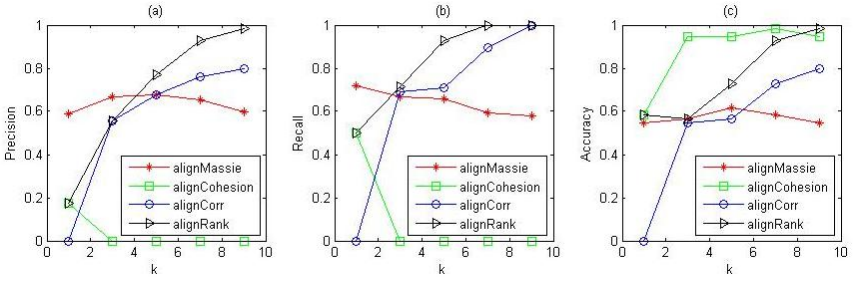


Fig. 3. Comparison of retrieval performance among different alignment measures

is not sensitive to the number of nearest neighbors, k. It is not coincident with human judgment that the more retrieved case is, the more accurate the prediction is. Therefore we are not sure about its contribution to prediction. By the way, most of time the performance alignMassie obtained can't match that alignRank did or alignCorr did.

It should be noted that we adopted the function f(x)=2x-1 to transform the value of both alignMassie and alignCohesion into [-1,1] that represented correlation relationship. This transformation for alignCohesion led to more than half of the retrieved set is covered with negative correlation that means the majority of cases in case base don't respect CBR hypothesis. Although the alignCohesion achieved higher accuracy than other three alignment measures, it is still unacceptable because its precision and recall is too low which means the cases in case base are negative correlation or the majority of cases in case base don't respect CBR hypothesis. It is necessary to reconsider the formula design from transformation to correlation. This would be our further work.

5.6 Textual Case Base Maintenance Based on Correlation

Rank correlation alignment could also be used to maintain case base. It is necessary to adapt the ICF (Iterative Case Filtering) and CBE (Case Base Editing) method implementation because maintenance algorithms in JCOLIBRI CBR framework are just designed for classification applications.

The earlier work in alignment measures or complexity measures suggested these measures could be applied to case base maintenance. For example, ref. [4] pointed out Case Alignment allows us to make informed maintenance decisions about individual cases. Ref. [8] mentioned these measures could identify neighborhoods with poorly aligned problem and solution concepts. However, none of them give an illustration about this issue. In this section, we will show an example application of alignment measures in textual case base maintenance.

A case in case base whose correlation is not identical to its k nearest neighbors is considered as a redundancy. Considering it as a criterion for maintaining case base, we can obtain following results shown in Table. 1.

Table 1. Comparison of accuracy of a predicated correlation before and after maintenance with extended ICF method

Alignment measure	Percent reduced	Accuracy in 3-NN case base maintenance (118 textual cases)	
		Before maintenance	After maintenance
alignMassie	53.39%	71.19%	43.64%
alignCohesion	96.61%	99.15%	75%
alignCorr	48.31%	54.24%	24.59%
alignRank	77.12%	93.22%	59.26%

The experimental results showed that it looks like correlation alignment don't support textual case base maintenance with our definition of redundancy for k nearest neighbors of a query. Surprisingly, alignCohesion outperformed the other alignment measures in the accuracy after maintenance. However, the percent reduced (96.61%) is too large to accept because there are little cases available for prediction. Of course, if there are too many cases that are not positive correlation, the prediction becomes meaningless because the majority of cases in case base don't respect CBR hypothesis. Further experiments will be needed to investigate the effect applying alignment to textual case base maintenance on more textual case bases in future work.

6 Conclusions

The main contributions of this paper are threefold. Firstly, we highlight the importance of case rankings both in problem side and in solution side and prove that it can contribute to the correctness of a predicted non-classification solution. In terms of this observation, we define three performance indicators for evaluating non-classification case-based systems. Secondly, we present a novel alignment/complexity measure based on Kendall profile metric for comparing partial rankings. Thirdly, we present a new algorithm for non-classification case retrieval and case base maintenance that can guarantee that the k nearest neighbors in problem space can be correctly solved by the k nearest neighbors in solution space without human feedback or human judgments. However, we are not sure about whether this method is applicable to textual case base maintenance because our preliminary results don't support our initial presupposition. It is necessary to redefine noise case and redundant case for textual case base according to alignment scores in future work.

Acknowledgments

We thank the anonymous ICCBR reviewers for their very valuable comments and Raghunandan M.A. for his proposal about the rectification in equation (5).

References

1. Leake, D.B.: CBR in context: The present and future. In: Leake, D.B. (ed.) Case based reasoning: Experiences, lessons, and future directions, pp. 3–30. MIT, Cambridge (1996)
2. Leake, D.B., Wilson, D.C.: When Experience Is Wrong: Examining CBR for Changing Tasks and Environments. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 218–232. Springer, Heidelberg (1999)
3. Lamontagne, L.: Textual CBR Authoring using Case Cohesion. In: TCBR 2006 - Reasoning with Text, Proceedings of the ECCBR 2006 Workshops, pp. 33–43 (2006)
4. Massie, S., Wiratunga, N., Craw, S., Donati, A., Vicari, E.: From Anomaly Reports to Cases. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 359–373. Springer, Heidelberg (2007)
5. Chakraborti, S., Beresi, U., Wiratunga, N., Massie, S., Lothian, R., Watt, S.: A Simple Approach towards Visualizing and Evaluating Complexity of Textual Case Bases. In: TCBR 2007 – Beyond Retrieval, Proc. of the ICCBR 2007 Workshops (2007)
6. Hullermeier, E.: Credible case-based inference using similarity profiles. IEEE Transactions on Knowledge and Data Engineering 19(6), 847–858 (2007)
7. Massie, S., Craw, S., Wiratunga, N.: When Similar Problems Don't Have Similar Solutions. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 92–106. Springer, Heidelberg (2007)
8. Raghunandan, M.A., Wiratunga, N., Chakraborti, S., Massie, S., Khemani, D.: Evaluation Measures for TCBR Systems. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 444–458. Springer, Heidelberg (2008)
9. Raghunandan, M.A., Chakraborti, S., Khemani, D.: Robust Measures of Complexity in TCBR. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 270–284. Springer, Heidelberg (2009)
10. Kolodner, J.L.: Case-Based Reasoning. Morgan Kaufmann, San Francisco (1993)
11. Faltings, B.: Probabilistic Indexing for Case-Based Prediction. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 611–622. Springer, Heidelberg (1997)
12. Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., Vee, E.: Comparing Partial Rankings. SIAM J. Discrete Math. 17(3), 628–648 (2006)
13. JCOLIBRI CBR Framework, Group for Artificial Intelligence Applications, Complutense University of Madrid, `http://gaia.fdi.ucm.es/projects/jcolibri/jcolibri2/index.html`
14. Recio-Gacia, J.A., Diaz-Agudo, B., Gonzalez-Calero, P.A.: Textual CBR in jCOLIBRI: From Retrieval to Reuse. In: TCBR 2007 – Beyond Retrieval, Proc. of the ICCBR 2007 TCBR Workshops (2007)
15. Delany, S.J., Bridge, D.G.: Catching the Drift: Using Feature-Free Case-Based Reasoning for Spam Filtering. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 314–328. Springer, Heidelberg (2007)

Case Based Reasoning with Bayesian Model Averaging: An Improved Method for Survival Analysis on Microarray Data

Isabelle Bichindaritz and Amalia Annest

University of Washington Tacoma, Institute of Technology
1900 Commerce Street
Tacoma, Washington 98402, USA
ibichind@u.washington.edu

Abstract. Microarray technology enables the simultaneous measurement of thousands of gene expressions, while often providing a limited set of samples. These datasets require data mining methods for classification, prediction, and clustering to be tailored to the peculiarity of this domain, marked by the so called 'curse of dimensionality'. One main characteristic of these specialized algorithms is their intensive use of feature selection for improving their performance. One promising method for feature selection is Bayesian Model Averaging (BMA) to find an optimal subset of genes. This article presents BMA applied to gene selection for classification on two cancer gene expression datasets and for survival analysis on two cancer gene expression datasets, and explains how case based reasoning (CBR) can benefit from this model to provide, in a hybrid BMA-CBR classification or survival prediction method, an improved performance and more expansible model.

Keywords: bioinformatics, feature selection, classification, survival analysis.

1 Introduction

Before genetic data of patients became available, the severity of a cancer was primarily determined by the stage of a tumor, based, in the clinical stage, on the morphology and spread of a tumor. Oncologists rely on the stage of a tumor to tailor a treatment plan adapted to the severity of the tumor, and particularly the amount and length of radiation therapy. When malignant tumors could be excised, pathologic analyses provided, in the pathologic stage, microscopic information to further evaluate the severity of the malignancy and treat the patient more or less aggressively as appropriate. However, both clinical staging and pathologic staging present limitations. First of all, not all tumors can be treated surgically, thus limiting stage determination by morphological and spread factors. Secondly, a number of low-risk patients turned up being misdiagnosed since their survival was not longer than that of the high-risk patients. Therefore oncologists are determined to find more reliable risk prediction methods and in particular genetic data available through microarray analysis have enhanced

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 346–359, 2010.

prediction reliability in many studies [1] – both in oncology and beyond in a variety of medical specialties.

Microarray technology provides a promising avenue. The availability of thousands of gene expression levels has enabled the pursuit of a new direction in cancer research. In particular, gene expression patterns can be thought of as multidimensional quantitative "expression phenotypes" which can in turn be correlated with clinical outcome. Because a single microarray can measure the expression levels of tens of thousands of genes simultaneously, the challenge lies in the development of data mining methods and tools to extract biological meaning from this immense amount of data. More specifically, the aim is to filter the expression dataset down to the smallest possible subset of accurate predictor genes. Reducing the number of predictor genes both decreases clinical costs and mitigates the possibility of overfitting due to high inter-variable correlations [2].

The most common approach to identifying a manageable group of predictor genes is called feature selection, in which a subset of relevant "features" (or variables) is selected from the larger dataset in order to produce a robust learning model [3, 4]. A well-designed feature selection algorithm will choose a small set of variables that is highly predictive of clinical outcome. Subsequent to or concurrent with the feature selection process, a supervised machine learning technique can be applied to generate a predictive function using the selected variables from a set of training data [5, 6]. In a supervised learning algorithm, the input is a set of training samples paired with the corresponding labels of those samples. The labels can be any predictable quantity of interest, such as a tumor subtype or a length of survival time. If the labels are exhaustive discrete classes to which the samples belong (e.g. "survived beyond five years" and "died before five years"), then the learning model is a *classifier* (see Kotsiantis 2007 for a review [7]). With microarray data, the most common approach is to apply a classification algorithm in which the patient data is split into subcategories corresponding to different prognoses or diagnoses. For survival analysis, the learning model is applied to predicting survival of patients in a *prediction* model.

This paper presents a case-based reasoning system (CBR) using the feature selection method called Bayesian Model Averaging (BMA) as a guide to its memory of cases for selecting the most similar cases in its case memory. In addition, the posterior probabilities provided by the BMA algorithm provide weights used by the CBR system for computing its similarity measure. Following, most similar cases can be used to either *classify* patients or *predict* their survival. This paper is organized as follows: after background information on the domain of microarray classification and survival analysis, methods developed in the BMA-CBR hybrid are presented, leading to an evaluation with accompanying results, a discussion, and a conclusion.

2 Background

One fundamental characteristic of biology is that it is a science based on observations and experiments. For that purpose, biologists gather data about natural species and phenomena, which taken together can be very large. In particular since the onset of genetic analyzes the amount of these data has become too large to afford for a human processing and analysis. Hence the applications of computer and information science

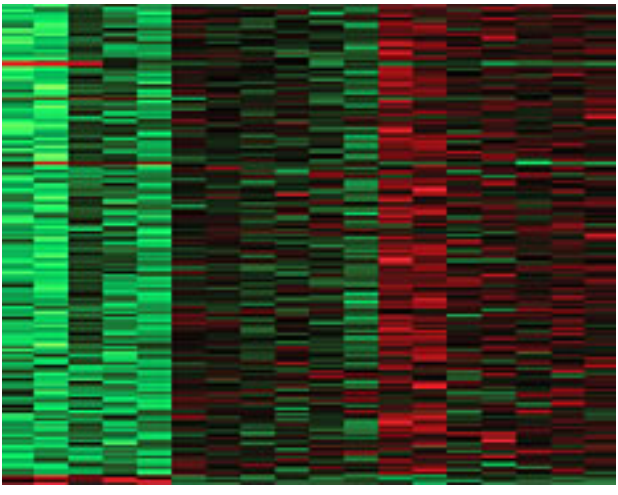


Fig. 1. A heatmap of microarray data

to biology have increased over time to allow for the interpretation of these every growing datasets. As a result, a new multidisciplinary science has emerged at the intersection of biology and computer science under the name of bioinformatics [8].

Gene Expression Data

Among the biosciences, three main areas have benefitted the most from computational techniques: genomics, proteomics, and phylogenetics. One of the most studied bioinformatics applications to date remains the analysis of gene expression data from genomics. Gene expression is defined as the process by which a gene's DNA sequence is converted into a functional gene product, generally a protein [1]. To summarize, the genetic material of an individual is encoded in DNA. The process of gene expression comprises two major steps: translation and transcription. During translation, excerpts of the DNA sequence are first encoded as messenger RNA (mRNA). Following during transcription, the mRNA is transcribed into functional proteins [9]. Since all major genes in the human genome have been identified, measuring from a blood or tissue sample which if these have been expressed can provide a snapshot of the activity going on at the biological level in an organism. The array of expressed genes, called an expression profile, at a certain point in time and at a certain location, permits to characterize the biological state of an individual. The amount of an expression can be quantified by a real number – thus expression profiles are numeric.

Among interesting questions studied in medical applications, are whether it is possible to diagnose a disease based on a patient's expression profile, or whether a certain subset of expressed genes can characterize a disease, or whether the severity of a disease can be predicted from expression profiles. Research has shown that for many diseases, these questions can be answered positively, and medical diagnosis and treatment can be enhanced by gene expression data analysis.

Microarray technologies have been developed to measure expression profiles made of thousands of genes efficiently. Following microarray-based gene expression

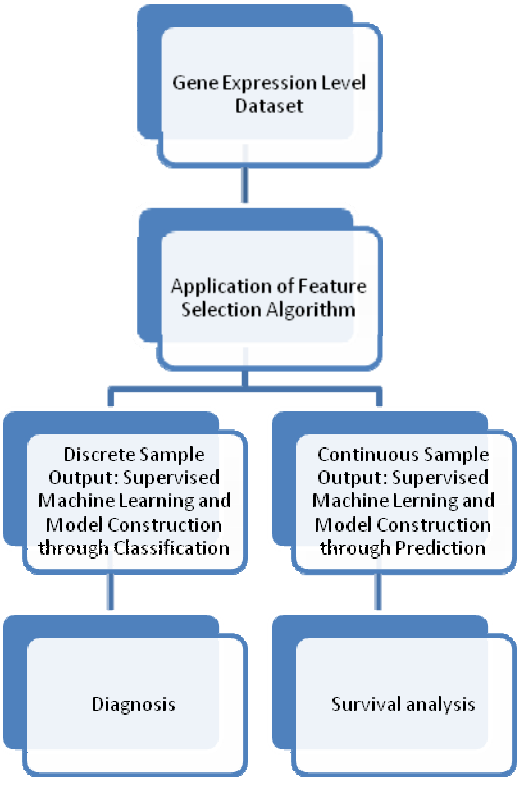


Fig. 2. Process-flow diagram illustrating the use of feature selection and supervised machine learning on gene expression data. Left branch indicates classification tasks, and right branch indicates prediction, with survival analysis as a special case.

profiling can be used to identify subsets of genes whose expression changes in reaction to exposure to infectious organisms, various diseases or medical conditions, even in the intensive care unit (ICU). From a technical standpoint, a microarray consists in a single silicon chip capable of measuring the expression levels of thousands or tens of thousands of genes at once – enough to comprehend the entire human genome, estimated around 25,000 genes, and even more [9]. Microarrays come in several different types, including short oligonucleotide arrays, cDNA or spotted arrays, long oligonucleotide arrays, and fiber-optic arrays. Short oligonucleotide arrays, manufactured by the company Affymetrix, are the most popular commercial variety on the market today [9]. See Fig. 1 for a pictorial representation of microarray data expressions.

Survival Analysis

Survival analysis is a statistical task aiming at predicting time to event information. In general the event is death or relapse. This task is a variant of a prediction task, dealing with continuous numeric data in the class label (see Fig. 2). However a distinction has

to be made between patients leaving the study for unrelated causes (such as end of the study) – these are called censored cases - and for cause related to the event. In particular in cancer research, survival analysis can be applied to gene expression profiles to predict the time to metastasis, death, or relapse. Feature selection methods are combined with statistical model construction to predict survival analysis. In the context of survival analysis, a *model* refers to a set of selected genes whose regression coefficients have been calculated for use in predicting survival prognosis [1, 10].

Feature Selection

Microarray data present a particular challenge for data miners, known as the curse of dimensionality. These datasets often comprise from tens to hundreds of samples or cases for thousands to tens of thousands of predictor genes. In this context, identifying a subset of genes the most connected with the outcome studied has been shown to provide better results – both in classification and in prediction. Therefore feature selection methods have been developed with the goal of select the smallest subset of genes providing the best classification or prediction. Similarly in survival analysis, genes selected through feature selection are then used to build a mathematical model that evaluates the continuous time to event data [11]. This model is further evaluated in terms of how well it predicts time to event. Actually, it is the combination of a feature selection algorithm and a particular model that is evaluated (see Fig. 2).

Annest et al. 2009 have studied in particular the task of survival analysis on microarray data [1]. They have identified that the problem with most feature selection algorithms used to produce continuous predictors of patient survival is that they fail to account for model uncertainty [1]. With thousands of genes and only tens to hundreds of samples, there is a relatively high likelihood that a number of different models could describe the data with equal predictive power. In their paper, the Bayesian Model Averaging (BMA) methods [12, 13] are applied to select a subset of genes for survival analysis on microarray data. Instead of choosing a single model and proceeding as if the data was actually generated from it, BMA combines the effectiveness of multiple models by taking the weighted average of their posterior distributions [1]. In addition, BMA consistently identifies a small number of predictive genes [10, 14], and the posterior probabilities of the selected genes and models are available to facilitate an easily interpretable summary of the output. Yeung et al. 2005 extended the applicability of the traditional BMA algorithm to high-dimensional microarray data by embedding the BMA framework within an iterative approach. Annest et al. 2009 further extended iterative BMA to survival analysis and implemented it as a Bioconductor package [1]. The results reveal that BMA presents with greater predictive accuracy than other algorithms while using a comparable or smaller number of genes, and the models themselves are simple enough to yield biological interpretation [1].

3 Methods

The system presented in this paper is an example of a hybrid between statistical methods and case-based reasoning. The first component of the system is a Bayesian

Averaging (BMA) feature selection system. Once a set of important features has been selected, these serve as indexing mechanisms to search through the case memory. The posterior probabilities provide weights for calculating the similarity measure. Actually, the system has two reasoning modalities – classification and survival analysis.

Bayesian Model Averaging

This section explains the BMA algorithm as it is described in Annest et al. 2009 [1]. The strength of BMA lies in its ability to account for model uncertainty, an aspect of analysis that is largely ignored by traditional stepwise selection procedures [12]. These traditional methods tend to overestimate the goodness-of-fit between model and data, and the model is subsequently unable to retain its predictive power when applied to independent datasets [10]. BMA attempts to solve this problem by selecting a subset of all possible models and making statistical inferences using the weighted average of these models' posterior distributions.

The core of the BMA algorithm is depicted in Equation (1) below [11]. Let Ψ denote the quantity of interest, and let $S = \{M_1, M_2, \ldots, M_n\}$ represent the subset of models selected for inclusion in the analysis. Then the posterior probability of Ψ given the training data *TD* is the weighted average of the posterior probability of Ψ given *TD* and model M_i, multiplied by the posterior probability of model M_i given *TD*. Summing over all the models in set *S*, we get:

$$\Pr(\Psi \mid TD) = \sum_{i \in S} \Pr(\Psi \mid TD, M_i) * \Pr(M_i \mid TD) \quad (1)$$

There are three issues to consider before Equation (1) can be applied: obtaining the subset *S* of models to be included, estimating the value of $\Pr(\Psi \mid TD, M_i)$, and estimating the value of $\Pr(M_i \mid TD)$ – which will be addressed in this section.

One challenge with BMA is the sheer number of models that could potentially be explored by the algorithm, especially when dealing with microarray data. If there are *G* candidate explanatory genes in the expression set, then there are 2^G possible models to consider. When working with tens of thousands of genes, such an undertaking is computationally intractable. In order to discard the noncontributory models and obtain a likely best subset, Raftery (1995) [12] proposed to use the regression by leaps and bounds algorithm from Furnival and Wilson (1974) [15]. This algorithm takes a user-specified input *"nbest"* and efficiently returns the top *nbest* models of each size (maximum 30 variables). Following application of the leaps and bounds algorithm, the Occam's window method of Madigan and Raftery (1994) [16] can be used to reduce the set of models. After identifying the strongest model returned by the leaps and bounds algorithm, the procedure can eliminate any model whose posterior probability is below the cutoff point in relation to the best model. The cutoff point can be varied, but the default is 20; that is, a model must be at least 1/20 as likely as the strongest model in order to be retained. Once this step is complete, the remaining group of models constitutes the set *S* to be used in Equation (1).

An exact calculation of the predictive distribution $\Pr(\Psi \mid TD, M_i)$ requires an integration over the vector of regression parameters θ_i:

$$\Pr(\Psi \mid TD, M_i) = \int \Pr(\Psi \mid \theta_i, TD, M_i) \Pr(\theta_i \mid TD, M_i) \, d\theta_i \quad (2)$$

Because this integral has no closed form solution for most censored survival models, the maximum likelihood estimate (MLE) $\hat{\theta}_i$ can be used as an approximation:

$$\Pr(\Psi \mid TD, M_i) \approx \Pr(\Psi \mid \hat{\theta}_i, TD, M_i) \quad (3)$$

While certain techniques such as the Markov Chain Monte Carlo (MCMC) methods have been used in survival analysis to obtain a more exact predictive distribution, the MLE requires fewer computational resources and has been deemed sufficient for the purpose of averaging over contending models [10].

Finally, a calculation of the posterior probability of model M_i given the training data *TD* involves an integral whose solution is impossible to evaluate exactly. Bayes' theorem yields Equation (4), which represents the posterior probability of model M_i given *TD*:

$$\Pr(M_i \mid TD) \propto \Pr(TD \mid M_i) \Pr(M_i) \quad (4)$$

where

$$\Pr(TD \mid M_i) = \int \Pr(TD \mid \theta_i, M_i) \Pr(\theta_i \mid M_i) \, d\theta_i \quad (5)$$

$\Pr(TD \mid M_i)$ is the integrated likelihood of model M_i, and θ_i is the vector of regression parameters $(b_0, b_1, \ldots, b_p)$ of model M_i. The Bayesian Information Criterion (BIC) can be used to approximate the integral in Equation (5). The Laplace method [17] is sufficient to accomplish this task when dealing with regular statistical models:

$$\log \Pr(TD \mid M_i) = \log \Pr(TD \mid \hat{\theta}_i, M_i) - (k_i/2) \log n + O(1) \quad (6)$$

In Equation (6), n represents the number of records in the data, k_i is the number of regression parameters in model M_i, and $O(1)$ is the error term. The Laplace method approximation is generally far more accurate than this final term (see [17] for discussion). The source code for the BMA implementation described above can be downloaded at http://www.research.att.com/~volinsky/bma.html, for both R and S-Plus software.

While this section has focused on the posterior probabilities of the models included in the BMA analysis, it may also be beneficial to obtain the posterior probabilities for each of the individual variables (genes) involved. This information is helpful in facilitating biological discussion as it reveals which of the genes are relevant predictors. Let the expression $(b_i \neq 0)$ indicate that the regression parameter for gene x_i exists in the vector of regression parameters θ_i for at least one model *M*. In other words, at least one model in the subset *S* includes gene x_i. Then the posterior probability that gene x_i is a relevant predictor can be written as:

$$\Pr(b_i \neq 0 \mid TD) = \sum_{M_S \text{ where gene i is relevant}} \Pr(M_S \mid TD) \quad (7)$$

In Equation (7), M_s refers to the set of all models within the subset *S* that include gene x_i. The posterior probability of gene x_i is a summation of the posterior probabilities of all models in M_s. This computation ensures that all statistically relevant predictor genes will be a part of at least one model in the subset.

Bayesian Model Averaging for Survival Analysis

A few previous studies has applied the Bayesian Model Averaging methods [1, 12, 13] to survival analysis. Volinsky et al. (1997) [10] assessed a patient's risk of stroke by using BMA to select variables in Cox Proportional Hazard Models [18]. The data was made available by the Cardiovascular Health Study and included 23 variables (e.g., age, smoking history, and blood pressure) that may contribute to a patient's chances of experiencing a stroke. BMA selected a total of 5 models and 11 predictive variables, including diuretic, aspirin use, diabetes, stenosis, and timed walk. Patient risk scores were calculated by taking the weighted average of the risk scores for each of the top five contending models. The patients were then assigned to either the high-risk, medium-risk, or low-risk group based on the empirical 33rd and 66th percentile cutoff points in the risk scores of the training set. To assess performance, Volinsky et al. created a log-score called the partial predictive score (PPS) [10]. The PPS for BMA was compared against the PPS for the top BMA model (that is, the single model of the top five BMA models with the highest posterior probability) and against the PPS of the model returned by stepwise backward elimination. BMA exhibited the highest PPS, with a prediction mechanism 15% more effective than the top model alone and 3.5% more effective than the stepwise procedure. Furthermore, the patients assigned to a risk group using BMA experienced fewer strokes in the low-risk group and more strokes in the high-risk group when compared with the other two methods.

Bayesian Model Averaging for Microarray Data

The BMA implementation described above is incompatible with microarray data. This is because the typical microarray dataset contains thousands or even tens of thousands of genes, but the leaps and bounds algorithm from Furnival and Wilson (1974) [15] can only consider a maximum of 30 variables when selecting the top *nbest* models to return to the user. Yeung et al. (2005) [14] developed an iterative BMA algorithm that takes a rank-ordered list of genes and successively applies the traditional BMA algorithm until all genes up to a user-specified number *p* have been processed.

In order to extend the iterative BMA method to survival analysis, a number of algorithmic modifications were implemented. First, the Cox Proportional Hazards Model [18] is used to rank each individual gene. Cox regression is a popular choice in the realm of survival analysis due to its broad applicability and capacity for handling censored data. It is a semi-parametric method that quantifies the hazard rate λ for a subject *s* at time T as follows:

$$\lambda(T \mid p_s) = \lambda_0(T) \exp(p_s \theta). \quad (8)$$

In this equation, $\lambda_0(T)$ is the baseline hazard function at time T, p_s is the vector of effect parameters (predictors) for subject *s*, and θ is the vector of unknown predictor coefficients. Cox observed that the baseline hazard function in Equation (8) could be left unspecified if the effect of a covariate on one individual remains the same for all times T (e.g., if an environmental variable doubles your personal risk of dying at time 5, it also doubles your risk at time 8). Therefore, an estimation of θ is all that is

needed. This approximation can be calculated using the partial likelihood, as explained in Annest et al. 2009 [1]. Once the regression parameters are estimated using the Cox model, the genes can be ranked in descending order of their log likelihood.

Following this step, the algorithm iterates through the user-specified *p* top-ranked genes, applying the traditional BMA algorithm for survival analysis [9] to each group of variables in the current BMA window (where the window size is denoted by *maxNvar*). This part of the procedure is similar to the classification method described previously; genes with high posterior probabilities are retained while genes with low posterior probabilities are eliminated. Following Yeung et al. (2005) [14], the 1% default threshold for inclusion has been adopted for this project. The algorithm relies on the elimination of at least one gene per iteration from the current BMA window, so the method cannot proceed if all genes in the window have a posterior probability ≥ 1%. Yeung et al. proposed an "adapted threshold" heuristic to account for this possibility, whereby the genes with the lowest posterior probabilities are removed to make room for subsequent variables. This heuristic has been incorporated into the iterative BMA method for survival analysis because the authors reported that its inclusion boosts predictive accuracy.

Case-Based Reasoning

The system presented in this paper is capable of analyzing microarray data for classification or survival analysis. It involves a feature selection step with BMA, followed by a case-based reasoning step

For classification purposes, the input test case to the system consists in a set of samples to classify in a discrete category, for example *m* different classes, corresponding in this domain to diagnostic categories. After applying the iterative BMA algorithm, case-based reasoning selects most similar examples (see Equation (9)) from each class, and compares their average similarity scores. The test case is then classified according to the class with highest similarity.

$$sim(c_i, c_j) = 1 - \sqrt{\frac{\sum_k w_k^2 (f_{i,k} - f_{j,k})^2}{\sum_k w_k^2}} \tag{9}$$

where c_i and c_j are two cases (one being the input test case, the other a memorized case), $f_{i,k}$ and $f_{j,k}$ are the *k* feature values from cases *i* and *j*, which are numeric in microarray data, and w_k is the weight associated with each feature *k*. The weights w_k are provided by the iterative algorithm, in the form of average posterior probability (see Equation (10)).

$$w_k = \frac{\sum_i Pr(M_i | TD)}{i} \} \tag{10}$$

where *i* corresponds to the number of models within the subset *S* that include gene *k*, and $Pr(M_i / TD)$ to the posterior probability of model M_i. As a matter of fact, weights are calculated in the same manner as the logistic regression coefficients.

Actually, the algorithm calculates one such similarity score S_k for each class $k \leq m$ by averaging similarity scores in each class (see Equation (11)). A parameter fixing the maximum number of nearest neighbors *nMax* allows for choice of the number of neighbors to select.

$$class = k \; such \; that \; \mathrm{k} = max_k \{ S_k \; where \; S_k = \frac{\sum_{nMax} sim(c_i, c_j)}{nMax} \} \tag{11}$$

For survival analysis purposes, the input test case in the system consists in a set of samples to classify into two risk groups (built by the algorithm) – the high risk group and the low risk group. First, the formula for selecting the *nMax* nearest neighbors is based on a similarity score like provided in Equation (10) where the weights are obtained from the linear regression parameters. Following, survival estimates are calculated based on the *nMax* nearest neighbors c_j (see Equation (12)) survival lengths. The formula for selecting the nearest neighbors is provided in Equation (13).

$$survival = \frac{\sum_{nMax} \; surv(c_j)}{nMax}\} \tag{12}$$

$$nearest\ neighbors\ = nMax\ c_k / max\ \{S_k\ such\ as\ S_k = sim(c_i, c_k)\} \tag{13}$$

4 Evaluation and Results

Results of both classification and survival analysis have been compared with Yeung et al. 2005 [14] and Annest et al. 2009 [1], respectively. The algorithms have been tested on completely independent training and test sets, which is the hardest evaluation possible.

Table 1. Summary of Leukemia 2 & 3 Datasets

Dataset	Total Number of Samples	# Training Samples	# Validation Samples	Number of Genes
Leukemia 2	72	38	34	3051
Leukemia 3	72	38	34	3051

Classification

Classification was evaluated on two datasets: a leukemia dataset with 2 classes and a leukemia dataset with 3 classes. These datasets were selected by the authors of previous studies because of the availability of comparative results. They comprise 72 samples described by 3051 genes, and diagnostic categories correspond to acute lymphoblastic leukemia (ALL), acute myeloid leukemia (AML) in the first dataset (see Table 1). The second dataset differentiates between two ALL subtypes (B-cell and T-cell).

On these data, results are very close to iterativeBMA (see Table 2), which uses logistic regression as a classification method. The BMA-CBR model does seem to improve the classification by being able to fit the test data more closely than logistic regression.

Survival Analysis

Once the risk scores for all patients in both the validation set and the training set have been calculated either with regression or with CBR, the algorithm employs the user-specified "*cutPoint*" for defining high versus low risk (e.g., a *cutPoint* of 60 means the lower 60% of scores will be deemed low risk, and the upper 40% will comprise the high risk group). Finally, predictive performance is measured by the p-value as calculated through the central chi-square distribution. Kaplan-Meier survival analysis curves [19] are also included as pictorial nonparametric estimators of the difference

between risk groups. All analyses in this study were conducted using R statistical software (http://www.r-project.org/). The Bioconductor packages for the iterative BMA algorithms for classification and survival analysis described in this paper are available for download from Bioconductor's website (http://www.bioconductor.org) as the *iterativeBMA* and *iterativeBMAsurv* packages respectively.

Table 2. Summary of classification results

Dataset	# classes	BMA-CBR	iterativeBMA	Other published results
Leukemia 2	2	#genes = 20 #errors = 1/34	#genes = 20 #errors = 2/34	#genes = 5 #errors = 1/34
Leukemia 3	3	#genes = 15 #errors = 1/34	#genes = 15 #errors = 1/34	#genes ~ 40 #errors = 1/34

Two datasets were used to compare results with iterative BMA: DLBCL and breast cancer (see Table 3). DLBCL comprises 240 samples with 7399 genes, while breast cancer comprises 295 samples with 4919 genes.

Table 3. Summary of DLBCL and Breast Cancer Datasets

DLBCL	Total Number	# Training Samples	# Validation Samples	Number Of Genes
DLBCL	240	160	80	7,399
Breast Cancer	295	61	234	4,919

On this dataset as well, BMA-CBR provides an improvement over iterativeBMA using linear regression (see Table 4). This improvement is shown on the lower p-value found between the two classes created by the algorithm: the high-risk group, and the low-risk group. Lower p-values indicate more separation between the two classes, therefore will result in higher accuracy when attempting to classify patients in the two risk groups.

Table 4. Summary of survival analysis results

Dataset	BMA-CBR	iterativeBMA	Best Other Published Results
DLBCL	#genes = 25 p-value = 0.00121	#genes = 25 p-value = 0.00139	#genes = 17 p-value = 0.00124
Breast cancer	#genes = 15 p-value = 2.14e-10	#genes = 15 p-value = 3.38e-10	#genes = 5 p-value = 3.12e-05

5 Discussion

Annest et al. (2009) explain that a growing number of studies have used a variety of statistical methodologies to perform survival analysis on high-dimensional microarray

data [1]. Beer et al. (2002) [20] developed a risk index based on 50 genes that classified lung cancer patients into either low- or high-rsk groups with a considerable degree of accuracy. The two groups differed significantly from one another in terms of survival rates (p=0.0006), and these results still held among patients with stage-1 tumors (p=0.028). Beer et al. tested their risk index through leave-one-out cross validation on the original data set. The index was then applied to an independent data-set with the same genetic expression information. Again, the high- and low-risk groups were found to differ significantly from one another, both overall (p=0.003) and among patients with stage-1 tumors (p=0.006).

In 2002, van't Veer et al. published a survival classification study involving primary invasive breast carcinomas [1]. They attempted to classify patients into two groups: those whose cancer recurred within five years of diagnosis and tumor resection (poor prognosis group), and those who remained disease-free beyond five years (good prognosis group). They used a three-step supervised classification method to develop a 70-gene predictive signature, which they applied to an independent test set of 19 patients. Their classifier correctly labeled 17 of the 19 samples, an accuracy of 89.5% (p=0.0018) [1].

Bair and Tibshirani (2004) [21] proposed a semi-supervised version of principal components analysis that is capable of generating a continuous predictor of patient survival. Their algorithm consistently selected fewer than 20 genes and successfully divided patients into high- and low-risk groups in four different datasets: lymphoma (p=0.00124), breast cancer (p=2.06e-05), lung cancer (p=1.47e-07), and acute myeloid leukemia (p=0.00136).

In 2009, Annest et al. applied the iterative BMA algorithm to survival analysis [1], after Yeung et al. validated their results applying this same algorithm to classification [14]. Their results and the ones presented in this paper are very comparable. The main reason is that feature selection coupled with regression coefficients provided by iterative BMA provides to the CBR algorithm a smaller list of features, associated with their respective weights, which are directly reusable by CBR.

Given the small dataset size, the reuse of this same subset of features suffices to provide very accurate classification or survival analysis performance, comparable to those of BMA plus regression. However, with larger number of samples, it is likely that the consideration of alternate genes would be favorable to better results. In this case, the subset of genes selected would retrieve a pool of candidate cases, which in turn could be considered with all their complete gene expressions when calculating the similarity measure. Although other approaches have been developed to apply CBR to microarray data analysis, the principle of applying a feature selection method prior to or concurrently with CBR is common for improving performance [22]. The advantage of resorting to BMA is to not only select features but also learn feature weights useful in similarity evaluation.

It is notable that BMA combinations are not all as successful. In comparison with other classification algorithms, CBR is able to take advantage not only of the gene selection, but also of the weights learned. Comparative results showing the advantage of CBR over other machine learning algorithms, when combined with BMA, will be published in another paper.

6 Conclusion

The BMA-CBR analysis framework provides promising results for classification and prediction on microarray data. There is a number of future directions for this work. First, we are considering testing on additional datasets. Second, we would like to refine the tests in survival analysis by calculating individual risk scores – and not only two risk classes. We are considering adding a meta-analysis level to combine results from different classifiers and different feature selection methods – it is notable that different feature selection methods find different lists of interesting genes. We would like to link an automatic interpretation component to gauge the usefulness of the genes selected. We are also planning to make this algorithm available in the Bioconductor package, once all preceding steps are complete.

References

1. Annest, A., Bumgarner, R.E., Raftery, A.E., Yeung, K.Y.: Iterative Bayesian Model Averaging: a method for the application of survival analysis to high-dimensional microarray data. BMC Bioinformatics 10, 10–72 (2009)
2. Jiangeng, L., Yanhua, D., Xiaogang, R.: A Novel Hybrid Approach to Selecting Marker Genes for Cancer Classification Using Gene Expression Data. In: The 1st International Conference on Bioinformatics and Biomedical Engineering ICBBE 2007, pp. 264–267 (2007)
3. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. The Springer International Series in Engineering and Computer Science, vol. 454. Springer, Heidelberg (1998)
4. Liu, H., Motoda, H. (eds.) Computational Methods of Feature Selection. Hall/Crc, Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC (2008)
5. Huang, T., Kecman, V., Kopriva, I.: Kernel Based Algorithms for Mining Huge Data Sets: Supervised, Semi-Supervised, and Unsupervised Learning. SCI, vol. 17. Springer, The Netherlands (2006)
6. Witten, I., Frank, R.: Data mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufman Series in Data Management Systems. Elsevier, Inc., San Francisco (2005)
7. Kotsiantis, S.: Supervised Machine Learning: A Review of Classification Techniques. Informatica 31, 249–268 (2007)
8. Cohen, J.: Bioinformatics – An Introduction for Computer Scientists. ACM Computing Surveys 36(2), 122–158 (2004)
9. Piatetsky-Shapiro, G., Tamayo, P.: Microarray Data Mining: Facing the Challenges. ACM SIGKDD Explorations Newsletter 5(2), 1–5 (2003)
10. Volinsky, C., Madigan, D., Raftery, A., Kronmal, R.: Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46(4), 433–448 (1997)
11. Hosmer, D., Lemeshow, S., May, S.: Applied Survival Analysis: Regression Modeling of Time to Event Data, 2nd edn. Wiley Series in Probability and Statistics. Wiley Interscience, Hoboken (2008)
12. Raftery, A.: Bayesian Model Selection in Social Research (with Discussion). In: Marsden, P. (ed.) Sociological Methodology 1995, pp. 111–196. Blackwell, Cambridge (1995)

13. Hoeting, J., Madigan, D., Raftery, A., Volinsky, C.: Bayesian Model Averaging: A Tutorial. Statistical Science 14(4), 382–417 (1999)
14. Yeung, K., Bumgarner, R., Raftery, A.: Bayesian Model Averaging: Development of an Improved Multi-Class, Gene Selection and Classification Tool for Microarray Data. Bioinformatics 21(10), 2394–2402 (2005)
15. Furnival, G., Wilson, R.: Regression by Leaps and Bounds. Technometrics 16, 499–511 (1974)
16. Madigan, D., Raftery, A.: Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window. Journal of the American Statistical Association 89, 1335–1346 (1994)
17. Raftery, A.: Approximate Bayes Factors and Accounting for Model Uncertainty in Generalised Linear Models. Biometrika 83(2), 251–266 (1996)
18. Cox, D.: Regression Models and Life Tables. J. Royal Stat. Soc. Series B 34, 187–220 (1972)
19. Kaplan, E., Meier, P.: Nonparametric Estimation from Incomplete Observations. Journal of the American Statistical Association 53, 457–481 (1958)
20. Beer, D., Kardia, S., Huang, C., Giordano, T., Levin, A., et al.: Gene-Expression Profiles Predict Survival of Patients with Lung Adenocarcinoma. Nature Medicine 8(8), 816–824 (2002)
21. Bair, E., Tibshirani, R.: Semi-Supervised Methods to Predict Patient Survival from Gene Expression Data. PLOS Biology 2(4), 511–522 (2004)
22. Jurisica, I., Glasgow, J.: Applications of Case-Based Reasoning in Molecular Biology. AI Magazine 25(1), 85–95 (2004)

User Trace-Based Recommendation System for a Digital Archive

Reim Doumat, Elöd Egyed-Zsigmond, and Jean-Marie Pinon

Université de Lyon,
LIRIS, INSA-Lyon, 7, avenue Jean Capelle
69100 Villeurbanne, France
{reim.doumat,elod.egyed-zsigmond,jean-marie.pinon}@insa-lyon.fr

Abstract. Precious collections of cultural heritage documents are available for study on the internet via web archives. The automatically added metadata on these scanned documents, are not sufficient to make a specific search. User effort is needed to add manual annotations in order to enhance document content accessibility and exploitability. Annotators have different experiences in dissimilar manuscript domains. Hence the reuse of users' experiences is constructive to accelerate the annotation process and to correct user mistakes. In this article we present our digital archive model and a prototype to collaboratively annotate online ancient manuscripts. Our system tracks important user actions, and saves them as traces composed of hierarchical episodes. These episodes are considered as cases to be reused by a recommender system.

Keywords: Trace-based reasoning, episodes of actions, recommender system, collaborative annotation, cultural heritage documents.

1 Introduction

Digitizing ancient texts, invaluable historical and cultural documents and artifacts (papyrus, stones and others) has become widespread; it offers the possibility to all concerned institutions or persons to conserve old documents, and to make them available for study and exploitation for a wide audience; such as librarians, historians, linguists, or any person interested in the ancient documents.

The question is "What is the next step after document digitization?", "How to understand the content of these documents?". Users need to access the digitized documents easily to study their content, search them, and enrich them by annotations or classifications. For this reason, a management system is needed to store, visualize, organize, search and annotate these documents. That is why many projects and digital libraries have been developed for these purposes, and that is why we are interested in developing a digitized manuscript's management system.

Nearly all documents in digital libraries overlook detailed annotations. Annotations are associated with documents either depending on metadata standards such as Dublin Core [1], METS [2], MARC21 [3], or depending on the application requirements. Many libraries use textual annotations resulted by applying OCR (Optical Character

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 360–374, 2010.

Recognition), or by applying handwriting segmentation algorithms [4], to extract the content of scanned documents.

Generally, in cultural heritage repositories, collections are sets of digitized images where the original papers are mostly handwritten documents. Document search and retrieval in these collections are annotation-based; it means that search tools use the descriptive metadata, transcriptions and other annotations to find the required document. Consequently, we perceive that their management system must allow adding annotations as metadata to facilitate the document retrieval. However, this work is laborious. Reading a manuscript page takes sometimes hours; thus a solution could be by publishing these documents online and offering annotation tools to facilitate the annotation creation. The problem is that the created annotations are not validated, and users can make many mistakes. Therefore, allowing users to correct or to comment on the annotations of other persons is useful. This is considered as a collaborative work, which is crucial to achieve good quality results in realizing difficult tasks. Furthermore, such a system must be intelligent enough, to capture users' interaction with the system, and to propose some assistance during the annotation process. In other words, observing users' actions can enhance simplicity and efficiency of application use, improve interaction quality, and save users time and effort.

The focus of our research is the development of an intelligent framework based on case-based reasoning for annotating and searching the content of handwritten manuscripts. The framework can reason from the registered traces of users' interaction with the system. We believe that this framework has wide applicability, but to date it has been applied only to the task of annotating and searching information in scanned manuscripts.

In this paper, we present an online digital archive application, to manage and to annotate ancient manuscripts manually and remotely, in a collaborative and assisted environment. The work in a collaborative environment necessitates tracking users' activity. A prototype of this web application has been developed, to exhibit the collections, to supply annotation tools, to trace user actions with the manipulated objects, as well as to assist users during the annotation process. User traces are harnessed by the assistant that uses case-based reasoning technique.

This article is organized as follows: Section 2 gives an overview about some tracing and decision systems. In Section 3, we present our web archive with a trace-based reasoning assistant. Section 4 shows our system evaluation results. At the end, we conclude our work and mention to some of our perspectives.

2 Background

Users who work to annotate manuscript pages have different experiences; an expert in one domain or language can be a novice in another. For this reason it is essential to track important user actions, to register these traces in a manner that eases their retrieval and reuse. In this section, we cite some projects that trace user interaction with the system to get information about how users exploit the application. These projects also propose to model the registered traces in order to reuse them.

Musette (Modelling USEs and tasks for tracing experience) [5], is an approach to build traces according to a use model. Traces in Musette are composed by entities,

events, and relations. Entities are objects being presented to the user in his interaction with the system, events are actions happening during the interaction, and relations are binary and can imply either entities or events. The *use model* describes what kind of entities, events and relations will actually be observable to produce a primitive trace. Trace components are not sufficient to build an observer, for this reason an *observation model* is needed to define the rules to produce traces. In Musette, the observation model is not specified.

Trèfle [6] is quite close to the Musette model but represents users, actions and objects in one graph. In addition, Trèfle introduces the notion of potential graph in order to filter the traces in comparable and reusable episodes. The basic assumption is that *users* manipulate *objects* through *procedures*. For each application the manipulated objects modeled in a graph forms the *use model*, the selected actions together with the users form another sub-graph called *observation model*. The objects are linked to the actions that manipulate them, each action having a set of objects as parameters. The use and observation models enable the tracing of the user-system interaction. The trace nodes are instances of users, objects and actions, they are attached to each other according to the relations defined in the observation and use model, and they are also attached to the nodes they instantiate from these models. As the whole forms one graph, Trèfle uses graph theory based methods to filter the traces. A trace filter is expressed as a partially specified graph: the *potential graph*. Finding isomorphic subgraphs in the traces gives episodes, which can be compared and adapted to provide recommendations. This is done according to the case-based reasoning paradigm.

Traces are also frequently used in learning systems because they track users activities, and help in improving the pedagogical scenarios. In [7] the authors talk about the necessity to help users based on traces, by collecting, transforming, analyzing the traces issued from observing human learning. For example, teachers define paths through documents that students have to follow depending on their learning objectives. Comparing teacher prescribed tasks with the user activities; traces participate to control the learning process. This is called **TBS** (Trace Based System) where traces are composed of observed elements associated to their trace model.

eMédiatèque is also used in e-learning to study and to extract the user's behavior and his activity's objective. Every specific use of a resource may become a valuable source of knowledge about how that resource can be used. According to [8], a trace is a computer object where data about the interaction between the user and the system are collected. The authors adopted the hypothesis that the visualization of actual user traces advances monitoring and analyzing processes during learning activities. They developed eMédiatèque, a collaborative platform with a tracing infrastructure to visualize actual user traces and to offer collaborative tracing tools. Visualizing a collective trace is in itself a collaborative tool since it permits to display the actions done by other users. The user in eMédiathèque can manipulate his trace to reveal the elements that he wants to be emphasized.

We can see that the use of traces in collaborative environments may be useful. Efficient trace exploitation is considerably better if the traces are conveniently modeled. It is possible to extract frequent items and correspondences using data mining methods, but if the tracing system has specific tools to cut the traces into comparable and reusable episodes, the capitalized experience will be retrieved faster and more

precisely. Such tools are based on trace models that are often based on the tasks a user performs.

In our application, we believe that reusing user traces is important to provide help to all users. In addition to the tracing system, there is a need for an intelligent system or an assistant, to read and to reuse these traces, to help users during the annotation and the search process.

For this reason, we examined the popular decision and intelligent techniques that allow computers to progress behaviors based on observed data from databases, such as Bayesian networks and Hidden Markov Models (HMMs). Actually, human subjective knowledge is very difficult to model using these techniques. Ancient manuscript annotation is a subjective task. Using capitalized user experience to assist this task comes naturally in view because humans despite their differences are closer in interpretation and annotation aspects to other humans than to computers. Case based reasoning has been successfully used in different domains such as project planning tasks [9], medical problems [10], and others. We believe that, in our system, CBR is the most suitable technique to assist users based on the knowledge represented in users' traces.

Next section describes in details the design of our web archive to annotate manuscripts, and to assist users during this process.

3 Using Trace-Base Reasoning to Assist Users

In this section, we present our archive model that manages the digitized manuscripts and their annotations on the Web, and offers appropriate tools to participate in the transcription of the manuscript content. The major concern is assisting users to annotate and to search manuscript's content.

3.1 Global View on Our Online Manuscript Archive

Our archive contains collections added by the archive administrator that are considered as *original collection*. Alternatively, collections created by the users are *personal collections*. Unlike the original ones, personal collections can be modified or deleted; they also have privacy rights to be seen by everyone, a group of users or just the collection creator.

Each manuscript page is scanned to be a digital image then image groups are organized into collections. Each image belongs to only one original collection. Moreover, a type of adjustable and movable rectangle can be attached to the image representing an image *fragment*. Fragments are added by registered users who have sufficient permission, to define a region in the image where they are interested to work on. An image may have numerous fragments.

All the previous defined types of documents are considered as document units; a *document unit* is an abstract document that may possibly represent a collection of images, a simple image or an image fragment (see Fig. 1.). In our application, we register user actions between his login and logout as a trace in the database. In the next section we talk in details about user traces.

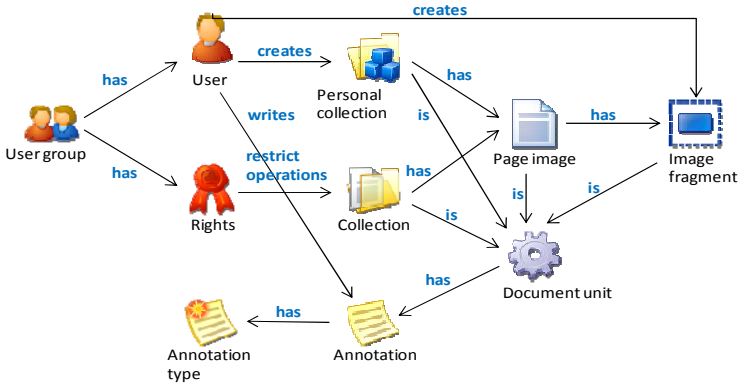


Fig. 1. The components and their relationships in our system

3.2 Trace-Based Reasoning

Trace-based reasoning is the technique of using traces as cases to reason the user observed knowledge. We store user actions as traces and attempt to reuse these cases; each case represents a sequence of actions.

Reusing user traces takes place in steps and first we begin by giving some definitions to the terms we use and describe how we register the cases.

Definitions

– Definition 1: an *Action (a)*, in our concept, is a user act that affects a document unit. Certain actions are tracked and registered with their parameters in order to be compared.

– Definition 2: *Action Parameter (P)*, parameters are the objects affected by an action. Parameters have types (*PType)* and values (*PValue)*. For example: variables that hold the value of metadata (annotations) or the values of coordinate points that define a fragment in an image are parameters. In general any object that can be manipulated by the system can be action parameter.

– Definition 3: *Action type* (*AType*) is a pre-defined value of the action. In our application not all actions are traced, the traced action types are: login, select, add, create, delete, search, modify, and logout.

– Definition 4: a *Trace (T)* in our notion is the record of user actions during a session of system exploitation. A session is defined between the user login and logout.

– Definition 5: an *Episode* (ep) is a group of user actions concerning the work done on one document unit during a session. An episode can contain sub-episodes; in this case it is called *parent episode*.

When the user changes the document unit another episode is being created. Since document unit may represent many types of documents (collection, image or fragment), registering the episode in the database requires identifying its document type For example, episode concerns the actions done on collections is called “collection episode” or ‘first level episode”, episode concerns an image is called “image episode” or “second level episode”, and episode concerns image fragment is called “fragment episode” or “third level episode”.

Furthermore, an episode may be *simple* or *complex*. *Simple episode* concerns only the work done on one document unit that does not contain another document unit (e.g. the ep2, ep4, and ep5 in Fig. 2.). *Complex episode* contains sub-episodes corresponding to the work done on its incorporated document units. Thus complex episodes have a hierarchical structure "levels" according to the document unit type. An example of a complex episode is (ep1, ep3 in Fig. 2.). We know that there can be other methods to cut a trace into episodes, but in our case the document unit type is a simple and efficient criteria.

Trace Composition

The system traces the actions of the connected user; traces are stored in a relational database that keeps track of the actions together with the affected objects (collections, pages, metadata…). As shown in **Fig. 2.**, a trace is composed of episodes ep; each episode is composed of an ordered list of actions a_i :

$$ep = (a_i, \quad i = 1, n)\,. \tag{1}$$

Actions have a type *AType* and a set of parameters Pi

$$a = \left(AType, P_{i\ (i=1,m)}\right). \tag{2}$$

For example: an action with the type: AddAnnotation has the parameters: AnnotationID and DocumentUnitID.

A parameter *P* has a parameter type *PType* and parameter value *Pvalue*

$$P = (PType, Pvalue)\,. \tag{3}$$

An example about a user trace is shown in **Fig. 2.**, demonstrating the trace composition of episodes.

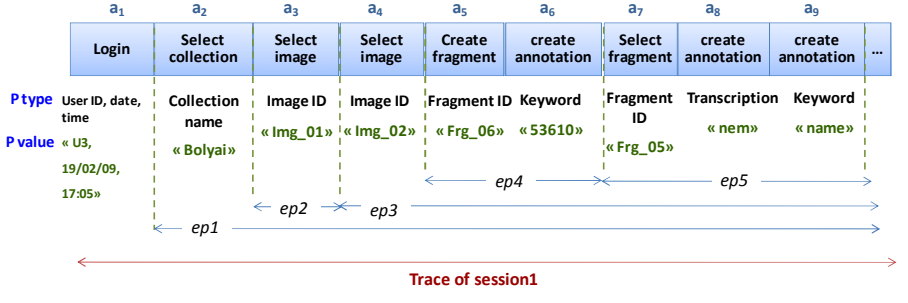


Fig. 2. Trace composition with the hierarchical episode structure

In Fig. 2, we can see trace hierarchical structure, it is composed of an episode-parent (ep1) about the collection "Bolyai". This episode contains two sub episodes (ep2) about image "img_01" and (ep3) about the image "img_02". Furthermore, (ep3) has two sub-episodes: (ep4) and (ep5) concerning two fragments of the image.

We consider that the actions (*create, select*) are similar when we compare the traces. The actions (*create, select, search*) are used to separate the trace episodes, these

actions begin new episodes. The action *Search* begins a new episode and a sequence of search actions is grouped together in one episode. Usually the actions (*delete* and *modify*) follow the *select* action. *Login begins a session trace and logout* terminates a session trace.

Trace episodes are generated at the moment when the user interacts with the system. The system captures a user's experience by registering the actions during his work; it arranges the actions sequence into episodes depending on the document unit *id*. Once traces are registered, they can be reused by the system. We treat episodes as reusable cases.

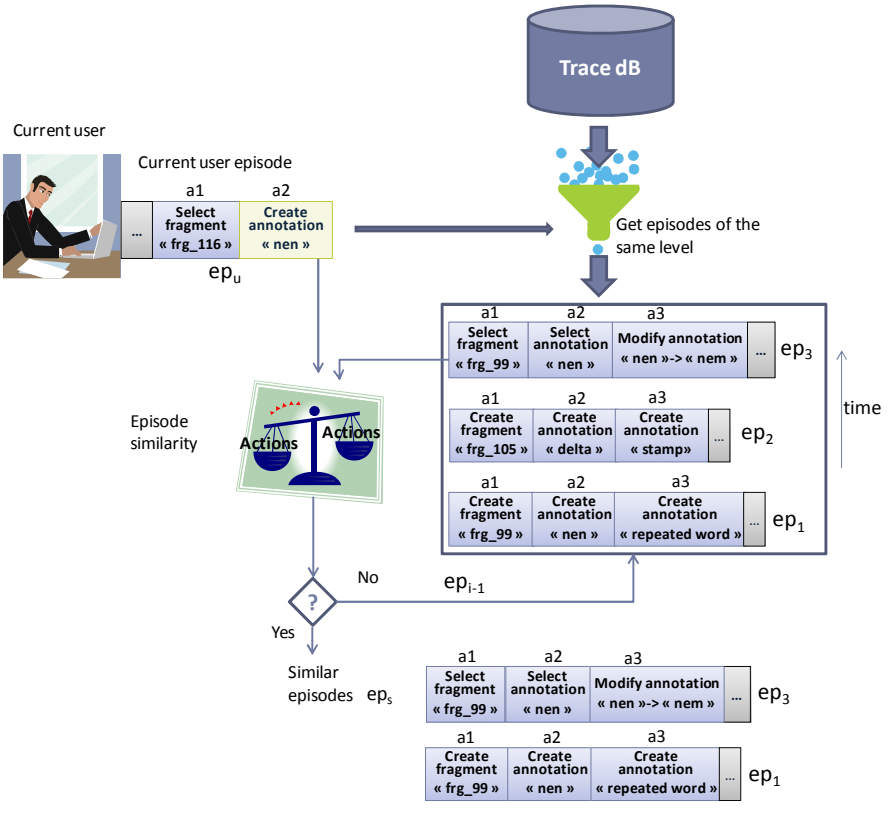


Fig. 3. Episode comparison process

Episode Comparison and Similarity

To assist a user during his session, we compare the last episode of his trace ep_u with the registered episodes of the same level to find similar ones (see Fig. 3).

We are more interested in the user last actions; the intention is to retrieve the most similar episodes to his current episode. When comparing episodes, the system compares episodes that have the same level. The target problem is the next action that the user has to/may do.

Throughout this paper we use the word *episode similarity* as a measure of likeness between two users' episodes. Episode similarity depends on similarity between their actions. In what follows, we define our comparison method for episodes. Comparing episodes enables finding similar situations to the current one. Once the most similar situations are identified, the system will calculate, as a next step, the recommendations to assist the user.

The next formula gives the similarity Sim_{ep} between two episodes *e1* and *e2*:

$$Sim_{ep}(e_1, e_2) = \sum_{i=1}^{|e_1|} \sum_{j=1}^{|e_2|} (Sim_{Action}(a1_i, a2_j)) \,. \quad (4)$$

$where\ a1_i \in e_1\ and\ , a2_i \in , e_2\ with\ i: 1..|e_1|, and\ j: 1..|e_2|$

And |ei| is the number of the actions in the episode

If Sim_{Action} is below a specified threshold, it will be considered as 0. This threshold is used to assure the quality of the recommendations. It was defined by our experimentations to 0.7 (see section 4).

To calculate the similarity Sim_{Action} between two actions a1, a2 we calculate the number of the parameters that they have, if the two actions have different parameter numbers then we consider that the actions are different and the similarity between the two actions equals to zero, otherwise if the two actions have the same number of parameters we get the number of the parameters and start to calculate the similarity between them depending on the formulate:

$$Sim_{Action}(a1, a2) = Sim_{Type}(AType_1, AType_2) * \frac{\sum_{i=1}^{m} Sim_{par}(P1_i, P2_i)}{2m}. \quad (5)$$

where m is the number of parameters. *AType* is the action type and *P* is the action parameter.

In equation (5), in order to have a similarity value in the range [0,1], we divide Sim_{par} by (2m) because this similarity is in the range [0,2], as a result of the equation (6)

The similarity between the two types of actions is given in Table 1:

Table 1. Table of similarity between action types

Action type	Login, logout	Select	Create	Delete	Modify
Login, logout	1	0	0	0	0
Select	0	1	1	0	0
Create	0	1	1	0	0
Delete	0	0	0	1	0
Modify	0	0	0	0	1

If $Sim_{Type} = 0$ then Sim_{Action} will be zero also, we do not continue to calculate the similarity between their parameters. If $Sim_{Type} = 1$, we compare the parameters of the two actions. For two actions of the same type, the parameters are stored in the identical order. Thus we can couple the parameters from the two actions in order to compare their types (equation 7) and values (equation 8). Once Sim_{ptype} and Sim_{pvalue} are calculated, we calculate Sim_{par} by adding these two values, the reason of using the addition

is to have a positive degree of similarity between parameters if they have the same parameter value of dissimilar parameter type. For example, two annotations that have the same value "nen" of annotation types "Keyword" and "Transcription".

$$Sim_{par}(P_1, P_2) = Sim_{ptype}(PType_1, PType_2) + Sim_{pvalue}(PValue_1, PValue_2)\ . \tag{6}$$

$$Sim_{ptype}(PType_1, PType_2) = \begin{cases} 0 \ \ if \ \ PType_1 \neq PType_2 \\ 1 \ if \ \ PType_1 = PType_2 \end{cases} . \tag{7}$$

If Sim_{ptype} is zero, we do not compare their *pvalue*. If it equals 1, we calculate the similarity between the *pvalue*. Sim_{pvalue} is calculated depending on the distance between the two values.

$$Sim_{pvalue}(PValue_1, PValue_2) = 1 - distance\ . \tag{8}$$

For parameters of number type, such as the coordinates of fragment points, the distance is given by the formula:

$$distance = \frac{|PValue_1 - PValue_2\ |}{(PValue_1 + PValue_2)}. \tag{9}$$

For parameters of string type such as annotation value, collection name and image name, we measure the distance using Levenshtein algorithm (also called edit distance). It can be also calculated by different algorithms or the use of an ontology that we think to integrate in the future.
For id parameters, the distance is calculated with the formula:

$$\text{distance} = \begin{cases} 1 \ \ if PValue_1 \neq PValue_2 \\ 0 \ if \ PValue_1 = PValue_2 \end{cases} . \tag{10}$$

In our system, to choose the best case from the retrieved cases, we use the recommender algorithm that takes as input the similar episodes to the current user episode, and suggest the user to do actions relative to his current work.

The recommender system takes into consideration the user profile and the current user episode to make a decision about the best actions to recommend, as shown in Fig. 4.

Recommendations Based on User Registered Traces

The objective of user/document recommendation is to reduce user's time and effort to reach and annotate documents. In our collaborative environment, the role of the assistant is to answer the question: *how to potentially reuse information created by one user to help another user*?

The goal of the trace based recommendation is to provide help in situations that are difficult to formalize. The problems aimed to be assisted are "what to do?" type problems rather than "how to do?" type problems. Transcribing a text fragment in an ancient manuscript needs interpretation, historical and linguistic knowledge and it is also possible that there can be several correct transcriptions. The assistance of tasks such as the transcription is a typical situation where trace based assistance can work.

The developed assistant (recommender system) calculates the most appropriate episode among the retrieved episodes for the actual user situation. It takes into consideration many factors to make its recommendations: the similar episodes (ep_S) that it found, the last action from the current user episode (ep_U), and the user profile (his groups). The recommender marks the most similar action in each episode (ep_S) to the user one, and then it gets the next action (a_n) to each marked action. The recommender examines the most repeated (a_n), and the user who left this action before generating the suggestions.

The assistant will recommend the user to do an action or to modify some parameters; it may recommend him using episodes of the same or of other levels.

We also need to mention that case recommendation cannot be static and it depends on current user tasks. We cannot couple (problem-solution) because traces are being added with each system exploitation, and new or important solutions could be found in new traces.

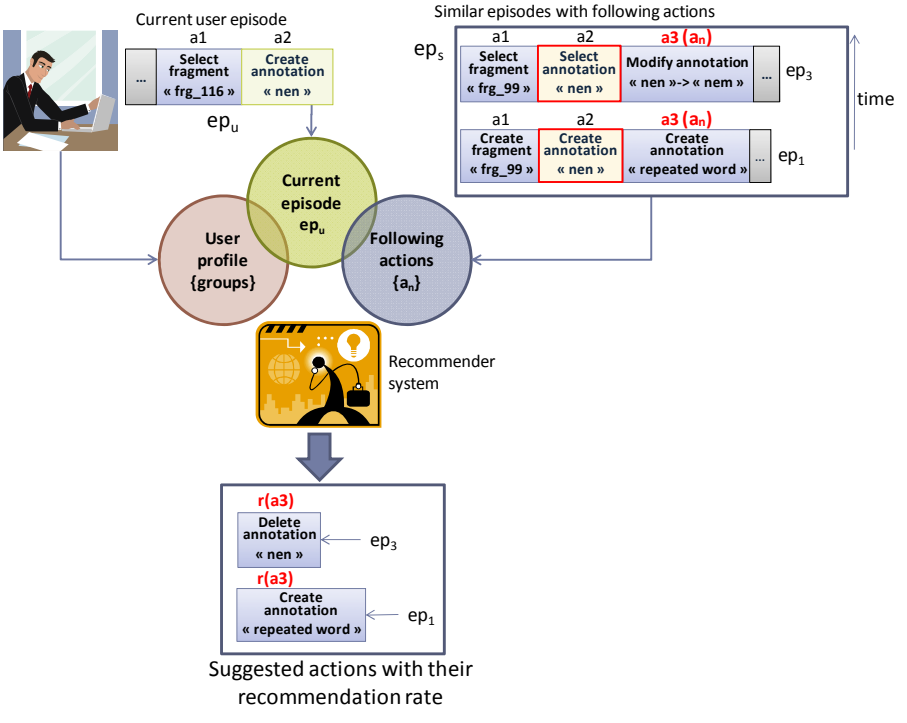


Fig. 4. Deciding which actions to be recommended to the user

Let us see an example about reusing a traced user action to assist another person (Fig. 5.). "User1" defines a fragment on an image using the annotation tool, then he adds an annotation "nen" of type "transcription" into his fragment (N_99) (Fig. 5.(A)). The system cannot recognize if the added annotations are correct or not.

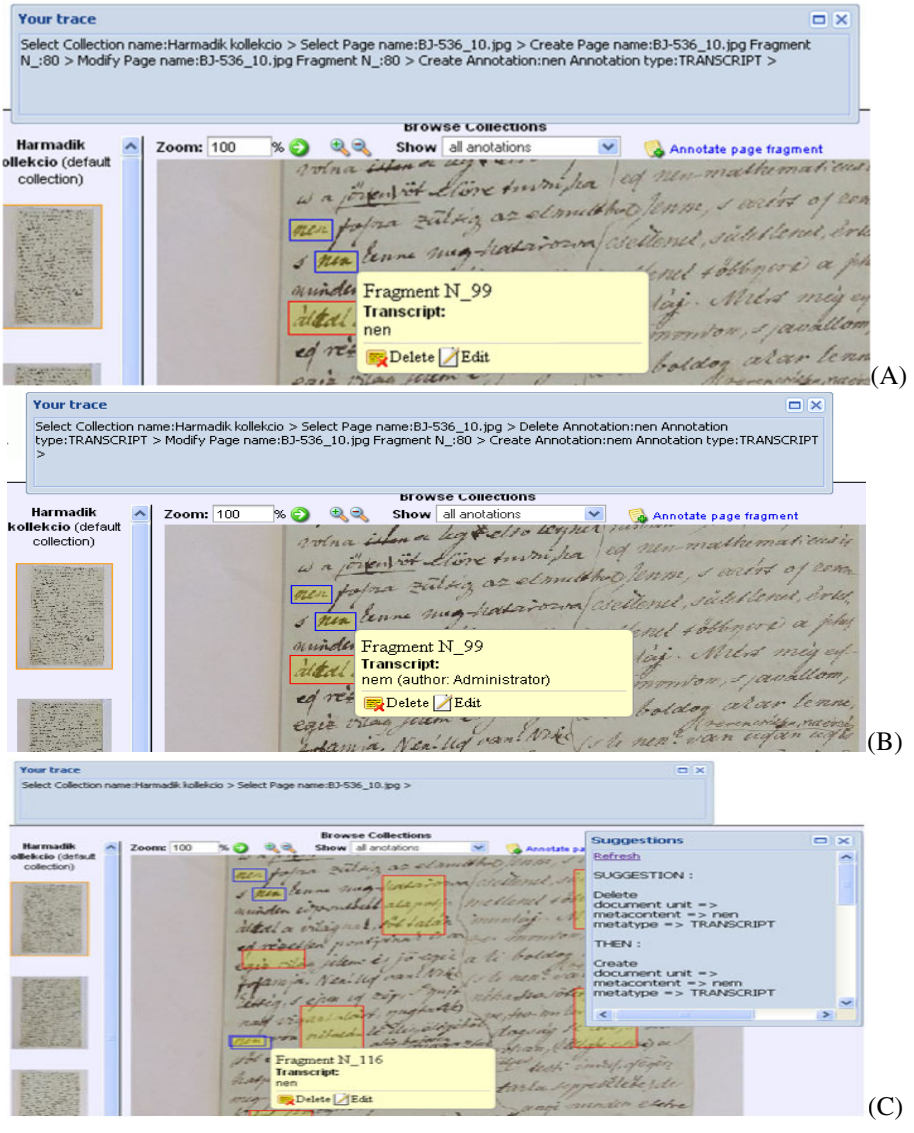


Fig. 5. Example of assisting the user during the annotation process

Another user "User2" corrects the annotation "nen" of the "User1" fragment into "nem", (Fig. 5.(B)), thus (Fragment N_99) was corrected by a user who knows the manuscript language. The tracing system registers these correction actions in the traces database. The recommender system will use these traces to recommend other users with related actions. When a user "User3" defines (Fragment N_116) (Fig. 5.(C)), and writes a word corresponding to an old value "nen" that has been modified, the assistant searches in the traces, gets similar episodes to the user, then recommends the user with the *next* actions, from the retrieved traces. In our example, an earlier

instance of the just entered keyword was corrected, the system proposes the new value "nem" (corrected one) held by the annotation. When the system makes suggestions, the user can refuse the assistant's proposition and continue his work.

By using the same principle, the assistant can also correct a newly created fragment suggesting deleting it, because another fragment has been deleted in almost the same position. This is done if the system evaluates a high similarity between the fragments size and position.

In (Fig. 6) we illustrate the details of user traces for our example in Fig. 5. The user "User3" is the current user. He annotates an image from the "Bolyai" collection. The assistant filters the traces depending on the type of the last document unit where "User3" is working (ep_X), then it extracts the most similar episodes by calling the similarity function for this episodes beginning with the most recent registered episode till it find a very similar case. The results of the comparison in the example are respectively episodes (ep_Y), (ep_Z) of "User1" and "User2" traces. By comparing fragment episodes, the system finds that "modify annotation" was made in this fragment after "Select annotation" since select annotation is equal to create annotation. The assistant will use this correction to propose the user "User3" the correction "nem" when he enters the annotation "nen". The assistant supposes that the user made a mistake according to the trace of "User2".

We consider that it is important to imply the user in the tracing and recommendation process. The user has to be aware that his actions are traced, and how this tracing is

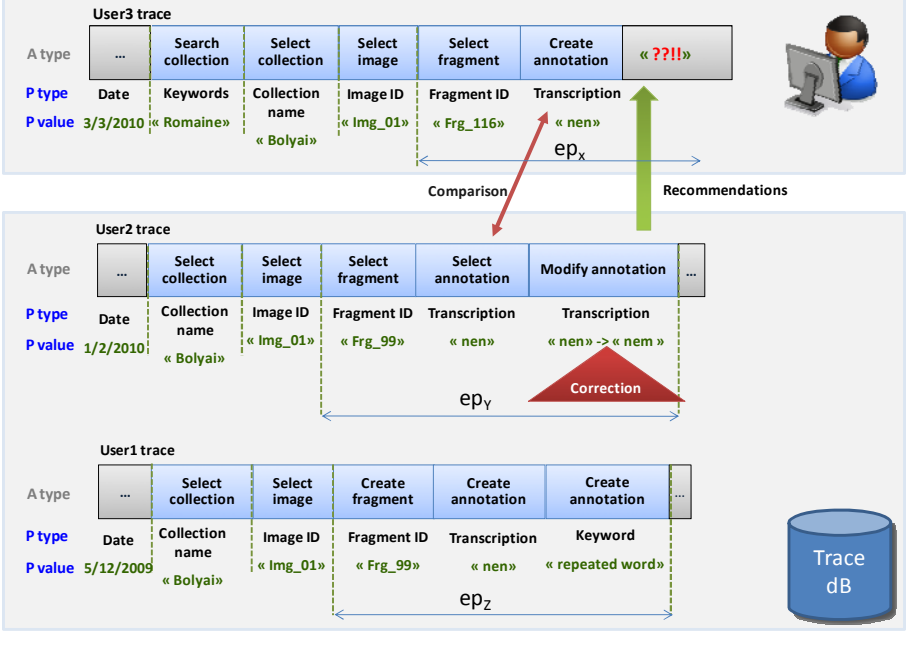


Fig. 6. Comparison between traces to make recommendations

done. We provide methods to show the trace in construction to the user, enabling him to stop the tracing. We also show for the recommendation the episode that gave them.

Of course, the tracing and the recommendation is to be shown in a non intrusive way on the user interface. The primary goal of the user is to carry out the annotation or the search. The currently created trace and the recommendation should appear somewhere at the edge of the main interface

4 Evaluation

Actually, we developed an online archive prototype called ARMARIUS to evaluate our system; it contains many collections of handwritten manuscripts. The more users work and leave traces, the more the system will have relative suggestions. This means that it is automatically fed.

Evaluating the performance of a digital archive application has different faces; we chose to evaluate the quality of user recommendations. Thus, we chose the most used metrics (*precision* and *recall*) to evaluate recommender system quality.

To realize the experimentation, we chose a sample trace database to evaluate the recommendations, and then we conducted a series of experiments using real user-data from different user profiles. Furthermore, we were particularly interested to insure the effectiveness of episode similarity degree presented by our technique, and the high quality recommendations issued from this similarity of knowledge. Thus, in recommendation quality measurement we applied the metrics on the same database sample and we changed the action similarity degree. Results of our experimentations are shown in Fig. 7 and Fig. 8.

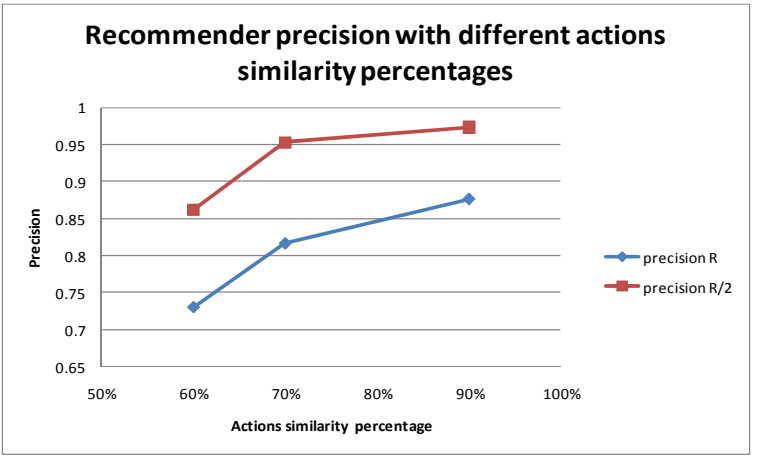


Fig. 7. Recommender system precision

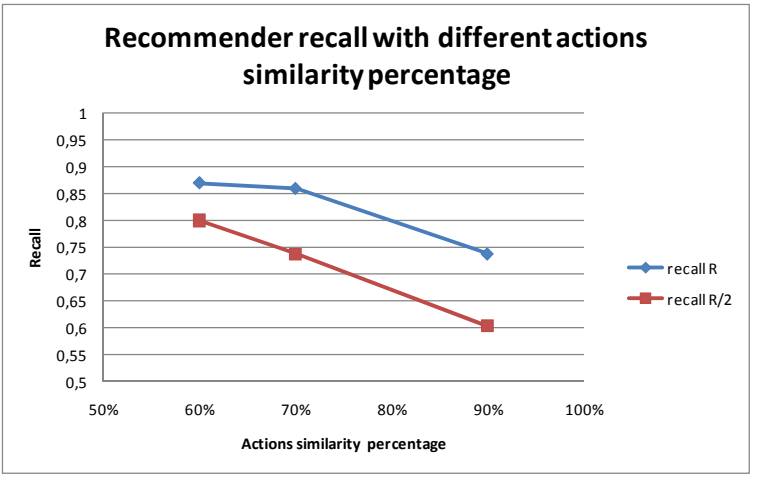


Fig. 8. Recommender system recall

The results show that the system gives best recommendations with a similarity degree between actions equal to 70%.

5 Conclusion and Further Works

In this article we presented a web application to expose and annotate the digitized images about old and handwritten manuscripts. The annotation process is done manually and remotely. Users work in collaboration to realize a correct annotation, and benefit from the experience of the others to annotate documents.

User interaction with the application is tracked to register the important actions and the manipulated objects. After that, registered user actions are used by the assistant, to help and to recommend users according to their actual context.

Subsequently the assistant recommends the actions that are in the similar episodes. A smarter assistant can study the context of the user episode and suggest more intelligent recommendations.

The episodic trace structure in our system eases the extraction and the selection of cases, the highest specificity of the cases selected facilitates their use.

On the other hand, the suggestion that is not used by any user (a wrong one or a non relative suggestion) will be not suggested frequently in the future, because the system will memorize the trace of the user who didn't accept the suggestion and this trace will be suggested by the assistance for other users.

We also find that the quality of our system is good, due to the filter that we apply on the traces database to retrieve only the episodes of the same type of the current document unit.

We do not register the couples (problem-solution) because traces are being added all the time, so a set of traces that is used to recommend the user with a solution will not be reused when new traces are added and that are more similar to the user actions. At last, hitherto we cannot predict user actions and interests.

In our future work we aim to discover the user intention to identify useful features relating to an actor's future actions without an explicit representation of the plan generation model.

References

1. Dublin Core Metadata Element Set, http://dublincore.org/documents/dces/
2. METS Official Website, http://www.loc.gov/standards/mets/
3. MARC Standards, http://www.loc.gov/marc/index.html
4. Gatos, B., Antonacopoulos, A., Stamatopoulos, N.: Handwriting Segmentation Contest. In: 9th International Conference on Document Analysis and Recognition ICDAR, pp. 1284–1288. IEEE Press, Brazil (2007)
5. Champin, P., Prié, Y., Mille, A.: MUSETTE: Modelling USEs and Tasks for Tracing Experience. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 279–286. Springer, Heidelberg (2003)
6. Egyed-Zsigmond, E., Mille, A., Prié, Y.: Club ♣ (Trèfle): A Use Trace Model. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 146–160. Springer, Heidelberg (2003)
7. Settouti, L.S., Prié, Y., Marty, J., Mille, A.: A Trace-Based System for Technology-Enhanced Learning Systems Personalization. In: 9th IEEE International Conference on Advanced Learning Technologies ICALT, pp. 93–97. IEEE Press, Latvia (2009)
8. Cram, D., Jouvin, D., Mille, A.: Visualizing Interaction Traces to improve Reflexivity in Synchronous Collaborative e-Learning Activities. In: 6th European Conference on e-Learning, pp. 147–158. Academic Conferences Limited, Copenhagen (2007)
9. Xu, K., Muñoz-Avila, H.: CaBMA: a case-based reasoning system for capturing, refining, and reusing project plans. In: Knowledge and Information Systems, vol. 15, pp. 215–232. Springer, London (2008)
10. Marling, C., Whitehouse, P.: Case-Based Reasoning in the Care of Alzheimer's Disease Patients. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 702–715. Springer, Heidelberg (2001)

On-the-Fly Adaptive Planning for Game-Based Learning

Ioana Hulpuş[1,2], Manuel Fradinho[1], and Conor Hayes[2]

[1] Cyntelix, Galway, Ireland
ihulpus@cyntelix.com, mfradinho@cyntelix.com
[2] Digital Enterprise Research Institute,
NUI Galway, Ireland
conor.hayes@deri.org, ioana.hulpus@deri.org

Abstract. In this paper, we present a model for competency development using serious games, which is underpinned by a hierarchical case-based planning strategy. In our model, a learner's objectives are addressed by retrieving a suitable learning plan in a two-stage retrieval process. First of all, a suitable abstract plan is retrieved and personalised to the learner's specific requirements. In the second stage, the plan is incrementally instantiated as the learner engages with the learning material. Each instantiated plan is composed of a series of stories - interactive narratives designed to improve the learner's competence within a particular learning domain. The sequence of stories in an instantiated plan is guided by the planner, which monitors the learner performance and suggests the next learning step. To create each story, the learner's competency proficiency and performance assessment history are considered. A new story is created to further progress the plan instantiation. The plan succeeds when the user consistently reaches a required level of proficiency. The successful instantiated plan trace is stored in an experience repository and forms a knowledge base on which introspective learning techniques are applied to justify and/or refine abstract plan composition.

1 Introduction

Business is becoming increasingly global, distributed, faster paced and fiercely competitive [12]. In this context, corporate training is a major concern and challenge for companies. Traditional training methods are often expensive as they involve getting experts and trainees together in a common place [21]. While online courses have become an alternative, they also have a problem that content may be unsynchronised with the dynamic needs of the business [18]. In this context, the interest in interactive games for learning has been growing [7,14,32]. Our paper focuses on how to retrieve and adapt learning-plans as the learner engages with the learning material. The basic unit of instruction in our system is what is known as a *serious game*, an interactive role-playing narrative designed to teach particular competencies. A learning-plan is a sequence of such narratives that is customised to the learners particular requirements and skill levels, for

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 375–389, 2010.

which there is generally insufficient data until the learner begins to play. Our goal is to research how the game-based learning process can be continuously adapted towards the efficient development of required competencies for each individual learner. This entails the use of a planner that collects feedback from the learner interaction with the suggested plans and uses this feedback to learn which parts of the plan are failing and how to recover. As described by Hammond [10], case-based planning (CBP) systems have the capability of learning from the interaction with the human users. They can also anticipate problems and learn how to avoid them. These features make CBP attractive for use in learning environments.

There has been a persistent interest in learning among case-based reasoning (CBR) practitioners since the early days of CBR. This has yielded good results particularly in intelligent tutoring systems (ITS) [28]. Drawing upon this previous work and recent learning theory, we take a step further and present a model for online-learning with serious games that is underpinned by a case-based planning strategy. We also research how alternative plans which target the same goals can be represented, and retrieved based on their outcomes for different learners. The retrieved plans are then incrementally instantiated during execution, taking into account the information derived from constant performance monitoring.

Our research is done within the context of the TARGET[1] European Project. TARGET's goal is to implement a new type of Technology Enhanced Learning(TEL) environment which shortens the time to competence for learners within enterprises. The targeted learning domains are innovation and project management. The TARGET platform is based on serious games.

The remaining of this paper is structured as follows. The next section summarises related work that leverages the use of CBR for the purpose of supporting human learning. Then, in Section 3 we present an overview of our research direction in the field of CBP and learning with serious games. In Section 4 we describe how our work relates to the family of CBP systems and provide a detailed description of our proposed solution. In Section 5 we present a discussion of the implications of our system, and then we conclude with some final remarks on future work.

2 Case-Based Reasoning and Human Learning

Using CBR to facilitate human learning has appealed to many researchers, partly due to its roots in cognitive science and its similarity to human problem-solving behavior [23,15,22]. In [23], Schank argues for a goal-based approach to education, in which case acquisition plays a central role. In [15], Kolodner suggests how CBR can enhance problem-based learning by recommending relevant problems to learners.

The research done by Jonassen and Hernandez-Serrano [13], also supports the use of CBR for instructional design based on problem solving. The authors argue for a story-based system supported by CBR that enables learning from

[1] http://www.reachyourtarget.org

other people's experiences. These experiences form a case library of narratives from employees that describe real-life work-related problems and their solutions. However, we are focused on how to create a suitable learning plan, rather than retrieve a similar narrative.

There are already promising results of using CBR in ITS. For example, ILMDA (Intelligent Learning Material Delivery Agent) [28] that combines CBR with system meta-learning in the domain of computer science for undergraduates. An approach that comes closer to serious games is presented in [8,9]. They present a metaphorical simulation of the Java Virtual Machine to help students learn the Java language and reinforce their understanding of object-oriented programming. These two systems operate in well defined domains, where problems have a direct mapping to correct solutions. However, we are creating a system for use in two very complex domains: Project Management and Innovation Support. In these domains, the problems are open-ended and the competencies required are complex and difficult to model. Therefore, our approach is to create an open environment capable of reasoning with very complex, poorly structured domain knowledge. Furthermore, we focus on long term learning goals. For this, a single learning episode is not enough; the system must design consistent and coherent *learning plans*. As such, we use a CBP approach rather than classical CBR.

3 Approach Overview

The term competency carries many definitions in the related literature [11]. The definition that matches the best the way we use and assess competencies is that they are a set of personal characteristics, knowledge, skills and abilities that help successfully perform certain tasks, actions or functions and are relatively stable across different situations [31]. The initial TARGET competency framework will use the IPMA Competence Baseline[2] and SHL Universal Competency Framework[3].

The competencies of an individual are characterised by a state of attainment (degree of mastery) which the system estimates by analysing the user's *performances*. Therefore we define a competency profile as a set of competency-level pairs. A learner in our system has assigned both a *current competency profile*, and a *targeted competency profile*.

The core unit of instruction is represented by *stories* which are interactive narratives the learner engages with as he assumes a specific role, with a specific mission. A learning plan is a personalised sequence of stories. Throughout a story execution, towards the achievement of his mission, the learner is put in various *situations* meant to develop and evaluate his competencies. Each story has at its core a *story template* which can accommodate several situations. In order to create a story starting from a story template, a sequence of these potential situations is selected, based on the learner needs and requirements. The learning

[2] http://www.ipma.ch/certification/standards/Pages/ICBV3.aspx
[3] http://www.shl.com/OurScience/Documents/SHLUniversalCompetencyFramework.pdf

plan is created in a two stage process. First, an *abstract plan* is generated, as a sequence of story templates. Then, the concrete learning plan is instantiated incrementally, each story being created starting from the corresponding story template, when the plan execution thread reaches it. This process being the central focus of our paper, we describe it in more detail in the following sections.

3.1 Example

The TARGET game engine has the responsibility of instantiating stories with required competency-training situations, player roles and level, non-player characters and narrative thread. While the main focus on the paper is how the CBP mechanism can select stories, adapt and link them to create coherent learning plans, we give in this section a brief example of a story template and its "child" stories. We will come back to this example throughout the paper, to illustrate how the CBP engine decides the instantiation of the learning plan.

Figure 1 illustrates an example of a story template with its corresponding possible situations, and the competencies trained and evaluated in each situation. The situations are labeled with letters A-G. The arrows which lead from one situation to another show the possible flows of situations. For example, situation A "*Partner does not produce*", can lead to one or both situations B "*Conflict between partners*" and F "*Tasks not achieved or postponed*". The dashed lines in the figure illustrate the links between situations and the related competencies. For example, situation B trains and evaluates *conflict resolution.*

For each story template, an instantiated story consists of one path through its graph of situations. The game engine will instantiate the story according to the requirements stated by the CBP module. The story instantiations consists of: (i) selection of the story situations, (ii) instantiation of story parameters. Given the example in Figure 1, we can consider a user who wants to train in *conflict resolution, crisis management* and *resource planning.* Then, a candidate story is created by switching on the situations B, C and D. To train the required competencies, the learner chooses the role of project coordinator. During his

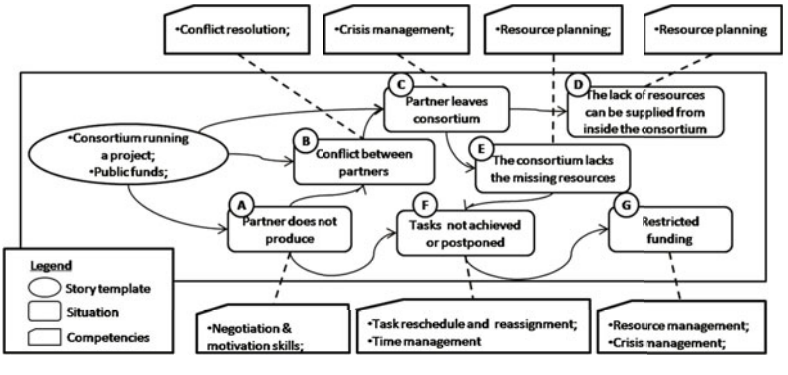


Fig. 1. Example of Story Template and Potential Situations

experience, the user is first evaluated on how he handles the conflict between the partners. Then he is evaluated on how he manages the situation where a partner leaves the consortium where other partners have sufficient resources to overcome the loss. Other candidate stories starting from the same template can be: $B \rightarrow C \rightarrow E$, or even $B \rightarrow C \rightarrow E \rightarrow F \rightarrow G$, which would suit a more experienced learner, or a learner who needs a more complex story.

Regarding the story parameters, for the given example such parameters would be the number of partners in the consortium, the number of partners involved in the conflict and the personality of the non-player-characters. All these contribute to an easier or more complicated story. Having the set of needed competencies, the case-based planner can choose the story templates to compose the abstract plan, but in order to know how to instantiate the stories (i.e, how to choose from the three stories we described above and instantiate the story parameters), it needs to know the performance of the user within the plan. Therefore, the story instantiated is delayed until the plan execution thread reaches its story template.

3.2 Principles of Linking Stories in a Learning Plan

A story represents a basic narrative unit for learning where closely related interactive situations focus on training the learner in a closely related competency set. However, a learning plan is a series of stories incrementally instantiated according to a learning strategy, but guided by a constant flow of feedback of the learner performance in each story. For example, a story may need to be repeated as a variation and situations added or removed and learning levels adjusted. The learner may be ready to move to a higher level in the next story or may not need particular situations to be instantiated. The learning plan must be created so that the flow of stories the user engages with lead him to the targeted competencies. For the creation of the learning plan we must consider the fact that the way learning episodes relate to each other is very important in order to keep the learner motivated and on the flow.

There are several aspects which we focus on in creating the learning plans. First of all, we have to consider if the company has a domain model where competencies have specific relations between them (e.g. decomposition, prerequisites. constraints). The learning model has to take these into account.

Secondly, it is important that the learning plan builds new competencies on top of existing ones. Following this principle, new competencies are developed in stories which also relate to possessed competencies. As well, within the learning plan the story complexity and the difficulty increases as the user performs.

The third principle is that learning needs practice, and often recursiveness and/or repetition. The variation theory of learning [19] and the cognitive flexibility theory [30] argue that practice of the same thing in different contexts, not pure repetition, leads to better learning outcomes. Following this principle, a learning plan should train the same competency in typical but varied situations until the learner reaches the desired level and also subject him to at least one atypical situation.

3.3 Reasoning with Abstraction to Build Learning Plans

In a game-based learning environment, a learning plan is an ordered list of stories meant to support the learner until he reaches the desired competency profile. The learning plan has to be adapted to the learner data like age, gender, cultural background. As well, it has to dynamically adapt based on the learner performances within the plan. This means that the planner does not have enough knowledge to create the whole plan in the initial stage of the planning. Therefore, at this stage several abstract plans are created, as sequences of story templates, and the learner can choose which one to execute. The story instances are created on-the-fly based on the story templates as the plan execution thread reaches them. At this stage the system has accumulated knowledge from the user's performances so far, and can personalise each story.

This methodology is inspired from the use of abstraction in case-based reasoning. By using abstraction, the less relevant features of a problem description are ignored in a first stage, leading to an abstract solution. Then, as the ignored features of the problem are being considered, the final concrete solution is derived from the abstract one [2]. In our case, the reasoner does not ignore features of the problem, but has to reason with an incomplete problem, which becomes complete as the solution is executed.

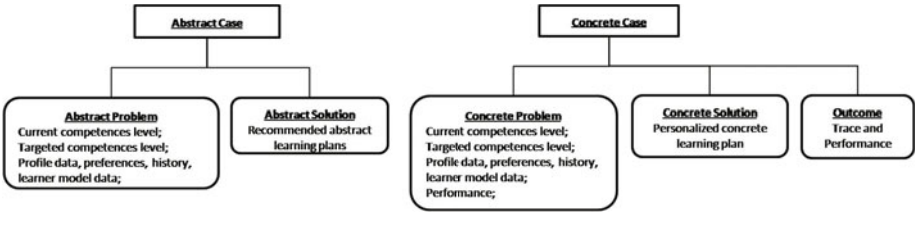


Fig. 2. Single Case Components

Following this hierarchical model, the abstract cases solve the problem requirements related to competency needs and learner profile data, by suggesting several abstract plans. The concrete cases have the problem enriched with the learner's performances, and therefore the solution is an iteratively created concrete learning plan. The two types of cases are represented in Figure 2.

4 Hierarchical CBP for Personalised Learning Plans

For planning on several layers of abstraction, many terms have been used in literature, the most common ones being hierarchical case-based reasoning [25] and stratified case-based reasoning [5]. The basic idea is that in the hierarchy of cases, only the "leaf" cases are concrete, and all the other nodes are "abstract cases". The studies made by Bergmann and Wilke [3], Branting and Aha [5] as well as

Smyth et al. [25], to name just a few, prove the advantages of this approach. Compared to classical case-based planning, it shows significant improvements in efficiency of retrieval and adaptation.

There are still differences in these approaches. In some of them, the abstract cases are created by abstraction and generalisation [1] of concrete cases. This is a bottom-up process which consists of merging concrete cases based on similar features. These are then discriminated based on their specific features, obtaining a hierarchical tree structure. In these systems the plans are retrieved entirely, and the new solutions are created by adapting them, e.g., in the PARIS system [3]. Other approaches create the abstract cases starting from task or goal decomposition. The concrete cases are the atomic actions which cannot be decomposed any more. The recent work of Lee, Chang and Liu in [17] uses such an approach. In these type of systems, each planning step is retrieved individually and then they are integrated to form the adapted solution. Smyth, Keane and Cunningham combine the two types of hierarchies in Déjà-Vu [25].

In our research, each planning step (a story instantiation) is not retrieved individually but is adapted by the user's previous interactions. Hence in our approach plan instantiation and the final steps of plan adaptation occur together. Generated abstract plans are presented to the user and he makes the choice of which one to follow. Every story generation is directly followed by user execution and system evaluation. The results are used to create new tasks for the subsequent steps.

In our solution, there are two levels of abstraction. In the systems which use abstraction it is common that the depth of the hierarchy is flexible, as the abstract cases are generated dynamically as soon as new cases share common features. The results of Branting and Aha in [5] show a significant improvement in efficiency when 3-4 levels of abstractions are used. If this proves to be valid in our system too, we will consider the option of using dynamic abstraction within each of the two current layers.

4.1 Abstract Plans

Abstract Case Representation. In order to represent the abstract cases we have to consider that there can exist multiple learning plans achieving the same learning goals. Consequently, all the plans which have similar initial states and goals are grouped under the same root. Then a description node is created for each abstract plan. This description node contains the users who executed the plan in the past, and a summary of their experiences (the plan outcome). This abstract-plan outcome includes information like the time the user needed to complete the plan, the average number of story repetitions, and the performances. It is important to note that this summary, although part of the abstract case representation, is extracted from the concrete layer. This way, we compensate for the loss of information which is inherent in reasoning with abstract cases [3]. Including this information in the description node gives us the possibility of combining CBR with collective filtering. In this scenario, collective performance information from similar learners will help in ranking candidate cases. The model

also supports the inclusion in the description of learners who were recommended the plan but did not choose to execute it. This information lends itself to providing explanations (e.g. 8 out of 10 learners selected this plan, 6 out of 8 learners completed this plan).

The description nodes have as children the abstract plans they describe. This hierarchy is illustrated in Figure 3. As mentioned in section 3.3, an abstract plan is a list of story templates. In Figure 3, the abstract plan 1 is composed of the story templates *ST1* → *ST2* → *ST3*, and the abstract plan 2 is *ST4* → *ST3*.

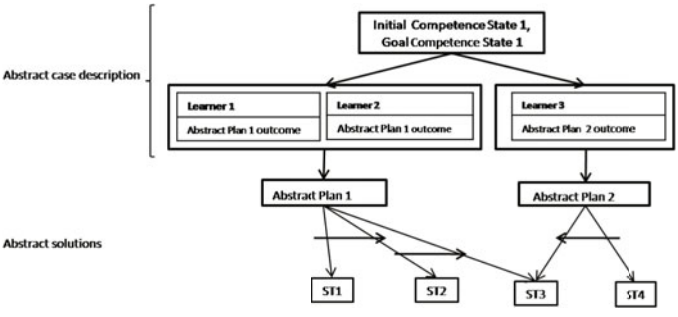


Fig. 3. Abstract Case Representation; ST - story template

Let us define the initial competency state as *(conflict resolution, beginner), (crisis management, beginner), (resource planning, upper average)* and the goal competency state as *(conflict resolution, average), (crisis management, average), (resource planning, expert)*. Then, the story template illustrated in Figure 1 is a good candidate for being part of the two abstract plans. Moreover, since it can bring together situations for all the goal competencies, it is a good candidate for being *ST3*.

Each story template in an abstract plan, has assigned the competencies it has to train and evaluate within that plan, forming an initial set of tasks. For example, let us consider the story template in Figure 1 is labeled *ST3* in Figure 3. Then, within the two abstract plans, the template is assigned the tasks to select situations which match *conflict resolution, crisis management* and *resource planning* since these are the competencies it was chosen for, even if it can support situations addressing other competencies as well (i.e., *negotiation*). This information is used when the story is created. It can be seen as an explanation why the template is part of the abstract plan. Still, this data is not enough for a personalised story. To personalise the story, more tasks to fulfill are assigned to the template as described later in section 4.2.

Abstract Plans Retrieval and Reuse. The retrieval of the abstract learning plan is a top-bottom traversal of the tree presented in Figure 3. This consists of two main steps: during the first step the system matches the current problem's initial state and goal to existing cases in the case base. Considering Figure 3,

this stage retrieves a set of nodes from the first level. During the second step the system retrieves the child nodes of the nodes returned after the first step. Then, for each such child it computes a *suitability* value, rather than a similarity value. The suitability value takes into consideration the learner similarity, the plan outcome for him and as well adaptation complexity [26].

After the most suitable abstract plans are retrieved, they are adapted so that they fulfill all the problem's requests: they fit the current competency of the learner, as well as his targeted competencies. The adaptation consists of adding/removing/replacing story templates from the original abstract plan.

4.2 Concrete Cases

Concrete Case Representation. Concrete case representation inherits from hierarchical representation used by Smyth et al. in Déjà-Vu [25]. The similarity comes from the fact that stories are generated step by step, therefore the final concrete solution is obtained by integrating the individual stories. Still, our suggested planner executes each step before instantiating the next. Both approaches permit multiple case reuse, which means that each planning step can be retrieved and reused from multiple cases.

As described in Figure 2, a component of the concrete problem is the set of previous user performances. Therefore, a learning plan that has been even partially executed by the learner is stored along with its performance score as a plan trace. The performances are analysed and depending on the result, the system selects and tailors the next story to play. Figure 4 shows the concrete plans layer, standing between the abstract plans layer and performance layer.

In the example in the figure, there are two abstract plans: $ST1 \rightarrow ST2 \rightarrow ST3$, and $ST4 \rightarrow ST3$. The first abstract plan has two instantiations, i.e., two concrete learning plans: $S1 \rightarrow S2 \rightarrow S3$ and $S1a \rightarrow S2 \rightarrow S3a$. The second abstract plan

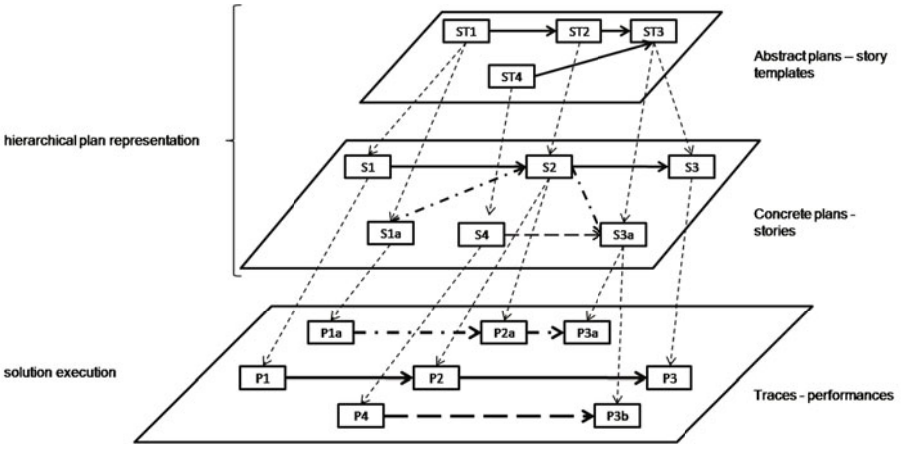


Fig. 4. Learning plans hierarchy; ST - story template; S - story; P - performance

has only one instantiation in the case base: *S4* → *S3a*. The arrows from the abstract plan to the concrete plan show how the story templates have been instantiated. The third layer shows how each concrete plan forms a trace as it is being executed. For example, the concrete plan *S1* → *S2* → *S3*, was executed once, leading to the trace: *P1* → *P2* → *P3*.

Let us consider the example in section 3.1, and take two learners, *L1* and *L2*. Because they have similar current and targeted competencies, they are recommended the same abstract plan: *ST1* → *ST2* → *ST3*. Let us further consider that the story template in Figure 1 is labeled *ST3* in Figure 4. Before the instantiation of *ST3*, the learner *L1* has executed two stories, *S1* and *S2*, with performances *P1* and *P2*. At the same stage, the learner *L2* has executed the stories *S1a* and *S2* with performances *P1a* and *P2a*. *L1* makes a good progress and successfully execute the tasks is short time. The planner can then decide to instantiate the *ST3* template to a complex story, therefore creating story *S3* as the flow of situations $B \rightarrow C \rightarrow E \rightarrow F \rightarrow G$. To make the story challenging, the planner also chooses to enforce a large consortium, with a spread conflict which determines a key partner to leave and cause a big resource gap. At the same time, *L2* has a slow progress, with blockages and long idle times, so the system decides to instantiate *ST3* into *S3a* as the flow of situations $B \rightarrow C \rightarrow D$. To make the story accessible, it defines a consortium of 4-5 partners with only two conflicting partners. A parter with a low contribution has to leave, and the lost resources can be covered from within the remaining consortium.

Planning on First Principles. The learning plan is created step by step, by instantiating the story templates of the abstract plan, at the moment they are needed or when the learner requests it.

Algorithm 1. NextStory(Learner, AbstractPlan, CurrentPlan, step)

```
   Input: Learner L, AbstractPlan AP, CurrentPlan CP, planning step n
 1 S_n = CP[n] ;                          /* S_n is the last executed story in CP */
 2 ST_n = AP[n];
 3 P_n = Performance(L, S_n);
 4 if P_n < ST_n.goal then
 5 |   T = GenerateSelfTasks(L,S_n,P_n);
 6 |   ST_n.tasks = ST_n.tasks ∪ T;
 7 |   S_n^a = CreateStory(ST_n);
 8 |   return S_n^a;
 9 else
10 |   if n + 1 < AP.length then
11 |   |   T = GenerateTasks(S_n, P_n, L);
12 |   |   DistributeTasks(AP, n + 1, T);
13 |   |   S_{n+1} = CreateStory(ST_{n+1});
14 |   |   return S_{n+1};
```

Algorithm 2. DistributeTasks(AbstractPlan, step, Tasks)

```
   Input: AbstractPlan AP, step n, Tasks T
 1 ST_n = AP[n];
 2 foreach task t ∈ T do
 3 |   if ST_n can satisfy t then
 4 |   |   ST_n.tasks = ST_n.tasks ∪ {t};
 5 |   |   T = T \ {t};
 6 if T ≠ ∅ and n + 1 < AP.length then
 7 |   DistributeTasks(AP, n + 1, T);
```

In order to create the next story, the system needs to interpret the previous performances and modify the remainder of the plan accordingly. Algorithm 1 illustrates how this is done. It uses a task creation and distribution mechanism shown in Algorithm 2: after a performance has been analysed a list of tasks is created. If the learner failed to reach the level planned for the current story, the planner can recommend him to replay a variation of the same story with a different difficulty level. Otherwise, the planner sends the package of tasks to the first subsequent story template. The story template keeps for itself the tasks which it can achieve and sends the rest further to the subsequent template in the plan, and so on. In case a story template cannot satisfy any new task, it is considered that it needs no further personalisation, and it is instantiated based on its initial set of tasks, set by the abstract plan.

An example of the concrete-plan creation process based on the task distribution is presented in Figure 5. The dashed arrows in the figure show how each performance triggers the delivery of tasks to the subsequent story template, which keeps for itself the tasks it can achieve and sends the rest forward.

As an example, let us consider that the story template *ST3* represents the story template illustrated in section 3.1, Figure 1. The template receives the tasks *T1* and *T3* due to the performance *P1*. *T1* states that the complexity of the story should be high, and *T3* requires that the *team management* competency should be approached so that it suits a beginner. *ST3* receives the two tasks but, because *T3* refers to a competency the story template cannot address, it can only keep *T1*. *T3* is sent further to the next story template. Due to previous performance *P2a*, *ST3* also receives tasks *T5* and *T6*. *T5* states that the competency *crisis management* should be approached so that it suits an average level learner. *T6* states that the set of competencies needed to successfully achieve the story mission must include the learner's targeted competencies, but not exclusively.

When *ST3* needs to be instantiated, it has to consider the tasks *T1*, *T5* and *T6*. Because of *T1* and *T6*, the story is created so that it brings a large number of situations. Because of *T5*, the situations have to be chosen and adapted so that *crisis management* is required in many situations, but not very demanding. This requirements lead to the instantiation of story *S3* as the flow of situations $B \rightarrow C \rightarrow E \rightarrow F \rightarrow G$ with parameters instantiated so that situations C and G cannot be handled unless the learner has an average level of proficiency in crisis management (due to *T5*).

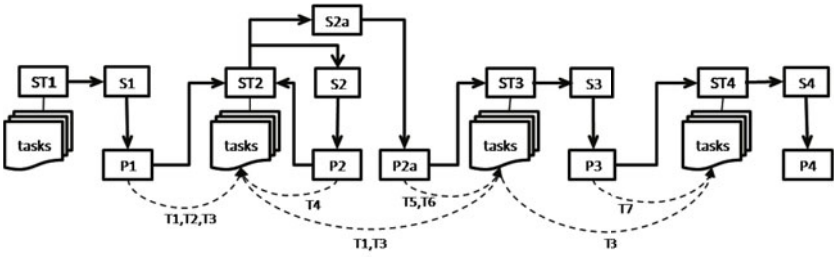


Fig. 5. Plan Instantiation Example

Using CBP Adaptation Techniques to Create Concrete Plans. The way the stories are created at this step from the story templates, can be either based on first-principles, or using one of the case-based-planning adaptation techniques like *derivational* or *transformational analogy* [20,29,6]. If the stories are generated on first-principles planning, then the system does not need to retrieve concrete plans from the case-base. They are created starting from the abstract plan, using only domain knowledge. Knowledge about how the tasks are generated from performances is needed. As well, how to instantiate a story starting from a story template and a set of tasks.

The transformational approach relies on the fact that the system saves entire plans and the new solution is created by reusing the old solution. If we use such an approach, then the system does not care to learn about tasks. If the old story's performance was partially similar to the current story's performance, then the system adapts the next story in the old plan to achieve the new story. In this approach domain knowledge is needed to be able to adapt a story to new previous performance.

On the other hand, using the derivational analogy, the cases are adapted based on the way the old solution has been built. Here, the system does not save the stories, but the tasks which lead to them. If the old story's performance is partially similar to the current story's performance, then the system adapts the old set of tasks and creates the new tasks. Using these tasks, it generates the story. Here, the system needs domain knowledge on how to integrate the tasks in the story creation.

Still, research and evaluation of the possible approaches has to be done before the best fitted solution can be selected.

5 Discussion and Future Work

By now, we have presented how learning plans are created for the learners and adapted to match their needs and performances. Another crucial part of case-based systems is the retain phase, during which the system adds the new cases to the case-base. The case base should avoid redundancy and be kept at a size which does not negatively influence the retrieval and adaptation efficiency. For this, we propose to keep all the traces and experiences in a separate storage, and then periodically carry out maintenance analysis [24] to make sure that only the cases which bring value to the case-base are retained.

Keeping the traces of successful and failed plans allows us to analyse the features and feature weighting that are leading to unsuccessful retrievals. Introspective learning techniques for feature weighting are designed to increase or decrease the weights of selected case features on the basis of problem solving performance [4]. Such techniques have also been used to facilitate easier adaptation of cases [16]. Analysing the repository of plan traces using introspective learning should allow us to improve the retrieval of abstract cases and their adaptation to the learner context.

Throughout this paper we mention the user data like age, gender and demographical details to be used for finding the suitable plan. Although it has been proven that cultural background, age and gender might influence a person's way of learning, we have to analyse if this data is relevant in our system. Therefore, we will use this data only for analysis during the early stages of the case base. If the analysis of cases proves any relation between learning and these parameters, we will consider them for plan retrieval.

Another aspect we have to consider when plans and stories are generated is diversity [27]. We need diversity both for the learner and the system. For the learner, it is important that recommended plans are varied and do not overlap with the user's already executed plans. For the system, it is important that it explores the efficacy of new plans as well, not only relying on old highly evaluated ones.

While the goal of this paper was to present a model of CBP and online learning using serious games, we should discuss our plans for implementation and evaluation. This work is being developed as part of the large European project TARGET, which contains academic and industrial partners. Although the case-based planning engine will be evaluated iteratively in small user trials, the full integration with the game engine and the evaluation of the overall system by TARGET industrial partners will take place in early 2011.

6 Conclusion

We have presented a methodological framework for creating personalised learning plans based on serious games - interactive narratives designed to teach particular competencies. We justified our reasons for proposing a novel a case-based planning approach and described in detail our hierarchical case structure and our iterative retrieval and adaptation process. We proposed that the learning process can be continuously adapted for each individual learner. We showed how alternative plans which target the same goals can be represented, and retrieved based on their outcomes for different learners. The retrieved plans are then adapted on-the-fly, based on an evaluation of the learner's performance. We proposed a hierarchical planning methodology which enables the planner to retrieve and personalise the learning plan for each user. We also examined how plan traces from all learners can be exploited to improve the case base of learning plans. This work is being developed as part of the European project TARGET and will be evaluated iteratively in small user trials. The full evaluation by the TARGET industrial partners is planned for early 2011.

Acknowledgments

This work was partly supported by the TARGET project under contract number FP7-231717 within the Seventh Framework Program, and by the Lion II. project funded by SFI under Grant SFI/08/CE/I1380. The authors also wish to thank Marcel Karnstedt for the many fruitful discussions.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations and system approaches. AI Communications 7(1), 39–59 (1994)
2. Bergmann, R., Wilke, W.: Building and refining abstract planning cases by change of representation language. Journal of Artificial Intelligence Research 3, 53–118 (1995)
3. Bergmann, R., Wilke, W.: On the role of abstraction in case-based reasoning. In: Advances in Case-Based Reasoning. LNCS (LNAI), pp. 28–43. Springer, Heidelberg (1996)
4. Bonzano, A., Cunningham, P., Smyth, B., et al.: Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 291–302. Springer, Heidelberg (1997)
5. Branting, K., Aha, D.W.: Stratified case-based reasoning: Reusing hierarchical problem solving episodes. In: IJCAI, pp. 384–390 (1995)
6. Cox, M.T., Munoz-Avila, H., Bergmann, R.: Case-based planning. The Knowledge Engineering Review 20(3), 283–287 (2006)
7. de Freitas, S.: Emerging technologies for learning. Tech. rep., Becta (2008)
8. Gómez-Martín, M., Gómez-Martín, P., González-Calero, P.: Game-driven intelligent tutoring systems, pp. 108–113. Springer, Heidelberg (2004)
9. Gómez-Martın, P., Gómez-Martın, M., Dıaz-Agudo, B., González-Calero, P.: Opportunities for CBR in learning by doing. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 267–281. Springer, Heidelberg (2005)
10. Hammond, K.: Case-based planning: A framework for planning from experience. Cognitive Science 14, 385–443 (1990)
11. Harzallah, M., Berio, G., Vernadat, F.: Analysis and modeling of individual competencies: Toward better management of human resources. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 36(1), 187–207 (2006)
12. IBM, Seriosity: Virtual worlds, real leaders: Online games put the future of business leadership on display. Tech. rep., IBM and Seriosity (2007)
13. Jonassen, D.H., Hernandez-Serrano, J.: Case-based reasoning and instructional design: Using stories to support problem solving. Educational Technology Research and Development 50(2), 65–77 (2002)
14. Kebritchi, M., Hirumi, A.: Examining the pedagogical foundations of modern educational computer games. Comput. Educ. 51(4), 1729–1743 (2008)
15. Kolodner, J.L., Hmelo, C.E., Narayanan, N.H.: Problem-based learning meets case-based reasoning. In: Second International Conference of the Learning Sciences (1996)
16. Leake, D., Kinley, A., Wilson, D.: Learning to improve case adaptation by introspective reasoning and CBR. LNCS, p. 229. Springer, Heidelberg (1995)
17. Lee, C.H.L., Cheng, K.Y.R.: A case-based planning approach for agent-based service-oriented systems. IEEE International Conference Systems, Man and Cybernetics, pp. 625–630 (2008)
18. Leyking, K., Chikova, P., Loos, P.: Competency- and process-driven elearning - a model-based approach. In: International Conference of E-Learning (ICEL 2007), New York (2007)
19. Marton, F., Trigwell, K.: Variatio est mater studiorum. Higher Education Research and Development 19(3), 381–395 (2000)

20. Munoz-Avila, H., Cox, M.T.: Case-based plan adaptation: An analysis and review. IEEE Intelligent Systems
21. Nebolsky, C.: Corporate training in virtual worlds. Journal of Systemics, Cybernetics and Informatics 2(6), 31–36 (2004)
22. Richter, M.M., Aamodt, A.: Case-based reasoning foundations. Knowledge Eng. Review 20(3), 203–207 (2005)
23. Schank, R.C.: Goal based scenario: Case-based reasoning meets learning by doing. In: Case-Based Reasoning: Experiences, Lessons and Future Directions, pp. 295–347 (1996)
24. Smyth, B.: Case-base maintenance. In: Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (1998)
25. Smyth, B., Keane, M.T., Cunningham, P.: Hierarchical case-based reasoning integrating case-based and decompositional problem-solving techniques for plant-control software design. IEEE Transactions on Knowledge and Data Engineering 13(5) (2001)
26. Smyth, B., Keane, M.T.: Adaptation-guided retrieval: questioning the similarity assumption in reasoning. Artificial Intelligence 102, 249–293 (1998)
27. Smyth, B., McClave, P.: Similarity vs. diversity. LNCS, pp. 347–361. Springer, Heidelberg (2001)
28. Soh, L.K., Blank, T.: Integrating case-based reasoning and meta-learning for a self-improving intelligent tutoring system. International Journal of Artificial Intelligence in Education 18, 27–58 (2008)
29. Spalazzi, L.: A survey on case-based planning. Artificial Intelligence Review 16, 3–36 (2001)
30. Spiro, R.J., Feltovich, R.P., Jacobson, M.J., Coulson, R.L.: Cognitive flexibility, constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. In: Constructivism and the Technology of Instruction: A Conversation, pp. 57–76 (1992)
31. Tobias, L.A.D.: Identifying employee competencies in dynamic work domains: Methodological considerations and a case study. Journal of Universal Computer Science 9(12), 1500–1518 (2003)
32. Van Eck, R.: Digital game-based learning: It's not just the digital natives who are restless. EDUCAUSE Review (2006)

A Case Based Reasoning Approach for the Monitoring of Business Workflows

Stelios Kapetanakis, Miltos Petridis, Brian Knight, Jixin Ma, and Liz Bacon

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

Abstract. This paper presents an approach for the intelligent diagnosis and monitoring of business workflows based on operation data in the form of temporal log data. The representation of workflow related case knowledge in this research using graphs is explained. The workflow process is orchestrated by a software system using BPEL technologies within a service-oriented architecture. Workflow cases are represented in terms of events and their corresponding temporal relationships. The matching and CBR retrieval mechanisms used in this research are explained and the architecture of an integrated intelligent monitoring system is shown. The paper contains an evaluation of the approach based on experiments on real data from a university quality assurance exam moderation system. The experiments and the evaluation of the approach is presented and is shown that a graph matching based similarity measure is capable to diagnose problems within business workflows. Finally, further work on the system and the extension to a full intelligent monitoring and process optimisation system is presented.

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

1 Introduction

Most activities within organisations require the coordination of tasks and the efficient and timely sharing of resources and artefacts between teams of people and the systems that interact and support them in order to achieve the particular aims of these activities. In order to standardise and ensure quality and consistency of such activities and their goals, businesses define their processes in terms of business workflows. These are formalised, followed and audited as part of a quality control management process. Modern business processes are increasingly being monitored and managed using computer systems. In order for this to happen effectively, business processes are usually formally defined and structured in terms of events, tasks and transitions involved in their operation. These are usually captured, audited and reported to the various business process stakeholders and managers.

Business processes are typically defined and represented in terms of a series of workflows and temporal relationships and constraints between them. Business processes can be defined using UML diagrams such as activity diagrams and represented

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 390–405, 2010.

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

Over the last few years, a number of standards have emerged and are widely accepted and supported by mainly Service Oriented Architecture (SOA) based enterprise technologies and systems. The OASIS Business Process Execution Language (BPEL), short for Web Services BPEL (WS-BPEL) is a key orchestration technology [2]. The Workflow Management Coalition (WfMC) backed XML Process Definition Language (XPDL) is a format standardised to interchange Business Process definitions between different workflow products and systems.

Modern enterprise systems are able to separate the definition of workflow based business processes from the software implementing the operation of these workflows, offering much more flexibility and agility than was possible in older systems. This allows the building of enterprise computer systems to monitor and control business processes and workflows within an organisation. Additionally, this allows for the agile changing and adaptation of workflows to adapt to the changing business needs of an organisation.

Case Based Reasoning (CBR) has been proposed as a natural approach to the recall, reuse and adaptation of workflows and knowledge associated with their structure. Minor et al [4] proposed a CBR approach to the reuse and adaptation of agile workflows based on a graph representation of workflows and structural similarity measures. Dijkman et al[13] have investigated algorithms for defining similarities between business processes focused on tasks and control flow relationships between tasks. Van der Aalst et al [14] compare process models based on observed behaviour in the context of Petri nets. The definition of similarity measures for structured representations of cases in CBR has been proposed [5] and applied to many real life applications requiring reuse of domain knowledge associated with rich structure based cases [6],[7].

A key issue associated with the monitoring and control of workflows is that these are very often adapted and manually overridden by to deal with unanticipated problems and changes in the operating environment. This is particularly the case in the aspects of workflows that directly interact with human roles. Additionally, workflows are liable to change as the business requirements change. Finally, in many case workflows involving processes from different parts of an organisation, or between collaborating organisations can conflict. Due to all of this, the problem of monitoring and controlling business processes involves reasoning with incomplete and uncertain data. Previous work on this area has focused on the uncertainty aspect of the workflow monitoring problem and showed the feasibility of the proposed approach [12]. This paper investigates the feasibility of monitoring workflows using real complex data from a real business process domain.

The flexibility and adaptability of workflows provides challenges for the effective monitoring of a business process. Typically, workflow management systems provide outputs in terms of event logs of actions occurring during the execution of a workflow. These could refer to actions (such as a sign-off action or uploading a document),

or communications (such as a transaction initiation or an email being initiated and sent). The challenge in monitoring and controlling workflows using event information is that even where the workflow structure is well defined and understood, the trace of events/actions does not usually contain all the information required to fully understand the context behind any decisions that caused these events/actions to occur. Additionally, there are often a lot of contextual information and communications that are not captured by the system. For example, some actions can be performed manually and informal communications/meetings between workflow workers may not be captured by the system. Knowledge of the workflow structure and orchestration of workflows does not necessarily define uniquely the choreography and operation of the workflows.

The approach proposed in this paper is based on a CBR process requiring the definition of similarity measures informed from knowledge discovery of norms and known problems from past operation. The CBR approach proposed uses a simple graph based representation of cases based on events, actions, intervals and their temporal relationships. An architecture capable to deal with the definition and orchestration of business processes as well as capable of providing monitoring and control services over workflows is key to allowing for the intelligent management of business processes within an organisation.

Section 2 discusses the chosen example of an exam moderation business process application domain that is used to explain and evaluate the approach. Section 3 presents briefly the proposed workflow and event log case representation and similarity measures used. Section 4 presents the architecture of the workflow intelligent monitoring system CBR-WIMS that has been developed to evaluate this work. Section 5 presents an evaluation based on two sets of workflow monitoring experiments.

2 The Exam Moderation Business Process Workflows

In order to evaluate the approach proposed in this research, it was decided to use a typical formal business process comprising of defined workflows and involving a number of actors interacting and communicating securely with a workflow monitoring system and involving artefacts tracked and changed by the workflow. The University of Greenwich, School of Computing and Mathematical Science exam moderation system is such a business process for which anonymised operational logs were made available to this research project. The exam moderation system is an automated web enabled secure system that allows course (module) coordinators, course moderators, exam drafters (typically senior managers), admin staff and external examiners to upload, modify, approve and lock student exam papers and orchestrates and captures the associated actions and communications. The system automates the whole process and provides an audit trail of events generated by workflow stakeholders and the system. The system orchestrates a formal process made up of workflows. The process can be defined and displayed formally in terms of a BPMN diagram (Fig. 1). The system tracks most workflow actions as timed log events. Most of these actions generate targeted email communications to workflow stakeholders, some for information and others requiring specific further actions from these stakeholders.

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

2.1 Uncertainty in Workflows

The overall exam moderation workflow process is formally defined and constrained by the orchestration of the system operation. There are also some limited facilities for manual override by system administrators. However, the overall process in conjunction with the actions and communications audit trail do not uniquely explain the exact cause of individual actions and cannot predict reliably what the next event/action will be and when this is likely to occur. Most of the uncertainty stems from the problem that a significant part of the workflow occurs in isolation from the system. The system does not capture all of the contextual knowledge associated with workflows. Many of the communications between workflow stakeholders can occur outside the system e.g. direct emails, physical meetings and phone calls adding to the uncertainty associated with past or anticipated events and the clear definition of the current state.

Discussions with workflow monitoring managers showed that patterns of events indicated, but not defined uniquely the current context and state of a workflow.

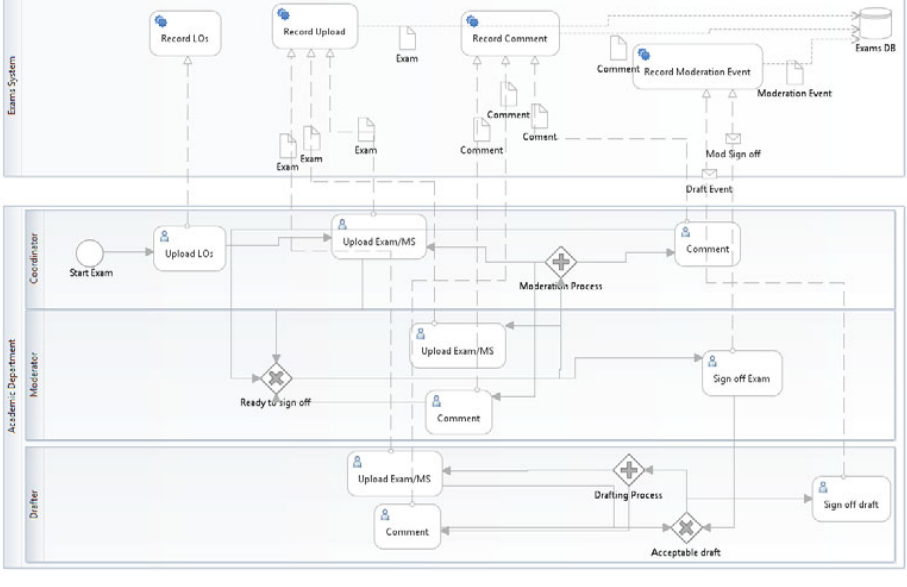


Fig. 1. BPMN representation of the exam moderation process activities and workflows (simplified)

Managers were generally able to guess from looking at the workflow events and communications audit what the context and current state of a workflow was and point to possible problems. Most problems occur due to human misunderstanding of the current state and confusion with roles and responsibilities and usually result to the stalling of a workflow. Managers will then try to restart the process by adding comments to the system, or initiate new actions and communications. However, this depends on managers realizing that such a problem has occurred in the first place.

A typical problem series of event could be one where a stakeholder has missed reading an email requiring an action. In that case, the workflow would stall until a manager or another stakeholder spots the problem and produces a manual action (such as sending an email) to get the workflow process moving again. For example, a course coordinator upload notification may have been missed by a moderator who would then not read the new version and either approve or try to amend by a new upload as s/he needs to do. In that case, the coordinator may take no further action and other stakeholders will not act expecting an action from the moderator to occur.

A key problem with uncertainty about the current status of a workflow is that due to the expected normal delay between workflow events/actions, it may not be clear at any given point in time whether the workflow has stalled or whether the moderator is just slow at responding to the original action of the coordinator upload. Based on the knowledge encapsulated in the system, this can only be resolved in a stochastic way based on retrieved knowledge from a similar series of events in past workflows for that moderator in addition to norms.

Discussions with system managers indicated that some of the uncertainty associated with expected response delays can be reduced by using past experience about response profiles and norms for individual stakeholders. Data mining or statistical analysis of the information obtained from past workflows for individual system users, in a particular workflow role, can provide the most likely response and likely response time for the user in a new workflow context. This can then be used to provide a more reliable similarity measure for the effective comparison between a new, unknown workflow state and past cases and retrieve useful associated knowledge as part of a case-based reasoning retrieval process.

The exams moderation business process is typical of many business processes in various business sectors involving processes that are supported and captured by a workflow control system typically integrated within an organisation's corporate Intranet system.

2.2 The CBR Workflow Monitoring System

The aim of the CBR Workflow Intelligent Monitoring System (CBR-WIMS) is to provide an automatic monitoring system that will notify managers and stakeholders of potential problems with the workflow and provide advice on actions that can remedy a perceived problem.

The monitoring system is designed to work based on experience of past event/action temporal sequences and the associated contextual knowledge and classification in a Case-Based Reasoning system. Similarity measures allow the retrieval of close matches and their associated workflow knowledge. This allows the classification of a sequence as a particular type of problem that needs to be reported to the

monitoring system. Additionally, it is intended that any associated knowledge or plan of action can be retrieved, adapted and reused in terms of a recommendation for remedial action on the workflow.

The CBR monitoring system uses similarity measures based on a simple linear graph representation of temporal events in a workflow normalized by experience from past behaviour on individual user workflow participation patterns.

3 Workflow and Event Log Representation and Similarity Measures

In CBR-WIMS workflows are defined using UML activity diagrams and mapped through Business Process Management Notation (BPMN)[1] into Web-Services Business Process Execution Language (WS-BPEL) [2] and stored within the system. The storage of workflows is temporal as a number of workflow versions can be stored to allow for modifications of the workflow following business process changes and their application to different contexts of use for a particular process. For example, variants of the exam process workflows can be defined to allow for specific types of exams, such as ones that require external validation or collaboration for courses delivered in partnership with other institutions. Similarity measures between workflow representations can be defined using a graph representation of workflow processes using an exhaustive graph similarity search algorithm based on the Maximum Common Subgraph [7]. This allows the reuse of knowledge about workflows between different workflow processes and variants however, this is beyond the scope of the work presented in this paper.

The workflows stored in WS-BPEL are used by CBR-WIMS to automatically orchestrate the execution of workflows in the system.

The representation of events in the workflow event log uses a general time theory, based on intervals [8]. In the theory used here, the temporal relationships have been reduced from the ones proposed by Allen [9] to just one, the "meets" relationship.

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

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

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

This approach has been shown to be suitable for defining temporal similarity measures in the context of a CBR system based on the graph representation of events and intervals and their temporal relationships and similarity measures based on graph matching techniques such as the Maximum Common Subgraph (MCSG)[11][7]. Additionally, such a graph can be checked for consistency of temporal references using linear programming techniques [11].

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

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

M = {Meets(t_1, t_2), Meets(t_1, t_3), Meets(t_2, t_5),
Meets(t_2, t_6), Meets(t_3, t_4), Meets(t_4, t_7),
Meets(t_5, t_8), Meets(t_6, t_7), Meets(t_7, t_8);

D = {Dur(t_2) = 1, Dur(t_4) = 0.5, Dur(t_6) = 0, Dur(t_8) = 0.3}
The graphical representation of temporal reference (T, M, D) is shown in Fig. 2:

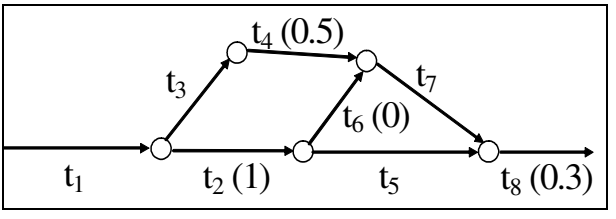


Fig. 2. Graph representation of temporal relationships

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

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

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

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

4 The Architecture of the Workflow Intelligent Monitoring System

CBR-WIMS is an Intelligent Workflow Monitoring System incorporating a CBR component. The role of the system is to assist the transparent management of workflows in a business process and to orchestrate, choreograph, operate, monitor and

adapt the workflows to meet changing business processes and unanticipated operational problems and inconsistencies. Fig. 3 below shows the overall architecture and components of CBR-WIMS. The system allows process managers to create, modify and adapt workflows to suit the changing business needs, and/or to allow for variations related to special business requirements. Workflow descriptions are stored in a temporal repository and can be used for looking up past business processes and to provide historical context for past event logs of operations. This allows for workflow based business processes to change in a system auditable way, allowing the system to preserve the context of captured operation logs.

The main part of the system controls the operation of the workflows. It responds to actions of various actors to the system and communicates messages about the operation of the system to them. The control system has a workflow orchestrator component that looks up the current workflow definition and orchestrates responses by invoking specific Web Services. The control component also manages and updates the data stored and current state of the workflow operation and provides an event audit log of the key events and actions that occur within the operation of the workflow.

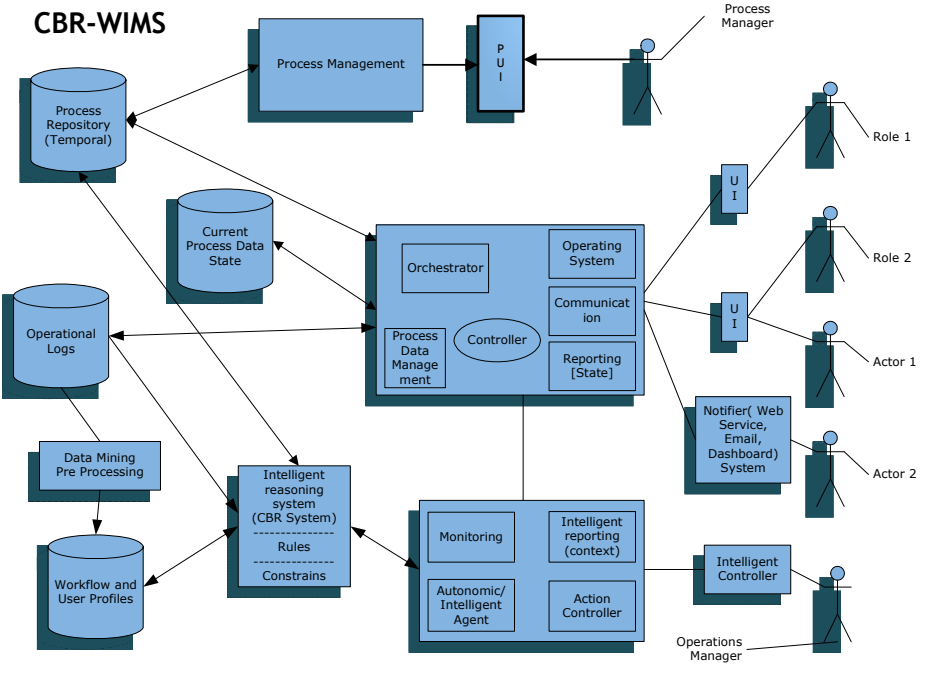


Fig. 3. The Intelligent Workflow Monitoring System Architecture

The workflow monitoring and intervention controller (controller in fig. 3) monitors, reports, and proposes possible remedial actions to the workflow operation manager. The monitoring system uses the CBR system to retrieve past useful experience about workflow problems occurred in the past by retrieving similar sequences of events/actions and context in the event log for a given workflow (or workflow part)

compared to the current state and recent sequence of events/actions in the operation of the workflow. If a fault or possible problem pattern is detected, this is reported to the workflow operations manager together with the retrieved similar cases and associated recorded experience of any known remedy/course of action.

In order to deal with the uncertain and contextual dimension of workflow similarity, the CBR system relies on knowledge discovered from past cases about workflow norms and user profiles created by statistical and data mining pre-processing. Workflow norms represent normal expected intervals between event actions for particular actors (e.g. the expected response time for a particular moderator) and for particular workflow contexts. The pre-processing component analyses operational logs and attempts to discover knowledge about norms and patterns of operation that can be used in the calculation of the similarity measures for the CBR process. This is particularly important for the monitoring process as any "interesting" or "abnormal" states need to be seen in the context of what has been normal or abnormal behaviour in past event sequence cases.

5 Workflow Monitoring Experiments and Evaluation

In order to evaluate the suitability of the approach proposed in this paper, a number of simple experiments were conducted using the CBR-WIMS system.

5.1 First Set of Experiments: Simulated Workflows

A simplified workflow process based on the exam moderation problem was constructed and a simulation was used to produce a series of workflow case studies. 320 simple event logs of workflows (exam processes) were produced to serve as cases in the case base. Each case was labelled by the simulation as either "stalled" or "not stalled" to indicate the presence or not of a problem in the workflow execution. Only exam upload actions were considered and only the last 3 such uploads in a series of workflow events were used to represent each case. A workflow event log audit trace is represented in its simplest form (no other contextual temporal relationships) as a simple graph segment:

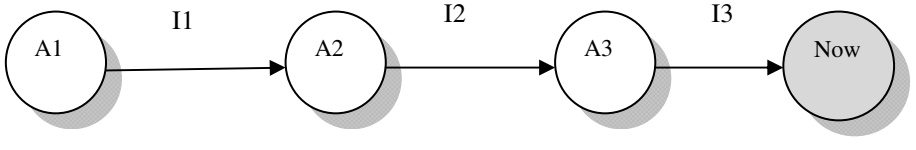


Fig. 4. A Simple workflow Event log segment

which can then be represented as:

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

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

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

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

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

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

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

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

C= (CoordUpload,John,3, ModUpload, Phil, 0,CoordUpload, John, 5) and
C'=(ModUpload,Phil,4, ModUpload, Phil, 0,CoordUpload, Mary, 3)

Assembling the MCSG:

1. σ(5, 3)=1-2/8=0.75
2. σ(CoordUpload, CoordUpload) = 1
3. σ(0, 0)=1
4. σ(ModUpload, ModUpload) = 1
5. σ(4, 3)=1-1/7=0.857
6. σ(CoordUpload, ModUpload) = 0 .. MCSG Matching stops

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

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

The 320 cases were split randomly into a case base of 300 cases and 20 test target cases.

Using the KNN algorithm for K=3, the three nearest neighbours to every target case were used to classify the target case as "stalled" or "not stalled" using simple voting. The results were compared against the known classification for the target cases. This evaluation run was repeated 10 times and the results of the classification were averaged over the 10 runs. Table 1. below shows the results of the evaluation runs:

Table 1. First Evaluation results – no normalisation for person profiles

	Average number of cases / 20	**%**
Target Cases Correctly classified	13.8	69
Missed positives	5	25
False positives	1.2	6

For the second set of experiments, the interval similarity measures were normalised to take into account the different rates of responses expected from different workflow actors.

A data analysis of the cases classified workflow actors into:

- Fast responders: 0-2 days
- Medium responders: 2-4 days
- Slow responders: over 4 days

For these cases, the interval duration I for each interval was replaced by the difference of the actual duration minus the nominal duration for the relevant type of workflow actor:

Fast responders: 1 day / Medium responders: 3 days / Slow responders: over 5 days

So assuming that if in the example above analysis of past behaviour has shown that John is a fast responder and Phil is a slow responder, the case is represented as:

C= (CoordUpload,John,2, ModUpload, Phil, 5,CoordUpload, John, 4)

This way the similarity measure is modified to provide a context based on knowledge discovered from past cases. The results of running a similar set of experiments as in the first iteration are summarised in Table 2.

Table 2. Second Evaluation results – normalised for person profiles

	Average number of cases / 20	**%**
Target Cases Correctly classified	15.3	76.5
Missed positives	3.8	19
False positives	0.9	4.5

It can be seen that the overall number of target cases correctly classified has increased, mainly by the corresponding reduction of missed positives.

5.2 Second Set of Experiments: Using Real Workflow Logs

The preliminary evaluation using simulated data was encouraging. To evaluate the approach and the CBR-WIMS system architecture further, the next step was to conduct a further evaluation using a larger dataset from actual (not simulated) workflow event audit logs. Larger segments of event log were used in the case representation involving a fuller set of possible exam moderation actions and events to predict the exact type of workflow disruption.

AUDIT TRAIL

Examination document Exam_COMPXX70_ver1_May_2009.DOC has been uploaded by Michael Peterson at 03/04/2009 12:21:43.
Email sent to Moderator: mcXX Office: cms-exams

Examination document Exam_MS_COMPXX70_ver1_May_2009.DOCX has been uploaded by Michael Peterson at 04/04/2009 00:06:03.
Email sent to Moderator: mcXX Office: cms-exams

Examination document Exam_COMPXX70_ver4_May_2009.DOCX has been uploaded by Ch&&&&g XX at 07/04/2009 13:15:22.
Email sent to Coordinator: pmXX Office: cms-exams Head of Department: pmXX

Examination comment has been added by Ch&&&&g Ma at 07/04/2009 13:18:32.
Email sent to Coordinator: pmXX Office: cms-exams Head of Department: pmXX

Moderator C&&&&g XX has signed off the Exam and Marking Scheme at 08/04/2009 00:07:17.
No Emails sent

Drafter SXtX McKXXXie has removed the Moderator Sign Off for the Exam and Marking Scheme at 08/04/2009 11:40:49.
Email sent to Moderator: mcXX Coordinator: pmXX Office: cms-exams Head of Department: pmXX

Fig. 5. An example excerpt of an event log showing for the exam moderation process (partially obfuscated)

Figure 5 above shows an excerpt of an event log for an exam paper. Each event corresponds to a workflow action. The actor initiating the action is identified, as well as communications sent to other stakeholders of the exam moderation process.

The evaluation involved looking at all events for each exam moderation process. This was a total of 1588 events involving 116 exam moderation workflow processes from one academic session. The analysis initially looked at all types of events (Uploads, Sign-offs, Comments, Reports and management overrides) involving all actor roles (Coordinator, Moderator, Drafter and Admin). A domain expert went through every exam process log and classified each exam moderation process using three classes, A, B and C as follows:

- A: The process completed but with problems (Typically with delays, stalling at some point and/or considerable confusion or disagreement between actors)
- B: The process completed with few or no problems.
- C: The process stalled and had not complete correctly at the point of observation

The evaluation was conducted by comparing the classification obtained by the kNN algorithm (for k=1,3 and 5) for each process to the one assigned by the expert. In addition to the Maximum Common Sub Graph similarity measure discussed earlier, a component event count similarity measure was also used. This measures the distance between two processes as the normalised sum of differences between the count of events for each type of events. This similarity measure does not take into account the comparative length of time intervals between events, but concentrates on the overall number of events of each type involved in the process.

A number of variants of the experiments were conducted to gain a better understanding of the effectiveness of the approach. In subsequent experiments, admin and reporting events were discounted as they do not directly affect the process. The next set of experiments involved removing also any comment posting events and finally Drafter events were removed, thus concentrating on the initial production stage of the exams process.

Finally, an algorithm was applied that smoothed event logs by consolidating similar events that happened in quick succession. For example it is very common that actors end up repeating an action due to user error (such as uploading the wrong file) or due to a system error. The following figures 6-10 summarise the results of the experiments and show the percentage of correct classifications for each class of exam workflows (A,B and C).

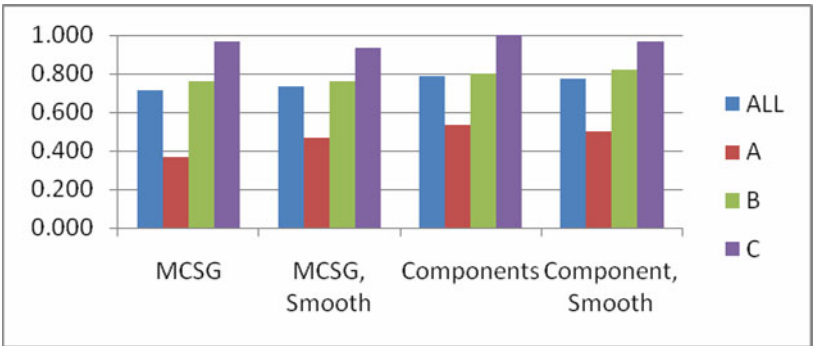


Fig. 6. Results for 1NN and all types of events

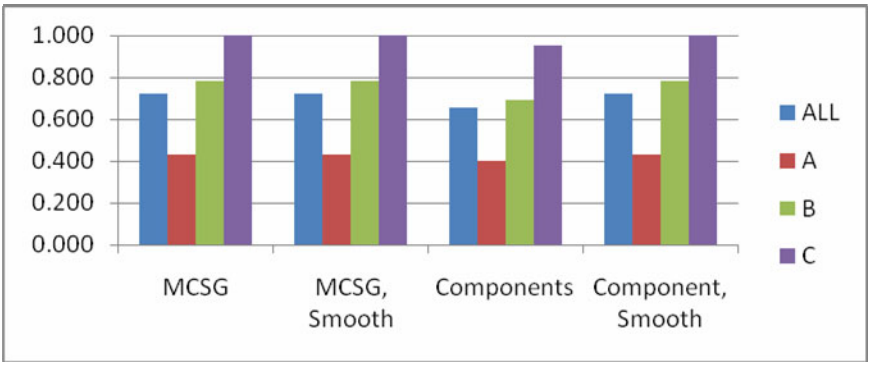


Fig. 7. Results for 1NN and all types of events except for reports and admin actions

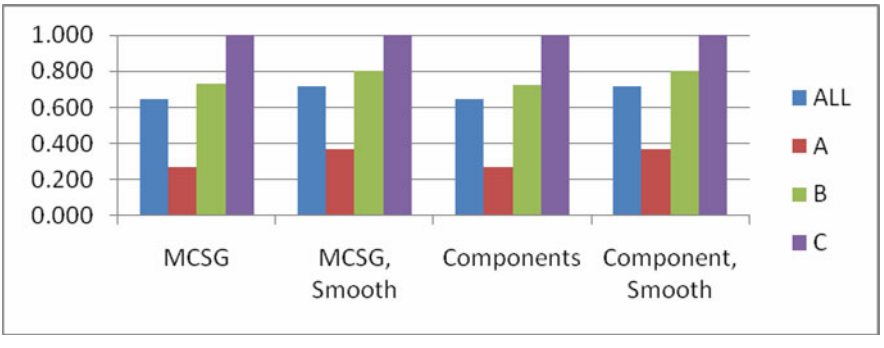


Fig. 8. Results for 1NN and no reporting/admin or comment events

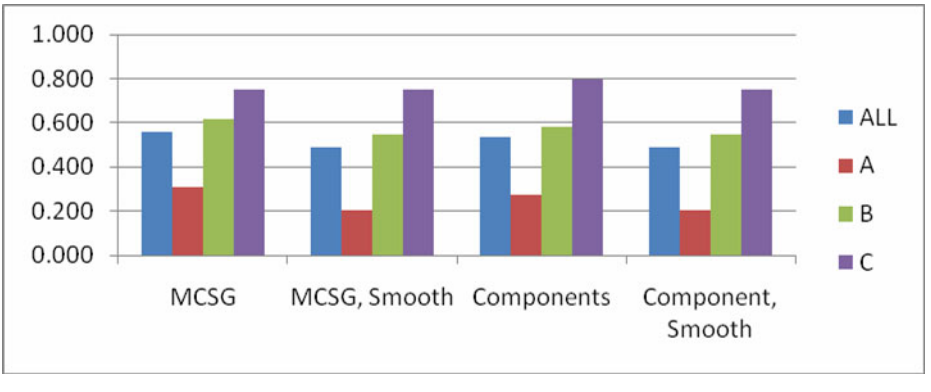


Fig. 9. Results for 1NN only for the production stage (no drafter involvement)

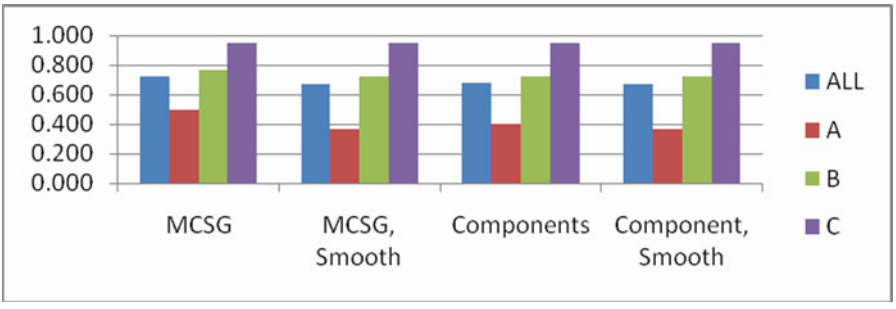


Fig. 10. Results for 3NN no admin and reporting events

The evaluation shows that in most cases the system predicts well the cases where there are substantial problems currently with the exam moderation process (C). It also generally predicts well the cases where the process has gone smoothly (B). The cases where some problems occurred at some past point of the process (A) are less well predicted, although it is evident that "filtering" of similar consecutive events (smoothing) and removing irrelevant admin and reporting events improves the performance of the classification process. This is especially the case when using the MCSG similarity measure. The approach is less able to spot problems during the initial phase of the process (figure 10), but this is probably because fewer problems occur then. The choice of k in the kNN algorithm seems to make little difference between k=1 and 3, but obviously due to the limited coverage of the case base considered there k=5 was not as efficient.

This evaluation did not take into account past norms of human actors and exam processes and only observed the processes at one given point in time. The case base used was also relatively small when compared to the full case base available. In this context, the results obtained are judged as very encouraging.

6 Conclusions

This paper discussed an approach for intelligent diagnosis and monitoring of workflows based on incomplete operation data in the form of temporal log data. This was based on a graph representation of workflows using temporal relationships. The workflow process is orchestrated by a software system using BPEL technologies in service oriented architecture in the CBR-WIMS system. The CBR-WIMS architecture was in the context of a system that monitors exam moderation workflows. The matching and similarity measures presented here showed a preliminary evaluation using real data from a typical workflow process. The evaluation showed that the approach is capable of classifying problems correctly in a workflow process. In particular it was shown that an analysis of past workflow event logs can provide norms and context that can reduce the uncertainty in similarity based matching and improve the efficiency of the reasoning process. The evaluation with real data provided from the exam moderation system event log has shown that the approach can provide some useful classification

of problematic workflows in real business workflow monitoring systems. The evaluation has shown that despite the theoretical computational complexity involved in the graph similarity calculations, the structural characteristics and typical sizes of many real workflows mean that effective monitoring of workflows using these techniques is feasible.

Further work will concentrate on further evaluation of the approach based on more complex case representation and similarity matching. In particular, work is currently under way on the MCSG similarity matching algorithm to make it operate on a moving window of observation of the workflows rather than the current static one. Work on further building and automating the CBR-WIMS system will allow the extension to provide intelligent advice to operators in addition to the existing simple monitoring action. Other work direction will cover the challenge of explaining the reasoning results and advice to the workflow operation managers, the combination of constraints and temporal consistency checking and the combination of workflow event log temporal knowledge with other uncertain temporal knowledge available about a workflow. Adaptation of solutions to retrieved similar workflows using local optimisation criteria and dealing with more complex control structures such as loops and XORs are also areas of further work. Finally, the reuse of knowledge across different workflows, concentrating on changed workflows and variants will be investigated.

References

1. Business Process Management Initiative (BPMI): BPMN 1.1: OMG Specification (February 2008), http://www.bpmn.org/ (accessed April 2009)
2. OASIS: BPEL, The Web Services Business Process Execution Language Version 2.0 (May 2006), http://www.oasis-open.org/apps/org/workgroup/wsbpel
3. Workflow Management Coalition (WfMC): XPDL 2.1 Complete Specification (Updated October 10, 2008), http://www.wfmc.org/xpdl.html (accessed April 2009)
4. Minor, M., Tartakovski, A., Bergmann, R.: Representation and Structure-Based Similarity Assessment for Agile Workflows. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 224–238. Springer, Heidelberg (2007)
5. Bunke, H., Messmer, B.T.: Similarity Measures for Structured Representations. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) EWCBR 1993. LNCS, vol. 837, pp. 106–118. Springer, Heidelberg (1994)
6. Mileman, T., Knight, B., Petridis, M., Cowell, D., Ewer, J.: Case-Based Retrieval of 3-D shapes for the design of metal castings. Journal of Intelligent Manufacturing 13(1), 39–45 (2002)
7. Wolf, M., Petridis, M.: Measuring Similarity of Software Designs using Graph Matching for CBR. In: Workshop Proceedings of AISEW 2008 at ECAI 2008, Patras, Greece (2008)
8. Ma, J., Knight, B.: A General Temporal Theory. The Computer Journal 37(2), 114–123 (1994)
9. Allen, J., Hayes, P.: Moments and Points in an Interval-based Temporal-based Logic. Computational Intelligence 5, 225–238 (1989)
10. Ma, J., Knight, B.: A Framework for Historical Case-Based Reasoning. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 246–260. Springer, Heidelberg (2003)

11. Ma, J., Knight, B., Petridis, M.: Deriving Explanations from Partial Temporal Information. In: Workshop Proceedings of ExACT 2008 at ECAI 2008, Patras, Greece (2008)
12. Kapetanakis, S., Petridis, M., Ma, J., Bacon, L.: Workflow Monitoring and Diagnosis Using Case Based Reasoning on Incomplete Temporal Log Data. In: Workshop Proceedings of the 8th International Conference on Case Based Reasoning, Seattle, USA (2009)
13. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) Business Process Management. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
14. van der Aalst, W., Alves de Medeiros, A.K., Weijters, A.: Process Equivalence: Comparing two Process Models Based on Observed Behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)

A Case-Based Reasoning Approach to Automating the Construction of Multiple Choice Questions

David McSherry

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

Abstract. Automating the construction of multiple-choice questions (MCQs) is a challenge that has attracted the interest of artificial intelligence researchers for many years. We present a case-based reasoning (CBR) approach to this problem in which MCQs are automatically generated from cases describing events or experiences of interest (e.g., historical events, movie releases, sports events) in a given domain. Measures of interestingness and similarity are used in our approach to guide the retrieval of cases and case features from which questions, distractors, and hints for the user are generated in natural language. We also highlight a potential problem that may occur when similarity is used to select distractors for the correct answer in certain types of MCQ. Finally, we demonstrate and evaluate our approach in an intelligent system for automating the design of MCQ quizzes called AutoMCQ.

Keywords: case-based reasoning, retrieval, similarity, multiple-choice questions, natural language generation.

1 Introduction

Multiple-choice questions (MCQs) are widely used for assessment in education and training [1-4], and are also used increasingly in on-line quizzes to enable users to test and/or extend their knowledge about a topic of interest (e.g., sports results, movies). However, the difficulty of authoring good MCQs is well recognized. Common pitfalls include badly worded questions and distractors for the correct answer that are easy for students/users to eliminate. In this paper, we present an approach to automating the generation of MCQs that combines case-based reasoning (CBR) with template-based natural language generation (NLG). We also demonstrate the approach in an intelligent system for automating the design of MCQ quizzes called AutoMCQ.

Measures of interestingness and similarity are used in our approach to guide the retrieval of cases and case features from which questions, distractors, and hints for the user are generated in natural language. The example domain that we use to illustrate the approach is international football (also known as soccer), not least because the core events of interest (i.e., matches) can easily be represented as cases that share common features (e.g., winning team, losing team, score). However, the proposed approach can be used to automate the generation of MCQs (and/or MCQ quizzes)

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 406–420, 2010.

related to any area of knowledge in which events or experiences of interest (e.g., historical events, movie releases) can be represented as cases.

In Section 2, we describe how CBR is combined with template-based NLG in our approach to automating the generation of MCQs. In Section 3, we present the measures of interestingness and similarity that guide the retrieval of cases and case features used to construct MCQs (and MCQ quizzes) in AutoMCQ. We also identify a potential problem that may occur when similarity is used to select distractors for the correct answer in certain types of MCQ. Our empirical results are presented in Section 4 and related work is discussed in Section 5. Finally, our conclusions are presented in Section 6.

2 Generating MCQs from Cases

In this section, we present the basic ideas in our CBR approach to automating the generation of MCQs from cases. We begin by describing the example case base in the domain of international football that we use to illustrate the approach. We then describe our approach to generating questions and hints in natural language from the descriptions of retrieved cases. We also use examples to illustrate how MCQs are created and presented to the user in AutoMCQ.

Example Case Base. Fig. 1 shows one of the 45 cases in our example case base, which covers the results of all matches in the knockout stages of the 1998, 2002, and 2006 FIFA World Cup championships. (However, the football domain is used only as an example and the author has no affiliation with FIFA or any other football organization.) The match features stored in each case are the winner, loser, score, round, and year. As we shall see, even this small set of features is enough to enable a variety of different MCQs to be generated from each case. The case description could be extended to include additional features such as the championship (e.g., Africa Cup of Nations, Copa América, FIFA World Cup, UEFA European Football Championship) in which the match was played or names of the goal scorers. Another feature of potential interest in the example domain is whether or not a match result was decided by a penalty shootout. However, there is no need to include this as an explicit case feature as it can be inferred from the match score (i.e., there must have been a penalty shootout if both teams scored the same number of goals).

MCQ Structure. The MCQs automatically generated from cases in our approach consist of 4 components: (1) question stem, (2) correct answer, (3) distractors (i.e., incorrect answers), and (4) a hint to be given if requested by the user.

Fig. 1. An example case in AutoMCQ

Selecting Plausible Distractors. A recognized feature of MCQs is that students/users can often identify the correct answer by a process of elimination. For this reason, an important challenge when constructing MCQs is how to choose distractors for the correct answer that cannot easily be eliminated. As discussed in Section 3, similarity-based retrieval plays an important role in our approach to selecting plausible distractors in automatically generated MCQs. However, we also identify a potential problem that may occur when similarity is used to select distractors for the correct answer in certain types of MCQ.

Providing Hints. Although hints are not typically provided in MCQs used for formal assessment, providing hints when requested may help to make a quiz based on MCQs more interesting and enjoyable for users. Ideally, the hint provided should help the user to identify the correct answer without giving the game away completely. One approach is to provide a clue that enables the user to eliminate one of the options from which she is asked to choose the correct answer. For example, if the question is *In which knockout round of the 1998 World Cup did France beat Croatia?*, then telling the user that France beat Brazil in the final enables the user to eliminate the final as the correct answer. Another approach in questions such as *Who beat England in the quarter-finals of the 2002 World Cup?* is to tell the user something about the team she is asked to identify (e.g., *The winners went on to win the final*). As we show in this paper, CBR is well suited to the task of automatically generating such hints when combined with template-based NLG.

Example MCQ. Fig. 2 shows an example MCQ in AutoMCQ and the hint provided if requested by the user. The correct answer (Korea Republic) and distractors selected by the system (Spain and Turkey) are presented to the user in random order. Also available to the user as a default answer is *No idea*. There is no penalty for an incorrect answer as in the negative scoring sometimes used to discourage guessing in MCQs used for assessment in education and training [3]. Conversely, there is no advantage in accepting the default option *No idea*, except perhaps for a user who simply wants to see the correct answer and move on to the next question. Immediate feedback for the user is provided after each question (i.e., correct answer, score so far). Typically, AutoMCQ presents a sequence of MCQs (or quiz) to the user that covers all cases in the case base, although the user can choose to exit from the quiz at any stage. Alternatively, the user can limit the scope of an AutoMCQ quiz to matches in a specific championship year (e.g., 2006) and/or matches involving a specific team (e.g., Brazil).

Interestingness and Diversity of MCQs. The interestingness and diversity of questions presented to the user are important considerations when designing an MCQ quiz. In AutoMCQ, the order in which questions are presented to the user is based on a measure of interestingness described in Section 3. As discussed below, the need for diversity when presenting a series of MCQs to the user in the form of a quiz is addressed in our approach by using a variety of different question templates.

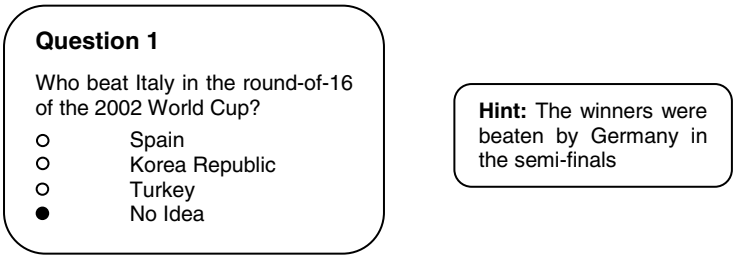


Fig. 2. An example question in AutoMCQ and the hint provided if requested by the user

Question Templates. AutoMCQ uses a variety of question templates to generate MCQs from cases in the example case base. Any of the question templates in Table 1 can be instantiated by the features of a retrieved match case to provide a question stem for a unique MCQ. There are two versions of the template that asks the user to identify the winning team in a retrieved match case. The shorter version is used if the match score was decisive, while the longer version is used if the match result was decided by a penalty shootout. There are also two versions of the *Match Score* template, one for matches in which the final score was decisive, and another for matches in which the two teams scored the same number of goals.

Hint Templates. For each question template, Table 1 shows the hint templates that AutoMCQ uses to generate a hint for the MCQ it is constructing from a retrieved case. One reason for having two or more hint templates for each type of question is to ensure that at least one hint template is applicable in the context of the retrieved case. Using a variety of different hint templates also helps to increase the diversity of MCQ dialogues. The hint templates for each type of question are shown in the order they are considered by AutoMCQ. For example, there are 5 hint templates for MCQs that ask the user to identify the losing team in a retrieved match case. The first hint template is not applicable if the retrieved case is a round-of-16 match, as this means that the losing team cannot have beaten another team in the knockout stage of the same championship. The second hint template can be used only if the losing team won the championship in another year. The third hint template is used only if the losing team progressed beyond the round-of-16 in another championship year. The fourth hint template does not apply if the losing team failed to reach the knockout stage in any of the other championships covered by the case base. The fifth hint template (which simply tells the user the match score) is perhaps the least helpful and is used only as a last resort.

Constructing an MCQ. Fig. 3 shows how a typical MCQ is constructed in AutoMCQ. The question and hint templates from Table 1 that we use to illustrate the approach are *Who beat* T_1 *in the* R_1 *of the* Y_1 *World Cup?* and *The winners were beaten by* T_2 *in the* R_2. The first step is to retrieve an interesting match case from the case base. As described in Section 3, our measure of match interestingness takes account of factors such as the knockout round in which the match was played and how surprising the result is given the track record of each team in the championships covered by the case base. In Fig. 3, the match retrieved in Step 1 is one in which Croatia

beat Germany in the quarter-finals of the 1998 World Cup. In Step 2, the features of the retrieved case are used to instantiate the selected question template (i.e., *Who beat Germany in the quarter-finals of the 1998 World Cup?*). The retrieved case also provides the correct answer (i.e., Croatia) for the MCQ in Step 3. In Step 4, the two teams that are most similar to the winning team (Croatia) among all teams that reached the knockout stage in 1998 are retrieved from the case base to be used as distractors in the MCQ. The similarity measure used for this purpose is described in Section 3. A hint to be shown if requested by the user is constructed in Steps 5 and 6. In Step 5, AutoMCQ attempts to retrieve a second match in which the winners of the first match (Croatia) were knocked out in the same championship. In this example, it succeeds, and the retrieved match is one in which Croatia was beaten by France in the semi-finals of the 1998 championship. In Step 6, the second retrieved case is used to instantiate the selected hint template (i.e., *The winners were beaten by France in the semi-finals*).

Table 1. Templates used by AutoMCQ to generate questions and hints in natural language from cases in the example case base

Question Template	Hint Templates
Winning Team: (1) Who beat T_1 in the R_1 of the Y_1 World Cup? or (2) Who beat T_1 in a penalty shootout in the R_1 of the Y_1 World Cup?	• The winners were beaten by T_2 in the R_2 • The winners went on to win the final
Losing Team: Who did T_1 beat in the R_1 of the Y_1 World Cup?	• The losers beat T_2 in the R_2 • The losers won the final in Y_2 • The losers reached the R_3 in Y_3 • The losers were beaten by T_3 in the round-of-16 in Y_4 • The final score in the match was G_1 - G_2
Championship Year: In what year did T_1 beat T_2 in the R_1 of the World Cup?	• T_1 (or T_2) did not reach the R_1 in Y_1 • T_1 was beaten by T_3 in the R_2 • T_1 went on to reach the final
Knockout Round: In which knockout round of the Y_1 World Cup did T_1 beat T_2?	• T_1 was beaten by T_3 in the R_1 • T_1 beat T_4 in the R_2
Match Score: (1) By what score did T_1 beat T_2 in the R_1 of the Y_1 World Cup? or (2) What was the final score when T_1 beat T_2 in the R_1 of the Y_1 World Cup?	• T_3 beat T_4 by the same score in the R_2 • The same final score occurred in the match between T_5 and T_6 in the R_3 • There was no other match in the knockout stage with the same final score

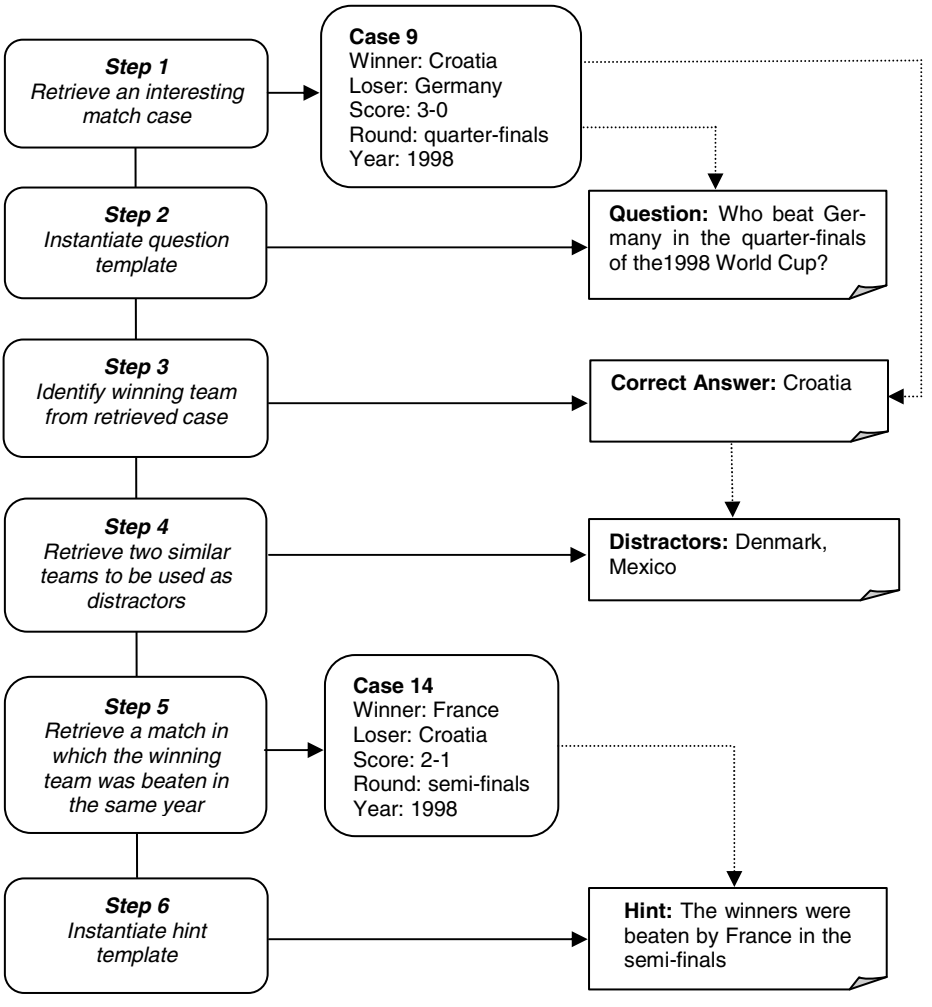


Fig. 3. Constructing an MCQ in AutoMCQ

The process that AutoMCQ uses to construct an MCQ is considerably simplified in Fig. 3. As shown in Table 1, there are several other question and hint templates from which AutoMCQ can choose when creating an MCQ. The hint template used in the example (i.e., *The winners were beaten by* T_2 *in the* R_2) cannot be used if AutoMCQ fails to retrieve a match in which T_1, the winning team from the first retrieved case, was knocked out in the same championship. In this situation, the hint created by AutoMCQ would be *The winners went on to win the final*. As mentioned earlier, the user can limit the scope of an AutoMCQ quiz to matches in a specific championship year and/or matches involving a specific team. In this case, the retrieval/ranking of interesting matches in Step 1 must also take account of these criteria.

Selecting Question Templates. When presenting an MCQ quiz, AutoMCQ cycles through the available question templates in the order shown in Table 1, keeping track of the number of times it has used each template. In a perfectly balanced quiz consisting of 15 MCQs, each of the 5 question templates would be used 3 times. However, AutoMCQ may choose not to apply some of the question templates to a retrieved match case if the user has specified a team about which she wishes to answer questions. For example, the *Winning Team* question *Who beat France in the final of the 2006 World Cup?* is an obvious giveaway if the user has specified that she would like to be asked questions only about matches involving Italy. In this situation, AutoMCQ would instead use the *Losing Team* template to generate the question *Who did Italy beat in a penalty shootout in the final of the 2006 World Cup?* For similar reasons, *Championship Year* questions are never asked if the user has specified a championship year about which she wishes to answer questions.

3 Measures of Similarity and Interestingness

In Section 3.1, we describe a simple measure of team strength that we use later in the discussion to define measures of team similarity and match interestingness. In Section 3.2, we describe the similarity measures used to guide the selection of distractors for winning teams, losing teams, and (decisive) match scores in the MCQs constructed by AutoMCQ. In Section 3.3, we identify a potential problem that may occur when similarity is used to select distractors for the correct answer in certain types of MCQ. We also describe our approach to distractor selection in MCQs that ask the user to identify the championship year or knockout round in which a retrieved match was played, or to identify an indecisive match score. In Section 3.4, we describe the measure of interestingness used to guide case retrieval in our CBR approach to automating the construction of MCQ quizzes in the example domain.

3.1 Measure of Team Strength

We measure the "strength" of a given team by the number of appearances it has made over all matches in the case base. Thus for each team T in the case base, we define:

$$strength(T) = |\{C : T = winner(C) \text{ or } T = loser(C)\}| \quad (1)$$

where for each case C, $winner(C)$ and $loser$(C) are the winning and losing teams in the match described by C. According to this measure, the 5 strongest teams based on their performances in the 1998, 2002, and 2006 World Cup championships are Brazil (10), Germany (9), France (8), Italy (7), and England (5). Note that the maximum number of appearances any team could have made in the knockout stages of the three championships is $3 \times 4 = 12$. Of course, there are other possible ways in which team strengths might be assessed from the available cases. For example, it might be considered that France and Italy, having won the championship in 1998 and 2006, should be ranked higher than Germany. On the other hand, Germany is the only team apart from Brazil to have reached the quarter-finals in all three World Cup championships.

3.2 Similarity Measures for Distractor Selection

We now describe the similarity measures used in AutoMCQ to select distractors for the correct answer in MCQs that ask the user to identify the winning team, losing team, or match score (if decisive) in a retrieved match case.

Team Similarity. Our measure of the similarity between two teams is based on their strengths as defined in (1). For any teams T_1 and T_2 in the example case base, we define:

$$similarity(T_1, T_2) = 1 - \frac{|strength(T_1) - strength(T_2)|}{S_{max} - S_{min}} \quad (2)$$

where S_{min} and S_{max} are the minimum and maximum strengths (1 and 10) over all teams in the case base. According to this measure,

$$similarity(\text{Brazil, England}) = 1 - \frac{|10-5|}{9} = 0.44 \quad (3)$$

while the two teams that are most similar to Brazil are Germany (0.89) and France (0.78).

Winning Teams. For MCQs that ask the user to identify a winning team (e.g., *Who beat Spain in a penalty shootout in the quarter-finals of the 2002 World Cup?*) the distractors chosen by AutoMCQ are the two teams that competed in the knockout stage of the same championship and are most similar in strength to the team that the user is asked to identify. In the example question about Spain's quarter-finals match in the 2002 World Cup, the correct answer is Korea Republic, and the most similar teams that reached the knockout stage in 2002 are Denmark, Mexico, and Turkey. (Note that ties between equally similar teams are broken randomly in the experiments reported in Section 4.) The reason for selecting only teams that reached the knockout stage of a given championship as distractors for a winning team is to avoid making it too easy for the user to eliminate distractors. For example, a user who knows that Croatia did not reach the knockout stage of the World Cup in 2002 can easily eliminate Croatia as a candidate for having beaten Spain in the 2002 quarter-finals.

Losing Teams. For MCQs that ask the user to identify a losing team, AutoMCQ selects distractors for the losing team in much the same way as distractors for a winning team. More specifically, the teams selected as distractors for a losing team are the two teams that competed in the knockout stage of the same championship and are most similar in strength to the losing team.

Decisive Match Scores. For MCQs that ask the user to identify a decisive match score (e.g., *By what score did Italy beat Ukraine in the quarter-finals of the 2006 World Cup?*), the selection of distractors for the correct answer is also based on similarity. Given a decisive match score $S_1 = (G_1, G_2)$ (i.e., one in which $G_1 > G_2$), we use a heuristic approach to generating two similar (and also decisive) match scores, S_2 and S_3, to be used as distractors. If $G_2 > 0$, the distractors generated by AutoMCQ are

$S_2 = (G_1, G_2 - 1)$ and $S_3 = (G_1 - 1, G_2 - 1)$. If $G_2 = 0$ and $G_1 > 2$, the distractors are $S_2 = (G_1 - 1, G_2)$ and $S_3 = (G_1 - 1, G_2 + 1)$. Finally, the distractors for $S_1 = (1, 0)$ are $S_2 = (2, 0)$ and $S_3 = (2, 1)$, and the distractors for $S_1 = (2, 0)$ are $S_2 = (1, 0)$ and $S_3 = (2, 1)$. In the example question about Italy's quarter-finals match against Ukraine in 2006, the correct answer is $S_1 = (3, 0)$, so the distractors are $S_2 = (2, 0)$ and $S_3 = (2, 1)$. Note that for all decisive match scores in the example case base, the distractors constructed in this way are also represented in the case base.

3.3 Alternative Approaches to Distractor Selection

For some MCQs, there may be no need for similarity assessment in distractor selection, for example if the number of available distractors is small (e.g., championship years in the example case base) and/or it is decided to present all available distractors to the user. In other situations, it may be natural to ask the user to select from a fixed set of answers (e.g., *knockout rounds*). Another reason for considering alternative approaches to distractor selection is a potential problem that may occur when similarity is used to select distractors for the correct answer in certain types of MCQ.

For example, consider an MCQ that asks the user to identify the number of goals scored by the winning team in a retrieved match case. Table 2 shows the options from which the user will be asked to select if the two (numerically) most similar scores are used as distractors for the correct score (and assuming that each score from 0 to 5, but no higher score, occurs in the case base). A potential problem with this strategy is that the user may quickly learn to recognize that the correct answer is usually the second highest of the scores from which she is asked to select. Note that randomizing the order in which answers are presented to the user does not solve the problem.

Table 2. Example to illustrate a potential problem when similarity is used to choose distractors in some types of MCQ

Correct Answer	Options
0	0, 1, 2
1	0, **1**, 2
2	1, **2**, 3
3	2, **3**, 4
4	3, **4**, 5
5	3, 4, 5

Championship Years. The type of MCQ that we used to illustrate a potential problem when distractor selection is based on similarity is not used in AutoMCQ. However, the same problem may occur in MCQs that ask the user to identify the championship year in which a retrieved match was played. For this reason, our approach to selecting distractors for a championship year (in general) is to retrieve all years that occur in the case base, and randomly select two that are different from the correct year. As our example case base covers only three championship years, the distractors for a given year (e.g., 2006) are simply the other two years (e.g., 1998, 2002).

Indecisive Match Scores. In MCQs that ask the user to identify an indecisive match score (i.e., one in which both teams scored the same number of goals) there is no need for distractor selection based on similarity, as only a small number of (distinct) indecisive scores tend to occur in practice. (There is also a risk of the problem identified above occurring if similarity is used to select distractors for indecisive match scores.) Instead, AutoMCQ selects distractors for a correct match score (if indecisive) in much the same way as distractors for a championship year. That is, it retrieves all indecisive match scores in the case base and randomly selects two that differ from the correct match score. As the only indecisive match scores in the example case base are (0-0), (1-1), and (2-2), the distractors for one of these scores are simply the other two scores.

Knockout Rounds. AutoMCQ also makes no use of similarity when selecting distractors for MCQs that ask the user to identify the knockout round in which a retrieved match was played (e.g., *In which knockout round of the 1998 World Cup did Netherlands beat Argentina?*). Instead, the distractors for the correct knockout round (i.e., quarter-finals) are simply the other three knockout rounds (i.e., round-of-16, semi-finals, final). However, a potential drawback with *Knockout Round* questions is that users can easily "play the odds" to increase their chances of guessing the correct answer. In any championship, there is only one final, while the numbers of matches in the semi-finals, quarter-finals, and round-of-16 are 2, 4, and 8 respectively. Thus by always choosing round-of-16, a user can increase her chance of guessing the correct answer from 25% to more than 50%. One way for the system to counter this strategy would be to ask the user to identify the knockout round only when round-of-16 is not the correct answer. However, one trade-off is a reduction in the number of *Knockout Round* questions that the user can be asked by more than one half. It is also possible that a user may learn to recognize the system's strategy and adapt her strategy to one of always guessing the quarter-finals (thereby increasing her chance of guessing the correct answer even more than previously). Currently, no such counter strategy is used in AutoMCQ. Instead, priority is given to maximizing the diversity of MCQs that the system generates from the available question templates.

3.4 Interestingness Measure

In the example domain of international football, it is natural for some match results to be considered more interesting than others, depending on factors such as the stage of the championship in which the match was played (e.g., the final), the strengths of the competing teams, and whether or not the match produced a surprising result. Moreover, users may quickly lose interest in an MCQ quiz if the questions they are asked early in the quiz do not hold their attention. Some measure of case (or match) interestingness is therefore needed to guide the retrieval and ranking of cases from which the questions presented to the user by AutoMCQ are generated. For any case C in the example case base, we define:

$$interestingness(C) = importance(round(C)) + 2 \times \frac{strength(loser(C))}{strength(winner(C))} \quad (4)$$

where $round(C)$ is the knockout round in which the match represented by C took place, $importance(round(C))$ is an importance score assigned to $round(C)$, $winner(C)$

and *loser*(*C*) are the winning and losing teams respectively, and *strength* is the measure of team strength defined in Section 3.1. The importance scores that we assign to the round-of-16, quarter-finals, semi-finals, and final of a given championship are 0, 1, 2, and 3 respectively.

According to this measure, the interestingness of the 1998 quarter-finals match in which Germany beat Croatia (i.e., Case 9 in the example case base) is:

$$interestingness(\text{Case 9}) = 1 + 2 \times \frac{strength(\text{Germany})}{strength(\text{Croatia})} = 1 + 2 \times \frac{9}{3} = 7 \quad (5)$$

In the event of a tie between cases with equal interestingness scores, AutoMCQ gives priority to tied cases, if any, that describe matches in which the result was decided by a penalty shootout. If none of the tied cases describe matches that ended with penalty shootouts, then AutoMCQ gives priority to tied cases that describe matches with the highest goal differences. If there is still a tie between two or more cases when these criteria have been applied, then AutoMCQ gives priority to tied cases that describe matches in which the total numbers of goals scored are highest.

At the start of an AutoMCQ quiz, the user can specify a championship year and/or team about which she would like to answer MCQs. For example, Table 3 shows the MCQs generated by AutoMCQ from the five most interesting matches involving Italy in the World Cup championships (1998, 2002, 2006) covered by the example case base.

Table 3. MCQs generated by AutoMCQ from the five most interesting matches involving Italy in the example case base

Question	Correct Answer	Distractors	Hint
Who did Italy beat in a penalty shootout in the final of the 2006 World Cup?	France	Brazil, Germany	The losers beat Portugal in the semi-finals
Who beat Italy in the round-of-16 of the 2002 World Cup?	Korea Republic	Spain, Turkey	The winners were beaten by Germany in the semi-finals
In what year did Italy beat Germany in the semi-finals of the World Cup?	2006	1998, 2002	Italy did not reach the semi-finals in 1998
In which knockout round of the 1998 World Cup did France beat Italy?	quarter-finals	round-of-16, semi-finals, final	France beat Brazil in the final
By what score did Italy beat Ukraine in the quarter-finals of the 2006 World Cup?	3-0	2-0, 2-1	Brazil beat Ghana by the same score in the round-of-16

4 Empirical Study

In this section, we investigate the hypothesis that for some types of MCQ, distractor selection based on similarity may help to reduce the scores that users can achieve by informed guesswork in the example domain of international football. For each of the question templates used by AutoMCQ (Table 1), we also investigate the success rates achieved by simulated users adopting different strategies when answering MCQs automatically generated from the example case base.

4.1 Individual Question Templates

For MCQs that ask the user to identify the winning team / losing team / match score in a retrieved match case, Table 4 shows the success rates (i.e., percentages of correct answers) achieved by a simulated user (Informed) that uses a strategy based on informed guesswork, and another simulated user (Random) that always selects a random answer. More precisely, when asked to identify a winning team or losing team, the first simulated user chooses the team that has made most appearances in the three championships covered by the example case base. When asked to identify a match score, it selects the score that occurs most frequently in the case base.

In the three question types, the simulated users select from 3 answers (correct answer and two distractors) for each question. The strategies used by the system to select distractors for the correct answer are Most Similar and Random (except in the case of indecisive match scores, for which only two distractors are available in the example case base). The similarity measures used in the Most Similar strategy are described in Section 3.2. The results for each question type are based on 10 replications in which the MCQs presented to the simulated users are generated from all (45) cases in the example case base.

Table 4. Success rates for simulated users adopting different strategies when answering MCQs that ask the user to identify the winning team / losing team / match score in a retrieved case

User Strategy	Winning Team Distractors		Losing Team Distractors		Match Score Distractors	
	Most Similar	Random	Most Similar	Random	Most Similar (if decisive)	Random (if decisive)
Informed	34.0%	76.7%	36.2%	44.7%	41.6%	58.2%
Random	32.2%	30.9%	35.3%	32.9%	35.3%	33.6%

The most striking features of the results are the high success rates in *Winning Team* and *Match Score* questions (77% and 58%) achieved by the simulated user that uses informed guesswork when distractors for the correct answer are chosen randomly by the system. However, the results also show the effectiveness of distractor selection based on similarity in reducing the success rates achieved by this simulated user, for example from 77% to 34% in *Winning Team* questions.

For MCQs that ask the user to identify the knockout round / championship year in which a retrieved match was played, Fig. 4 shows the success rates achieved by a simulated user (Random) that always selects a random answer. The results for each question type are based on 10 replications in which the MCQs presented to the simulated user are generated from all (45) cases in the example case base. The success rate for another simulated user (Informed) that always selects the most likely answer in *Knockout Round* questions (i.e., round-of-16) is also shown.

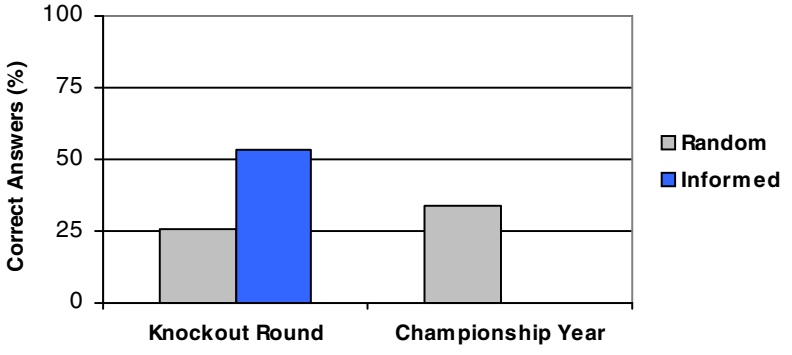


Fig. 4. Success rates (i.e., percentages of correct answers) for simulated users when answering MCQs that ask the user to identify the knockout round / championship year in which a retrieved match was played

Not surprisingly, the success rate for the simulated user that always selects a random answer is close to 25% in *Knockout Round* questions (4 options) and close to 33% in *Championship Year* questions (3 options). However, as predicted by our analysis in Section 3.3, the success rate achieved by the simulated user that always selects the most likely answer in *Knockout Round* questions (53%) is much higher.

4.2 Success Rates in an MCQ Quiz

Table 5 shows the success rates for two simulated users in an MCQ quiz which includes equal numbers of MCQs that ask the user to identify winning teams, losing teams, championship years, knockout rounds, and match scores. The first simulated user (Informed / Random) uses informed guesswork (as described in Section 4.1) in *Winning Team*, *Losing Team*, *Knockout Round*, and *Match Score* questions. In *Championship Year* questions it selects a random answer. The second simulated user (Random) selects a random answer for all questions. The strategies used by the system to select distractors for winning teams, losing teams, and decisive match scores are Most Similar and Random. In questions that ask the user to identify championship years, knockout rounds, or indecisive match scores, the simulated users select from a fixed set of answers (e.g., 1998, 2002, 2006 in the case of championship years). The results are based on 10 replications in which the MCQs presented to the simulated users are generated from all (45) cases in the example case base in random order.

Table 5. Success rates for simulated users in an MCQ quiz with equal numbers of questions that ask the user to identify winning teams, losing teams, championship years, knockout rounds, and match scores

User Strategy	**Distractor Selection Strategy** (Winning Teams, Losing Teams, Decisive Match Scores)	
	Most Similar	Random
Informed / Random	40.7%	56.7%
Random	31.3%	29.3%

Again the results show the benefits of similarity-based distractor selection. Though used only in *Winning Team*, *Losing Team*, and most *Match Score* questions, it has reduced the overall success rate achieved by the first simulated user from 57% to 41%. In summary, the results presented in this section support our hypothesis that distractor selection based on similarity may help to reduce the scores that users can achieve by informed guesswork in certain types of MCQ.

5 Related Work

One example of the use of template-based NLG in CBR is Díaz-Agudo *et al.*'s [5] work on story plot generation. As in our CBR approach to automating the generation of MCQs, NLG is used to support the CBR process in their approach to story generation. It is also interesting to note the work by Fan and Kendall [6] and Francisco *et al.* [7] on the use of CBR to improve the effectiveness of template-based NLG.

Automating the construction of MCQs is a challenge that has attracted the interest of artificial intelligence researchers for many years. For example, Carbonell [8] presents an intelligent system for computer-assisted instruction called SCHOLAR that uses a semantic network to generate questions on a specific topic such as the geography of South America. SCHOLAR generates a combination of true/false, fill-in-the-blank, and multiple-choice questions (e.g., *What language is spoken in Chile?*).

In more recent work, Mitkov *et al.* [9] present a system that uses natural language processing techniques to generate MCQs related to concepts identified from text. Their approach uses language resources such as corpora and ontologies, with excerpts from a linguistics textbook being used to evaluate the system. MCQs created by the system are post-edited by a domain expert before being used for assessment, and may be discarded if considered unsuitable by the expert. Brown *et al.* [10] present an approach to automating the generation of MCQs for vocabulary assessment from WordNet [11], while Papasalouros *et al.* [12] propose an approach to automating the generation of MCQs from domain ontologies.

An important benefit of our CBR approach to automating the generation of MCQs from cases describing events or experiences of interest is the relative ease with which new cases can be added to the case base. As we have shown in the example domain of international football, a small set of case features may be enough to enable a variety of different MCQs to be generated from each case. Another advantage of our approach is that there is no need for MCQs to be post-edited by a domain expert.

6 Conclusions

We presented a CBR approach to automating the construction of MCQs in which measures of interestingness and similarity are used to guide the retrieval of cases and case features from which questions, distractors, and hints are generated in natural language. We also demonstrated the approach in an intelligent system for automating the design of MCQ quizzes called AutoMCQ, and showed that distractor selection based on similarity, where appropriate, may help to reduce the MCQ scores that users can achieve by informed guesswork in the example domain. However, we also identified a potential problem that may occur when similarity is used to select distractors for the correct answer in certain types of MCQ. In future work we would like to investigate the potential impact of providing feedback after each question in a series of automatically generated MCQs on the overall scores that users can achieve, for example by making it easier to eliminate distractors in subsequent questions.

References

1. Collins, J.: Education Techniques for Lifelong Learning: Writing Multiple-Choice Questions for Continuing Medical Education Activities and Self-Assessment Modules. RadioGraphics 26, 543–551 (2006)
2. Harper, R.: Multiple-Choice Questions - A Reprieve. Bioscience Education E-journal, 2–6 (2003)
3. Moss, E.: Multiple Choice Questions: Their Value as an Assessment Tool. Current Opinion in Anaesthesiology 14, 661–666 (2001)
4. Tarrant, M., Ware, J., Mohammed, A.M.: An Assessment of Functioning and Non-Functioning Distractors in Multiple-Choice Questions: a Descriptive Analysis. BMC Medical Education 9, 40 (2009)
5. Díaz-Agudo, B., Pablo Gervás, P., Federico Peinado, F.: A Case Based Reasoning Approach to Story Plot Generation. In: González-Calero, P.A., Funk, P. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 142–156. Springer, Heidelberg (2004)
6. Fan, Y., Kendall, E.: A Case-Based Reasoning Approach for Speech Corpus Generation. In: Dale, R., Wong, K.-F., Su, J., Kwong, O.Y. (eds.) IJCNLP 2005. LNCS (LNAI), vol. 3651, pp. 993–1003. Springer, Heidelberg (2005)
7. Francisco, V., Hervás, R., Pablo Gervás, P.: Dependency Analysis and CBR to Bridge the Generation Gap in Template-Based NLG. In: Gelbukh, A. (ed.) CICLing 2007. LNCS, vol. 4394, pp. 432–443. Springer, Heidelberg (2007)
8. Carbonell, J.: AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction. IEEE Transactions on Man-Machine Systems MMS 11, 190–202 (1970)
9. Mitkov, R., Ha, L.A., Karamanis, N.: A Computer-Aided Environment for Generating Multiple-Choice Test Items. Natural Language Engineering 12, 177–194 (2006)
10. Brown, J.C., Frishkoff, G.A., Eskenazi, M.: Automatic Question Generation for Vocabulary Assessment. In: Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp. 819–826. Association for Computational Linguistics, Morristown (2005)
11. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
12. Papasalouros, A., Kanaris, K., Kotis, K.: Automatic Generation of Multiple Choice Questions from Domain Ontologies. In: IADIS International Conference e-Learning, pp. 427–434. IADIS Press (2008)

Towards Case-Based Adaptation of Workflows

Mirjam Minor, Ralph Bergmann, Sebastian Görg, and Kirstin Walter

Business Information Systems II
University of Trier
54286 Trier, Germany
{minor,bergmann,goer4105,walt4701}@uni-trier.de

Abstract. Creation and adaptation of workflows is a difficult and costly task that is currently performed by human workflow modeling experts. Our paper describes a new approach for the automatic adaptation of workflows, which makes use of a case base of former workflow adaptations. We propose a general framework for case-based adaptation of workflows and then focus on novel methods to represent and reuse previous adaptation episodes for workflows. An empirical evaluation demonstrates the feasibility of the approach and provides valuable insights for future research.

1 Introduction

Today, many companies and organizations must be able to quickly adapt their business according to newly arising opportunities and demands from the customer. This requires a means to flexibly adapt the current business processes which guide the core activities of the organization. Workflow technology is widely used to control the execution of business processes. *Workflows* are "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [1]. The basic constituents of workflows are *tasks* that describe an activity to be performed by an automated service (e.g. within a service-oriented architecture) or a human (e.g. an employee). The procedural rules for the tasks are usually described by routing constructs like sequences, loops, parallel and alternative branches that form a control flow for the tasks. A significant limitation of traditional workflow technology is its missing flexibility of the workflow concept and its management by workflow engines. Workflows are meant to be described in a modeling phase during the setup of the workflow system. Once described, they are executed repeatedly in the same manner over a long period of time. In order to address today's requirements concerning easy and fast adaptation of workflows, a new class of workflow systems is currently emerging, called *agile workflows* systems [2, 3]. Agile workflow systems break the traditional separation between workflow modeling and execution. Workflows can be created (for example based on a template) for a particular demand or business case. Workflows can also be adapted even after they have already been started, for example if some unforeseen events occur. Hence, the creation and adaptation of workflows has now become a very important activity for which urgent support is needed.

Recent work on case-based reasoning (CBR) addresses these needs. Workflow retrieval [4, 5, 6, 7, 8] provides assistance to a workflow modeling expert. Instead of

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 421–435, 2010.

composing new workflows from scratch, retrieved workflows or portions of workflows can be reused. This already reduces the modeling effort significantly. However, the adaptation of the retrieved workflows during modeling or the adaptation of already started workflows is still a challenging task for workflow modelers. It is also an important and widely unexplored research topic for CBR.

This paper presents on a novel framework for the adaptation of workflows. The adaptation of a workflow itself is performed in case-based manner, which enables the reuse of adaptation experience. We collect the experience from previous workflow adaptation episodes in a dedicated adaptation case base. An adaptation case stores the aim of the performed adaptation (change request description), the original workflow, the adapted workflow as well as the difference of both workflows in terms of add and delete lists containing workflow elements. When a new adaptation request for a workflow occurs (as part of an initial workflow modeling activity or as consequence of an unforeseen event to a workflow that is already under execution) this adaptation is performed in a case-based way, i.e., by reuse of an adaptation case.

The next section of the paper discusses relevant related work. Then, we introduce the representation of adaptation cases and present in Sect. 4 the proposed CBR cycle for case-based adaptation. Sect. 5 describes briefly the retrieval of adaptation cases, while Sect. 6 provides the details of the workflow adaptation by adaptation case reuse. Finally, Sect. 7 describes the current state of the implementation of our approach as well as the results of an empirical study.

2 Related Work

Workflow-oriented CBR deals with CBR methods for cases representing workflows. The reuse of workflow templates is widely spread in recent commercial workflow management systems. Before a workflow is enacted, a new workflow instance is derived from the workflow template. CBR is a means to go beyond this kind of assistance. Case-based retrieval is employed to dynamically select and execute suitable workflows to support collaboration in business [9]. Montani [10] proposes the use of CBR to support the management of exceptions in business process execution. Within the WINGS workflow system a method for the semantic retrieval of data-centric workflows from a repository has been developed [7]. Conversational CBR has been applied in the tool CBRFlow [4] to guide the user in adapting a workflow to changing circumstances. Leake and Morwick [8] evaluate the execution paths of past workflows in order to support user extension of workflows that are under construction. Several approaches exploit the relationships between plan and workflow representations and apply case-based planning (CBP) methods (see [11, 12] for a survey on CBP) for the construction of workflows. The CODAW system [5] supports the incremental modeling of workflows by similarity-based retrieval and workflow composition using an HTN planner. Xu and Munoz-Avila [13] apply case-based HTN planning for the construction of project plans. Most approaches of this kind rely on a first principles planner as underlying methodology being enhanced by an adaptation approach. Thus, the strong requirement of a formal planning model (e.g. some kind of STRIPS-like formalization) holds as well for the workflow steps.

This paper focuses on adaptation in the context of workflow cases. Several techniques for adaptation in CBR have been proposed so far (for a review see [14]). The most basic distinction between different adaptation methods is whether transformational adaptation or generative adaptation is applied. Transformational adaptation adapts the solution directly based on the differences between the current problem and the problem stated in the case. This is a knowledge intensive approach since domain specific adaptation knowledge is required that describes how differences in the problem should be compensated by modifications of the solution. Various methods have been proposed which differ in the representation and processing of such adaptation knowledge. Generative adaptation methods require a generative problem solver (e.g. an AI planner) that is able to solve problems based on a complete and sound domain model. This is a very strong assumption that inhibits the application of this approach in many typical CBR application domains where no such domain model can be built. More recent approaches to adaptation in CBR are motivated by the fact that the acquisition of explicit adaptation knowledge for transformational adaptation is a very difficult and time consuming task. Several methods have been developed that exploit the knowledge already captured in the cases as source of adaptation knowledge. In our own earlier previous work [15], we propose a general framework for learning adaptation knowledge from cases. Meanwhile, several successful examples of such methods exist that either apply inductive learning to extract adaptation knowledge [16, 17, 18] or that apply a case-based adaptation approach [19].

The adaptation approach proposed in this paper is a kind of case-based adaptation as it is based on a dedicated case base of previous successful adaptations. On the other hand, each adaptation case describes a particular kind of workflow modification that can be transferred to new situations. Hence, the adaptation case base can be considered experiential transformational adaptation knowledge. In particular, we do not assume any formal semantic representation of the workflow steps (tasks).

3 Representation of Workflow Adaptation Cases

The adaptation knowledge for workflows is represented in case-based manner. Before the case structure is discussed, the workflow modeling language we have developed within the CAKE framework [3, 11] is briefly introduced. However, our adaptation approach can be extended to other control-flow-oriented workflow languages.

The CAKE workflow modeling language (compare [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 element, an end element, 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. For example, the workflow depicted in box 2 of Fig. 1 has the following workflow elements: A start element, a task “Ask for invoice number”, an AND-block containing two further tasks, a task “Send confirmation”, and an end element. Each workflow element has a unique identifier (ID) and an execution state. The execution state “ACTIVE” is assigned to elements that are currently executed. “READY” means that a workflow element has not yet been executed, and “COMPLETED” denotes the status of workflow elements that have already finished execution.

An *adaptation case* can now be described as follows. It is the representation of a previous adaptation episode of a workflow. Like a traditional case, it is structured into a problem part and a solution part:

1. The *problem part* consists of
   a. a semantic description of the *change request* (what causes the change) and
   b. the *original workflow* prior to the adaptation.
2. The *solution part* contains
   a. the *adapted workflow* and
   b. the description of the *adaptation steps* (added and deleted workflow elements) that have been executed to transform the original workflow into the adapted workflow.

The change request is a semantic description of the cause of the request. This part of the problem description makes use of traditional case representation approaches, e.g. a structural representation or a textual representation. Workflows are represented as described before. The representation of the adaptation steps (2b) deserves some special attention. Similar to STRIPS operators, the adaptation steps are described by an *add* and a *delete* list. Each list contains a set of chains of workflow elements. A chain encapsulates a set of connected workflow elements with one entry point and one exit point. Further, each chain records a pair of *anchors*. A *pre anchor* is the workflow element (in the original workflow) after which workflow elements from the chain have been added or deleted. A *post anchor* is the workflow element from the original workflow following the last element of the chain. Hence, the pre anchor

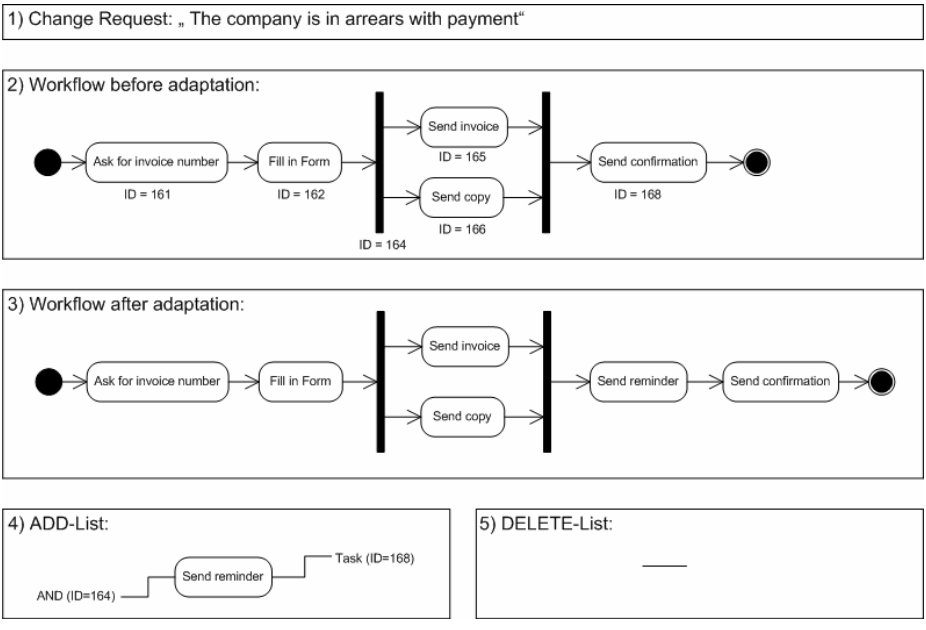


Fig. 1. Sample workflow adaptation case for the workflow "Send an invoice"

describes the position after which the adaptation starts and the post anchor describes the first position in the original workflow that is not anymore affected by the adaptation described in the chain. These anchors are further used during the reuse of the workflow adaptation to identify similar points in new workflows at which the proposed adaptation can be applied. Please note that the overall description of the adaptation steps can be automatically computed from the original workflow and the adapted workflow or alternatively it can be captured as part of a modeling tool that is used for manual workflow adaptation.

Fig. 1 illustrates a sample workflow adaptation case. The considered workflow describes the business process of invoicing something to a business partner. The topmost box of Fig. 1 contains a sample change request (in a textual representation) that occurred during the execution of the invoice workflow: The recipient of the invoice is in arrears with the payment. The original workflow is depicted in box 2 of Fig. 1 in UML activity diagram notation. This workflow is adapted to fulfill the change request, by sending a reminder to the partner. The workflow in box 3 shows the adapted workflow. The original workflow has been modified by adding the newly created task "Send reminder". In boxes 4 and 5 the according adaptation steps are represented by an add and a delete list. The added task "Send reminder" is listed together with the pre anchor (AND with element ID 164) and the post anchor (Send confirmation Task with element ID 168).

4 CBR Cycle as Framework for Case-Based Adaptation

We now propose a general framework for the case-based adaptation of workflows by applying the traditional CBR cycle [20] to adaptation cases. Hence, the case base consists of adaptation cases (see Sect. 5). The similarity measure, which must be able to assess the similarity between two problem descriptions of an adaptation case must therefore be able to consider the *change request similarity* as well as the *workflow similarity*.

The new problem to be solved (query) consists of a *target workflow* (which may be already partially executed) and the description of the *current change request* of the change to be made on this target workflow. Fig. 2 illustrates the framework. In the first step the most relevant adaptation cases are *retrieved* from the case base by similarity-based retrieval employing the similarity measure. One or possibly several adaptation cases are selected for reuse. In the *reuse* phase, the adaptation steps contained in the adaptation case(s) are applied to the target workflow. This occurs in two distinct steps. First, the concrete location in the target workflow is determined that needs to be changed. This is necessary as there are usually many different positions within the workflow at which a chain from the add list can be inserted or to which the deletions in the delete list can be applied. Second, the changes are applied to the target workflow at the determined locations. The resulting *adapted workflow* is then the proposed solution. During the subsequent *revise* phase the user can validate the workflow adaptations proposed: she can either confirm them or revise them by manually performing appropriate adaptations herself. The current graphical workflow modeling component of our CAKE system [21] supports such a manual workflow adaptation. The resulting confirmed solution then represents the correctly adapted

target workflow that the workflow engine continues to enact. Finally, a new validated adaptation case has been constructed, which can then be *retained* as part of the adaptation case base.

We see that the application of the CBR cycle to adaptation cases is quite straight forward. However, the most crucial part is the reuse phase which requires novel methods for transforming previous workflow modifications in order to apply them to the target workflow. Therefore, our research in this paper focuses particularly on this phase. Only after the characteristics of the new reuse methods have been investigated, we will be able to re-consider the similarity assessment as the similarity assessment must ensure the retrieval of adaptable cases and hence it is dependent on the particular reuse methods. Despite of the focus on reuse methods we will now briefly give some thoughts on similarity measures for workflows (Sect. 5) as they are also needed within the reuse methods being described in Sect. 6.

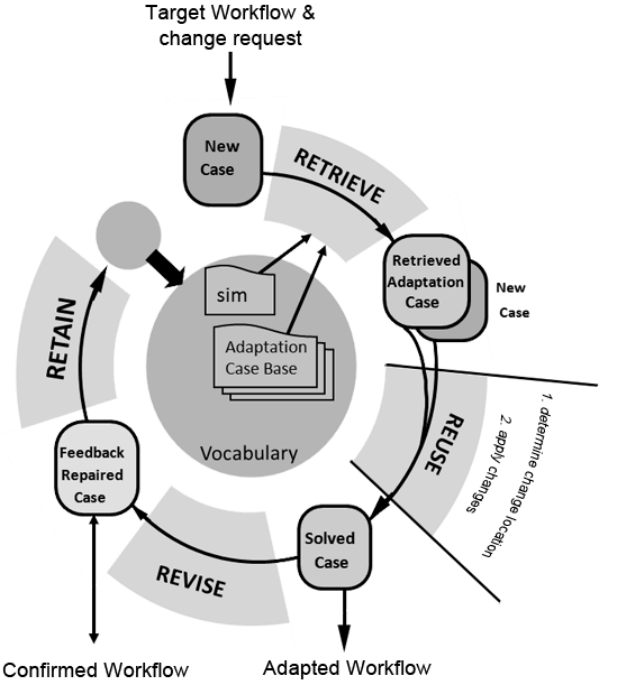


Fig. 2. Visualization of the adaptation cycle

5 Workflow Retrieval

The retrieval of workflow adaptation cases requires similarity measures for the different components of the problem part: a *change request similarity* and a *workflow similarity.* Both similarity values are aggregated (we use a weighted sum) to determine the overall similarity of adaptation cases. As the focus of this paper is on adaptation, we

only sketch a very simple similarity measure, which we use in the subsequent case study. Please refer to our previous work [3] and to the literature [5] for more sophisticated similarity measures for workflows that could be used as well. For the purpose of this work, we restrict our representation of the change request to a textual description of the purpose of the change. Change request similarity is then derived from the traditional Levenshtein distance measure. 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. Future work will elaborate the use of a change request ontology (and a related similarity measure) to include more semantics.

The workflow similarity measure used aggregates the similarity of the set of occurring workflow elements and the similarity of the abstracted control flow structure. The similarity measure for workflow elements (which is also employed during the adaptation, see Sect. 6) compares tasks by means of their task names and control-flow elements (AND, XOR, etc.) by an exact matching. The similarity of task names is again simply computed by a Levensthein distance. In future work, more sophisticated measures for the similarity of workflow elements could be investigated. For instance, further properties of the workflow elements could be included like the input and output parameters specifying the data flow.

In order to compute the similarity of the abstracted control flow the control flow graph is serialized into a string containing a specific character for each workflow element (see Fig. 3 for an example). For reasons of simplicity, the current serialization only considers the utmost sequence of tasks and blocks, but can be easily extended to include inner blocks as well. The similarity between two serialization strings is again measured by applying the Levenshtein distance. For the sample in Fig. 3, the Levenshtein distance is 2, which would induce a similarity value of 0.5.

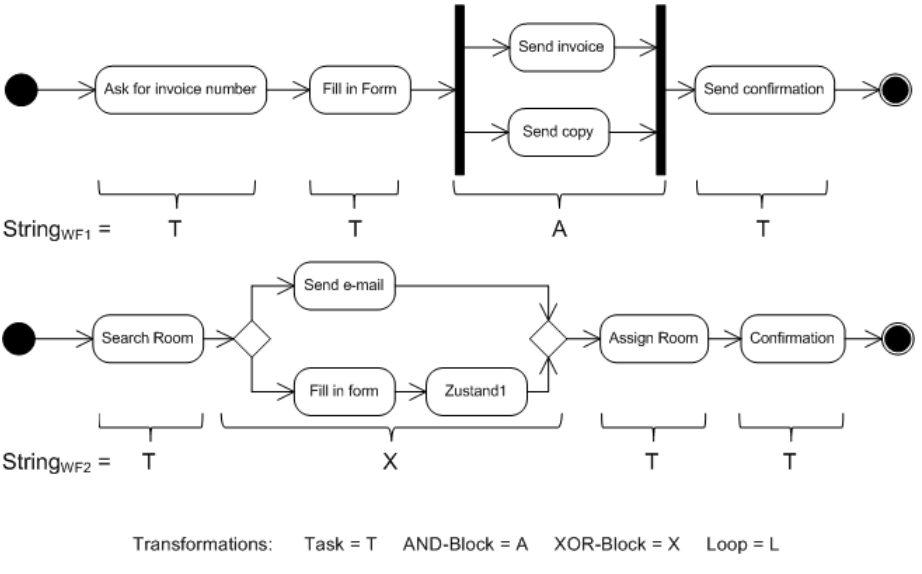


Fig. 3. Simple serialization of the control flow

6 Workflow Adaptation

The result of the workflow retrieval is one or more workflow adaptation cases. In the current approach, we accept only one adaptation case per reuse phase. The adaptation knowledge from this case is to be transformed and applied to the new workflow. The assumption is that the lists of edit operations from the retrieved case (add and delete list) will provide appropriate structural changes when adopted to modify the new workflow. Two main issues have to be solved for this: First to determine the locations where the adaptation steps should be applied to the new workflow and, second, to execute the adaptation.

6.1 Determine Change Location

In a manual adaptation scenario the workflow modeler knows where a workflow needs to be adapted. For the scenario of an automatic workflow adaptation, the positions for applying the edit operations in the target workflow have to be determined for every chain. This problem can be solved by means of the anchor concept. The anchors of the chains have to be mapped to appropriate positions in the target workflow.

The anchor mapping method consists of three steps: (1) First it determines *candidate positions* for the anchors in the target workflow. (2) Then it assesses the candidate positions to restrict the set of potential positions to a set of *valid candidate positions* for each anchor. (3) Last, the mapping from the anchors to the set of valid candidate positions is constructed. The resulting mapping might be partial but it has to satisfy the following consistency criterion: A mapping of anchors to positions in the target workflow is called *consistent* if it preserves the inner order of anchor pairs. That means that for any mapped anchor pair the position of the pre anchor must be before to the position of the post anchor in the target workflow. A pair of candidate positions is called *consistent pair of candidate positions* if it does not violate the consistency of a mapping. Fig. 4 illustrates the mapping method by the sample anchor pair from Fig. 1. The target workflow provides six candidate positions. The result of the mapping method is only a partial mapping as the pre anchor could not be mapped to an appropriate position. In the following, the three steps of the mapping method will be described in more detail.

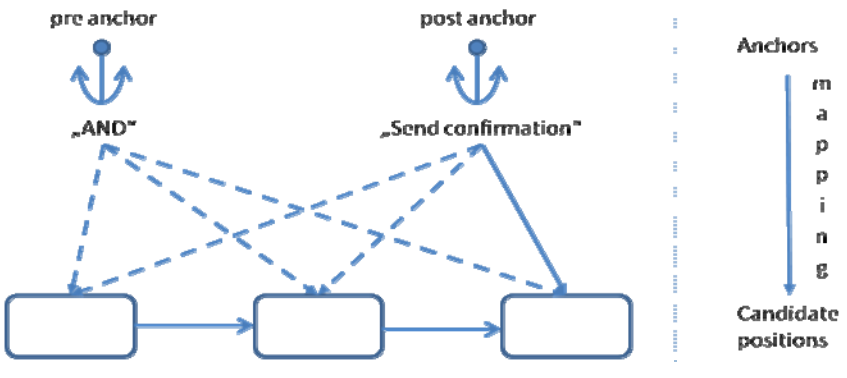


Fig. 4. Sample candidate positions for a pair of anchors

(1) The first task of determining candidate positions uses a filter for workflow elements that provides candidate positions. It would not be useful to take workflow elements into consideration that have already completed their execution. Hence, the execution state of the workflow elements is evaluated to filter out workflow elements that are not editable any more (those with the status ACTIVE or READY remain). The remaining workflow elements provide the candidate positions. This first step of the mapping method can be omitted in case the workflows have not yet started execution. In this case, the positions of all workflow elements in the target workflow are regarded candidate positions.

(2) The second step of the mapping method assesses the candidate positions for a particular anchor. The assessment step employs the similarity measure for workflow elements to filter out invalid candidate positions. The similarity of the element at the candidate position to the element at the anchor position in the chain is computed. All candidate positions for which this similarity value is above a threshold called the *validity threshold* are considered valid candidate positions for an anchor. The other candidate positions are refused.

(3) The third step selects valid candidate positions to construct a consistent mapping of anchors to candidate positions. Four different mapping methods have been investigated (compare Sect. 7):

Pre anchor mapping (PRAM): The post anchors are ignored, while for each pre anchor the best matching candidate position is selected according to the workflow similarity measure. In case a candidate position is the best matching position for multiple pre anchors, one pre anchor is chosen arbitrarily and the others are mapped to the next best matching candidates.

Post anchor mapping (POAM): The pre anchors are ignored, while the best matching positions for the post anchors are selected analogously to the PRAM method.

Composite anchor mapping (CAM): The mapping chooses candidate positions from the set of consistent pairs of valid candidate positions and from the set of single candidates as follows. First the set of pairs is split into two sets, namely to the set of *tight pairs*, that means that the candidate position of the pre anchor is a direct successor to the candidate position of the post anchor, and to the set of non-tight pairs. The set of tight pairs provides candidates for mapping the anchors from the add list, while the set of non-tight pairs is considered for anchors from the delete list. Then both sets of pairs are ordered according to the sum of the workflow element similarities at the pre anchor positions and at the post anchor positions. The similarity value of the best matching pair (in case there is one) is compared to the similarity value of the best matching single candidate according to the PRAM and PROM method. The pair or single candidate with the highest similarity value is chosen. In case a candidate is selected several times, one anchor pair is selected arbitrarily and the others are mapped to the next best matching pairs or single candidates.

Maximum anchor mapping (MAM): This method is very similar to the CAM method. Just the ordering of candidate pairs is different: The sum aggregating the similarity values for the pre and post anchor positions of a pair is replaced by a maximum function.

The mapping result plays a different role for chains from add lists and from delete lists: In case of a chain from the add list, the anchor positions are the only criterion to determine the insert positions within the target workflow. In case of a chain from the delete list, the main criterion to determine the positions of workflow elements that have to be deleted from the new workflow is a mapping of the workflow elements from the delete list to workflow elements in the target workflow. Again, the similarity measure for workflow elements is applied. The best matching workflow elements are candidates for being deleted. The fitting anchor positions are a secondary criterion to prevent premature delete operations.

6.2 Application of Changes

The positions of the anchors determined by the mapping method specify where to apply the changes to the target workflow. The add operations are executed immediately for all chains of which at least one anchor has been mapped successfully. In case there is a position for the pre anchor the add operations are performed starting at this point. Otherwise, the add operations are performed ending at the position of the post anchor. Chains whose anchors could not be mapped cannot be applied automatically. They might be applied by the user in the revise phase (see Sect. 4).

Chains from the delete list are applied only if the mapping of their elements is complete and the similarity values for the individual workflow elements are above a threshold called *delete threshold*. Additionally, it is required that the elements to be deleted are organized in exactly the same control flow than those in the chain. Thus, the delete operations have to fulfill stronger constraints than the add operations to be applied.

The above list of four mapping methods may be extended by further methods. None of the above methods guarantees completeness but partial mappings allow to apply parts of the adaptation steps. As the evaluation has shown (see below), the application of the four methods has been quite successful already for some test cases.

7 Implementation and Evaluation

7.1 Collaborative Agent-based Knowledge Engine (CAKE)

The described methods have been fully implemented as part of the Collaborative Agent-based Knowledge Engine (CAKE) [9, 21] developed at the University of Trier within several research projects. CAKE provides modeling and enactment support for agile workflows. It uses a light-weight process modeling language for the agile workflows at the user level by extending the UML activity diagram notation by own symbols (see also Sect. 3). The workflow technology offered allows late-modeling and manual adaptation of ongoing workflows. Some typical ad-hoc changes are to reorder some parts of a workflow instance or to insert an additional task. CAKE also offers data modeling facilities using a common object-oriented data model and a comprehensive structural case representation framework, a library of configurable similarity measures, and a similarity-based retrieval component. As part of our current research CAKE has been extended as follows:

- it now supports representation of adaptation cases
- the application of the existing similarity measures for the retrieval of adaptation cases, and
- the reuse of adaptation cases as described in Sect. 6.

7.2 Evaluation Goal and Domain

The extended CAKE system has then been used in a first empirical evaluation of the proposed adaptation methods. The evaluation itself is a big challenge, since immediately useful test data about workflow cases is not available. Although some public workflow repositories exist, none of them considers and records workflow changes. Therefore, we needed to spend a significant effort to develop and elaborate a particular test scenario for this evaluation. To evaluate our approach, it was necessary to obtain a case base of adaptation cases and a set of new test problems each of which consists of a target workflow and a change request. We decided to perform this evaluation in the domain of administrative processes as they typically occur in University offices. In several structured interview sessions with office secretaries, we first obtained a set of 19 typical office workflows as they are executed on a regular basis in our University. Then, for each workflow the interviewees were asked to describe a common change request and the resulting change in the workflow [22]. From the records of these interviews, a case base of adaptation cases has been manually constructed using the CAKE system. Then, in an independent interview session, a set of nine new test problems together with reference solutions (adapted workflows) was also elicited from the secretaries. These test queries have been obtained in a similar manner than the adaptation cases, but care has been taken that the workflows are different from those in the case base. We are aware that the size of the case base and the number of target problems is quite small and does not allow applying statistical methods to evaluate the results, but the large effort of such an experiment with real users does not yet enable a larger experimental data set in this phase of research. However, an initial evaluation and an assessment of the applicability of this approach for the domain of office administration should be possible.

As the target of our research is the reuse phase, this evaluation particularly aims at providing first insights about the four variants of workflow adaptation. We want to investigate whether the four methods are able to correctly apply the adaptation steps in the adaptation cases and whether the four methods differ in the degree of correctness of the adaptation. For this purpose we executed two experiments described below.

7.3 Experiment 1: Adaptation Case Reconstruction

The aim of the first experiment was to evaluate whether each of the proposed reuse methods is able to correctly reconstruct the adaptations described in each of the 19 adaptation cases. For each adaptation case the four variants of reuse methods have been applied to the contained original workflow, i.e. the description of the adaptation steps is used to compute an adapted workflow from the original workflow. By comparing the computed adapted workflow with the correct adapted workflow noted in the adaptation case, the correctness of the reuse method can be measured. The correctness then mainly depends on the capability to correctly reconstruct the anchor

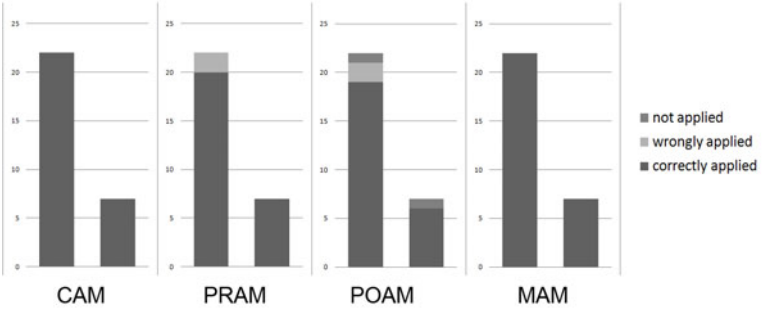


Fig. 5. Results of Experiment 1

positions to which the adaptations have to be applied. As a quality criterion we measured the average number of *correctly* applied add chains, *wrongly* applied add chains, and *not* applied add chains as well as the average number of *correctly* applied *delete* chains, *wrongly* applied delete chains, and *not* applied delete chains. Fig. 5 shows the results of this experiment. There are two bars for each variant. The left bar shows the results for the 22 add chains and the right bar shows the results for the 7 delete chains occurring altogether in the 19 adaptation cases. Only CAM and MAM are able to correctly reconstruct all adaptation cases. If only one anchor is used as in PRAM and POAM, not all anchor positions can be identified correctly.

7.4 Experiment 2: Adaptation of Test Problems

In the second experiment, we investigated the capability of the four reuse variants to correctly adapt the workflows of the 9 test problems. As we only aim at evaluating the reuse phase, we skipped the retrieval phase of our adapted CBR cycle from Fig. 2 to diminish the impact of an improper similarity measure on the assessment of the reuse methods. Instead, for each test problem, the best matching adaptation case has been selected manually with support from an office secretary. Then, the adaptation steps in the selected case are applied to the target workflow of the test problem. In this experiment, we used a high validity threshold of 0.9 for add lists. Hence, adaptations are only performed if there is high evidence that the positions are correct. Finally, the resulting adapted workflow is compared with the reference solution noted within the test problem description. Then, the same quality criterion is applied as in experiment 1. Fig. 6 illustrates the results of this experiment. As in experiment 1 the left bar shows the results for the add chains and the right bar the results for the delete chains. The best results are achieved by CAM and MAM. However, there is no difference of the four variants applied to the delete chains and there is only little differences of CAM and MAM applied to the add chains.

To investigate the influence of the validity threshold, we lowered the validity threshold to 0.7 and run the experiment again. With a lower threshold, adaptation steps are transferred even if the matching of the anchors is less perfect, leading to increased matching opportunities. If the similarity measure is appropriate, we expect also an increasing number of correctly applied adaptation steps.

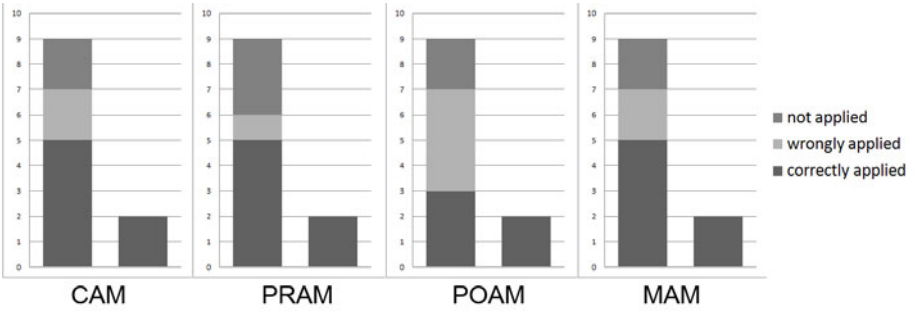


Fig. 6. Results of Experiment 2 - validity threshold=0.9.

The results are shown in Fig. 7. The best result was obtained by CAM leading to 7 correctly applied and none not applied add chains. With all four methods, the number of correctly applied adaptation steps is increased with the lower threshold, while the number of wrongly applied steps is not changed. These results indicate that at least in this setting a lower validity threshold improves finding proper anchors. It is quite likely that this effect is strongly dependent on the similarity measure used. This issue must be investigated in more depth in future experiments with improved similarity measures.

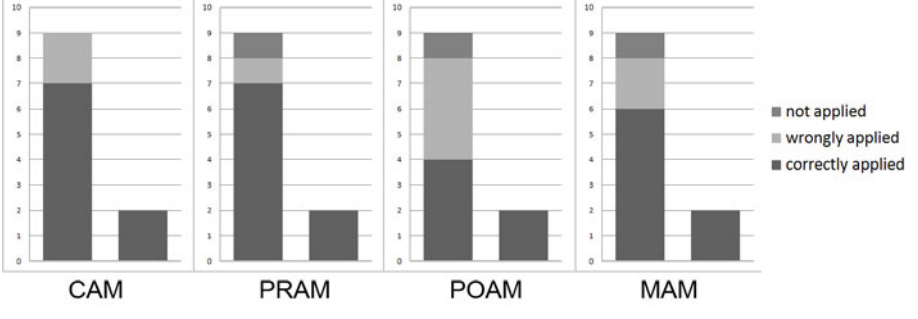


Fig. 7. Results of Experiment 2 - validity threshold=0.7

8 Conclusion and Open Issues

In this paper, we presented a novel framework for case-based adaptation of workflows together with first results from an empirical evaluation of a sample implementation. Workflow adaptation cases store pairs of workflows and a change request describing the cause why the first workflow has been converted into the second. The adaptation steps are described as edit operations (add and delete steps) that are applicable to similar workflows with similar change requests as well. The reuse phase consists of two tasks: First to determine the change location in the target workflow and second to apply the changes from the retrieved case. A novel anchor concept has been introduced to find appropriate locations for chains of edit operations. The evaluation on

sample workflows from the office domain has shown that considering two anchors per chain performs better results than single anchors only. The anchor concept has turned out to be suitable for transforming adaptation steps from one workflow to another in a real-world domain. Our work so far has confirmed the intention to develop a framework for case-based adaptation of workflows. Novel open issues arise that concern the reuse phase (e.g. more complicated anchors integrating several workflow elements and data flow) as well as general issues like the acquisition and maintenance of the adaptation cases. Further domains like e-science workflows or cooking could be investigated. We believe that our results could be useful for the procedural knowledge in many further application areas.

References

1. Workflow management coalition glossary & terminology, http://www.wfmc.org
2. Weber, B., Wild, W.: Towards the agile management of business processes. In: Althoff, K.-D., Dengel, A.R., Bergmann, R., Nick, M., Roth-Berghofer, T.R. (eds.) WM 2005. LNCS (LNAI), vol. 3782, pp. 409–419. Springer, Heidelberg (2005)
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), 80–98 (2008)
4. Weber, B., Wild, W., Breu, R.: Cbrflow: Enabling adaptive workflow management through conversational case-based reasoning. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 434–448. Springer, Heidelberg (2004)
5. Madhusudan, T., Zhao, J.L., Marshall, B.: A case-based reasoning framework for workflow model management. Data and Knowledge Engineering 50, 87–115 (2004)
6. Minor, M., Tartakovski, A., Bergmann, R.: Representation and Structure-Based Similarity Assessment for Agile Workflows. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 224–238. Springer, Heidelberg (2007)
7. Gil, Y., Kim, J., Florenz, G., Ratnakar, V., Gonzalez-Calero, P.: Workflow Matching Using Semantic Metadata. In: Proc. of the 5th Int. Conf. on Knowledge Capture (K-CAP), pp. 121–128. ACM, New York (2009)
8. Leake, D.B., Kendall-Morwick, J.: Towards Case-Based Support for e-Science Workflow Generation by Mining Provenance. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 269–283. Springer, Heidelberg (2008)
9. Bergmann, R., Freßmann, A., Maximini, K., Maximini, R., Sauer, T.: Case-based support for collaborative business. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 519–533. Springer, Heidelberg (2006)
10. Montani, S.: Prototype-based management of business process exception cases. Applied Intelligence, Springer Science + Business Media, LLC 2009 (2009) (Published online)
11. Cox, M.T., Muñoz-Avila, H., Bergmann, R.: Case-based planning. The Knowledge Engineering Review 20(3), 283–287 (2005)
12. Muñoz-Avila, H., Cox, M.T.: Case-based plan adaptation: An analysis and review. Intelligent Systems 23(4), 75–81 (2008)
13. Xu, K., Munoz-Avila, H.: CaBMA: Case-Based Project Management Assistant. In: Proceedings of The Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI 2004), pp. 931–936. AAAI Press, Menlo Park (2004)

14. Lopez Mantaras, 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. The Knowledge Engineering Review 20(03), 215–240 (2005)
15. Wilke, W., Vollrath, I., Bergmann, R.: Using knowledge containers to model a framework for learning adaptation knowledge. In: European Conference on Machine Learning (MLNet) Workshop Notes – Case-Based Learning: Beyond Classification of Feature Vectors (1997) (published online)
16. Hanney, K., Keane, M.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS (LNAI), vol. 1266, pp. 179–192. Springer, Heidelberg (1997)
17. Craw, S., Wiratunga, N., Rowe, R.C.: Learning adaptation knowledge to improve case-based reasoning. Artificial Intelligence 170(16-17), 1175–1192 (2006)
18. Badra, F., Cordier, A., Lieber, J.: Opportunistic adaptation knowledge discovery. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 60–74. Springer, Heidelberg (2009)
19. McSherry, D.: Demand-driven discovery of adaptation knowledge. In: IJCAI 1999, pp. 222–227. Morgan Kaufmann, San Francisco (1999)
20. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
21. Minor, M., Schmalen, D., Weidlich, J., Koldehoff, A.: Introspection into an agile workflow engine for long-term processes. In: WETICE 2008. IEEE, Los Alamitos (2008)
22. Minor, M., Schmalen, D., Kempin, S.: Demonstration of the Agile Workflow Management System Cake II Based on Long-Term Office Workflows. In: CEUR Proceedings of the BPM 2009 Demonstration Track, vol. 489 (2009) (published online)

A Method Based on Query Caching and Predicate Substitution for the Treatment of Failing Database Queries

Olivier Pivert[1], Hélène Jaudoin[1], Carmen Brando[2], and Allel Hadjali[1]

[1] Irisa – Enssat, University of Rennes 1
Technopole Anticipa, 22305 Lannion Cedex, France
pivert@enssat.fr, jaudoin@enssat.fr, hadjali@enssat.fr
[2] IGN/Cogit
73, Avenue de Paris, 94165 Saint-Mandé Cedex, France
carmen.brando-escobar@ign.fr

Abstract. This paper proposes an approach aimed at obviating empty answers for a family of conjunctive queries involving value constraints. Contrary to the approaches based on a relaxation of the predicates involved in the query, the principle suggested here consists in replacing the query by a similar one which has been processed previously and whose answer is known to be non-empty. This technique thus avoids the combinatory explosion induced by classical relaxation-based approaches.

1 Introduction

In the context of database querying, users may find themselves in the situation where their queries do not return any answers. Since the late 80s, some cooperative answering approaches have focused on this problem. Some of these approaches aim at identifying the cause of the failure and rely on the concept of false presuppositions (a presupposition being any statement implied by the query). A query with a false presupposition necessarily fails (i.e., leads to an empty set of answers), and the idea is to identify these false presuppositions so as to help the user modify his/her query [1]. For instance, the statement "the king of France is bald" has as a presupposition "there is a king of France" which is false. Godfrey [2] considers any subquery as a presupposition of the query itself, and the approach he proposes searches for Minimal Failing Subqueries (MFSs) of a failing query. In this paper, we consider the situation where a set of answers is empty because the database just does not contain any information that fits the requirements expressed in the query (we do not consider any knowledge about the data — in the form, e.g., of integrity constraints — and external to them). The main issue is then to provide the user with some alternative data corresponding to a *similar* information need.

Among the aforementioned cooperative approaches, many rely on the *relaxation* paradigm [1], whose aim is to expand the scope of a query so as to get answers which are in the neighborhood of the original user's query and consists

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 436–450, 2010.

in replacing some query conditions by more general conditions or in eliminating some of them (see, e.g., [3,4,2,5,6,7]). The main problem with this type of approach is that it induces an important combinatorics in the case of conjunctive queries since one has to scan the lattice of all possible relaxed queries (which can be huge) and process many of them (which is expensive) before finding a non-failing one.

In this paper, we aim to avoid this combinatory explosion by exploring a *case-based reasoning* approach. The idea is to take advantage of the queries previously submitted to the system for efficiently finding a non-failing query as similar as possible to a failing one. In this study, we consider conjunctive queries of the form "Q = P_1 and P_2 and ... and P_n" where each P_i is of the form "$attribute_i \in E_i$" and E_i is either a finite set or an interval defined in the set of real numbers. Basically, the idea is to look for a "good" global substitute to a failing query Q among queries previously submitted to the system (and stored in a repository — also called *query cache* in the following — D^+) whose sets of answers are known to be non-empty. Such an approach implies to have available a resemblance measure over every attribute domain involved in the database considered so as to define the notion of *semantic proximity* between queries. With this approach, one has the guarantee to obtain a non-empty answer in one step since only one query — known to be non-failing — needs to be processed. Notice that this approach is relevant mainly for databases over which deletions and updates are rare or inexistant (a typical example is DBLP, or any archiving system). Otherwise the repository would have to be updated too, which would be a complex and expensive task.

The remainder of the paper is organized as follows. Section 2 presents some background notions related to semantic proximity. A predicate substitution approach is presented in Section 3. Section 4 describes a query replacement strategy suited to the treatment of failing conjunctive queries. Implementation aspects are dealt with in Section 5, which also presents some experimental results (notably about the efficiency of the approach). Section 6 is devoted to a comparison of our proposal with some related works. Finally, Section 7 concludes the paper by recalling its main contributions and outlining perspectives for future work.

2 About Fuzzy Sets and Semantic Proximity

Fuzzy sets [8] aim at describing vague classes of objects, i.e., classes whose boundaries are not clear-cut. They generalize classical sets by viewing membership as a graded concept. The membership degree of an element x to a fuzzy set A, denoted by $\mu_A(x)$, takes a value in the interval [0, 1]: $\mu_A(x) = 0$ (resp. 1) indicates that x does not belong at all (resp. completely belongs) to A. The *support* S_A and the *core* C_A of a fuzzy set A defined over a universe X are two crisp sets defined respectively as:

$$S(A) = \{x : x \in X, \mu_A(x) > 0\}$$

and

$$C(A) = \{x : x \in X, \mu_A(x) = 1\}.$$

A continuous fuzzy set is often defined by means of a trapezoid membership function (t.m.f.) (cf. Fig. 1). It is defined by the limits of its support (resp. a and d), and those of its core, (resp. b and c).

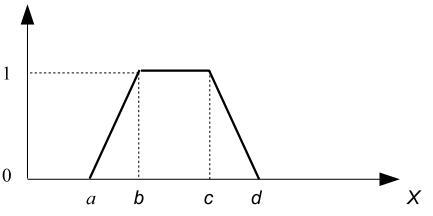


Fig. 1. Trapezoidal membership function

A proximity relation can be used for expressing the degree of closeness between two elements of a given domain. The general definition is as follows [9]:

Definition 1. A proximity relation is a *fuzzy* relation *prox* on a domain U, such that for u, $v \in U$,

1. $\mu_{prox}(u, u) = 1$ (reflexivity),
2. $\mu_{prox}(u, v) = \mu_{prox}(v, u)$ (symmetry).

The quantity $\mu_{prox}(u, v)$ evaluates the proximity, or resemblance, between elements u and v. When categorical values are dealt with, the proximity relation can be given in extension (cf., e.g., Table 1). On the other hand, when scalar domains are concerned, at least two approaches are possible. One can evaluate either

- the extent to which $u - v$ is close to 0 (*absolute* comparative approach), or
- the extent to which the u/v is close to 1 (*relative* comparative approach).

Definition 2. *Relative closeness* may be modeled by a fuzzy relation *prox* such that:

$$\mu_{prox}(x,\ y) = \mu_M(x/y),$$

where M is a fuzzy term meaning "close to 1", such that:

1. $\mu_M(1) = 1$ (since x is perfectly close to x);
2. $\mu_M(t) = 0$ if $t \leq 0$ (assuming that two numbers close to each other should have the same sign);
3. $\mu_M(t) = \mu_M(1/t)$ (symmetry). Then, M is a symmetric fuzzy number which means that $S(M) = [1 - \epsilon, 1/(1 - \epsilon)]$ where ϵ is a real number.

M is called a tolerance parameter. Strict equality is recovered for $M = 1$ defined as $\mu_1(x/y) = 1$ if $x = y$ and $\mu_1(x/y) = 0$ otherwise.

Definition 3. *Absolute closeness* may be modeled by a fuzzy relation *prox* such that:

$$\mu_{prox}(x,\ y) = \mu_Z(x - y),$$

where Z is a fuzzy set centered in 0, such that:

1. $\mu_Z(r) = \mu_Z(-r)$;
2. $\mu_Z(0) = 1$;
3. its support $S(Z)$ is bounded; it is denoted by $[-\delta, \delta]$, where $\delta \in \mathbb{R}^+$.

Classical equality is recovered for Z defined as $\mu_Z(x-y) = 1$ if $x = y$, 0 otherwise. In the sequel, we assume that every proximity relation follows either an absolute comparative approach or a relative comparative approach.

3 Predicate Substitution

Let $P = (A \text{ in } E)$ be a failing predicate w.r.t. a relation r ($\forall t \in r$, $t.A \notin E$), where A is an attribute and E is a finite set or an interval, and $P' = (A' \text{ in } E')$ a non-failing predicate w.r.t. r ($\exists t \in r$, $t.A \in E'$). We denote by *prox* the proximity relation defined over the domain of A.

3.1 Case Where E and E' Are Finite Sets

In order to assess the "quality" of P' as a substitute to P, a measure is needed. It is not strictly speaking a proximity measure, since the symmetry property is not desired here. Indeed one wants to know whether P' is a good substitute to P, but not the reciprocal. Several possible substitutivity measures (denoted by sbs_i later on) are discussed hereafter.

1st idea: one assesses the extent to which every element from $(E'-E)$ resembles at least one element from E:

$$sbs_1(P,\ P') = inf_{x' \in (E'-E)}\ \{sup_{x \in E}\ \{prox(x,\ x')\}\}. \tag{1}$$

The problem with this measure is that the worst element "masks" the others, as illustrated in the next example. In the following, we assume available the resemblance relation on animals depicted in Table 1.

Table 1. Proximity relation over attribute *animal*

	rooster	hen	duck	turkey	cow
rooster	1	0.9	0.6	0.7	0
hen	0.9	1	0.6	0.7	0
duck	0.6	0.6	1	0.5	0
turkey	0.7	0.7	0.5	1	0
cow	0	0	0	0	1

Example 1. E = {hen, duck, turkey}, E'_1 = {hen, turkey, cow}, E'_2 = {cow, rooster}. We get $sbs_1(P,\ P'_1) = sbs_1(P,\ P'_2) = 0$ but since there are neither hens nor turkeys in the database — otherwise P would not be a failing predicate —, it seems reasonable to claim that E'_2 should be a better substitute than E'_1. However, in the computation of $sbs_1(P,\ P'_2)$, the element cow "masks" rooster.$\diamond$

2nd idea: one assesses the extent to which there is an element from $(E' - E)$ which resembles at least an element from E:

$$sbs_2(P,\ P') = sup_{x' \in (E'-E)}\ \{sup_{x \in E}\ \{prox(x,\ x')\}\}. \tag{2}$$

Here, the difficulty is that the "winning set" may include elements which are very distant from those desired by the user.

Example 2. $E = \{\text{hen}\}$, $E'_1 = \{\text{hen, cow, turkey}\}$, $E'_2 = \{\text{duck}\}$. Here, E'_2 should win, since it includes only elements close to the desired ones, contrary to E'_1, which includes "cow". However, it is E'_1 which wins since $sbs_2(P,\ P'_1) = 0.7$ while $sbs_2(P,\ P'_2) = 0.6$.$\diamond$

3rd idea: one mixes the quantitative and the qualitative aspects by measuring the average resemblance degree between an element from $(E' - E)$ and an element from E. For each element x' from $(E' - E)$, the corresponding measure looks for the maximal proximity between x' and an element x from E, computes the sum of these maximal proximities, and divides this sum by the number of elements present in $(E' - E)$:

$$sbs_3(P,\ P') = \frac{\sum_{x' \in (E'-E)}\ sup_{x \in E}\ \{prox(x,\ x')}{|E' - E|}\}. \tag{3}$$

Example 3. $E = \{\text{hen, duck, turkey}\}$, $E'_1 = \{\text{hen, turkey, cow}\}$, $E'_2 = \{\text{cow, rooster}\}$. We get: $sbs_3(P,\ P'_1) = 0$ and $sbs_3(P,\ P'_2) = 0.45$.$\diamond$

Since measure sbs_3 appears the most satisfactory, it will be used in the following.

3.2 Case Where E and E' Are Intervals

Let us first consider the simple case where proximity between two intervals x and y is defined in a Boolean manner:

$$res(x,\ y) = 1 \text{ if } |x - y| \leq \delta, 0 \text{ otherwise.} \tag{4}$$

Let us consider two intervals: $E = [m,\ M]$ — that from P —, and $E' = [m',\ M']$ — that from the potential substitute P'. Using the tolerance value δ, one extends interval E into $E'' = [m-\delta, M+\delta] = [m'', M'']$. One has to compute the extent to which E' is close E'', and the expression of the substitutivity measure becomes:

$$sbs_3(P,\ P') = \frac{|(E' - E)\ \cap\ E''|}{|E' - E|} \tag{5}$$

Example 4. Let E be $[10, 15]$, $\delta = 2$, $E' = [7, 12]$ (cf. Fig. 2). We get $E'' = [8, 17]$ and $sbs_3 = \frac{|[7,10[\ \cap\ [8,17]|}{|[7,10[|} = 2/3$.$\diamond$

The case where proximity is defined by means of a fuzzy tolerance indicator M or Z (cf. Section 2) is slightly more complex. In such a situation, interval E is

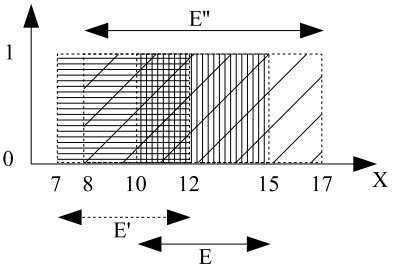


Fig. 2. Representation of intervals E, E' and E''

dilated into a fuzzy set E'' according to the calculus described in [9] and the definition of $sbs_3(P,\ P')$ becomes:

$$sbs_3(P,\ P') = \frac{\int_{x \in (E'-E)} \mu_{E''}(x)}{|E' - E|} \tag{6}$$

Example 5. Let E be $[10, 15]$, and E' be $[7, 12]$ and assume a fuzzy tolerance indicator Z with a triangular membership function of support $[-2, 2]$ (cf. Fig. 3). We get $sbs_3 = 1/3$.$\diamond$

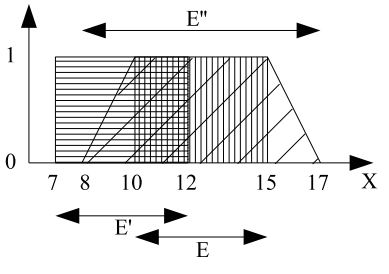


Fig. 3. Representation of intervals E, E' and of the fuzzy interval E''

3.3 Case Where E Is an Interval and E' Is a Set

When P involves an interval E and P' involves a set E', $sbs_3(P,\ P')$ writes:

$$sbs_3(P,\ P') = \frac{\sum_{x' \in E' \wedge x' \notin E} sup_{x \in E}\ prox(x,\ x')}{|\{x \in E' \mid x \notin E\}|}. \tag{7}$$

On the other hand, the dual case (set in P and interval in P') is much more tricky and cannot be captured by Formula (6) defining sbs_3 when the attribute domain is continuous. Consequently, we introduce the constraint that a finite set can only be replaced by another finite set.

4 Query Replacement Strategy

4.1 Position of the Problem

Let Q be a failing user query and let us consider the set S_Q of predicates $\{A_i \in E_i\}$ from Q. We assume a repository D^+ of non-failing queries. Each query Q' of D^+ is associated with the set $S_{Q'}$ of its predicates $\{A_i \in E'_i\}$. The problem is to find the best substitute of Q among all queries of D^+ .

Definition 1 (substitute of a query). *A substitute to a query Q is a query Q' from D^+ such that:*

i) Q' is addressed to the same relation(s) as Q,
ii) Q' shares at least one attribute from its "where" clause with that from Q,
iii) Q' involves at least one predicate $A_i \in E'_i$ which is not subsumed by $A_i \in E_i$.

Item *ii*) of the definition above guarantees a semantic proximity between Q and Q' while item *iii*) is based on the following remark. Even if query Q' involves a predicate which is subsumed by its counterpart in Q, query Q' can be an interesting substitute to Q since the other predicates must also be taken into account. For instance, if query Q = (A in {rabbit, hen} and B in {wheat, cabbage}) returns an empty answer, it is still possible that query Q' = (A in {rabbit} and B in {wheat, oats}) returns a non-empty one whereas the predicate on A in Q' is subsumed by that in Q. However, for a query Q' to be a possible substitute, it is necessary that Q' involves at least one predicate which is not subsumed by the corresponding one in Q (but notice that if it were not the case, the answer to Q' would be empty — since the answer to Q is — and Q' would therefore not be in D^+).

The approach that we propose consists of the following three steps:

1. select the queries from the repository D^+ which satisfy Definition 1, and adapt these queries (see algorithm below),
2. compute the proximity degrees between the queries retained and the user query Q through the measure sbs_3,
3. determine the closest substitute to Q and process it.

4.2 Replacement Mechanism

It is worth noticing that the predicates from Q' which are strictly subsumed by those from Q can be replaced by the latter ones. Indeed, such a predicate from Q (taken individually) could not have been the reason for the query failure. Moreover, for every predicate $A_i \in E_i$ from Q which is not "covered" by Q', i.e., which concerns an attribute on which there is no constraint in Q', one may compute the proximity between P and the entire domain of the attribute considered.

The conjunctive combination of the proximities related to the atomic predicates can be performed by means of a triangular norm (for instance the minimum), so as to obtain the overall proximity between two queries. The idea is to

assess the extent to which every predicate of the substitute query is close to the corresponding predicate from the initial failing query. Notice that alternative solutions could also be possible, for instance one might use a mean operator. The substitution algorithm is outlined hereafter.

Let Q' be a query from D^+ which is addressed to the same relation(s) as Q and shares at least one attribute from its "where" clause with that from Q. The five steps of the algorithm are:

1. replace the "select" clause from Q' by that from Q;
2. remove from Q' every predicate that concerns an attribute absent from the "where" clause from Q;
3. replace every predicate from Q' which is strictly subsumed by the corresponding one from Q by the latter;
4. for the other predicates, compute the proximity between the predicate from Q' and the corresponding one from Q, by means of measure sbs_3, and replace the predicate from Q' by its union with that from Q. As to the predicates from Q which are not covered by Q', one computes their substitutivity degree relatively to the entire domain of the attribute involved;
5. aggregate the local proximities by means of a triangular norm.

4.3 Detailed Example

Let Q be the following failing user query:

```
SELECT #id
FROM Farms
WHERE veg in {corn, rapeseed} AND city in {Lannion, Caouennec, Prat}
      AND area in [60, 100];
```

Let us assume that the domain of "veg" is: {corn, rapeseed, sunflower, wheat, cabbage, broccoli, potato, rutabaga} and that the associated proximity relation is given in Table 2.

Table 2. Proximity relation over attribute *veg*

	co	ra	su	wh	ca	br	po	ru
co	1	0.4	0.3	0.8	0.1	0.1	0.6	0.4
ra	0.4	1	0.9	0.6	0.2	0.2	0.1	0.1
su	0.3	0.9	1	0.5	0.1	0.1	0.3	0.3
wh	0.8	0.6	0.5	1	0.2	0.1	0.5	0.4
ca	0.1	0.2	0.1	0.2	1	0.9	0.6	0.7
br	0.1	0.2	0.1	0.1	0.9	1	0.4	0.6
po	0.6	0.1	0.3	0.5	0.6	0.4	1	0.8
ru	0.4	0.1	0.3	0.4	0.7	0.6	0.8	1

Let Q'_1 be the following query from D^+:

```
SELECT #name
FROM Farms
WHERE veg in {wheat, rapeseed, sunflower} AND city in {Lannion, Prat}
      AND area = 125 AND animal in {cow, pig};
```

The query Q''_1 obtained by adapting Q'_1 according to the algorithm above is:

```
SELECT #id
FROM Farms
WHERE veg in {corn, rapeseed, wheat, sunflower}
      AND city in {Lannion, Caouennec, Prat} AND area in [60, 100];
```

The degree computed by sbs_3 for the substitution of {corn, rapeseed} by {wheat, rapeseed, sunflower} equals:

$$\frac{max(0.8,\ 0.6) + max(0.3,\ 0.9)}{2} = 0.85.$$

Let us assume that the proximity over the areas is based on a fuzzy tolerance indicator Z with a triangular membership function of support $[-50,\ 50]$. The substitution of [60, 100] by 125 is assigned the degree 0.5 (i.e., the proximity degree between 100 and 125, cf formula 7). Finally, the degree computed for Q''_1 using the t-norm minimum is:

$$min(0.85, 0.5) = 0.5.$$

Let us now consider another query, denoted by Q'_2, from D^+:

```
SELECT #name
FROM Farms
WHERE city = Caouennec AND area in [80, 180] AND
      animal in {sheep, goat};
```

Altering Q'_2 according to the algorithm yields Q''_2:

```
SELECT #id
FROM Farms
WHERE city in {Lannion, Caouennec, Prat} AND area in [60, 180];
```

As to the condition on attribute *veg* we get:

$$sbs_3(veg \in \{corn,\ rapeseed\},\ veg \in domain(veg)) \\ = (0.9 + 0.8 + 0.2 + 0.2 + 0.6 + 0.4)/6 = 0.52.$$

As to the condition on attribute *area*, we get (cf. formula 6):

$$sbs3(area \in [60,\ 100],\ area \in [80,\ 180]) \\ = ((150 - 100)/2)/(180 - 100) = 25/80 = 0.31.$$

Thus, the degree attached to Q''_2 is $min(0.52, 0.31) = 0.31$ and Q''_1 is a better substitute to Q than Q''_2.

Remark. In case of ties, one could take into account the cardinality of the result of each candidate query so as to break these ties, provided that these cardinalities are stored in D^+.

5 Implementation and Optimization

We chose to model the cache (as well as the "container" of proximity measures) as a relational database, for the following reasons. First, it provides a logical representation of these structures which is as independent as possible from the application which will handle them. Second, such a modeling makes it possible to take advantage of the indexing structures available in any DBMS to improve the access to the candidate queries. The functional architecture of the system is presented in Figure 4.

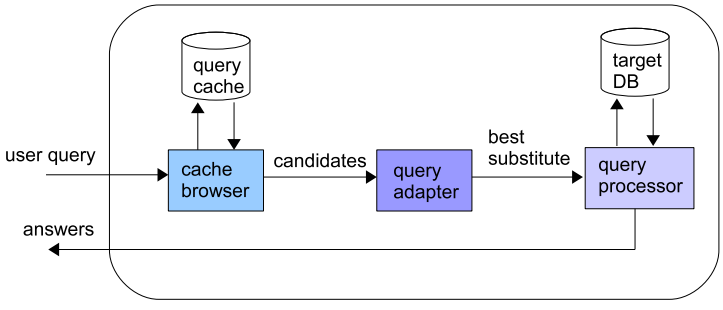


Fig. 4. Functional architecture of the system

The main objective of the experiments reported hereafter was to optimize the access to the cache and the computation of the best substitute query. The experiments were performed using the DBMS Oracle^{TM} 10g release 2 Enterprise Edition, installed on a workstation running Windows XP^{TM} with a processor's frequency of 2.67 GHz and 256 MB of RAM. We used the Oracle^{TM} database sample schema SH [10] which represents information related to sales. It involves, among others, the following relations:

Sales(prod_id, cust_id, time_id, channel_id, promo_id, qty_sold, amount_sold),
Customers(cust_id, cust_first_name, cust_last_name, ..., cust_city, ...), and
Times(time_id, day_name, day_number_in_week, day_number_in_month, ...)

which contain respectively 918,843, 55,500, and 1,826 tuples.

So far, only a few queries have been tested and the experiments still need to be pursued. However, the results obtained already reveal interesting phenomena, as illustrated in the next section where a typical query involving selections and joins is considered.

5.1 Example of a Test Query

The failing query Q_1 detailed hereafter involves three selection predicates and three join conditions; it retrieves the products whose amount sold is between \$700 and \$800, bought by customers living in *Bondy* or *Valbonne*, around the middle of any month. The corresponding SQL expression is:

```
SELECT p.prod_name, t.day_number_in_month, s.amount_sold
FROM times t, sales s, customers c, products p
WHERE t.time_id = s.time_id AND
      s.prod_id = p.prod_id AND
      c.cust_id = s.cust_id AND
      s.amount_sold between 700 and 800 AND
      t.day_number_in_month between 14 and 16 AND
      c.cust_city in ('Bondy','Valbonne');
```

Attribute *cust_city* is a string-type attribute. In this experiment, we only considered French cities (there are 70 in the database). We defined a proximity measure for attribute *cust_city* based on the relative distances between each pair of cities:

$$\forall x \in C,\ \forall y \in C,\ prox(x,\ y) = 1 - \frac{d(x,\ y)}{d_{max}}$$

where $d(x, y)$ gives the distance between x and y, and d_{max} is the longest distance between two cities in C.

Attributes *day_number_in_month* and *amount_sold* are numeric. The domain of the former is [1, 31] whereas for the latter it is [6, 1783]. For these attributes, we used non-fuzzy proximity measures with a tolerance value of 4 and 200, respectively.

In order to feed the cache, we randomly generated queries the following way:

1. the number of predicates was either 1, 2, or 3;
2. the attributes considered for these predicates were: *day_number_in_month*, *amount_sold* and *cust_city*;
3. in the case of numeric attributes, the intervals were chosen randomly in the domain of values;
4. in the case of string-based predicates, both the elements and the number of elements in the set were also chosen randomly.

Of course, only the non-failing queries were inserted into the cache.

5.2 Improving the Performances of the System

The system is composed of PL/SQL procedures and functions which execute SQL queries to retrieve the data from the database. The objective of the experiment is to optimize these queries, which implies choosing an appropriate physical design for the cache. Three versions have been tested.

Version 1. This version corresponds to the system without almost any improvements (it uses the B-tree indexes that the system creates by default for the keys of the relations). The total execution time obtained for query Q_1 equals 412 seconds with a cache containing 10,000 related queries.

Version 2. In order to optimize the access to the proximity measures, we replaced the heap tables *StringResemblance* and *NumericResemblance* by two index-organized tables (IOT) — let us recall that an IOT is a variant of a B-tree.

Unlike an ordinary (heap-organized) table whose data is stored as an unordered collection (heap), data for an index-organized table is stored in a B-tree index structure in a primary-key-sorted manner. Besides storing the primary key values of an index-organized table row, each index entry in the B-tree stores the non-key values as well.

This modification led to a decrease of the execution time from 412 to 385 seconds, with a cache containing 10,000 related queries. The queries which access the proximity tables performed less disk reads, from 51 to 38 blocks.

Version 3. The algorithm was modified so as to load the two proximity tables (for numeric and string attributes) into main memory. By doing so, each time a failing query is processed by the application, the algorithm reads the measures once; then, they are directly accessed from the main memory.

This last modification made it possible to decrease the execution time from 385 to 111 seconds. Using this last physical model, we considered four data loads and we measured execution time. The results are showed in Table 3. We observe that the execution time varies linearly in the number of the candidate queries to the substitution of Q in the cache. The algorithm finds the best substitute in less than two minutes when the cache contains $10,000$ possible candidates. This tends to show that the approach is tractable and scalable. Let us emphasize that the performance measures reported here indicate the number of queries in the cache which "have to do" (in the sense of the approach) with the initial one, and not the overall number of queries in the cache. So, the case "10,000 distinct queries" (and its associated time) is already rather extreme, and the user should not have to wait 2 minutes in general, but much less.

Table 3. Execution time of query Q_1 using the optimized model

# of candidate queries in the cache	processing time (seconds)
100	2.26
1,000	12.62
10,000	110.98
100,000	1,149.97

5.3 Discussion about the Quality of the Replacement Queries

The experiments are still in a preliminary phase. So far, we have tested several queries on several databases, and the replacement queries obtained all seemed relevant, but of course, the "quality" of the best replacement query (i.e., its semantic closeness to the initial query) depends on the number of queries in the cache which "have to do" with that query. Establishing a relation between the size of the cache and the average quality of a replacement query is not easy, since it depends on which queries are in the cache and which is submitted (some statistical knowledge about the queries addressed to a given database is necessary). It is an important objective for further work, though. In any case,

let us recall that every candidate replacement query is associated with a degree which expresses its semantic closeness (in the sense of a fuzzy proximity) to the initial one, so the user is aware of how tolerant he/she has to be if he/she fires one of these replacement queries.

6 Related Work

Some related work can be found in both domains of databases and information retrieval, including web search.

The *semantic caching* approach (see, e.g., [11,12,13,14]) proposes to keep in a cache some previously executed queries along with their results and checks whether the answers to a given user query can be retrieved from the cache for optimization purposes. It uses the notion of query containment (i.e. a query Q is contained in a query G, if all answers to Q are also answers to G) to find the answers in the cache. In a similar spirit, Ghosh *et al.* [15] propose a query clustering approach aimed at optimizing queries by reusing execution plans computed for similar queries.

On the other hand, the approach of *query rewriting using views* aims at finding view-based queries which are equivalent to (or contained in) a given user query. View-based query rewriting was first introduced using materialized views for query optimization purposes [16]. Afterwards, it was brought to the domain of data integration systems [17,18] where the objective is to find *certain answers* to a query in a decentralized database context.

However, none of these approaches deals with the empty answer problem: they are concerned either in optimizing the access to information or in computing the set of certain answers to a query from distributed data sources. In our case, given a failing query, we are not interested in finding neither contained nor equivalent queries, since those would produce empty answers too.

In the *information retrieval* domain, other techniques are based on similarity measures and make use of previous executed queries to improve web search (see, e.g., [19,20]). The measures underlying these approaches are strongly based on relationships between keywords, whereas we deal with a more general type of conditions than those expressed by a set of keywords, namely value constraints.

There are also relevant work in the domain of case-based reasoning, which integrates past user experience to solve current queries. In particular, Fouqué *et al.*[21] propose to include additional information in the answer to a user query, using an approach that they call *associative query answering*. The basic idea is to extend the "select" clause of a user query with attributes which appeared in similar queries previously submitted by other users. As we do, the authors use a nearness measure between every pair of values of each attribute, which is used to evaluate similarity between the user query and those previously executed. However, that work does not deal with failing queries. A CBR approach in a social web search context is also proposed in [22], but this approach considers keyword queries only, as the IR methods mentioned above.

Bidault *et al.* [23] tackle the empty answer issue and use a repository of predefined queries in the context of mediation systems. Although their approach

shows some similarity with ours, their solution is still quite different. They build a set of predefined successful queries for each source to offer a substitute for a failing user query. Similarity degrees between the initial user query and predefined successful queries are computed on the basis of a *hierarchy of concepts*. Finally, some heuristics make it possible to select the "best" substitute to the initial user query. In this approach, the similarity degree concerns the extent to which two concepts share the same characteristics whereas the substitutivity measure we defined is based on the proximity between the domain values of the attributes involved in the queries.

7 Conclusion

In this paper, we have outlined an approach aimed at obviating empty answers to queries involving value constraints. The method proposed uses a query repository and is based on the adaptation of the past non-failing query which is *the most similar* (in the sense of some semantic proximity relations defined on the attribute domains) to the user query considered.

The perspectives for future work are manifold. First, it would be worth investigating the possibility of reusing the execution plans of past queries in order to optimize the evaluation of the selected substitute query, in a spirit similar to [15]. Another point worthy to study concerns the substitutivity measure which is at the heart of the approach. The measure advocated here should be compared with some others adapted from classical similarity measures, cf. [24,25]. Finally, we intend to generalize this approach to a broader class of queries, including for example approximate join conditions.

References

1. Gaasterland, T., Godfrey, P., Minker, J.: Relaxation as a platform for cooperative answering. Journal of Intelligent Information Systems 1, 293–321 (1992)
2. Godfrey, P.: Minimization in cooperative response to failing database queries. Int. Journal of Cooperative Information Systems 6(2), 95–149 (1997)
3. Chu, W., Yang, H., Chiang, K., Minock, M., Chow, G., Larson, C.: Cobase: A scalable and extensible cooperative information system. Journal of Intelligent Information Systems 6(2-3), 223–259 (1996)
4. Gaasterland, T.: Cooperative answering through controlled query relaxation. IEEE Expert 12, 48–59 (1997)
5. Huh, S., Moon, K., Lee, H.: A data abstraction approach for query relaxation. Information and Software Technology 42, 407–418 (2000)
6. Motro, A.: SEAVE: A mechanism for verifying user presuppositions in query systems. ACM Trans. on Off. Inf. Syst. 4(4), 312–330 (1986)
7. Muslea, I.: Machine learning for online query relaxation. In: Proc. of the Int. Conf. of Knowledge and Discovery and Data mining (KDD 2004), pp. 246–255 (2004)
8. Zadeh, L.: Fuzzy sets. Information and Control 8, 338–353 (1965)
9. Bosc, P., Hadjali, A., Pivert, O.: Towards a tolerance-based technique for cooperative answering of fuzzy queries against regular databases. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 256–273. Springer, Heidelberg (2005)

10. Oracle Corporation: Oracle database sample schemas 10g release 2 (10.2), http://www.oracle.com/pls/db102/homepage
11. Arens, Y., Knoblock, C.: Intelligent caching: selecting, representing and reusing data in an information server. In: Proc. of CIKM, pp. 433–438 (1994)
12. Chidlovskii, B., Borghoff, U.: Semantic caching of web queries. VLDB Journal 9, 2–17 (2000)
13. Godfrey, P., Gryz, J.: Answering queries by semantic caches. In: Bench-Capon, T.J.M., Soda, G., Tjoa, A.M. (eds.) DEXA 1999. LNCS, vol. 1677, pp. 485–498. Springer, Heidelberg (1999)
14. Miranker, D., Taylor, M., Padmanaban, A.: A tractable query cache by approximation. In: Proc. of SARA 2002, pp. 140–151 (2002)
15. Ghosh, A., Parikh, J., Sengar, V., Haritsa, J.: Plan selection based on query clustering. In: Proc. of VLDB 2002, pp. 179–190 (2002)
16. Chaudhuri, S., Krishnamurthy, R., Potamianos, S., Shim, K.: Optimizing queries with materialized views. In: Proc. of ICDE 1995, pp. 190–200 (1995)
17. Beeri, C., Halevy, A., Rousset, M.C.: Rewriting queries using views in description logics. In: Proc. of PODS 1997, pp. 99–108 (1997)
18. Jaudoin, H., Petit, J.M., Rey, C., Schneider, M., Toumani, F.: Query rewriting using views in presence of value constraints. In: Proc. of the Int. Workshop on Description Logics (DL 2005), pp. 250–262 (2005)
19. Raghavan, V., Sever, H.: On the reuse of past optimal queries. In: Proc. of ACM SIGIR, pp. 344–350 (1995)
20. Wen, J., Nie, J.Y., Zhang, H.J.: Query clustering using user logs. ACM Transactions on Information Systems 20(1), 59–81 (2002)
21. Fouqué, G., Chu, W., Yau, H.: A case-based reasoning approach for associative query answering. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1994. LNCS, vol. 869, pp. 183–192. Springer, Heidelberg (1994)
22. Smyth, B., Briggs, P., Coyle, M., O'Mahony, M.P.: A case-based perspective on social web search. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 494–508. Springer, Heidelberg (2009)
23. Bidault, A., Froidevaux, C., Safar, B.: Similarity between queries in a mediator. In: Proc. of ECAI 2002, pp. 235–239 (2002)
24. Cross, V., Sudkamp, T.: Similarity and Compatibility in Fuzzy Set Theory – Assessment and Applications. Physica-Verlag, Heidelberg (2002)
25. Balfe, E., Smyth, B.: A comparative analysis of query similarity metrics for community-based web search. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 63–77. Springer, Heidelberg (2005)

Case Acquisition from Text: Ontology-Based Information Extraction with SCOOBIE for myCBR

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

[1] Knowledge Management Department,
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Straße 122, 67663 Kaiserslautern, Germany
[2] Knowledge-Based Systems Group, Department of Computer Science,
University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern
{firstname.lastname}@dfki.de

Abstract. *myCBR* is a freely available tool for rapid prototyping of similarity-based retrieval applications such as case-based product recommender systems. It provides easy-to-use model generation, data import, similarity modelling, explanation, and testing functionality together with comfortable graphical user interfaces. SCOOBIE is an ontology-based information extraction system, which uses symbolic background knowledge for extracting information from text. Extraction results depend on existing knowledge fragments. In this paper we show how to use SCOOBIE for generating cases from texts. More concrete we use ontologies of the Web of Data, published as so called Linked Data interlinked with *myCBR*'s case model. We present a way of formalising a case model as Linked Data ready ontology and connect it with other ontologies of the Web of Data in order to get richer cases.

Keywords: Textual Case-Based Reasoning, Ontology-based Information Extraction, Linked Open Data, Web of Data.

1 Introduction

Structural Case-Based Reasoning relies on cases described by attributes and corresponding values. Structural CBR systems organise attributes in various ways, e.g., as flat attribute lists or in an object-oriented way. This organisation is often called domain ontology. The structural approach is useful in domains where additional knowledge such as complex similarity measures must be used in order to produce good results, and where a case model is easy to acquire.

myCBR[1] [22] is an Open Source (Structural) CBR tool developed at the German Research Center for Artificial Intelligence (DFKI). Key motivation for implementing *myCBR* was the need for a compact and easy-to-use tool for rapidly building prototype CBR applications in teaching, research, and small industrial

[1] http://www.mycbr-project.net

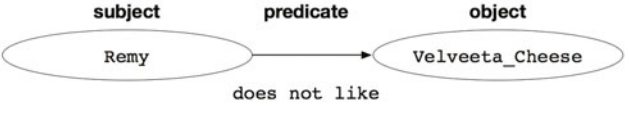
I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 451–464, 2010.

projects with low effort. Moreover, the tool is easily extendable in order to facilitate the experimental evaluation of novel algorithms and research results.

In domains where large collections of semi-structured documents are readily available a different CBR approach is used: Textual CBR (TCBR). TCBR systems aim at managing information contained in semi-structured documents and providing means for content-oriented retrieval. The approach is especially useful in domains where the intended user is able to immediately make use of the knowledge contained in the respective documents.

There is no standard or consensus about the structure of a textual CBR system. One way of employing TCBR is to determine the structure of cases by a case model and use a pre-processor to fill case templates from semi-structured texts, thus providing a structured representation of the documents. The quality of the cases obviously depends on the abilities of the pre-processor.

In this paper, we describe how to combine the ontology-based information extraction tool SCOOBIE [3] as such a pre-processor with *myCBR*, where an ontology is an explicit, formal specification of a conceptualisation, i.e. a particular vocabulary that can be employed for describing aspects of real domains [16]. SCOOBIE strongly depends on the existence of an ontology to provide formal extraction results. In the presented approach, one such ontology is the case model of *myCBR*. As standard ontology representation formalism we chose to express the case model in the Resource Description Framework language (RDF) [23] where meaning is expressed by facts encoded in sets of triples [7].




```
<rdf:Description
  rdf:about="http://dbtropes.org/resource/Main/Ratatouille#Remy">
  <does-not-like
    rdf:resource=
      "http://mycbr-project.net/models/Recipe#velveeta_cheese"/>
</rdf:Description>
```

Fig. 1. RDF represents information as graph in form of triples. Triples consist of subject, predicate and object values. RDF can be serialised in XML.

Triples are like elementary sentences composed of subject, predicate, and object. Subjects, predicates, and objects are given as names for entities, also called resources or nodes. Entities represent something like a contact, an appointment, a website, etc. Names are either literals or Uniform Resource Identifiers (URI) [8], which are global in scope, always referring to the same entity in any RDF document in which they appear. Fig. 1 shows an example of an RDF triple describing which cheese Remy does not like.

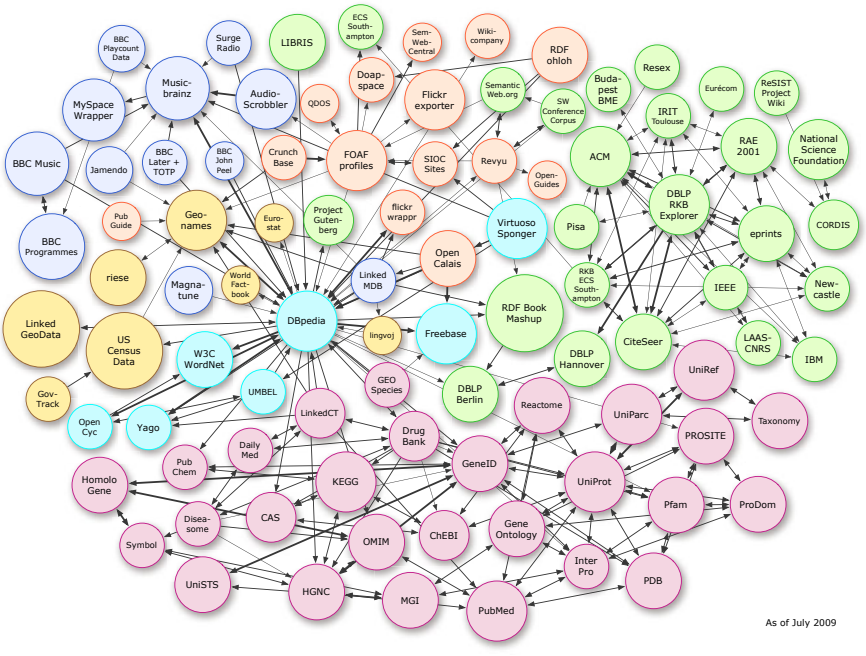


Fig. 2. Existing linked open data sets (as of July 2009) [9]

Representing a case model in RDF results in URIs for each attribute and 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 (Fig. 2)[2] or the Web of Data [9].

We show that case models published as Linked Open Data can be interconnected with information from various data sources. Thus, they can be enriched with additional knowledge from other ontologies. Combining SCOOBIE and *myCBR* allows leveraging available knowledge of the Web of Data for acquiring cases from texts.

The rest of the paper is structured as follows: The next section provides an overview of previous and related work regarding ontology-based information extraction and Textual CBR. Section 3 briefly recapitulates relevant features of *myCBR*. In Sect. 4 we describe the ontology-based information extraction tool SCOOBIE before we put SCOOBIE to use for *myCBR* (Sect. 5). Finally, Sect. 6 summarises the combination of the two tools and concludes the paper with an outlook on further work.

[2] Graphic source URL of the LOD cloud: `http://www4.wiwiss.fu-berlin.de/bizer/pub/lod-datasets_2009-07-14_colored.png`

2 Related Work

In their survey article on Textual CBR, Weber et al. [24] formulate four research questions that TCBR addresses. TCBR looks at how to *assess similarity between textually represented cases*, *map from texts to structured case representations*, *adapt textual cases*, and *automatically generate representations for TCBR*.

In this classification schema our approach clearly addresses the second question, of how to map texts to structured case representations and work along the line of Lenz's framework, which targets semi-structured documents [18]. In this our approach is similar to what empolis' information access suite [14] and jColibri[3] [6,19] are offering: intelligent access to all sorts of documents.

In [17], Lenz describes a layered model that divides text processing into several stages, i.e., Keyword Layer, Phrase Layer, Thesaurus Layer, Glossary Layer, Feature Value Layer, Domain Structure Layer, and Information Extraction Layer. SCOOBIE is built upon a standard information extraction pipeline architecture [2]. In general, Lenz' layered model of textual CBR subsumes SCOOBIE's pipeline architecture, moreover SCOOBIE can be incorporated into parts of the layer model, i.e., Keyword Layer, Phrase Layer, Thesaurus Layer, and the Information Extraction Layer.

SCOOBIE is based on information extraction (IE) principles that are well described in [5]. Comparable and approved OBIE systems are the General Architecture for Text Engineering (GATE)[4] [10] and the SmartWeb Ontology-Based Annotation component (SOBA) [12]. In contrast to these, SCOOBIE does not use the ontology as input gazetteer, i.e., a plain list of relevant labels, but as a model for semantic analyses such as instance disambiguation and discourse analysis. This technique of using existing domain ontologies as input for information extraction tasks and extraction results for ontology population and therefore knowledge acquisition was presented already in [20].

SCOOBIE also supports the population of extraction templates. The concept of querying ontologies with IE templates in order to extract information from text is completely missing in existing OBIE systems. Other OBIE approaches can be found in the proceedings of the OBIES workshop 2008 [4].

3 Similarity-Based Retrieval with myCBR

myCBR, in its current version[5], focuses on the similarity-based retrieval step of the CBR cycle [1], as this is still the core functionality of most CBR applications. A popular example of such retrieval-only systems are case-based product recommender systems [11]. While the first CBR systems were often based on simple distance metrics, today many CBR applications make use of highly sophisticated, knowledge-intensive similarity measures [21]. Such extremely

[3] `http://gaia.fdi.ucm.es/projects/jcolibri`
[4] `http://gate.ac.uk`
[5] *myCBR*, Version 2.6

domain specific similarity measures enable the improvement of the retrieval quality substantially. However, they increase the development effort significantly.

myCBR is a plug-in for the Java-based Open Source ontology editor Protégé[6] [15], which was chosen as the modelling platform for *myCBR* and which existing functionality for creating case models and instances was extended for modelling similarity measures and testing similarity-based retrieval.

The major goal of *myCBR* was to minimise the effort for building CBR applications that require knowledge-intensive similarity measures. Therefore, it provides comfortable graphical user interfaces for modelling various kinds of attribute-specific similarity measures and for evaluating the resulting retrieval quality. In order to reduce also the effort of the preceding step of defining an appropriate case representation, it includes tools for generating the case representation automatically from existing raw data.

myCBR is intended for structural CBR applications that make use of rich attribute-value based or object-oriented case representations. Although Protégé provides powerful graphical user interfaces for modelling attribute-value based and object-oriented representations, their manual definition remains a laborous task. It includes the definition of classes and attributes and the specification of accurate value ranges required for a meaningful similarity assessment.

3.1 CSV Import

In order to ease the definition of case representations, *myCBR* provides a powerful CSV[7] data import module. CSV files are widely used to store attribute-value based raw data in pure ASCII format. For example, in the Machine Learning community example data sets are usually exchanged by using CSV files[8]. Using the CSV importer, the user has the choice to import data instances into an existing Protégé data model, or to create a new model automatically based on the raw data. In the latter case, *myCBR* generates a Protégé slot[9] for each data column of the CSV file automatically. After the CSV data has been imported, the user may further modify the generated case model (e.g., extend it to an object-oriented representation) to meet the application specific needs. The final case model together with the case base is stored by *myCBR* in XML files.

3.2 Cases from Texts

For retrieving texts with *myCBR* a similar approach to CSV import is used. But where the entries in the CSV file are used at face value without interpreting them[10], texts as raw material for cases need to be searched for the occurrence of certain concepts and phrases, i.e., attribute values.

[6] http://protege.stanford.edu/
[7] Comma Separated Values
[8] See, for example, `http://archive.ics.uci.edu/ml/`
[9] In Protégé, attributes are called *slots*.
[10] The importer analyses each column to determine the value ranges for slots. For text data, symbol or string type slots are created depending on a user defined threshold.

Concepts may be expressed in different ways. They may vary grammatically (singular, plural, case) or semantically (synonyms, hyponyms, hypernyms). The different representations can be seen as triggers for concepts. Concepts can also be measured values, i.e., numbers with a unit.

Another difference to the CSV import is that a case model cannot be derived automatically from texts. At least a basic case model is required to start building cases from texts. Section 5 describes the process in detail. First we will have a closer look at SCOOBIE.

4 Ontology-Based Information Extraction with SCOOBIE

Ontology-based information extraction (OBIE) extracts formal facts from text resources. In terms of RDF, facts may be triples representing attribute knowledge about a resource (e.g., *:Cheddar_ Cheese skos:prefLabel "Cheddar Cheese"*)[11] or relations between resources (e.g., *:Remy :does-not-like :velveeta_ cheese*). OBIE algorithms incorporate those relevant bits of knowledge from an input ontology that support information extraction inside a pipeline of cascading extraction tasks [2]. Conceiving the extraction pipeline of the SCOOBIE system [3] as black box, mandatory input parameters are:

1. The *ontology*, which comprises vocabularies and schemes, used to represent entities of the SCOOBIE domain model. The classes (e.g., ingredients, or cooking steps), datatype properties of these classes (e.g., ingredient label or recipe name) and object properties between instances of these classes (e.g., persons not liking ingredients) define a search space of possible instances and facts that may be extracted from text.
2. *Entities* of SCOOBIE's domain model, which are represented as instances of the ontology. The given datatype property values of these instances (e.g., the symbolic slot value of a recipe's ingredient called *"cheese"*) are used for extracting instances from text. Object properties between instances are used for disambiguating instances with similar datatype property values or ranking the relevance of extracted instances.

As shown in Fig. 3, the ontology and its instances are analysed during an offline pre-processing and training phase. Results are index structures (e.g., suffix arrays, B*-trees) and learning models (e.g., Conditional Random Fields, K-Nearest Neighbour Classifiers) that can now be used by efficient extraction tasks inside the extraction pipeline:

1. *Normalisation*: Extracts document metadata and plain text data from textual or binary file formats. The language of the plain text is detected by applying statistics about n-gram distributions of letters in different languages.

[11] In order to preserve readable inline examples we prefered writing RDF triples in Turtle syntax (please refer to `http://www.w3.org/2007/02/turtle/primer/`).

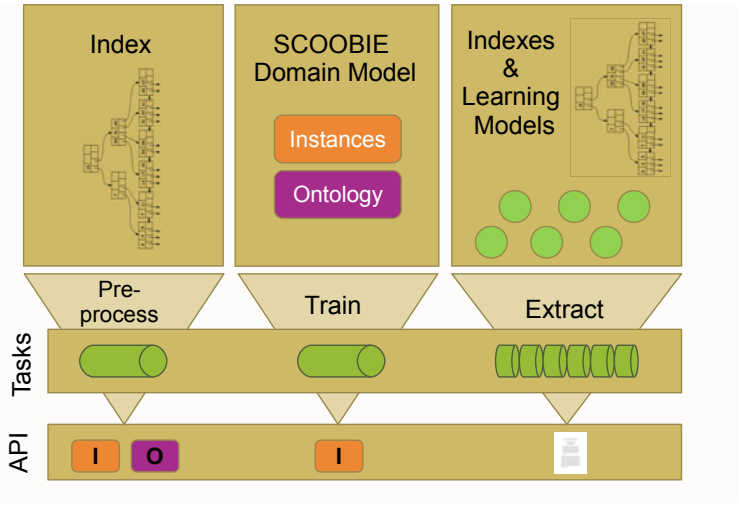


Fig. 3. Architecture of the ontology-based information extraction system SCOOBIE

2. *Segmentation*: Partitions plain text into segments: paragraphs, sentences, and tokens. With respect to the detected language, a POS tagger tags each token with its part of speech (POS).
3. *Symbolisation*: Recognises datatype property values in text by matching phrases in text and values of datatype properties of the domain model.
   For example, assume having the triple *:Cheddar_ Cheese skos:prefLabel "Cheddar Cheese"*, then *"Cheddar Cheese"* may be recognised as content symbol.
   By performing noun phrase chunking, noun phrases expressing candidates for names without any structure in syntax (e.g., names) are detected.
4. *Instantiation*: For each recognised datatype property value the Instantiation resolves instance candidates.
   For example, assume having the triple *:Cheddar_ Cheese skos:prefLabel "Cheddar Cheese"*, then *"Cheddar Cheese"* is resolved as *skos:prefLabel* of instance *:Cheddar_ Cheese*. An instance candidate recognition resolves possible candidates for recognised datatype property values. Here, ambiguities may occur if more than one instance possesses the same datatype property values (e.g., determining wether "onions" symbolise "red onions" or "green onions"). Candidates are disambiguated by counting resolved instances in the domain model that are related directly with an object property or indirectly via another instance of the domain model. As result, the ambiguous instance with a higher count of related and recognised instances is taken.
5. *Contextualisation*: Extracts facts (RDF triples) about resolved instances. At first, a fact candidate extraction computes all possible facts between resolved instances. Then, a set of fact selectors rates these facts according to heuristics. A known fact selector heightens rates of extracted facts that exist as triples inside the domain model.

6. *Population*: Creates scenario graphs in RDF format. They contain extracted values, i.e., HTTP URIs of resolved instances with those datatype property values that match with text sequences and RDF triples about object properties between these resolved instances. Scenario graphs can be filtered and ordered by confidence values in range between zero and one.

5 Generating Structural Cases from Texts

Using SCOOBIE as pre-processor for *myCBR* comprises four steps as illustrated by Fig. 4: Based on the *case model* the *information extraction model* is generated (1). This is then fed into SCOOBIE *for* myCBR together with *linked open data* and the text files from which the cases are to be extracted (2). The extracted concepts (expressed in RDF format) are then combined into cases (3), which can eventually be used for similarity-based retrieval (4). We will detail and exemplify each step in the following by providing our fictive user Remy with a similarity-based search engine for recipes.

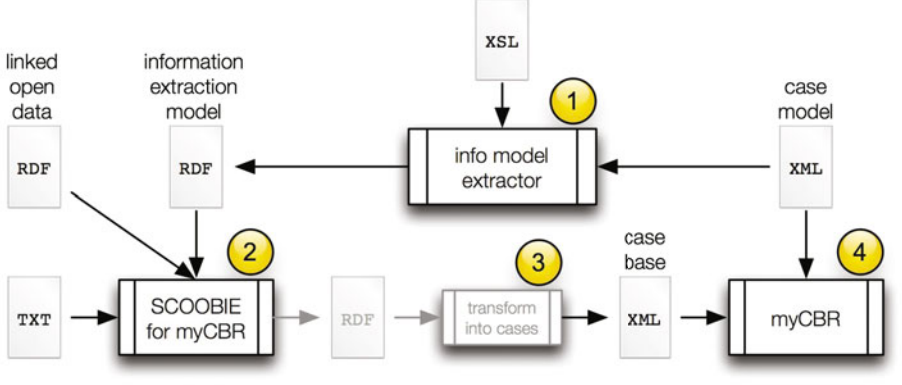


Fig. 4. Process overview: generate information extraction model from case model (1), extract case information from texts (2), transform result into cases (3), and use generated cases for retrieval (4)

5.1 Generating the Information Extraction Model

Cases in *myCBR* are class instances modelled with Protégé. In this first version only flat attribute-value lists are considered, but the approach can be easily extended to more complex class structures by adapting the application logic in SCOOBIE for *myCBR* accordingly.

Our guiding example is recipe search. Remy likes to cook but sometimes forgets to buy ingredients and needs to deal with what is left in the fridge and in his storage room. Remy needs a system for retrieving recipes that he can cook

with the ingredients at hand. As recipe collection we used recipes of the Second Computer Cooking Contest[12].

For the similarity-based search we modelled a class *Recipe* with such attributes as *title, ingredients_ meat, ingredients_ vegetables, ingredients_ fish, ingredients_ pasta, ingredients_ cheese, ingredients_ spices, preparation_ steps*. The title is not used for retrieval, but for identifying the recipe. All other attributes are of type *Symbol* and can hold multiple values. For example, allowed values for *ingredients_ meat* are *chicken, pork,* and *bacon* which can occur together in one recipe. *myCBR* provides various options to configure similarity measures for set type attributes [22]. Depending on the chosen settings, the mapping between query values and case values is calculated differently. For example, a set of values may have an "and" or an "or" semantic. The size of the query and case sets also may have different impact on the similarity.

The *information model extractor* (circle (1) in Fig. 4) takes the case model and transforms it into an information extraction model with the help of XSL transformations[13]. The Extensible Stylesheet Language (XSL)[14] is used for describing how to transform and render XML documents.

```
<SMFunction smfname="default" model_instname="ingredients_cheese"
    type="Symbol" active="true" simMode="Table">
  <QuerySymbol symbol="blue_cheese">
    <CBSymbol sim="1.0" symbol="blue_cheese" />
  </QuerySymbol>
  <QuerySymbol symbol="velveeta_cheese">
    <CBSymbol sim="1.0" symbol="velveeta_cheese" />
  </QuerySymbol>[...]
</SMFunction>
```

Fig. 5. Section of *myCBR* case model

Figure 5 shows a few lines of the case model.[15] Two of the allowed values for the attribute *ingredients_ cheese* are shown: *blue_ cheese* and *velveeta_ cheese*.

SCOOBIE needs its input ontologies represented in RDF format. For transforming the case model into RDF, using XSL was an obvious choice. We use SKOS, the Simple Knowledge Organisation Systems family of formal languages as representation formalism. SKOS is designed for exactly our purpose of representing a structured controlled vocabulary.[16] SKOS is built upon RDF. Its main objective is to enable easy publication of controlled structured vocabularies for the Semantic Web.

[12] CCC at ICCBR 2009: `http://www.wi2.uni-trier.de/ccc09/index.php`

[13] `http://www.w3.org/TR/xslt`

[14] `http://www.w3.org/Style/XSL/`

[15] Actually, the XML snippet is part of the similarity measure functions file, which duplicates the information contained in the proprietary, LISP/Protégé file format.

[16] `http://www.w3.org/TR/2009/REC-skos-reference-20090818/`

```
<xsl:for-each select="QuerySymbol">
  &lt;skos:Concept
      rdf:about="<xsl:value-of
        select='concat("http://mycbr-project.net/models/Recipe#",
          @symbol)'/>"&gt;
    &lt;skos:prefLabel&gt;
      <xsl:value-of select='@symbol'/>
    &lt;/skos:prefLabel&gt;
    &lt;rdf:type
      rdf:resource=
        &quot;<xsl:copy-of select="$slot_name_for_type"/>&quot;/&gt;
  &lt;/skos:Concept&gt;
</xsl:for-each>
```

Fig. 6. Section of XSL transformations

Figure 6 shows the main section of the XSL file. The XSL transformations are nearly domain independent. Following the Linked Open Data principles with HTTP accessable URIs the RDF ontology file needs to exist at the given URL (e.g., `http://mycbr-project.net/models/Recipe`) and needs to be provided in the XSL stylesheet. And there is the basic requirement that just one class exists (which, in turn, implies that the case model is flat). Then, for each *QuerySymbol*, i.e., allowed value, one SKOS concept is created.

```
<skos:Concept
    rdf:about="http://mycbr-project.net/models/Recipe#blue_cheese">
  <skos:prefLabel>blue cheese</skos:prefLabel>
  <rdf:type rdf:resource="ingredients_cheese"/>
</skos:Concept>
```

Fig. 7. Section of transformation result: initial information extraction model

Each attribute value (of type Symbol) becomes a SKOS concept. In order to keep the relation between attribute and value the attribute name is treated as a SKOS concept's RDF type. The allowed value becomes the preferred SKOS label (and is used for the information extraction task in the next step). Figure 7 shows the result of the transformation process. The concept with URI `http://mycbr-project.net/models/Recipe#blue_cheese` is of RDF type *ingredients_ cheese* and has the preferred label *blue cheese*.

5.2 Extract Case Information from Texts

The resulting information extraction model is the core ontology. It is used to extract concepts from input texts, but it would perform very poorly if used alone. SCOOBIE would only be able to find exact matches to preferred labels. In order to find more concepts additional ontologies are needed.

```
<http://mycbr-project.net/models/Recipe#red_pepper">
  owl:sameas <http://dbpedia.org/resource/Cayenne_pepper>
<http://mycbr-project.net/models/Recipe#blue_cheese">
  owl:sameas <http://dbpedia.org/resource/Blue_cheese>
```

Fig. 8. Example expressions connecting the case model to DBpedia

Further intelligence is brought into the system by, first, extending the number of labels (alternative labels), which act as triggers for the associated symbols and, second, by linking SKOS concepts, and thus symbols, to Linked Open Data (e.g., DBpedia[17]). This last step then enhances the case generation step drastically. Figure 8 shows a sample snippet of the respective expressions.

SCOOBIE then extracts attribute values according to the information extraction model and all additional ontologies linked to the information extraction model. A configuration model determines which portion of the texts are analysed into which attribute.

```
<RECIPE>
  <TI>"Blue" Fettuccine</TI>
  <IN>4 oz Danish blue cheese or 8 oz. Danish blue
    Castello cheese, chilled</IN>
  <IN>1/4 c Marinated, dried tomatoes</IN>
  <IN>8 oz Green fettuccine or spinach egg noodles</IN> [...]
  <IN>1/4 c Chopped fresh parsley</IN>
  <PR>[...]
    <STEP>Meanwhile, in small skillet over medium heat, heat
    reserved oil; add shallots and garlic. Saute until shallots
    are limp but not brown.</STEP>
    <STEP>Add wine, basil and reserved tomatoes. Heat through
    and keep hot.</STEP> [...]
    </PR>
</RECIPE>
```

Fig. 9. Example recipe text

The recipes of the Computer Cooking Contest are given in XML format. Figure 9 shows a section of a recipe. Each recipe is divided in title <TI>...</TI>, ingredients <IN>...</IN> and preparation steps <STEP>...</STEP>. The title is just copied into the attribute *title* and used as an identifier for a case. The ingredients are analysed into the attributes *title, ingredients_meat, ingredients_vegetables, ingredients_fish, ingredients_pasta, ingredients_cheese, ingredients_spices*. The preparation steps go into the attribute *preparation_steps*. All attributes are multi-valued, as already said above. The order of steps cannot be taken into account. This is due to the set semantics for multi-valued attributes.

[17] "DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web." `http://dbpedia.org/About`

5.3 Building the Case Base

The resulting RDF extracts are transformed into *myCBR*'s case base format (illustrated in Fig. 10). An instance is created, and the title is copied from the recipe text into the respective slot.

```
<Instance model_instname="recipes_Class10000">
  <slotvalue slot="title" value="&quot;blue&quot; fettuccine" />
  <slotvalue slot="ingredients_cheese" value="blue_cheese" />
  <slotvalue slot="ingredients_pasta" value="egg_noodles" />
  <slotvalue slot="ingredients_pasta" value="noodles" />
  <slotvalue slot="ingredients_pasta" value="spinach_noodles" />
  <slotvalue slot="preparation_steps" value="heat" />
  <slotvalue slot="preparation_steps" value="saute" /> [...]
</Instance>
```

Fig. 10. Snippet of an example (recipe) case

From each extracted SKOS concept the slot name, encoded in each of the concept's RDF type, is taken (e.g., *ingredients_ cheese* in Fig. 7) and filled with the respective concept's name (e.g., *blue_ cheese* from the concept's URI *http://mycbr-project.net/models/Recipe#blue_ cheese*).

Finally, the newly generated case base can be used for searching recipes that match a given set of available ingredients and preferred way of preparation. Using the title of the case the original recipe text can be retrieved for Remy.

6 Conclusion and Outlook

Weber et al. [13] describe the combination of knowledge elicitation for a case model and information extraction techniques for case generation in a knowledge management scenario. SCOOBIE and *myCBR* very well fit into this scenario.

In this paper we described how to combine the ontology-based information extraction tool SCOOBIE with the similarity-based, structural CBR system *myCBR*. Ontology-based information extraction systems strongly depend on the existence of symbolic background knowledge for generating relevant results. Such knowledge is available in *myCBR* and can be used as seeding knowledge for feeding the powerful information extraction system SCOOBIE. Linking this core knowledge to available linked open data on the Web of Data provides new opportunities for knowledge modelling and retrieval.

A case model usually is constructed top down after analysing a domain and the respective documents. From the case model we automatically generated an information extraction model consisting of SKOS concepts. For each allowed value of symbol type attributes we constructed a unique SKOS concept and used the allowed value as preferred label. This IE model already allows for simple text extraction. Linking the IE model to the Web of Data makes the difference.

As we have shown, SCOOBIE has a lot more to offer (e.g., extracting facts, or even new instances), but the current implementation only uses a fraction of SCOOBIE's abilities. A GUI, based on the OSGi[18] platform, is under development to ease the use of developing the case model in combination with case generation from texts. This close interaction will, on one hand, help to extend the case model with symbols previously not recognised (i.e., extend the range of allowed values), and, on the other hand, help to extend the information extraction model by linking allowed values to concepts of other ontologies on the Web of Data.

myCBR is an ongoing project. We encourage others to try out *myCBR* in their own research and teaching projects and to contribute to the further development by implementing their own extensions and experimental modules.

Acknowledgments

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

References

1. Aamodt, A.: Explanation-driven case-based reasoning. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) EWCBR 1993. LNCS, vol. 837, Springer, Heidelberg (1994)
2. Adrian, B., Dengel, A.: Believing finite-state cascades in knowledge-based information extraction. In: Dengel, A.R., Berns, K., Breuel, T.M., Bomarius, F., Roth-Berghofer, T.R. (eds.) KI 2008. LNCS (LNAI), vol. 5243, pp. 152–159. Springer, Heidelberg (2008)
3. 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. LNCS (LNAI), vol. 5803, pp. 249–256. Springer, Heidelberg (2009)
4. Adrian, B., Neumann, G., Troussov, A., Popov, B. (eds.): Ontology-based Information Extraction Systems, OBIES 2008 (2008), http://CEUR-WS.org/Vol-400/
5. Appelt, D., Israel, D.: Introduction to information extraction technology: A tutorial prepared for ijcai-99. SRI International (1999)
6. Bello-Tomás, J., González-Calero, P.A., Díaz-Agudo, B.: JColibri: An Object-Oriented Framework for Building CBR Systems. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 32–46. Springer, Heidelberg (2004)
7. 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)
8. Berners-Lee, T., Fielding, R.T., Masinter, L.: Rfc 3986: Uniform resource identifier (uri): Generic syntax (2005), http://www.ietf.org/rfc/rfc3986.txt (Last access: 2010-02-26)
9. 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)

[18] http://en.wikipedia.org/wiki/OSGi: [Last access: 2010-05-03]

10. Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H.: Evolving GATE to meet new challenges in language engineering. Natural Language Engineering 10(3-4), 349–373 (2004), http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=252241&fulltextType=RA&fileId=S1351324904003468 (Last access: 2010-02-26)
11. Bridge, D., Göker, M.H., McGinty, L., Smyth, B.: Case-based recommender systems. Knowledge Engineering Review 20(3) (2006)
12. Buitelaar, P., Cimiano, P., Racioppa, S., Siegel, M.: Ontology-based Information Extraction with SOBA. In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), pp. 2321–2324. ELRA (2006), http://www.aifb.uni-karlsruhe.de/WBS/pci/Publications/buitelaaretal_lrec06.pdf (Last access: 2010-02-26)
13. David, R.W., Aha, D.W., Sandhu, N., Munoz-Avila, H.: A textual case-based reasoning framework for knowledge management applications. In: Schnurr, H.P., Staab, S., Studer, R., Stumme, G., Sure, Y. (eds.) Proceedings of the Ninth German Workshop on Case-Based Reasoning, pp. 244–253. Shaker Verlag, Aachen (2001)
14. empolis: Information Access Suite — e:IAS Text Mining Engine. Technical white paper (2008), empolis IAS 6.2 (Publication Date: 8 December 2008) Build: 5976
15. Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., Tu, S.W.: The evolution of Protégé an environment for knowledge-based systems development. Int. J. Hum.-Comput. Stud. 58(1), 89–123 (2003)
16. Gruber, T.R.: Toward principles of the design of ontologies used for knowledge sharing. International Journal of Human and Computer Studies 43, 907–928 (1995)
17. Lenz, M.: Defining knowledge layers for textual case-based reasoning. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 298–309. Springer, Heidelberg (1998)
18. Lenz, M.: Knowledge sources for textual CBR applications. In: Lenz, M., Ashley, K. (eds.) Textual Case-Based Reasoning: Papers from the AAAI 1998 Workshop, pp. 24–29. AAAI Press, Menlo Park (1998); Technical Report WS-98-12
19. Recio-García, J.A., Díaz-Agudo, B., Gómez-Martín, M.A., Wiratunga, N.: Extending jcolibri for textual CBR. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 421–435. Springer, Heidelberg (2005), http://dblp.uni-trier.de/db/conf/iccbr/iccbr2005.html#RecioDGW05
20. Sintek, M., Junker, M., Elst, L.V., Abecker, A.: Using information extraction rules for extending domain ontologies. In: Maedche, A., Staab, S., Nedellec, C., Hovy, E. (eds.) Position Statement for the IJCAI 2001 Workshop on Ontology Learning (2001), http://CEUR-WS.org/Vol-38/
21. Stahl, A.: Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning. Ph.D. thesis, University of Kaiserslautern (2003)
22. Stahl, A., Roth-Berghofer, T.R.: Rapid prototyping of CBR applications with the open source tool myCBR. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 615–629. Springer, Heidelberg (2008)
23. W3C: Rdf primer (February 2004), http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (Last access: 2010-02-26)
24. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual case-based reasoning. Knowledge Engineering Review 20(3), 255–260 (2005)

Similarity-Based Retrieval and Solution Re-use Policies in the Game of Texas Hold'em*

Jonathan Rubin and Ian Watson

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

Abstract. In previous papers we have presented our autonomous poker playing agent (SARTRE) that uses a *memory-based* approach to create a betting strategy for two-player, limit Texas Hold'em. SARTRE participated in the 2009 IJCAI Computer Poker Competition where the system was thoroughly evaluated by challenging a range of other computerised opponents. Since the competition SARTRE has undergone case-based maintenance. In this paper we present results from the 2009 Computer Poker Competition and describe the latest modifications and improvements to the system. Specifically, we investigate two claims: the first that *modifying the solution representation results in changes to the problem coverage* and the second that *different policies for re-using solutions leads to changes in performance*. Three separate *solution re-use policies* for making betting decisions are introduced and evaluated. We conclude by presenting results of self-play experiments between the *pre* and *post* maintenance systems.

1 Introduction

Games offer a well suited domain for Artificial Intelligence (AI) investigation and experimentation [1] due to the fact that a game is usually composed of several well-defined rules which players must adhere to. Most games have precise goals and objectives which players must meet to succeed. For a large majority of games the rules imposed are quite simple, yet the game play itself involves a large number of very complex strategies. Furthermore, a performance metric is naturally embedded into the game itself. Success can therefore easily be measured by factors such as the amount of games won or the ability to beat certain opponents.

Games are often classified by the amount of information available to the players. If a player has access to all the information they require about the game during play then the game can be classified as having *perfect information*. However, if some of that information is hidden from the player the game is known

* If you wish to challenge the latest version of SARTRE, you may do so at http://www.cs.auckland.ac.nz/poker/

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 465–479, 2010.

as having *imperfect information*. Take for example the game of chess. Chess is a game of *perfect information* because each player can look down upon the board and obtain all the information necessary to make their playing decisions. On the other hand, the game of poker is a game of *imperfect information*. In poker players are given cards which only they can see, therefore players now have to make decisions based on hidden information because they cannot see their opponents' cards.

Games can be further classified as either *deterministic* or *stochastic*. If a game contains chance elements, such as the roll of a dice, this introduces randomness into the game. These types of games are known as *stochastic* games and examples include bridge, backgammon and poker. The absence of these chance elements ensures the game is *deterministic*. Games such as chess, checkers and go are examples of deterministic games.

Poker is a *sequential*, *stochastic* game with *imperfect information*. It is *sequential* because players choose their actions in sequence. It is *stochastic* because the shuffling of cards introduces randomness into the game. It is a game of *imperfect information* because players cannot see their opponent's cards, therefore players need to make decisions based on hidden information. Given the relatively simple rules of the game there is an enormous amount of subtle and sophisticated scenarios that can occur during a hand of play (this is particularly true of the Texas Hold'em variation).

In previous papers [2,3] we introduced our autonomous poker agent SARTRE (Similarity Assessment Reasoning for Texas hold'em via Recall of Experience) that plays the game of 2-player, limit Texas Hold'em. SARTRE constructs a poker betting strategy using a *memory-based* approach where cases are stored in a database which describe past game state information, along with the betting decisions made and the eventual outcome of those decisions. SARTRE chooses a betting action by consulting this memory of past games and retrieving similar game states. Once the most similar game state has been found its solution is re-used.

In 2009 we submitted SARTRE to the IJCAI Computer Poker Competition, where the system was thoroughly evaluated by challenging many different types of opponents. In this paper we present the results of the 2009 limit competitions in which SARTRE participated. Since the competition, case-base maintenance [4] has been conducted on the SARTRE system. This maintenance phase resulted in changes to SARTRE's solution representation. In this paper we introduce the latest modifications and improvements to the system. In particular, we address the following two claims:

Claim 1: Changes in solution representation results in changes to problem coverage.

Claim 2: Different policies for re-using solutions leads to changes in performance.

We introduce 3 separate policies for making betting decisions, which we label *solution re-use policies*. The performance of each *solution re-use policy* is evaluated

and the experimental results presented, along with the results of self-play experiments between the *pre-maintenance* and *post-maintenance* systems.

The paper proceeds as follows. Section 2 briefly describes the rules of Texas Hold'em. Section 3 reviews related work and introduces two broad types of poker strategies. An overview of our *memory-based* approach is given in Section 4, along with details regarding the maintenance that the system has undergone since the 2009 Computer Poker Competition. For the purposes of this paper we will refer to the version of SARTRE that participated in the 2009 competition as SARTRE-1 and the updated, post-maintenance system as SARTRE-2. Section 5 presents the results of the 2009 Computer Poker Competition along with further self-play experiments between SARTRE-1 and SARTRE-2. Conclusions of the work are discussed in Section 7 and avenues for future research discussed in Section 8.

2 Texas Hold'em

Here we briefly describe the game of Texas Hold'em, highlighting some of the common terms which are used throughout this work. For a more in depth introduction to Texas Hold'em consult [5].

The game of Texas Hold'em is played in 4 stages – **preflop**, **flop**, **turn** and **river**. During the preflop all players at the table are dealt two **hole cards**, which only they can see. Before any betting takes place, two forced bets are contributed to the pot, i.e. the **small blind** and the **big blind**. The big blind is typically double that of the small blind. The player to the left of the big blind, known as **under the gun**, then begins the betting by either folding, calling or raising. The possible betting actions common to all variations of poker are described as follows:

Fold: When a player contributes no further chips to the pot and abandons their hand and any right to contest the chips which have been added to the pot.

Check/Call: When a player commits the minimum amount of chips possible in order to stay in the hand and continue to contest the pot. A check requires a commitment of zero further chips, whereas a call requires an amount greater than zero.

Bet/Raise: When a player commits greater than the minimum amount of chips necessary to stay in the hand. When the player could have checked, but decides to invest further chips in the pot, this is known as a bet. When the player could have called a bet, but decides to invest further chips in the pot, this is known as a raise.

In a **limit** game all bets are in increments of a certain amount. In a **no limit** game, players can wager up to the total amount of chips they possess in front of them. Once the betting is complete, as long as at least two players still remain in the hand, play continues onto the next stage. Each further stage involves the drawing of **community cards** from the deck. Players combine their hole cards with the public community cards to form their best 5 card poker hand.

The number of community cards revealed at each stage is as follows: **flop** – 3 community cards, **turn** – 1 community card, **river** – 1 community card. Each stage also involves its own round of betting and as long as there are players left who have not folded their hands, play continues. A **showdown** occurs after the river where the remaining players reveal their hole cards and the player with the best hand wins all the chips in the pot. If two or more players have the same best hand then the pot is split amongst the winners. The list of poker hand categories in ascending order of strength is as follows: *high-card*, *one-pair*, *two-pair*, *three-of-a-kind*, *straight*, *flush*, *full-house*, *four-of-a-kind*, *straight-flush*.

3 Related Work

There are two main types of strategies that a poker agent may employ:

1. A **Nash equilibrium** strategy, or
2. an **exploitive** strategy

A **strategy** in this context refers to a mapping between game states and the actions that an agent will take at that game state. Typically, an agent's strategy consists of specifying a **probability triple** at every game state. A probability triple, *(f,c,r)*, specifies the proportion of the time an agent will either fold (f), check/call (c) or bet/raise (r) at a particular point in the game.

A **Nash equilibrium** is a robust, static strategy that attempts to limit its exploitability against a worst-case opponent. In general, a set of strategies are said to be in *equilibrium* if the result of one player diverging from their equilibrium strategy (while all other players stick to their current strategy) results in a negative impact on the expected value for the player who modified their strategy [6]. Currently, it is intractable to compute exact Nash equilibria for full-scale Texas Hold'em [7], but by applying simplifying abstractions to the game it is possible to approximate a Nash equilibrium strategy. The University of Alberta Computer Poker Research Group[1] (CPRG) were the first to approximate a Nash equilibrium for the full-scale game of two-player Texas Hold'em [8]. One of the outcomes of this research was the poker agent Sparbot, which is available within the commercial software product **Poker Academy Pro 2.5**[2]. GS1 [9] and GS2 [10] developed by Andrew Gilpin and Thomas Sandholm at Carnegie Mellon University are also examples of early attempts to derive approximate equilibrium solutions for limit hold'em.

Over the years it has become possible to construct and solve larger mathematical models of the poker game tree via improved iterative procedures, such as counterfactual regret minimization (CFRM) [11,7]. Typically, by applying fewer simplifying abstractions to the game model, stronger Nash equilibrium strategies have been produced. The University of Alberta CPRG's latest Nash based agent is Hyperborean-Eqm and it was constructed via the CFRM algorithm. The

[1] http://poker.cs.ualberta.ca/
[2] http://www.poker-academy.com/

winner of the limit equilibrium division at the 2009 IJCAI Computer Poker Competition was GGValuta. GGValuta was developed by a team of students from the University of Bucharest based on the CFRM algorithm [12].

As a Nash equilibrium strategy assumes an unknown, worst-case opponent, it will limit its own exploitability at the expense of taking advantage of weaker opponents. Hence, while this sort of strategy may not lose, it will also not win by as much as it could against weaker opponents. On the other hand, a player that attempts to isolate the weaknesses of their opponent and capitalize on those weaknesses is said to employ an **exploitive** (or **maximal**) strategy. This is typically achieved by constructing a model of an opponent and using it to inform future actions. A consequence of an exploitive strategy is that it no longer plays near the equilibrium and hence is vulnerable to exploitation itself, especially if the model of the opponent is incorrect or no longer valid. Vexbot [13] is an example of an exploitive poker agent that has been created using opponent modeling and imperfect information game tree search. Vexbot is also available within Poker Academy Pro 2.5.

Other approaches used to construct exploitive agents include the use of Monte-Carlo simulation [14,15] and the Monte-Carlo Tree Search algorithm [16], which involve the evaluation of nodes in the game tree by drawing repeated random samples. More recent exploitive strategies such as Restricted Nash Response (RNR) [17,7] and Data Biased Response (DBR) [18] attempt to optimise the trade-off between the robust nature of Nash equilibrium strategies and the power of exploitive strategies that rely on opponent modelling. Finally, various machine learning approaches have been used to construct strong poker agents. For example, the winner of the limit bankroll division of the 2009 IJCAI Computer Poker Competition was MANZANA, developed by hobbyist Marv Andersen. MANZANA employed the use of artificial neural networks trained on data from the best agent of the previous year's competition [12].

3.1 CBR Motivation

The focus of this paper is on generating case-based poker strategies. When a new problem is encountered similar cases are retrieved from the case-base of poker hands and their solutions are adapted or re-used to solve the problem. As Nash equilibrium-based strategies are very large a current goal of our research is to determine how close this type of strategy can be approximated with a compact case-base that relies on finding similar cases and generalising the observed actions. Case-based strategies can easily be used to approximate the play of a selected "expert" or group of "experts" via observation and case generation. Expert players can be human players or other computerised agents. By simply updating the case-base, different types of players can be modelled without relying on the creation of complicated mathematical models or algorithms.

CASPER [19,20], is an example of a previous poker agent constructed using a case-based strategy. CASPER was designed to play 10-player limit hold'em. Our latest poker agent, SARTRE [2] has been specialised to play 2-player, limit

hold'em. We refer to the approach used by SARTRE as a *memory-based* approach and it is introduced in the next section.

4 Our Approach

4.1 Overview

An overview of the *memory-based* approach used by SARTRE is as follows:

- A database of cases is built by observing actual poker hands. Each case consists of a collection of attribute-value pairs that encapsulate game state information.
- Separate case-bases are constructed for each round of play (*preflop*, *flop*, *turn*, *river*).
- When SARTRE is required to make a betting decision a **target case** is created to describe the current state of the game and the appropriate case-base (collection of **source cases**) is searched to locate similar cases.
- A betting decision is made by employing 1 of 3 solution re-use policies: *probabilistic*, *majority-rules* or *best-outcome* (refer to Section 4.5).

The details of the above approach are now presented.

4.2 Case Representation

Table 1 depicts the post-flop case representation[3]. Every case is made up of 3 attributes that capture information about the current game state. The attributes were selected by the authors as they are believed to encapsulate the salient aspects of the game in a concise manner. The first attribute (**hand type**) classifies the 5-card hand of a player into a category which represents information about its current strength and ability to improve (such as whether the opponent has a flush or straight draw). The next attribute (**betting sequence**) simply enumerates the betting that has been witnessed up till the current point in the hand. An *r* stands for a bet or a raise, a *c* stands for a check or a call and a hyphen delimits the betting rounds. The final attribute (**board texture**) highlights important information about the public community cards such as whether there are 3 or more cards of the same suit present on the board and hence a player could have a flush (i.e. ***Flush-Possible*** and ***Flush-Highly-Possible***)[4].

Each case also contains a solution, which is made up of an **action** triple and an **outcome** triple. The **action** triple suggests a probability distribution, *(f, c, r)*, over betting actions that the agent should select given the current state of the game. The **outcome** triple records the average quantitative profit or loss that has been observed in the past given the various betting decisions. Outcomes that have never been observed are labelled with -∞.

[3] Pre-flop cases are slightly different in that they only contain **hand type** and **betting sequence** features, where **hand type** refers to 1 of the standard 169 pre-flop equivalence classes.

[4] A complete listing of all the **hand type** and **board texture** categories can be found at www.cs.auckland.ac.nz/research/gameai/sartreinfo.html

Table 1. A case is made up of three attribute-value pairs which describe the current state of the game. A solution consists of an action triple and an outcome triple, which records the average numerical value of applying the action in this case (-∞ refers to an unknown outcome).

Attribute	Type	Example
1. Hand Type	Class	*High-Card*, *Pair*, *Two-Pair*, *Set*, *Flush*, *Pair-with-Flush-Draw*, *High-Card-with-Straight-Draw*, ...
2. Betting Sequence	String	*rc-c*, *crrc-crrc-cc-*, *r*, ...
3. Board Texture	Class	*No-Salient*, *Flush-Possible*, *Straight-Possible*, *Flush-Highly-Possible*, ...
Action	Triple	(0.0, 0.5, 0.5), (1.0, 0.0, 0.0), ...
Outcome	Triple	(-∞, 4.3 , 15.6), (-2.0,-∞,-∞), ...

Claim 1 in section 1 states that: *changes in solution representation results in changes to problem coverage*. In particular, SARTRE-1 represented actions and outcomes as single characters or numerical values, respectively. At decision time all similar cases were retrieved and the solutions combined to form a probability triple. Using this representation SARTRE-1 was forced to store many duplicate cases within each of its case-bases. SARTRE-2 now uses full probability triples as its solution representation. To derive the probability triples, SARTRE-2 must first pre-process the training data. By pre-processing the training data SARTRE-2 allows a much more compact case-base size, due to the fact that it no longer retains many duplicate cases (see Table 2).

4.3 Similarity-Based Retrieval

The k-nearest neighbour algorithm (k-NN) is used to retrieve the most similar case from the case-base. The k-NN algorithm involves positioning the target case in an n-dimensional search space of source cases. Similarity between the target and source cases individual attributes is calculated using specific similarity metrics, described in detail below. The target case is compared to every case in the appropriate case-base and similarity computed for each attribute in the case.

For SARTRE-1 the value of k could vary between 0 to N, where N was only bounded by the number of cases in the case-base. If 0 cases were retrieved a default policy of Always-Call was adopted. Furthermore, SARTRE-1 required exact matches, otherwise the default policy was used. Given SARTRE-2's updated representation, k is now set to 1. SARTRE-2 no longer requires a default policy as the solution of the most similar case is always used, no matter what the similarity.

4.4 Similarity Metrics

In order to compute global similarity, each of the attributes represented in Table 1 requires their own local similarity metric. The attributes' specific local similarity metrics are described below, where the values within the brackets indicate the allowable similarity values:

Hand Type [0, 1]: Either a combination of cards are mapped into the same category as another combination of cards, in which case a similarity value of 1.0 is assigned, or they map into a separate category, in which case a similarity value of 0 is given. This same metric is used for both SARTRE-1 and SARTRE-2.

Board Texture [0, 1]: As above.

Betting Sequence [0, 0.8, 0.9, 1]: SARTRE-1 used a simplistic all or nothing similarity metric for the *betting sequence* attribute, where similarity was either 0 or 1. SARTRE-2 improves upon this simplistic metric by assigning stepped levels of similarity. The first level of similarity (level0) refers to the situation when one betting sequence exactly matches that of another. If the sequences do not exactly match the next level of similarity (level1) is evaluated. If two distinct betting sequences exactly match for the active betting round and for all previous betting rounds the total number of bets/raises made by each player are equal then level1 similarity is satisfied and a value of 0.9 is assigned. Consider the following example where the active betting round is the *turn* and the two betting sequences are:

1. *crrc-crrrrc-cr*
2. *rrc-rrrrc-cr*

Here, level0 is clearly incorrect as the sequences do not match exactly. However, for the active betting round (*cr*) the sequences do match. Furthermore, during the preflop (1. *crrc* and 2. *rrc*) both players made 1 raise each, albeit in a different order. During the flop (1. *crrrrc* and 2. *rrrrc*) both players now make 4 raises each. Given that the number of bets/raises in the previous rounds (preflop and flop) match, these two betting sequences would be assigned a similarity value of 0.9.

If level1 similarity was not satisfied the next level (level2) would be evaluated. Level2 similarity is less strict than level1 similarity as the previous betting rounds are no longer differentiated. Consider the river betting sequences:

1. *rrc-cc-cc-rrr*
2. *cc-rc-crc-rrr*

Once again the sequences for the active round (*rrr*) matches exactly. This time, the number of bets/raises in the preflop round are not equal (the same applies for the flop and the turn). Therefore, level1 similarity is not satisfied. However, the number of raises encountered for all the previous betting rounds combined (1. *rrc-cc-cc* and 2. *cc-rc-crc*) are the same for each player, namely 1 raise by each player. Hence, level2 similarity is satisfied and a similarity value of 0.8 would be assigned. Finally, if level0, level1 and level2 are not satisfied level3 is reached where a similarity value of 0 is assigned.

4.5 Solution Re-use Policies

Once the above similarity metrics have been applied and a similar case retrieved, a betting decision is required. Claim 2 of this paper states: *different policies for re-using solutions leads to changes in performance*. To investigate this claim we have experimented with three separate approaches for making betting decisions, which we refer to as **solution re-use policies**. They are as follows:

Probabilistic: this solution re-use policy probabilistically selects a betting decision based on the *(f, c, r)* proportions specified by the solution of the retrieved case or cases. Betting decisions that have greater proportion values will be made more often then those with lower values.

Majority-Rules: the majority-rules solution re-use policy will re-use the decision that was made the majority of the time, as specified by the proportions contained within the solution of the most similar case or cases.

Best-Outcome: rather than re-use a case's action triple, the best-outcome solution re-use policy will make a betting decision based on the values in the outcome triple of the most similar case or cases. The betting decision which has the greatest average outcome overall is the one that is chosen.

4.6 Case-Base Construction

A beneficial property of the approach described above is the ability to easily *"plug-in"* different case-bases that have been trained on the hand history logs of various *"expert"* players. Typically, data obtained from a single Nash equilibrium-based player is chosen to train the system. A single expert is chosen, rather than multiple experts to avoid conflicting decisions. The expert's original strategy is then approximated by constructing cases and generalising the observed actions by retrieving the most similar case at decision time. The case-base used by SARTRE-1 was trained on the hand history logs of Hyperborean-Eqm, who was the winner of the limit equilibrium division at the 2008 Computer Poker Competition [21]. SARTRE-2 was trained on the hand history logs of MANZANA, which was the winner of the limit hold'em bankroll competition at the 2009 Computer Poker Competition [22]. Table 2 depicts the number of cases recorded for each separate betting round for both SARTRE-1 and SARTRE-2.

Given SARTRE-1's requirement to store a large number of cases, the amount of data used to train the system had to be restricted due to the corresponding storage costs. On the other hand, the use of probability triples along with preprocessing the training data ensures SARTRE-2 stores a more compact case-base. The reduced storage costs associated with SARTRE-2 allows a larger set of data to be used to train the system. The final result of this increased training is that SARTRE-2 encounters (and stores cases for) a wider range of problems than SARTRE-1.

Table 2. Total cases recorded per round for SARTRE-1 and SARTRE-2

Round	Total Cases (SARTRE-1)	Total Cases (SARTRE-2)
Preflop	201,335	857
Flop	300,577	6,743
Turn	281,529	35,464
River	216,597	52,088
Total	1,000,038	95,152

5 Experimental Results

5.1 2009 IJCAI Computer Poker Competition

The Annual Computer Poker Competition[5] (CPC) has been held each year either at the AAAI or IJCAI conference since 2006. The CPC involves separate competitions for different varieties of Texas Hold'em, such as both limit and no-limit competitions as well as heads-up and multiple-opponent competitions. Entrance into the competition is open to anyone and the agents submitted typically represent the current state of the art in computer poker. The CPC uses a duplicate match structure. For a heads-up match a duplicate match proceeds as follows: N hands are played between two agents after which the agent's memories are wiped and the N hands played again, but in the reverse direction, i.e. the cards that were initially given to player A are instead given to player B and vice-versa. This way both players get to play both sets of N cards and this minimises the variance that is involved in simply playing a set of N hands in one direction only. In the 2009 competition, $N = 3000$ was used and many duplicate matches were played in order to achieve significant results.

The annual CPC typically involves two winner determination methods. The first is known as the **equilibrium competition** and the second is the **bankroll competition**. The equilibrium competition analyses the results of a set of matches in a way that rewards agents that play close to a Nash equilibrium. The bankroll competition rewards agents that are able to maximally exploit other agents, resulting in increased bankrolls.

Measurements are made in **small bets per hand** (sb/h), where the total number of small bets won or lost are divided by the total hands played. For example, assuming a \$10/\$20 hold'em game, imagine after 3000 hands a player has made a total profit of \$1800. To calculate the sb/h value, first the total profit is divided by the small bet (\$10) which is then divided by the total hands (3000) which gives a final value of +0.06 sb/h.

Final results for the 2009 IJCAI Computer Poker Competition are depicted in Table 3 for the 2-player limit hold'em bankroll and equilibrium divisions, respectively. For this competition SARTRE-1 used a majority-rules solution reuse policy and a default always-call policy when no similar cases were retrieved.

[5] http://www.computerpokercompetition.org/

Table 3. 2009 limit heads up bankroll and equilibrium results, respectively

Place	Competitor	sb/h
1	MANZANA	0.186 ± 0.002
2	Hyperborean-BR	0.116 ± 0.002
3	GGValuta	0.110 ± 0.002
4	Hyperborean-Eqm	0.116 ± 0.002
5	Rockhopper	0.103 ± 0.002
6	*Sartre*	0.097 ± 0.002
7	Slumbot	0.096 ± 0.002
8	GS5	0.082 ± 0.002
9	AoBot	-0.002 ± 0.003
10	dcurbHU	-0.07 ± 0.002
11	LIDIA	-0.094 ± 0.002
12	GS5Dynamic	-0.201 ± 0.002

Place	Competitor
1	GGValuta
2	Hyperborean-Eqm
3	MANZANA
4	Rockhopper
5	Hyperborean-BR
6	Slumbot
7	*Sartre*
8	GS5
9	AoBot
10	GS5Dynamic
11	LIDIA
12	dcurbHU
13	Tommybot

Table 3 (left) shows that SARTRE-1 placed 6th out of 12 competitors in the bankroll division, achieving an average profit of 0.097 sb/h. Table 3 (right) depicts the final placings for the equilibrium competition, which rewards agents that play closer to a Nash equilibrium. In this division, SARTRE-1 placed 7th out of 13 competitors.

5.2 Self-play Experiments

This section presents the results of self-play experiments between:

- The 3 solution re-use policies of SARTRE-2, and
- The *pre* and *post* maintenance systems, SARTRE-1 and SARTRE-2.

The purpose of these experiments is to determine the validity of the claims put forward in section 1 and to investigate whether the maintenance process had any affect on the actual performance of the system.

Solution Re-Use Policies. Table 4 presents the first set of results where the 3 solution re-use policies (described in Section 4.5) were played against each other. A round robin tournament structure was used, where each policy challenged every other policy. The figures presented are in small/bets per hand. Each match consisted of 3 separate duplicate matches, where $N = 3000$. Hence, in total 18,000 hands of poker were played between each competitor. All results are statistically significant.

Table 4 shows that the majority-rules policy outperforms its probabilistic and best-outcome counterparts. Of the three, best-outcome fairs the worst, losing all of its matches.

Table 4. Results of self play experiments between 3 solution re-use policies of SARTRE-2

	Majority-rules	Probabilistic	Best-outcome	**Average**
Majority-rules		0.011 ± 0.005	0.076 ± 0.008	$\mathbf{0.044 \pm 0.006}$
Probabilistic	-0.011 ± 0.005		0.036 ± 0.009	$\mathbf{0.012 \pm 0.004}$
Best-outcome	-0.076 ± 0.008	-0.036 ± 0.009		$\mathbf{-0.056 \pm 0.005}$

SARTRE-1 *Vs.* SARTRE-2. Given the above results we selected the majority-rules solution re-use policy for SARTRE-2 and conducted a further set of experiments where SARTRE-2 challenged SARTRE-1. Table 5 presents the outcomes of 10 duplicate matches, where $N = 3000$. The figures presented are the small bets per hand SARTRE-2 won against SARTRE-1. In total 60,000 hands of poker were played between the two systems.

Table 5. Results of self play matches between SARTRE-2 and SARTRE-1. Each match consists of 6000 hands. Results for SARTRE-2 are recorded in **sb/h**.

Match	SARTRE-2 (**sb/h**)
1	−0.034
2	0.30
3	0.016
4	0.005
5	0.001
6	−0.013
7	−0.004
8	0.011
9	0.004
10	0.000
Mean:	**0.0286**
Std dev:	**0.096368**
95% CI:	**[-0.04034, 0.09754]**

Table 5 shows that for the 10 duplicate matches played, SARTRE-2 won on average 0.0286 sb/h against SARTRE-1. Out of the 10 matches SARTRE-2 won 6 and lost 3. The final match was a draw. A 95% confidence interval was calculated on the sample using a student's t-distribution. The result shows that while SARTRE-2 achieves a greater average profit than SARTRE-1, further evaluation is required to achieve statistical significance.

6 Discussion

The results indicate that simply re-using the decision made the majority of the time results in better performance than mixing from a probability triple and that choosing the decision that resulted in the best outcome was the worst solution re-use policy. Moreover, while we have not presented results against other

computerised agents, our initial testing appears to suggest the same outcomes are observed.

One of the reasons for the poor performance of **best-outcome** is likely due to the fact that good outcomes don't necessarily represent good betting decisions and vice-versa. The reason for the success of the **majority-rules** policy is less clear. We believe this has to do with the type of opponent being challenged, i.e. Nash equilibrium-based agents. As an equilibrium-based strategy doesn't attempt to exploit any bias in its opponent strategy, it will only gain when the opponent ends up making a mistake. Therefore, biasing actions towards the decision that was made the majority of the time is likely to go unpunished when challenging an equilibrium-based agent. Furthermore, sticking to this decision avoids any **exploration** errors made by choosing other actions. On the other hand, against an exploitive opponent the bias imposed by choosing only one action is likely to be detrimental to performance and therefore it would become more important to mix up decisions.

By modifying the way knowledge is encoded within the case-base knowledge container, SARTRE-2 allows a significant reduction in case-base size. In particular, SARTRE-2 stores 904,886 fewer cases than SARTRE-1. Futhermore, the results of experiment 2 (SARTRE-1 Vs. SARTRE-2) show that SARTRE-2 appears to perform a little better than SARTRE-1. Once again, initial results against other computerised agents (not included in the paper) appear to support this observation. There are many factors that could have contributed to this difference in performance, including training on different "expert" players, the updated betting sequence similarity metric, as well as the improved problem coverage of SARTRE-2 as a result of the updated solution representation.

7 Conclusions

We have presented the SARTRE system. Given hand log training data the SARTRE system stores the betting decisions in a case-base and generalises the observed decisions to inform a betting strategy. During game play, actions are chosen by searching the appropriate case-base and retrieving cases similar to the present situation. SARTRE selects a betting action based on 1 of 3 solution re-use policies. A version of the SARTRE system, which we have labelled SARTRE-1, was submitted to the 2009 IJCAI Computer Poker Competition. The competitors in this competition typically represent the current state-of-the-art in computer poker agents. In the 2009 competition, SARTRE placed 6th out of 12 in the limit bankroll event and 7th out of 13 in the equilibrium event. Given the simplistic nature of the approach the results are quite positive.

Case-base maintenance was applied to the system. In this paper we referred to the post-maintenance system as SARTRE-2. Observations made during the maintenance process provided support for claim 1 that *changes in solution representation results in changes to problem coverage*. In particular, updates to SARTRE-2's solution representation resulted in the removal of duplicate cases from the case-base. Reducing the number of cases required for storage had the

effect of increasing the amount of scenarios SARTRE-2 was able to encounter and use for training purposes. This led to a more comprehensive case-base being generated.

SARTRE-2's case representation is concise which allows a compact representation of a betting strategy for limit Texas Hold'em. The attribute-value pairs that make up SARTRE's case representation were described along with the betting sequence similarity metrics used by the system, which allow graceful degradation when an exactly matching case is unable to be retrieved.

The second claim investigated was that *different policies for re-using solutions leads to changes in performance*. Empirical results, presented in Table 4, were used to support this claim. Three *solution re-use policies* were introduced and comparatively evaluated. The results showed that the *majority-rules* policy achieved the greatest profit during self-play experiments. Given this result, SARTRE-1 was challenged by SARTRE-2 using a majority-rules decision re-use policy. The results showed that on average SARTRE-2 achieved a greater profit than SARTRE-1. However, as the systems are still closely related in strength, many more hands are required in order to display a significant difference.

8 Future Work

In the future we wish to introduce opponent modeling. Currently SARTRE attempts to approximate a robust betting strategy that is able to perform well against a range of opponents without paying any attention to how the opponent plays. We wish to investigate possible ways of augmenting the current system with opponent modeling information, which could be used to exploit weaker opponents.

References

1. Billings, D., Papp, D., Schaeffer, J., Szafron, D.: Poker as Testbed for AI Research. In: Mercer, R.E. (ed.) Canadian AI 1998. LNCS, vol. 1418, pp. 228–238. Springer, Heidelberg (1998)
2. Rubin, J., Watson, I.: A Memory-Based Approach to Two-Player Texas Hold'em. In: Nicholson, A., Li, X. (eds.) AI 2009. LNCS, vol. 5866, pp. 465–474. Springer, Heidelberg (2009)
3. Rubin, J., Watson, I.: SARTRE: System Overview, A Case-Based Agent for Two-Player Texas Hold'em. In: Workshop on Case-Based Reasoning for Computer Games, Eighth International Conference on Case-Based Reasoning (ICCBR 2009), Springer, Heidelberg (2009)
4. Leake, D.B., Wilson, D.C.: Categorizing Case-Base Maintenance: Dimensions and Directions. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 196–207. Springer, Heidelberg (1998)
5. Texas Hold'em Article, http://en.wikipedia.org/wiki/Texas_Holdem
6. Morris, P.: Introduction to game theory. Springer, New York (1994)
7. Johanson, M. B.: Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player. Masters thesis, University of Alberta (2007)

8. Billings, D., Burch, N., Davidson, A., Holte, R.C., Schaeffer, J., Schauenberg, T., Szafron, D.: Approximating Game-Theoretic Optimal Strategies for Full-scale Poker. In: IJCAI 2003, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 661–668. Morgan Kaufmann, San Francisco (2003)
9. Gilpin, A., Sandholm, T.: A Competitive Texas Hold'em Poker Player via Automated Abstraction and Real-Time Equilibrium Computation. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence (AAAI 2006), pp. 1007–1013. AAAI Press, Menlo Park (2006)
10. Gilpin, A., Sandholm, T.: Better Automated Abstraction Techniques for Imperfect Information Games, with Application to Texas Hold'em Poker. In: 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), IFAAMAS, pp. 192–200 (2007)
11. Zinkevich, M., Johanson, M., Bowling, M.H., Piccione, C.: Regret Minimization in Games with Incomplete Information. In: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, pp. 1729–1736. MIT Press, Cambridge (2007)
12. 2009 Computer Poker Winners, http://www.cs.ualberta.ca/~pokert/2009/results/winners.txt
13. Billings, D., Davidson, A., Schauenberg, T., Burch, N., Bowling, M., Holte, R.C., Schaeffer, J., Szafron, D.: Game-Tree Search with Adaptation in Stochastic Imperfect-Information Games. In: van den Herik, H.J., Björnsson, Y., Netanyahu, N.S. (eds.) CG 2004. LNCS, vol. 3846, pp. 21–34. Springer, Heidelberg (2006)
14. Billings, D., Castillo, L. P., Schaeffer, J., Szafron, D.: Using Probabilistic Knowledge and Simulation to Play Poker. In: AAAI/IAAI, pp. 697–703. AAAI Press, Menlo Park (1999)
15. Schweizer, I., Panitzek, K., Park, S.H., Furnkranz, J.: An Exploitative Monte-Carlo Poker Agent. Technical report, Technische Universitat Darmstadt (2009)
16. Van den Broeck, G., Driessens, K., Ramon, J.: Monte-Carlo Tree Search in Poker Using Expected Reward Distributions. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS, vol. 5828, pp. 367–381. Springer, Heidelberg (2009)
17. Johanson, M., Zinkevich, M., Bowling, M. H.: Computing Robust Counter-Strategies. In: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, pp. 721–728. MIT Press, Cambridge (2007)
18. Johanson, M., Bowling, M.: Data Biased Robust Counter Strategies. In: Twelfth International Conference on Artificial Intelligence and Statistics, pp. 264–271 (2009)
19. Rubin, J., Watson, I.: Investigating the Effectiveness of Applying Case-Based Reasoning to the Game of Texas Hold'em. In: Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, pp. 417–422. AAAI Press, Menlo Park (2007)
20. Watson, I., Rubin, J.: CASPER: A Case-Based Poker-Bot. In: Wobcke, W., Zhang, M. (eds.) AI 2008. LNCS (LNAI), vol. 5360, pp. 594–600. Springer, Heidelberg (2008)
21. 2008 Poker Bot Competition Summary, http://www.cs.ualberta.ca/~pokert/2008/results/
22. 2009 Poker Bot Competition Summary, http://www.cs.ualberta.ca/~pokert/2009/results/

Experience-Based Critiquing: Reusing Critiquing Experiences to Improve Conversational Recommendation

Kevin McCarthy, Yasser Salem, and Barry Smyth

CLARITY: Centre for Sensor Web Technologies
School of Computer Science and Informatics
University College Dublin, Belfield, Dublin 4, Ireland
{kevin.mccarthy,yasser.salem,barry.smyth}@ucd.ie
http://www.clarity-centre.org

Abstract. Product recommendation systems are now a key part of many e-commerce services and have proven to be a successful way to help users navigate complex product spaces. In this paper, we focus on critiquing-based recommenders, which permit users to *tweak* the features of recommended products in order to refine their needs and preferences. In this paper, we describe a novel approach to reusing past critiquing histories in order to improve overall recommendation efficiency.

1 Introduction

Today, e-commerce services rely on recommender systems to help users to navigate complex product spaces. Amazon's use of recommendation technologies is well documented [5] and the recent Netflix competition [1] highlights the value of sophisticated recommendation techniques in the commercial world. The recommender systems community has explored a diverse space of different recommendation techniques, from *collaborative filtering* methods, which rely on simple ratings-based profiles to generate recommendations from similar users, to *content-based* methods, which rely on the availability of product knowledge, generally in the form of detailed descriptions, to make recommendations.

Most deployed recommender systems are *single-shot*, in the sense that the job of the recommender is to produce a single list of recommendations for the user [4,14,15]. The single-shot strategy is well-adapted to the recommendation of simple products and services, such as ringtones, movies, books, etc., but it is not so well suited to recommending more complex items. In more complex recommendation scenarios it is appropriate to offer the user an opportunity to provide feedback, to refine their needs and preferences, based on recent recommendations. In response researchers have developed so-called *conversational recommendation* strategies [2,8,9,16] to support this type of recommendation scenario. Briefly, users participate in a *recommendation dialogue*, receiving recommendations and providing feedback in order to inform a new set of recommendations. Different approaches to conversational recommendation can be distinguished by their use

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 480–494, 2010.

of different types of feedback and one type of feedback that forms the basis of this work is *critiquing*. Critiquing in a simple form of feedback which facilitates a *"show me more like this but ..."* type of response. Recently there has been renewed interest in critiquing [3,7,10,11,13,19] because of its many advantages: it is simple to implement for a variety of interface types, and it can be appropriate for users who are not experts in a given product domain. However, as we shall see, traditional approaches to critiquing can lead to protracted dialog sessions leading researchers to seek out ways of improving the performance of critiquing-based recommender systems [7,19].

In this paper, we are interested in improving the efficiency of critiquing-based recommender systems without introducing additional critiquing complexity. Our starting point is the idea that the critiquing histories, or experiences, of users carry important information about feature preferences and trade-offs and we consider how these experiences can be usefully reused to bias the recommendation process. In the following sections, we describe one such technique for harnessing critiquing experiences during recommendation and demonstrate its effectiveness, relative to conventional critiquing, on a real-world restaurant dataset.

2 Related Work

Recommender systems are a common way to promote products or services that may be of interest to a user, usually based on some profile of interests. The single-shot approach, which produces a ranked list of recommendations, is limited by design. It works well when a user's needs are clear, but it is less suitable when a user's needs are not well known, or where they are likely to evolve during the course of a session. In these scenarios it is more appropriate to engage the user in a recommendation dialog so that incremental feedback can be used to refine recommendations. This type of conversational recommender system is much better suited to help users navigate more complex product spaces.

Various forms of feedback are used in conversational recommender systems: value elicitation, ratings-based feedback, preference-based feedback and critiquing [17]. Systems which employ value elicitation expect users to input specific values for product features, e.g. *hard-drive = 320GB*. This is a rich form of feedback so the user must possess a high level of domain knowledge to use it effectively. In contrast, ratings-based feedback is a much simpler form of feedback, preferred by most collaborative filtering systems. Users assign a simple rating, e.g. *3 stars out of 5*, to indicate their satisfaction with the recommendation; see, for example, [4]. With ratings-based feedback, the user does not require detailed domain knowledge, since they are not commenting on specific features. Instead they simply provide an overall recommendation rating. Preference-based feedback is a special case of this in which, instead of rating a set of recommendations, the user simply indicates their preferred recommendation [8]. It is a low cost form of recommendation that requires minimal domain knowledge, just an ability to distinguish good from bad recommendations. However, it is clearly limited in its information content, as it is not always apparent why the user has selected one recommendation over others. In this paper, we look at an altogether different form of feedback; critiquing,

which strikes a balance between ease of feedback and the information content of the feedback. Simply put, critiquing allows users to indicate a *directional preference* with respect to a particular product feature. For example, in a restaurant recommender a user might respond to a given recommendation by asking for a new suggestion that is *cheaper*. In this case the user is critiquing the *price* feature, asking for a new restaurant with a lower price; this is the standard form of critiquing, which is also called *unit critiquing* because the user critiques a single feature at a time. The critique acts as a filter over the remaining recommendable items. The next recommendation will be compatible with the critique while being maximally similar to the previous recommendation.

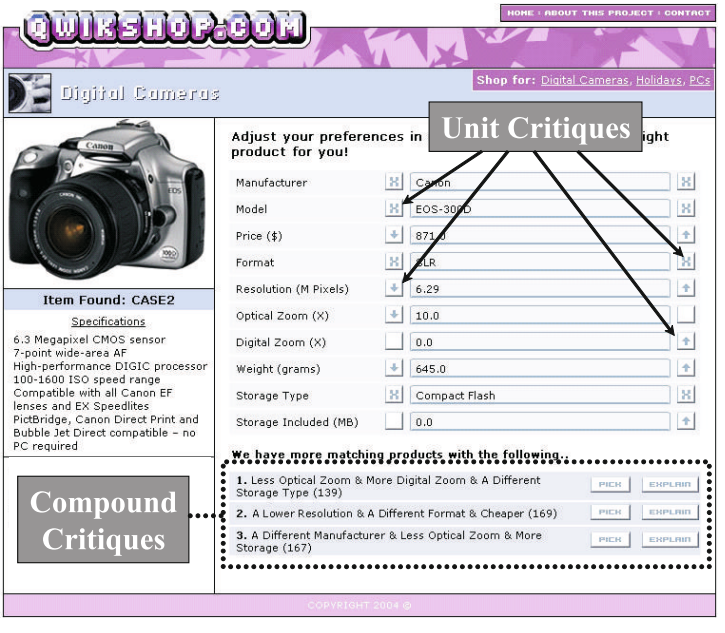


Fig. 1. Critique-based recommender system specialising in digital cameras

Critiquing has been evaluated in many e-commerce situations, from choosing a restaurant to purchasing a camera [3,7,10]. For example, Figure 1 shows a screen-shot of a typical critique-based recommender system for recommending digital cameras. The current recommendation is displayed in the main window, described by a range of features (i.e. manufacturer, resolution, price, etc.), and the critiques are presented on either side of the feature values (marked "Unit Critiques"). The user can choose one of these critiques to refine their query (e.g. More than 6.29M Pixels). The recommender system uses the critique as a constraint over the value-space of the feature when choosing the next recommendation. A key advantage is that the user does not need to provide a specific value for a product feature, so it demands a lower level of product knowledge

and domain expertise. Critiquing also has a simple interaction mechanism which can be accommodated in a variety of interface styles. However, while critiques help the recommender to narrow its search, progress through a complex product space can be slow, leading to protracted sessions and unhappy users [11]. Moreover, users often change their mind within a session, which can limit the effectiveness of certain approaches to critiquing; see [12].

Recently a number of researchers have started to look at how to make critiquing more efficient by allowing users to provide feedback on multiple features with a single critique. For instance, the work of [7,11,13,19] describes the generation of so-called *compound critiques* from collections of individual *unit critiques*. The screenshot of the recommender system in Figure 1 shows compound critiques which allow the user to cover multiple feature critiques at the same time (e.g. *"A Lower Resolution & A Different Format & Cheaper"*). Not only do compound critiques allow the user to make larger jumps through the product space, they also help to clarify the different trade-offs that exist between features, which can help to improve a user's domain knowledge. Compound critiques are dynamically generated during each recommendation cycle. For example [7,11] use association rule mining in order to discover compound critiques whereas [19] use a version of multi-attribute utility theory (MAUT). Live user studies of compound critiquing indicate that when users utilise compound critiques (in 35% - 55% of cycles) they go on to benefit from much shorter sessions (20%-50% reductions) compared to users who ignored compound critiques [6]. While such benefits speak to the potential value of compound critiquing, in practice these benefits are limited to those users who avail of compound critiques and many users do not. Hence, in this paper we will describe an alternative solution to this problem, one that does not rely on new feedback options, but rather focuses on improving the traditional unit critiquing. We introduce a new source of knowledge to the critiquing process, namely the critiquing experiences of other users, on the assumption that these experiences may encode meaningful patterns of critiques which may help us to short-cut the standard critiquing process.

3 Experience-Based Critiquing

Inspired by ideas in case-based reasoning our proposed experience-based critiquing technique attempts to harness a new source of knowledge during the critiquing process, namely the critiquing experiences of other users. We will focus on past critiquing sessions that have been successful – in the sense that they have led to a purchase decision, for example, and our basic assumption is that these successful experiences must encode useful patterns of critiques, which may help us to short-cut the standard critiquing process for future users. This strand of research borrows from work on mining web logs for request sequences, to use for predicting web pages for caching and prefetching [18]. In what follows we will describe a novel technique to leverage these experiences as part of a conventional critiquing-based recommender system and we will go on to demonstrate the potential of these experiences to significantly improve the efficiency on standard unit critiquing.

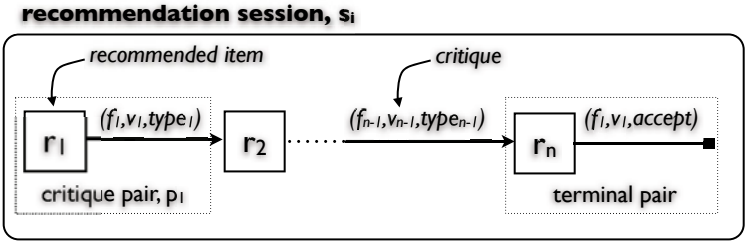


Fig. 2. Each recommendation session is made up of a sequence of recommendation-critique pairs. Each recommendation pair is comprised of a recommended item and an individual critique (feature, value, type) applied to that item.

3.1 Recommendation Sessions

In a typical critiquing session the user will start with a high-level understanding of their needs. For example, when choosing a restaurant they might start by indicating a price-range and a location. During the course of a session this will be refined, as the user critiques the features of recommended restaurants, perhaps indicating that they are looking for somewhere that is less formal but more expensive than earlier recommendations. Thus, during a particular critiquing session a user may provide feedback on a range of different features.

We can model each recommendation session, s_i, as a sequence of *recommendation critique pairs*, as shown in Figure 2 and Equations 1-2; each r_i represents a recommendation and c_i is the critique that is applied by the user to that recommendation. Each c_i is represented as a triple, $(f_i, v_i, type_i)$, where f_i refers to the feature $f_i \epsilon r_i$ that is the focus of the critique, v_i is the value of f_i in r_i ($r_i.f_i$), and $type_i$ is the type of critique that is applied (typically, $type_i \epsilon \{<, >, =, <>\}$); see Equation 4. For now we can assume that each session terminates (see Equation 3) when the user chooses to *accept* a recommendation, indicating that they are satisfied with the recommendation, or when they choose to *stop* a given session, presumably because they have grown frustrated with the recommendations received. Thus we can add *accept* and *stop* to the set of permissible critique types such that every session terminates with one or other of these types.

$$s_i = \{p_1, ..., p_n\} \tag{1}$$

$$p_i = (r_i, c_i) \tag{2}$$

$$terminal(s_i) = p_n = (r_n, c_n) \tag{3}$$

$$c_i = (f_i, v_i, type) \tag{4}$$

In general, the many users of a given critiquing-based recommender system will produce a large collection of critiquing sessions ($S = \{s_1, ..., s_k\}$) as they engage with the recommender system. The sessions reflect the *experience* of these

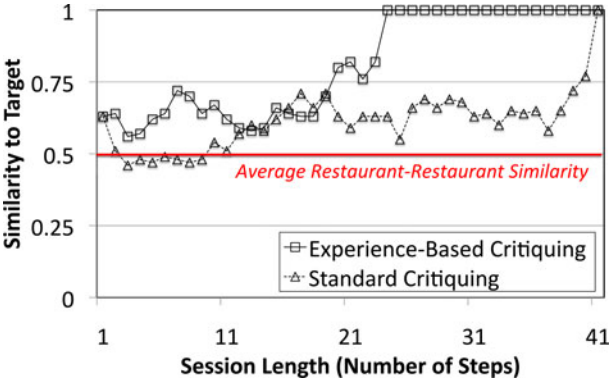


Fig. 3. Example session showing mean similarity to the target per critique step

users and capture potentially useful knowledge about their preferences and the different trade-offs they tend to make. In this paper, we are interested in the potential for these experiences to inform the recommendation process itself. In other words, these critiquing sessions are the cases in a case base of critiquing experiences. For the remainder of this paper we will assume that only *successful sessions* — that is, those sessions where the user *accepts* a recommendation — are stored in the case base. Then we can treat this accepted recommendation as the *solution* of the case and the critique pairs that proceed it as the *specification* part. We will describe how to harness these critiquing experiences to improve the efficiency of the recommendation process by using a modified critique-based recommendation algorithm, called *experienced-based critiquing*, which differs from the traditional approach to critiquing in the manner in which new recommendations are generated; see Figure 4 for a brief overview.

3.2 Conventional Critiquing

According to conventional critiquing, when a user applies a critique c_i to an item r_i, the recommender responds by retrieving an item, r_T, which is compatible with c_i, in the sense that the item satisfies the critique c_i, and which is maximally similar to r_i, as in Equations 5-6[1]. Note that the form $r.f$ indicates the value of feature f in recommended item r and $apply(type, u, v)$ is true if and only if the predicate denoted by $type$ is satisfied by the values u and v; for example, $apply(<, 25, 40)$ is $true$ whereas $apply(=, casual, formal)$ is not.

$$r_T = Recommend(r_i, c_i) = \underset{\forall\, r_j \epsilon\, items \,\wedge\, satisfies(c_i, r_j)}{argmax} \Big(sim(r_i, r_j) \Big) \qquad (5)$$

[1] Implementations will differ on issues such as the similarity metric used and also on whether items which have already been critiqued in a session are available for repeat recommendation.

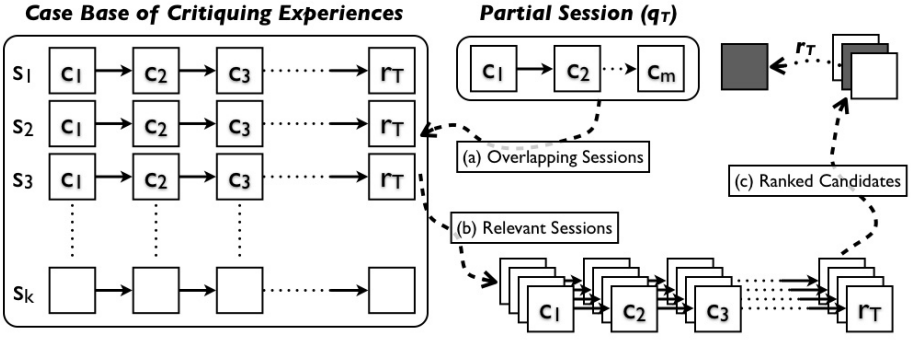


Fig. 4. Experience-based critiquing reuses past successful sessions to identify terminal items that have proven popular in sequences of critiques that are similar to the user's critiques. These items are a source recommendation candidates for the current user session.

$$satisfies(c_i, r_j) \leftrightarrow apply(type_i, r_j.f_i, r_i.f_i) \tag{6}$$

Figure 3 shows an example session profile for a standard critiquing session from the evaluation in Section 4. For each step in the session, we note the similarity of the recommended item (in this case a restaurant) to the target, which is known to the evaluation below. Here the session takes 41 steps to reach the target. Notice too how the similarity to the target does not rise monotonically as the session progresses. Often the user will select a critique but the new recommendation will be less similar to the target item than the critiqued case; the critiqued feature may be a better match to the target but this often comes at the expense of other features. In fact, notice how in the early part of the session (steps 3-9 inclusive) the similarity to the target dips below the average item-item similarity in our restaurant dataset; during these steps a random restaurant would likely have been a better overall match to the target than the recommended one.

3.3 Harnessing Critiquing Experiences

Experience-based critiquing extends conventional critiquing by reusing past sessions to guide the critiquing process. Instead of retrieving a new item that is maximally similar to the current recommendation, and compatible with the user's critique, we recommend one of the items that past users have ended up purchasing/accepting under similar critiquing scenarios. This can be best understood in terms of three basic steps: (1) building experience cases from past users' sessions; (2) identifying past critiquing sessions that are similar (i.e., relevant) to the current user session; (3) ranking recommendation candidates from the terminal items of these similar sessions.

Building Experience Cases: Before a case base of previous users' critiquing experiences can be used in the recommendation process, the experience cases

must be built from the historical recommendation sessions (see Figure 2). This process is not as simple as extracting the critiques from the recommendation-critique pairs since in many scenarios users are liable to change their mind mid-session and as a result sequences of critiques can be in conflict [12]. For example, a user might start by looking for a product that is *cheaper than $100* only to later shift towards looking for a product in the $100 - $150 range, once they start to recognise the different tradeoffs that exist in the target product space, and so eliminating recommendation candidates in the $100 - $150 range, based on the earlier critique, would be inappropriate. Accordingly we edit the current user's critiques by working backwards through the session starting with the last critique before target acceptance. If the current critique conflicts with a less recent critique (that has already been processed) then it is eliminated. This leaves a set of core critiques which represent the *boundaries* of the user's preferences with respect to the features that have been critiqued. Once this process is complete we also extract the final case as accepted by the user upon completion of their recommendation session. This case will serve as the candidate target for this particular critiquing experience (see Figure 4).

Identifying Relevant Critiquing Sessions: When a user applies a critique c_i to a recommended item r_m we will use the user's current (partial) critique session, $c_1, ..., c_m$, as a query (q_T), over the case base of past critique sessions, in order to identify a set of *relevant sessions*; see (a) and (b) in Figure 4. Briefly, a relevant session is one which has at least some overlap with the current query (see Equation 8), based on a particular overlap metric. In this case, we propose the simple overlap score shown in Equation 7, which computes the square of the number of critiques in q_T that are also present in a given session; the use of the square function here introduces a strong bias in favour of greater overlaps. Note that in the case that there are no relevant sessions, and thus no candidates to recommend, then we revert to standard critiquing and retrieve a new item that is maximally similar to the current recommendation and compatible with the user's critique.

$$OverlapScore(q_T, s_i) = \left[\sum_{c_i \in q_T} \sum_{c_j \in s_i} match(c_i, c_j) \right]^2 \quad (7)$$

$$S^{REL} = RelevantSessions(q_T, S) = \left\{ s_i \epsilon S : OverlapScore(q_T, s_i) > t \right\} \quad (8)$$

Ranking Recommendation Candidates: These relevant sessions (S^{REL}) correspond to sequences of critiques that have previously led a user to a successful outcome. Each relevant session will terminate with some final, accepted recommendation case. The final recommendation case from the experience session with the largest overlap score intuitively makes a good recommendation for the current user (as shown in Equation 9). The recommended case, r_T, will be compatible with the last critique made by the user, will have overlapping critiques (the minimum amount of overlap depending on the threshold, t) and will have been previously selected as a final recommendation.

$$r_T = Recommend(S^{REL}) = \underset{\forall\ s_i \epsilon\ S^{REL}}{argmax} \left(OverlapScore(q_T, s_i) \right) \quad (9)$$

In summary then, when a user critiques an item, instead of just presenting that most similar remaining item that satisfies the critique, the experience-based critiquing technique recommends an item which has been frequently accepted by past users under similar critiquing conditions. This recommended item is usually not the most similar item to the critiqued item, allowing the recommender system to take larger steps through the product space and, hopefully, improve overall recommendation efficiency. For instance, returning to Figure 3, we also show the session profile for the corresponding experience-based critiquing session (with a case base of almost 3000 session cases and $t = 15$). In this example case, the session reaches the target item at step 24, a 41% reduction in session length compared to the standard critiquing approach. Moreover, by and large, the similarities of the individual recommendations are typically 25% more similar to the target item than their corresponding recommendation from the standard critiquing session, and they never fall below the average item-item similarity level.

4 Evaluation

In conventional critique-based recommendation systems, new recommendations are influenced by the sequence of critiques in the current session. These critiques help to focus the recommender within the recommendation space. According to the new technique presented in this paper, the critiquing experiences of other users can also play a role in guiding the session. We evaluate this new technique, by comparing it to conventional critiquing. To do this we have developed a restaurant recommender system, based on a comprehensive database of Dublin restaurants.

4.1 Datasets

There are two key datasets used in this evaluation: *restaurants* (recommendation items) and critiquing *experience sessions*. The former stems from a recent crawl of an online restaurant database for Dublin. A total of 632 individual restaurants have been extracted and each is represented by 28 different features (e.g. price, quality, etc.), including 2 nominal, 7 numeric, and 19 binary features.

Ideally, we would like to be able to evaluate experience-based critiquing using real users. However, this is not currently feasible since it would require a major live deployment over an extended period of time. In the alternative, we adopt the approach taken by [7,11] to automatically generate experience critiquing sessions based on the behaviour of rational users, using the standard approach to critiquing described in Section 3.2. We do this by selecting a random restaurant as our *target*. From this target we automatically create a *query*, by selecting 4-8 features from the target at random, which acts as a starting point for each session. Each session begins by the recommender retrieving a best-matching

restaurant for the query. From here the 'user' must select a feature to critique. To do this we automatically select one of the features of the recommended restaurant and critique it *in the direction* of the target restaurant. For example, if the target restaurant has a price range of €20-€30 and the retrieved case has a price range of €40-€50, then if the price feature is selected for critiquing we will apply the *cheaper* (<) critique. Moreover, features are selected for critiquing based on a probability model that favours nominal and numeric features over binary features to simulate a more realistic critiquing session. Each session terminates once the target case has been recommended. We can repeat this process to generate an arbitrary number of critiquing sessions. In the case of this experiment, we generate several different queries for each of the 632 restaurant cases to generate a total of 2928 distinct critiquing sessions. These sessions range in length from 8-15 steps, with an average length of 11.42, and on average each session involves a critique of about 10 unique features. These sessions (or a random selection of these sessions) can then be used as the case base for our experienced-based critiquing technique.

4.2 Algorithms and Methodology

We are interested in comparing the performance of a *standard* critiquing-based recommendation algorithm (as described in Section 3.2) to our *experience*-based algorithm. We generate a separate set of 500 target problems as our *test set* by using the aforementioned technique to generate a query-target pair. Next, we 'solve' each target problem, simulating the actions of a rational user using: (a) standard critiquing; and (b) the experience-based approach. In the case of the latter we use different sized case bases and overlap thresholds. A target problem is deemed to be solved once the problem's target restaurant has been recommended, at which point we note the session length.

4.3 Results

The key performance issue to consider is whether the experience-based critiquing technique leads to earlier target recommendations, when compared to standard critiquing, and thus shorter sessions? If it does then this can lead to tangible benefits both for the user and for the recommendation service provider, since all other things being equal, shorter sessions mean less effort for the user and improved conversion rate for the service provider. To examine this, Figure 5 presents the average session length (across the 500 unseen test problems) for the standard critiquing and experienced-based methods. We show the performance variation across different case base sizes, ranging from 500 cases to the full set of 2928 cases; the performance of the standard critiquing method presents as a dashed straight line since it is unaffected by case base size. We differentiate the performance of the experience-based method for different overlap thresholds (see Equation 8) in order to understand the benefits, or otherwise, of more strict thresholds. We ran simulations from $t = 0$ (must have at least one overlapping critique) to $t = 48$ (must have at least seven overlapping critiques).

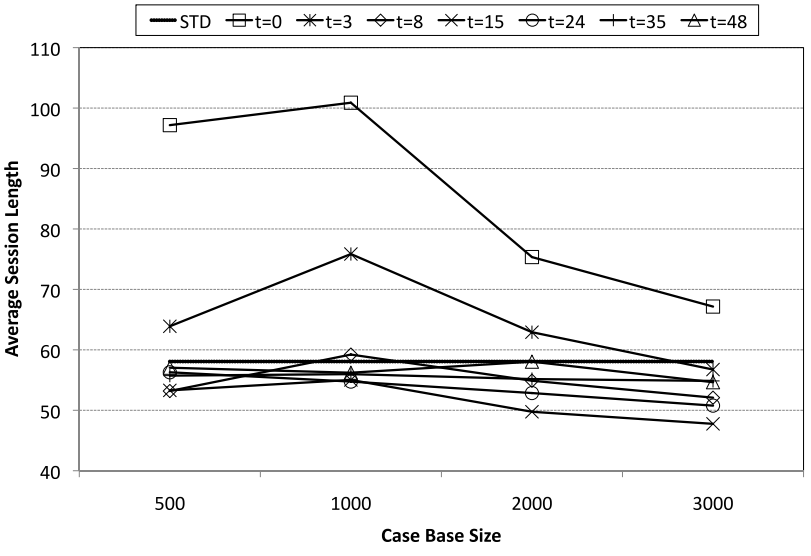


Fig. 5. The average session length results

The first point to notice is the performance of the standard approach to critiquing. On average its sessions extend to 58 steps. The experience-based critiquing techniques outperform the standard approach in some cases. Mostly these occur when a larger case base of experiences is available and when an appropriate threshold is chosen.

Across the various case base sizes results generally improve (except for case base with 1000 sessions). This is what is expected as larger case bases give more opportunity to find larger relevant overlapping experiences which leads to better case recommendation. Each individual case base (500, 1000, 2000) is made up of a random selection from the larger 3000 case-base. The sessions selected for the 1000 case base perform worse than those of 500 case base, even though the test problem coverage was 92% and 68% respectively for the two case bases. An examination of best practices for selecting previous sessions for experience cases is left for future work.

When we examine the various threshold settings for experience-based critiquing, we find that thresholds of 0 and 3 (at least one and two overlaps respectively) can not better the results of standard critiquing. Sessions with small overlaps do not generate beneficial recommendation candidates. When thresholds of 8 and greater (must have at least 3 overlapping critiques) are employed the average session lengths are lower than standard across the various case base sizes. Generally the trends improve as case base size increases. At the largest experience case base size, $t = 15$ out performs standard by over 10 cycles. This reduction of 18% is seen as a good starting point for future experience-based critiquing research.

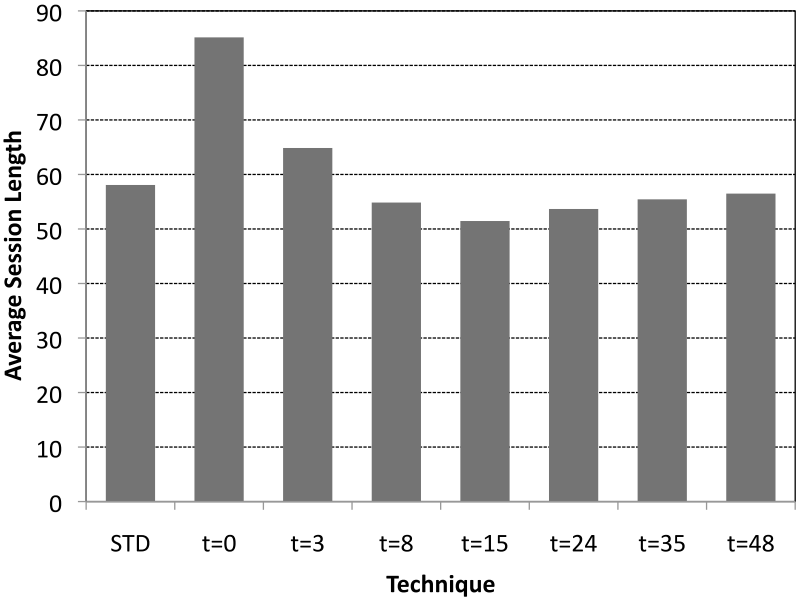


Fig. 6. Session lengths averaged across various CB sizes

For a clearer picture of the impact of varying t on session length, we have combined the various experience case base size results and presented the average session length across all of the techniques employed (Figure 6). As expected the value of t has a large effect on average session results. When t is low (at 0 and 3 - corresponding to at least 1 and 2 overlaps respectively) targets are recommended to the user that are not necessarily helpful in the long run and this leads to more protracted session lengths. However, as the threshold is made larger, the average session lengths improve and start to outperform standard critiquing. As the values of t are increased beyond $t = 24$ the average session length also begins to increase. There is a tipping point where the minimum number of overlaps acceptable starts to affect the performance of the recommender. Figure 6 shows that for this simulation dataset, recommending items from experience cases which have between 3 and 5 overlapping critiques with the current session, offer the most benefit.

4.4 Discussion

These results indicate that the experience-based critiquing can enjoy benefits compared to conventional critiquing. When we set the threshold to about four overlaps, our experience-based approach can lead to sessions that are shorter than those produced by standard critiquing. This means a cost saving for users, bringing them to the target recommendation in less steps than needed by conventional critiquing.

It is worthwhile to compare this new approach to critiquing to other approaches which aim to reduce critiquing effort. In Section 2 we described *compound critiquing*, designed to short-cut sessions by presenting users with dynamically generated compound critiques. These techniques do reduce session length but they also carry an extra overhead for the user in the form of more complex critiques to review. Indeed, in live-user trials a significant percentage of users tended to ignore compound critiques [6]. A key benefit of our new approach is that it can potentially deliver session length reductions but without introducing any new interface elements, such as extra compound critiques, and hence these benefits are likely to be available to all users and not just a fraction.

These results are based on artificial user data. Every effort has been made to ensure that these sessions are plausible — by following an offline evaluation protocol that has been successfully used in the past [7] — but they have not been generated by real users. Thus, these results should be considered as *preliminary* and need to be verified on live users. Nevertheless, the results do speak to the *potential* for experienced-based critiquing to improve recommendation efficiency.

5 Conclusions

Critiquing-based recommendation techniques are useful when it comes to helping users to navigate complex product spaces. However, they can lead to protracted sessions and a high session-failure rate for end-users. While conventional approaches have been extended to deliver more efficient recommendation sessions (e.g. [7,11,13]), these extensions typically introduce an additional cost to the user, often in the form of a more complex interface and/or feedback options. Our goal in this work has been to improve the efficiency of critiquing-based recommender systems, but without introducing additional interface components and/or costs for the end-user. To this end we have described a novel critiquing strategy, which reuses a case base of prior critiquing experiences on the grounds that these past experiences are liable to encode important users preferences and feature trade-offs that may help to improve recommendation efficiency. We have described how these past experiences can be used to influence recommendation generation and the results of an offline evaluation demonstrate the potential benefits of this experience-based recommendation approach.

Future work will focus on a number of important areas, including the examination of experience case base session generation and selection, as well as new overlap scoring mechanisms. Also, a more comprehensive evaluation of the experience-based critiquing approach, involving live-users in a realistic online recommendation scenario is needed. In addition, the technique we have presented here has been based on *successful critiquing sessions* only; that is, by design the critiquing sessions stored in the case base all represent successful recommendation sessions, where the user eventually reached their target restaurant. In reality there is the potential to include failed sessions — where a user failed to get to an acceptable restaurant — as an additional source of critiquing experience, and in the future we will consider how these experiences can also be used to bias recommendation.

Acknowledgement. This work is supported by Science Foundation Ireland grant 07/CE/I1147.

References

1. Bennett, J., Lanning, S.: The Netflix Prize. In: Proceedings of the KDD Cup and Workshop (2007)
2. Bridge, D.: Product Recommendation Systems: A New Direction. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080. Springer, Heidelberg (2001)
3. Burke, R., Hammond, K., Young, B.: The FindMe Approach to Assisted Browsing. Journal of IEEE Expert 12(4), 32–40 (1997)
4. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM 40(3), 77–87 (1997)
5. Linden, G., Hanks, S., Lesh, N.: Interactive assessment of user preference models: The Automated Travel Assistant. In: Jameson, A., Tasso, C.P.,, C. (eds.) User Modeling: Proceedings of the Sixth International Conference, pp. 67–78. Springer, Wien (1997)
6. McCarthy, K., McGinty, L., Smyth, B., Reilly, J.: On the evaluation of dynamic critiquing: A large-scale user study. In: Veloso, M., Kambhampati, S. (eds.) Proceedings of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI-2005), pp. 535–540. AAAI Press / The MIT Press (2005)
7. McCarthy, K., Reilly, J., McGinty, L., Smyth, B.: On the dynamic generation of compound critiques in conversational recommender systems. In: De Bra, P.M.E., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137, pp. 176–184. Springer, Heidelberg (2004)
8. McGinty, L., Smyth, B.: Comparison-Based Recommendation. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 575–589. Springer, Heidelberg (2002)
9. McSherry, D.: Incremental Relaxation of Unsuccessful Queries. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 331–345. Springer, Heidelberg (2004)
10. Pu, P., Faltings, B.: Decision Tradeoff Using Example-Critiquing and Constraint Programming. Constraints 9(4), 289–310 (2004)
11. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Dynamic critiquing. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 763–777. Springer, Heidelberg (2004)
12. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Incremental critiquing. Knowledge-Based Systems 18(4-5) (2005)
13. Reilly, J., Zhang, J., McGinty, L., Pu, P., Smyth, B.: A comparison of two compound critiquing systems. In: IUI 2007: Proceedings of the 12th international conference on Intelligent user interfaces, pp. 317–320. ACM Press, New York (2007)
14. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW 1994), August 1994, pp. 175–186. ACM Press, North Carolina (1994)

15. Shardanand, U., Maes, P.: Social Information Filtering: Algorithms for Automating "word of mouth". In: Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI 1995), pp. 210–217. ACM Press/Addison-Wesley Publishing Co., Denver (1995)
16. Shimazu, H.: ExpertClerk: Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. In: Nebel, B. (ed.) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001), pp. 1443–1448. Morgan Kaufmann, San Francisco (2001)
17. Smyth, B., McGinty, L.: An Analysis of Feedback Strategies in Conversational Recommender Systems. In: Cunningham, P. (ed.) Proceedings of the Fourteenth National Conference on Artificial Intelligence and Cognitive Science, AICS 2003 (2003)
18. Yang, Q., Zhang, H.H., Li, T.: Mining web logs for prediction models in www caching and prefetching. In: KDD 2001: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 473–478. ACM Press, New York (2001)
19. Zhang, J., Pu, P.: A comparative study of compound critique generation in conversational recommender systems. In: Wade, V.P., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 234–243. Springer, Heidelberg (2006)

Improving Pervasive Application Behavior Using Other Users' Information

Mike Spence and Siobhán Clarke

Distributed Systems Group
School of Computer Science and Statistics
Trinity College Dublin
spencem@scss.tcd.ie, Siobhan.Clarke@scss.tcd.ie

Abstract. The behavior of a pervasive application is much improved with access to accurate, relevant information. While other users' devices are a promising source of current information for pervasive applications, the relevance of information describing other users is not always apparent. To date, CBR has been successfully used to select information of relevance from the previous experience of the application's user. This paper describes how CBR techniques can be used to select accurate, relevant information from other users as well. We address the challenges that arise due to the set of other users from which information is available being dynamic and potentially sparse, the potential pitfalls of completely ignoring the previous experience of the application's user while using context from other users, and the elicitation of essential feedback distracting the potentially mobile user. This paper presents an examination of the use of information from other users through simulations run on three existing pervasive datasets.

1 Introduction

Pervasive applications provide behavior appropriate to a user's situation and run on devices that travel with the user or that are embedded in the environment [1]. The environments in which these applications run are constantly changing due to device mobility and evolving situations. Dynamic, pervasive applications must be aware of the environment and situation in which they operate. This capability is termed context-awareness, where context is defined as any information that can be used to characterize the situation of an entity [2]. The more accurate and relevant the context information is, the closer the application's situation can be modeled and, in turn, the more appropriate the application's behavior can be.

As an example to aid our discussion, we introduce a scenario of a commuting user (as in [3]), currently riding a bus. The user owns a device which stores and updates the user's context information via sensors or remote services. The user's current situation is modeled as a collection of attribute-value pairs with both static and dynamic context information, e.g., the user's age is 31 and the current ambient temperature is 10 degrees. There are also a number of other users in the environment, e.g., other bus passengers, drivers of adjacent automobiles,

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 495–509, 2010.

and employees in nearby office buildings, all with their own collection of context information that describes their unique situation. One particular application on the user's device adjusts the volume of an aural notification (the behavior) when an incoming message is received according to the amount of ambient noise (the context information). The consequences of inaccurately modeling the ambient noise for the user are missing a message if the notification is too quiet for the user to hear or potential embarrassment if the notification is inappropriately loud for the situation.

Acquiring accurate context information, while essential to operation, can be difficult in pervasive environments as pre-assigned sources of context information cannot always be relied upon, e.g., if the audio sensor fails or is otherwise obstructed. Sensors often degrade or fail entirely, and sources of context information that are useful in one situation are often ineffective in others, e.g., attempting to use a GPS sensor while indoors.

With equipment failures and changing situations, it is therefore necessary to have adaptable methods of context acquisition. In this paper we will focus on general, knowledge-light acquisition methods as we are not assuming any predefined knowledge in the face of the potentially highly dynamic situations of pervasive applications. One such method is *context selection*, where the context information desired by the application's user is chosen from a source according to certain criteria such as accuracy, trustworthiness, timeliness, and retrieval cost [4,5]. Another method is high-level data fusion, where current and historical context information is used to derive the desired context information. These include reasoning methods such as Naive Bayes, Dempster-Shaffer, Decision Trees, and instance-based methods like Case-Based Reasoning (CBR) [6].

We define *other users' context information* as information on other devices that describes those devices' users. Other users' context information is a promising source of information for the application's user for a variety of reasons. The devices of other users may have context information from sensors that are not present or are no longer functioning appropriately on the device of the application's user. Additionally, these devices' context information is especially useful when an application is functioning in an unfamiliar situation. *Unfamiliar situations* are those in which the application's knowledge, reasoning, and experience are insufficient to aid accurate context acquisition. The context information from other devices can be useful in these situations as they are derived from different knowledge, reasoning, and experience than the application's context. Other users' context information is particularly useful in unfamiliar situations if the other users are familiar with operating in the current environment. Unfortunately, the relevance of the context information describing other users is not always apparent. In our scenario, for example, using the ambient noise from users in adjacent automobiles or nearby office buildings might not be appropriate for the user on the bus.

To attempt to alleviate these difficulties, this paper presents a context acquisition method that applies the relevance-estimation of Case-Based Reasoning (CBR) techniques to select the most relevant information from the collection

of other users' context information. This technique augments a context acquisition method with the capability to select relevant, accurate context information. Specifically, it utilizes the information describing other users to help an application navigate situations that are unfamiliar to it but might be common to the other users. Additionally, our method attempts to deal with the fact that the set of other users from which information is available is dynamic and potentially sparse in pervasive environments.

The remainder of this paper describes related work including context selection methods and existing pervasive applications that use CBR (Section 2). CBR's potential use for selecting relevant context information from other users is presented along with our acquisition method to do so (Section 3). The experiment setup is detailed (Section 4), and the results of the experiments are presented and analyzed (Section 5). Finally, a summary and discussion of future work (Section 6) conclude the paper.

2 Related Work

While the diversity and range of the tools available on other users' devices make their potential great for aiding more accurate context acquisition, they also make it more difficult to determine the relevance of the context information to the current situation of the application's user. There are many context selection methods that consider context information from remote devices, e.g., [4,5]. These generally select the appropriate context information using utility functions that maximize criteria such as precision, communication cost, trustworthiness, and timeliness. Timeliness gives higher preference to more recent context information. Beyond the consideration of timeliness, however, they all assume that the context information from these devices explicitly describes the current situation of their user. This is not always a valid assumption as the context information describing other users can take on a diverse range of values that cannot be guaranteed to be similar to the application's actual value. Even when considering values from the application user's previous experience and not from other users, the most recent information is highly favored even though less recent information might more closely describe the user's current situation, e.g., a situation identical to one that occurred the previous day. Additionally, the selection methods that select from only remote devices do not perform well in the sparse environments that are common for pervasive applications.

Some techniques that can aid in determining relevance of context information to a situation beyond timeliness are those of Case Based Reasoning (CBR). Case-based reasoning [7] is defined as "... [solving] a new problem by remembering a previous similar situation and reusing information and knowledge of that situation[7]." CBR has been shown to handle context information in a natural manner, i.e., representing cases as a collection of context information, called *attributes* [8]. When applied to context acquisition, the "problem" is the type of context information desired by the application along with the other context attributes of the user's current situation, and a "previous similar situation" is the

vector of context attributes describing a previous situation of the user that is similar to the situation described by the problem. We refer to the vector of context attributes that describes a situation at a specific point in time as a *situation instance* (i.e., a *case* in CBR terminology).

Specifically, the existing context acquisition methods using CBR employ a global similarity measure to determine the relevance of each desired context from a previous situation by comparing the overall similarity of the context attributes of each previous situation instance to the context attributes of the current situation. The overall similarity is a result of the *global similarity measure*, which is a weighted sum of the local similarity measures for a given set of context attributes, e.g., location, role, and temperature. A *local similarity measure* compares the similarity of a single attribute, e.g., comparing the similarity of two locations or two roles. It is assumed that each context attribute has a local similarity measure, regardless of how it is implemented. There can be multiple and varied implementations of local similarity measures depending on the type of attribute and application domain, e.g., distance-based methods, such as Euclidean or city block distances, or via a similarity table for categorical attributes [9].

There are a few pervasive computing applications that utilize case-based reasoning and the context information of other users to help the user's application function. The LISTEN Project [8] presents a mobile, context-aware application for a museum that aims to enrich a patron's experience by providing them with automatic exhibit recommendations derived from an underlying CBR framework. There is no automatic support for unfamiliar situations, as the cases must be manually elicited per new exhibit, e.g., the type of exhibit and location. There is also no discussion of learning the weights of the global similarity measure. While making assumptions about the importance beforehand may be sufficient for this application, in many pervasive applications the importance of the similarity of each attribute may change dynamically based on the actual situation. Global similarity weight learning methods generally require supervised learning and this often takes the form of user feedback from experts [10]. Unfortunately, manual elicitation of cases and user feedback requires an expertise of which the user is not always capable and distracts the user from his main task.

AmICREEK [11] is a pervasive architecture that uses knowledge-intensive CBR to identify the current situation and recommend appropriate tasks for those situations. "Knowledge-intensive" here means that the CBR process is aided by general domain knowledge. While they demonstrate the utility of using CBR to determine the current situation in a healthcare domain, all of the cases were elicited manually and there is no discussion of learning attribute importance.

Finally, the personalized route planning application in [12] uses CBR to determine a user's preferred route between two locations. Route planning problems in geographical areas that are unfamiliar to the application's user are solved by identifying other users that have taken similar previous routes to routes taken by the application's user and that also have experience in the unfamiliar area of the problem. The most favorable other user is then tasked with providing a

solution to the route planning problem. Their approach is specific to one type of context, i.e., routes, so feedback for weight learning of each context type's importance is not applicable. They also require manual elicitation of preferred routes or an initial period for all users that are participating to create a store of previous routes in order for the similarity measure to function appropriately.

In summary, in completely unfamiliar situations current pervasive applications using CBR techniques and other users' information require a user to label new situations or add them manually. Without automatic case elicitation and automatic feedback for weight learning methods, the amount of time and expertise required of the user for these actions can be significant. Additionally, even if the user is willing and capable of manually providing appropriate new cases and feedback, this is not always possible in pervasive applications due to the potential danger of distraction while the user is mobile.

3 Automatic Context Acquisition for Pervasive Applications in Unfamiliar Situations

We present an automatic context acquisition method that meets the challenges listed in the last section by combining the selection of other users' context information with existing CBR techniques to improve the behavior of pervasive applications in unfamiliar situations. Our method utilizes the CBR techniques of a global similarity measure to determine the relevance of context information and automatic learning methods that do not require user feedback for learning the measure's weights. Additionally, the equipment, experience, knowledge, and reasoning of other users in the environment are indirectly used by the sharing of situation instances to add unfamiliar situations to the situation instance set, precluding the necessity of case elicitation or situation labeling from the user.

The CBR techniques of identifying the most relevant context information via the global similarity measures and automatic learning of their weights improve the existing selection methods' assessment of relevance by using situation similarity rather than just timeliness. This allows them to better consider the relevance of other users as well as to provide a more sophisticated consideration of the previous experience of the application's user. This expands the number of situation instances available. This outcome is especially important in sparse and disconnected environments, when the collection of available other users might not provide sufficient problem-space coverage for accurate context acquisition.

The automatic consideration of other users' context information augments existing pervasive applications using CBR by providing a new set of situation instances. These situation instances from other users also acting in the environment provide instances that cover situations potentially unfamiliar to an application's user. This allows the applications to function in a greater number of unfamiliar environments also requiring the users to label or eliciting entirely new cases from them. These situations and labels can come from a variety of means specific to the individual other user, such as predefined, domain-specific knowledge and reasoning, previous experience in that environment, different or

functioning sensors and communication capabilities, or user feedback. In this way, the situation instances available become a collective, distributed case base that covers more of the problem space.

Finally, the dynamic weight learning algorithm that eliminates the need for eliciting user feedback is discussed in the next subsection.

3.1 Weight Learning Algorithm

For learning the relative importance of each attribute's similarity in the global similarity metric, the weight learning component learns the weights directly from a situation instance set without the aid of user feedback. To automatically perform the weight learning methods, a leave-one-out approach where the removed instance from the instance set is the query and the remaining instances are judged based on their usefulness in handling that query.

The algorithm is a specialization of the weight learning algorithm presented by Stahl [10]. Stahl's algorithm performs an iterative search for the weights that best minimize their average error function using a conjugate gradient algorithm. The error function is based on the difference between the ideal ordering for a set of queries on a case base and the ordering that the weights given by the current iteration return. The ideal ordering is given by a "teacher" that is able to order cases according to their utility in solving each query. The "teacher" is deliberately vague as it is assumed to be application-specific which means it can be anything from a known utility function to feedback from a human domain expert. We use the local similarity measure of the desired context attribute as the "teacher".

To eliminate the necessity for user feedback, the weight learning algorithm uses the fact that the context information desired by the application has a local similarity measure and that the value of the desired context information for the training instance is known. This similarity measure can be used as an assessment of how well the learning method is performing. Adding a local similarity measure for the desired context information is given for free as it is already assumed that each attribute has a local similarity measure in order to apply the global similarity measure. Acquiring the feedback directly from the context information instead of the user also eliminates the challenges associated with acquiring user feedback in mobile, pervasive environments.

4 Experiment Setup

The overall goal of the experiments is to measure the benefit of using other users' context information for satisfying a pervasive application's context acquisition needs. We compare the differing experiment permutations of methods (i.e., random and relevance) and instance sets (i.e., the application user's previous experience, other users, and both instance sets simultaneously) on how accurately they can be used to model the user's situation. We use three benchmark datasets from pervasive environments to provide the underlying data for our simulations.

For each test we estimate the desired context value using the application user's current and historical context information in addition to the current context information from other users. We then compare the estimated value to the actual value to determine the average estimation error of each experiment permutation over all the attributes.

4.1 Datasets and Assumptions

A major challenge facing evaluations in pervasive environments is the lack of benchmark datasets common to older, more established branches of computer science, especially AI. An initial attempt to rectify this is the collection of datasets introduced in a workshop at Pervasive 2004: Benchmarks and a Database for Context Recognition.[1] Most of these datasets track the context information of a single user performing a scenario, such as travelling along a path or performing a sequence of tasks. Often the user repeats the scenario a number of times, each referred to as a run. The dataset may then have a single user performing the different runs and scenarios or it may have multiple users doing so (although not simultaneously).

Unfortunately, none of these are multi-user, simultaneous datasets. Such datasets track the situations of multiple users acting simultaneously in an environment. Multi-user, simultaneous datasets are vital for research questions like ours that pertain to other users' context information. The chief problem in generating these datasets is the deployment expense in terms of users and equipment. To overcome this limitation, we present a treatment of the existing datasets in order to simulate multi-user, simultaneous datasets.

Table 1. Information and statistics for each dataset

Dataset Name	Scenarios	Total Runs	Average Iterations	Attributes (Nominal)	Description
Nokia	2 (direction)	41	5,132.1	9 (3)	User traveling to and from work
Locomotion	4 (user)	13	34,307.5	11(1)	Users walking along a fixed path
Mäntyjärvi	5 (activity)	240	191.9	23 (0)	Users repeating sequence of actions

We have used three datasets from the Pervasive 2004 workshop in order to run our tests: Nokia Context Data [3],[2] the Locomotion dataset [13],[2] and the Mäntyjärvi dataset [14].[3] A summary of each dataset is shown in Table 1. The datasets are unchanged except discarding an extremely short run from the

[1] http://www.pervasive2004.org/program_w5.php
[2] http://www.pervasive.jku.at/Research/Context_Database/
[3] http://www.cis.hut.fi/~jhimberg/contextdata/

Table 2. Portion of a run from the Locomotion dataset

Time	Attribute1	Attribute2	Attribute3	Attribute4	...	Attribute11
T1:	-0.305	969.4	-1.8171	1.063	...	1
T2:	-0.188	969.4	-1.8183	1.7901	...	1
T3:	0.073	969.4	-1.8159	2.4679	...	1
...	...	...	...	...	...	...
T38501:	0.258	969.4	-1.8122	2.4691	...	1

Nokia dataset. In addition, any temporal attributes, attributes that remained static throughout all runs, and complimentary attributes, i.e., attributes that could directly predict other attributes of which they were the compliment, were ignored.

Every run in all three of the datasets has the same structure, as depicted in Table 2 using an example from the Locomotion dataset. A time iteration is a point in time that has values for a number of context attributes which can be nominal (Attribute11) or continuous (Attribute1 through Attribute10), and reflect the situation of the user at that time. There are several time iterations that make up a run. The run shown in Table 2 has 38,501 time iterations.

We repurpose existing datasets to model actual multi-user, simultaneous environments as far as possible. We simulate multiple users by assuming each separate run of a dataset is a different user. We also assume that the users perform simultaneously in a single environment by giving each user the same start time and the same time intervals for recording attribute information. For example, the situation of each user at time iteration 3 is represented by the context attributes in each corresponding run next to the row T3.

Our treatment using existing benchmark datasets taken from actual pervasive environments has two key advantages over simulations that randomly generate data from probability distributions. First, it allows for realistic relationships between different types of information, e.g., temperature is highly dependent on location. Second, it allows for the information describing a user's situation to evolve over time in a realistic manner. These properties are necessary in order to run our experiments that evaluate the utility of other users' information in unfamiliar situations.

Despite these advantages, there are three main reasons why our treatment of the separate runs as simulating a simultaneous, multi-user environment does not perfectly reflect these environments. Firstly, any shared environment context that should be similar for all users, such as temperature or air pressure, will not be similar as these readings are taken at different times and in some cases from different days. Any disparity in shared environment context in our treatment can be seen as a worst-case scenario, both in that the desired context information and the attributes used to determine similarity will be farther off than would normally be the case in an actual simultaneous, multi-user environment. This means that any results that favor the remote sources involving these attributes are conservative and do so in spite of this unfavorable bias. It is worth noting that

values selected from the previous experience of the same user will be unaffected by this concern.

Secondly, by placing repeated runs of the same scenario alongside the test scenario, we are seemingly guaranteeing that there will be a similar user available from which to retrieve a similar value. Treating different runs of the same scenario as the different users is not the same as having the exact same run, however. The disparity in shared environment context listed previously and variations between runs of a scenario (especially with regard to exact time iteration that each action takes place) mean that these potentially similar runs can be markedly different. The fact that there are no static attributes, such as user age or role, in any of the datasets used makes identifying different runs of the same user even less likely. Even without this variation between runs, the presence of similar users and users doing similar tasks is not an anomaly but common among several everyday tasks, e.g., doing an activity with friends and family, collaborating with coworkers, or travelling along the same route as strangers. The objection to the presence of similar runs is also mitigated by the presence of runs from other scenarios, so that we are testing the identification of the correct context information when there are several users doing potentially very different tasks.

Thirdly, the treatment does not model the interaction between the users, i.e., how users can affect the situation of each other. While interactions are interesting phenomena to study, they are not important to a number of tasks. In fact, as seen in Table 1, all of the datasets used in this simulation involve minimal interaction with other users, i.e., performing a sequence of actions and travelling to and from work.

While not as ideal as datasets generated from actual multi-user, simultaneous environments, treating different runs of a dataset as multiple users allows us to reasonably simulate some environments fairly common to pervasive environments. The concerns are not completely forgotten in our experiments, however. Where applicable, they are discussed in the analysis presented in Section 5.

4.2 Structure of Experiments: Simulating Context Source Failure

The ability to judge the relevance of other users' context information is particularly important in unfamiliar situations. We simulate a common problem in pervasive computing where the sensor or communication channel that provides the context information stops functioning. For the application to deliver appropriate behavior, this context information must be estimated. Using the experience up until the time of the failure to estimate the current context information can often be sufficient, but as time goes on the likelihood of an unfamiliar situation where previous experience is not applicable increases.

To simulate unfamiliar situations, we divide a run into two subsets, as shown in Figure 1. Subset 1 is the previous experience instance set representing when the source of context information was still available, i.e., the context source fails just after time T_h. Subset 2 is used as the collection of test cases for which we attempt to estimate the missing value. We test the different experiment permutations to see how well they estimate the missing context information for

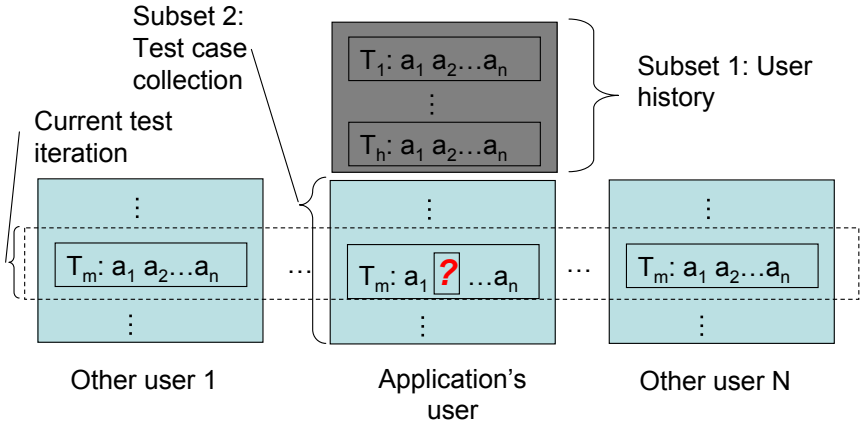


Fig. 1. Treatment simulating multi-user, simultaneous datasets

each time iteration of Subset 2, i.e., the collection of test cases. The only sources of desired context information are that of the application's previous experience (Subset 1 of the run) and the situation instances of other available users at the time iteration being tested (the dashed box in Figure 1).

We repeat these experiments for every run using a leave-one-out methodology, similar to [15], where the run left out represents the application's user and the remaining runs represent the other users. In addition, for each run we also simulate the failure of every possible attribute. For each iteration of the test we compare the estimated value to the actual value using the difference of the magnitude for continuous attributes (divided by twice the standard deviation of that context type across the entire dataset to control for type-specific variation as suggested in [16]) and classification error for nominal attributes (i.e., a value of 0 for identical values and a value of 1 otherwise). For each experiment permutation, we then average these values over all the time iterations and runs for each attribute. An example of the output is shown in Figure 2, where the lower values indicate a better overall accuracy.

5 Evaluation Results and Analysis

The experiments attempt to answer four main research questions (RQs) concerning using other users' context information. For all experiments, the weights for the global similarity measure are learned from the application user's previous experience using the specialized version of Stahl's algorithm discussed in Section 3. The local similarity measures for continuous attributes are one minus the normalized dissimilarity measure given by dividing the magnitude of the difference between the two values being compared by the magnitude of the maximum range of that attribute across all runs of the dataset. For nominal attributes, if the values are equal then the similarity is 1, otherwise it is 0.

In addition, the number of previous experience instances for all runs is 20% of the shortest run within a dataset. The number of iterations is 80% of the shortest run within a dataset, divided by the iteration time interval for that dataset. Using the shortest dataset ensures that all application users within a dataset will have an equal number of previous experience instances and an equal number of test instances.

Each of the following experiments reports on the estimation error averaged over all attributes and runs for each approach. The means for each test and dataset can be seen in Figure 2, where a more favorable approach would have a lower value, i.e., a lower estimation error. To statistically quantify the difference between approaches, ANOVA was run to determine if there was indeed a difference in overall average estimation error. Since this was the case in every experiment, Tukey's range test with a familywise error rate of 5 was then run against all pairs of means to discover which means were statistically different from one another.

RQ-1: Is using relevance to select context information from other users more accurate than existing context selection methods that do not use relevance?

This experiment tests which approach is more accurate for selecting from other users' context information: using existing selection methods that do not employ relevance or using the global similarity measure to select context information from the most similar user. As there is nothing to differentiate values from other users when not using relevance in our experiment, the existing selection methods are implemented by randomly selecting another user from which to retrieve a value.

The difference between the two approaches can be seen in the first and fourth interval in the interval plots in the left column of Figure 2. For all three datasets, the average difference between using existing selection methods on other users and using relevance to select the other user's context information significantly favors using relevance.

RQ-2: Is using relevance to select context information from other users more accurate than using relevance to select context information from previous experience?

This experiment tests which approach is more accurate: using just other users' information or using only the previous experience of the application's user. The average differences in error can be seen in the first and second intervals in the interval plots in the left column of Figure 2. They show that the answer to RQ-2 is less straightforward than RQ-1. On the one hand, the results of the Nokia and Mäntyjärvi datasets show that there can be an additional benefit to accuracy when using the context information of other users as opposed to previous experience. On the other hand, the locomotion dataset shows that this is not always the case. These differing results mean that the best instance source, i.e., previous experience or other users, for overall accuracy is dataset-specific. This would seem to indicate that knowledge to aid case dispatching [15], i.e., knowledge of the best instance source from which to choose, is required to select

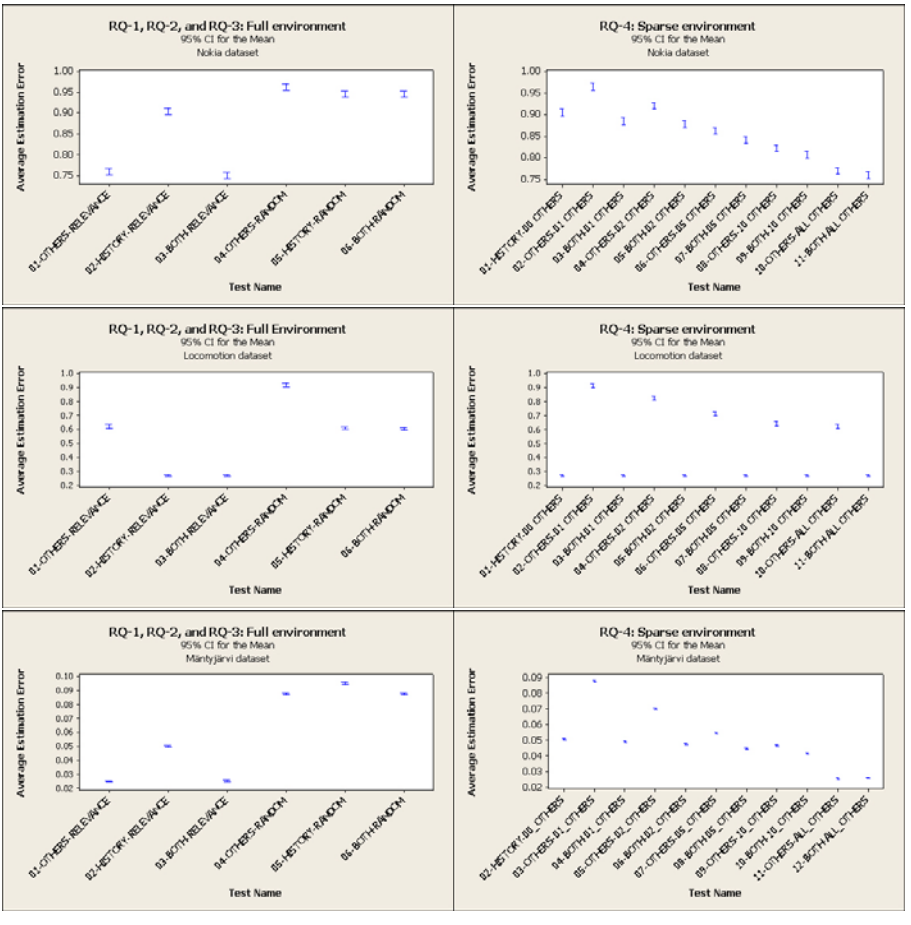


Fig. 2. Average estimation error versus approach for each dataset. The interval plots in the left column are for RQ-1 through RQ-3 and those in the right column are for RQ-4.

the most accurate overall context information. However, RQ-3 shows that such case dispatching knowledge might not be necessary.

RQ-3: What is the effect of also using the previous experience of the user in addition to the context information of other users?

This experiment answers which approach is most accurate for estimating the desired context information: using just other users' information, using only the previous experience of the application's user, or using both together. This is similar to the experiment presented in [15]. The average differences in estimation error for each dataset can be seen in the first, second, and third intervals in the interval plots in the left column of Figure 2. Originally, we hypothesized that combining the previous experience instance set along with other users' instance set would give an average estimation error lower than considering either of the two instance sets separately. While this is true across the three datasets, within each dataset the approach using both instance sets is statistically equivalent to the best individual instance set, i.e., the approach using the previous experience set in the Locomotion dataset and the approach using other users in the Nokia and Mäntyjärvi dataset. That is, the approach using both instance sets is no better and no worse than the approach using the more accurate instance set.

While the approach using both instance sets does not perform better than the best instance set on its own, it is still a useful result. It suggests that if a pervasive application is operating in an environment in which the most appropriate instance set to select from is unknown, selecting from a combination of both instance sets is equivalent to selecting the best set. While this is not as efficient processing-wise as selecting from the best set, it guarantees the lowest overall estimation error without necessitating case dispatching knowledge.

RQ-4: What is the most accurate approach for estimating context information in sparse environments, i.e., when there are a small number of other users available?

This experiment tests the utility of using only the context information of the limited number of other users versus also using the previous experience of the application's user. The experiment is run for the following number of other users: 1, 2, 5, 10, and all other users. The subset of other users is selected randomly at each iteration to give the target restricted number of users.

The results for this experiment are summarized in the three interval plots in the right column of Figure 2. The first interval is the approach using only the previous experience instances. Each subsequent interval pair shows the estimation error for sparse environments consisting of 1, 2, 5, 10, and all other users for both the approach using only other users and for the approach using both instance sets, i.e., the respective instance subset of other users and the instance set representing the previous experience of the application's user.

As in RQ-2, the datasets have very different results. The Locomotion dataset behaves exactly as expected given the results of RQ-2 and RQ-3, i.e., the estimation error using only previous experience is not affected at all by adding context information from other users. The experiments using the Nokia and Mäntyjärvi datasets also behave exactly as expected. As the environment

becomes less sparse, the error of both the approaches using the other users instance set and the approach using both instance sets decreases or gets no worse. Overall, this test shows that utilizing the user's previous experience in addition to other users' information in sparse environments does not decrease accuracy and in many cases improves it.

6 Summary and Future Work

In this paper, we have shown how the use of context information from other users can improve the accuracy of a pervasive application's context acquisition, which in turn improves the application's context-aware behavior. For our context acquisition method we have used techniques from CBR in order to identify the most relevant context information both from other users as well as from the previous experience of the application's user.

Through experiments involving three existing datasets, this context acquisition method was shown to perform more accurately than existing context selection methods. For two of the datasets the approach using other users' context information also improved the accuracy over just using the previous experience of the application's user. The estimation accuracy of these two datasets was also improved by the utilization of other users' context information even in sparse environments. The third dataset was unaffected by the addition of any number of other users. Finally, in all three datasets the accuracy of using both instance sets together was statistically equivalent to whichever the more accurate solitary instance set was. This means that the approach using both instance sets is a good choice if there is no prior case dispatching knowledge for non-sparse environments and that for these three datasets there was no benefit in accuracy beyond the best individual instance set from the potential increase in problem space coverage provided by the combination of the two sets.

While the results of the experiments performed in this paper show that other users' context information is a promising source of accurate context information for the application's user, further work is needed to determine the applicability of using other users' context information in pervasive environments. As noted, these tests should ideally be conducted in an actual multi-user, simultaneous environment or, failing that, on a larger range of pervasive datasets. Finally, while estimation error results for our experiments are favorable and selecting from other users' context information may ultimately improve accuracy, there are several other concerns that emerge as a result of using other users' context information rather than the previous experience. Privacy, trustworthiness, and the communication and processing expenses of retrieving context information from other users are all concerns that need to be weighed against any benefits of accuracy.

Acknowledgments. This work was supported, in part, by the Science Foundation of Ireland grant 03/CE2/I303_1 to Lero – the Irish Software Engineering Research Centre (www.lero.ie). We would like to thank Brett Houlding for guidance with respect to the statistical analysis of this work. We are also grateful

for the originators of the pervasive datasets used in this paper both for their creation and especially for making them public. Finally, we would like to thank the anonymous reviewers for their helpful comments.

References

1. Weiser, M.: Ubiquitous computing. Computer 26(10), 71–72 (1993)
2. Dey, A.K.: Understanding and using context. Personal Ubiquitous Comput. 5(1), 4–7 (2001)
3. Flanagan, J.A., Murphy, D., Kaasinen, J.: A nokia context recording database with synchronized user interaction. In: Benchmarks and a Database for Context Recognition: Workshop Proceedings Pervasive 2004, Linz/Vienna, Austria (2004)
4. Chantzara, M., Anagnostou, M.E., Sykas, E.D.: Designing a quality-aware discovery mechanism for acquiring context information. In: AINA, vol. (1), pp. 211–216 (2006)
5. Huebscher, M.C., McCann, J.A.: Adaptive middleware for context-aware applications in smart-homes. In: MPAC 2004: Proceedings of the 2nd Workshop on Middleware for Pervasive and ad-hoc Computing, pp. 111–116. ACM, New York (2004)
6. Das, S.: High-Level Data Fusion. Artech House, Inc., Norwood (2008)
7. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications 7(1), 39–52 (1994)
8. Zimmermann, A.: Context-awareness in user modelling: Requirements analysis for a case-based reasoning application. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 718–732. Springer, Heidelberg (2003)
9. Lesot, M.J., Rifqi, M., Benhadda, H.: Similarity measures for binary and numerical data: a survey. Int. J. Knowl. Eng. Soft Data Paradigm. 1(1), 63–84 (2008)
10. Stahl, A.: Learning feature weights from case order feedback. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 502–516. Springer, Heidelberg (2001)
11. Kofod-Petersen, A., Aamodt, A.: Case-based reasoning for situation-aware ambient intelligence: A hospital ward evaluation study. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 450–464. Springer, Heidelberg (2009)
12. McGinty, L., Smyth, B.: Collaborative case-based reasoning: Applications in personalised route planning. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 362–376. Springer, Heidelberg (2001)
13. Junker, H., Lukowicz, P., Tröster, G.: User activity related data sets for context recognition. In: Benchmarks and a Database for Context Recognition: Workshop Proceedings Pervasive 2004, Linz/Vienna, Austria (2004)
14. Mäntyjärvi, J., Himberg, J., Kangas, P., Tuomela, U., Huuskonen, P.: Sensor signal data set for exploring context recognition of mobile devices. In: Benchmarks and a Database for Context Recognition: Workshop Proceedings Pervasive 2004, Linz/Vienna, Austria (2004)
15. Leake, D.B., Sooriamurthi, R.: When two case bases are better than one: Exploiting multiple case bases. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 321–335. Springer, Heidelberg (2001)
16. Gelman, A.: Scaling regression inputs by dividing by two standard deviations. Statistics in Medicine 27(15), 2865–2873 (2008)

a.SCatch: Semantic Structure for Architectural Floor Plan Retrieval

Markus Weber[1,2], Christoph Langenhan[3], Thomas Roth-Berghofer[1,2], Marcus Liwicki[1,2], Andreas Dengel[1,2], and Frank Petzold[3]

[1] Knowledge Management Department,
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Straße 122, 67663 Kaiserslautern, Germany
[2] Knowledge-Based Systems Group, Department of Computer Science,
University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern
[3] Chair of Architectural Informatics
Faculty of Architecture
Technical University of Munich
Arcisstrasse 21, 80333 Munich, Germany
{firstname.lastname}@dfki.de, lastname@tum.de

Abstract. Architects' daily routine involves working with drawings. They use either a pen or a computer to sketch out their ideas or to do a drawing to scale. We therefore propose the use of a sketch-based approach when using the floor plan repository for queries. This enables the user of the system to sketch a schematic abstraction of a floor plan and search for floor plans that are structurally similar. We also propose the use of a visual query language, and a semantic structure as put forward by Langenhan. An algorithm extracts the semantic structure sketched by the architect on DFKI's Touch& Write table and compares the structure of the sketch with that of those from the floor plan repository. The a.SCatch system enables the user to access knowledge from past projects easily. Based on CBR strategies and shape detection technologies, a sketch-based retrieval gives access to a semantic floor plan repository. Furthermore, details of a prototypical application which allows semantic structure to be extracted from image data and put into the repository semi-automatically are provided.

Keywords: semantic building design, architecture, image understanding, case based reasoning, graph theory.

1 Introduction

During design processes, architects use existing buildings and building designs as references. These reference drawings are used to guide solutions for similar architectural situations. However, current electronic searches use textual information rather than graphical information.

The a.vista concept suggested by Langenhan [1] concerns geometrical search strategies rather than today's keyword-based search methods. The configuration

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 510–524, 2010.

of space and the relations between physical structures are hard to represent using keywords, in fact transforming these structural configurations into verbally expressed typologies tends to produce fuzzy and often imprecise descriptions of architecture. The project investigates the limits of architectural spaces represented in drawings. By recording space boundaries in a database, information about the specific project is transferred to both the descriptive world of architecture and the computer. Every building in the database features a digital fingerprint, which shows the architectural situation of the building in terms of its space boundaries and their characteristics. By providing a sketch of the required architectural configuration, the user creates a digital search fingerprint for the query. The search fingerprint can then be compared with the fingerprints in the database.

The a.vista concept includes a semantic structure describing the fingerprint of a floor plan within a graph representation. This formal representation is the foundation of the a.SCatch system. In Section 2 we discuss related work and current methods. Section 3 discusses the semantic structure used to formalise the content of a floor plan and Section 4 describes the a.SCatch system and how the semantic structure is used for similarity-based retrieval. In Section 5 we compare the results to existing projects and concepts. Section 6 presents an evaluation of the visual query language. Possible future developments and a summary can be found in Section 7.

2 Related Work

Since the middle of the 1990s the approach of applying Case-Based Reasoning (CBR) to design and architectural tasks has been known as Case-Based Design (CBD). The case-base contains information on buildings that have already been built or designed, enabling the computer to adapt solutions accordingly, on its own or with help from the architects. Two studies have been published by Heylighen in 2001 [2] and by Richter et al. in 2007 [3].

Table 1 gives an overview of CBD systems from the past 15 years of research activity in the field of Case-Based Design. The table concentrates on the development of different CBD applications with regards to the features supported by the software. The marked fields show whether the appropriate feature was realised in the application.

Six of the CBD prototypes, (CADRE [4], FABEL [5], IDIOM [6], SEED Layout [7], SL_CB [8] and TRACE [9]) aim for a partially or completely automated generation of building layouts by applying the retrieved solution. Two out of these prototypes (CADRE and IDIOM) leave the selection of the reference project to the user. The remaining four, FABLE, SEED, SL_CB and TRACE, apply the solution to the given architectural problem automatically and generate building layouts independently with only a few user inputs.

The PRECEDENTS [10] project can be seen as a counter example to these concepts. As the name already indicates, the architect is to be helped in finding reference projects. This approach is conceptually close to classical verbally driven

Table 1. Overview Case-Based Design systems

CBD application / application feature	Data Storage			Input System			Output System						
	Floor plans + text	Abstraction	Topology	Graphic	Verbal	Adaptation	Reference projects	Applying solutions	Graphical Information	Learning	Subproblems	Semantic net	Analogy
Archie-II	X	X			X		X		X		X	X	
CADRE	X	X	X	X	X	X		X	X	X	X		
FABEL	X	X	X	X	X	X	X	X	X		X		X
IDIOM		X	X	X	X	X		X	X				
PRECEDENTS	X	X			X		X		X		X	X	
SEED Layout		X			X	X	X	X	X		X		
SL_CB	X	X			X	X	X	X	X				
TRACE		X	X	X	X		X	X	X				
CaseBook	X		X	X	X		X						
MONEO	X	X			X		X		X				X
CBA	X	X			X		X					X	
DYNAMO	X	X		X	X		X		X	X		X	

architectural databases. However the graphic inputs of CaseBook [11] appear to be more suitable to formulating the retrieval query due to the visual way architects work.

An important feature of CBR is the ability of such applications to learn. This feature is addressed in CADRE and DYNAMO [12]. DYNAMO proposes a kind of manual reindexing. User input allows the parameters of the database to be changed or added to according to how it is to be interpreted. Another important feature is that of dividing a problem into sub-problems, which was realised in FABEL. This can also be an approach to deal with the ambiguity that frequently occurs in the description of architecture. By allowing meaningful fragmentation into small units, a 100% match of the total problem is no longer necessary in all cases: it is sufficient to have a 100% match of several fragments, for example a 100% match in 60% of the fragments. A measure for the similarity between the inputs and the stored projects can thus be determined. Handling similarities is brought up for discussion in MONEO [13] and CADRE. Archie-II [14] is a Case-Based Design Aiding System (CBDA) which looks at the early phase of the design process and leaves the reasoning process to the user. A semantic data representation can be found in CBA [15].

3 Methods

The use of metadata enhancing digital floor plans with additional information offers the opportunity to create smart objects that allow users to have easier access to the planning material. Enriching geometrical data with semantic information allows the application and hence the user to identify rooms, doors or chairs. Today we are looking at semantic models to describe this data representation. Examples of semantic models are BIM, IFC and the digital fingerprint. When transferring data from one format to another (interoperability), it is much easier to have smart objects than just lines or points.

When it comes to enhancing CAD models of buildings with semantic information, one of the major approaches in architecture is that of Building Information Modeling (BIM). Charles Eastman documented it in his books [16,17], and Jerry Laiserin made it popular and declared it to be an industry standard [18]. According to Eastman et al. [17], BIM is the process of generating and managing building information in an interoperable and reusable way. A BIM system is a system or a set of systems that enables users to integrate and reuse building information and domain knowledge throughout the life cycle of a building.

Furthermore, they emphasise that 3D knowledge-rich parametric modeling systems are central to BIM and in the life cycle of a building. As buildings are composed of geometric components, geometric information forms a substantial part of BIM. In addition, further domain knowledge is added to the BIM, such as project information, light analyses, or quantities and properties of building components and so forth. BIM is considered to have several stakeholders: architects, engineers, project managers and building-owners.

Modern architectural design is done using Computer Aided Design (CAD) software. Several vendors, such as Autodesk (Revit Architecture), Graphisoft (ArchiCAD) and Nemetschek (Allplan) offer software packages with their own data formats to store information about the building. However, according to BIM interoperability is important in reducing costs and supporting all stakeholders.

The International Alliance for Interoperability (IAI) [19] established an open specification that is not controlled by a single vendor. The file format Industry Foundation Classes (IFC) is an interoperable BIM standard for CAD applications.

Langenhan's proposed semantic structure [1], the digital fingerprint, is a spatial, functional, semantic structure and is used to formalise the structure of a floor plan. Langenhan introduces four main concepts to describe housing construction spaces and their relations, following the paradigm of incremental space:

1. **Room** - the most atomic structure in a formal representation,
2. **Zone** - consists of several rooms and describes the functionality of the grouped rooms, for instance a 'sleeping zone',
3. **Unit** - groups zones and also has a functional meaning, such as 'apartment' or 'terrace',
4. **Level** - the current floor level of the building.

A building and its corresponding floor plan are separated into levels. Each level is divided into multiple units, which could be an apartment or a terrace, for instance. Units can be further divided into zones. Examples of zones are the 'living zone' the 'sleeping zone' and the 'function zone'. A zone groups together different rooms which are the most atomic part of the structure.

Today however, there is not always a strict division of the function that spaces can serve. A single physical space can therefore have several functions and be a combination of multiple functional spaces, such as a living room combined with a kitchen. Table 2 provides an overview of some structural entities.

But today there is not always a strict division of the function that spaces can serve. Thus a single physical space can have several functions and thus be a combination of multiple functional spaces, such as a living room combined with a kitchen. An overview of some structural entities is illustrated in Table 2.

Table 2. Taxonomy - Example of entities

Concepts	Level	Unit	Zone	Room
Entities	Attic Floor	Circulation	Circulation Zone	Bedroom
	Upper Floor	Apartment	Living Zone	Workroom
	Ground Floor	Terrace	Function Zone	Bathroom
	Basement	Balcony	Sleeping Zone	Kitchen
				Corridor
				Staircase

After having introduced the different concepts, the connections between them need to be discussed. The instances of a concept with the same type can have either a direct, adjacent or no relation. If two spaces have a common wall and a door which links the spaces, this is defined as a connection. An adjacent relation is indicated by a shared wall without an alley.

A floor plan contains the level of a building, the root node always is a level node. The level will be hierarchically divided into units, zones and rooms via *part-of* relations. The resulting structure could be represented as a tree, but as already discussed, structural concepts of a layer either have a direct, adjacent, or no connection. This means that there are three different types of vertexes. Section 4.3 will discuss the graph structure in more details.

4 The a.SCatch System

Usability is the main aim of Human-Computer Interaction (HCI) research. It is essential for designing interfaces that allow users to work and interact with applications intuitively. Appropriate metaphors and devices must be used to allow fast and easy interaction.

Architects prefer to sketch in their initial design phase. A pen gives them more freedom than using a mouse with Computer Aided Design (CAD) software. Using the Touch&Write pen device to draw in a digital environment allows more

immediate interaction, and the architects benefit from the digital representation of their drawings.

As the pen and touch paradigm is more intuitive for an architect to use, a prototype is being implemented for the Touch&Write table [20] developed at DFKI. The Touch&Write table, illustrated in Fig. 1, combines the paradigms of multi-touch input devices and a pen input device. The table is a novel rear-projection tabletop which detects touching by using frustrated total internal reflection (FTIR) proposed by Han [21] and a high resolution pen technology offered by Anoto Group AB.

Fig. 1. The Touch&Write table

The a.SCatch system offers user interfaces for:

- Manual editing of the automatic extraction results,
- Sketch-Based retrieval.

For Sketch-Based Retrieval, a visual query language was proposed by Langenhan [1]. This abstract representation of floor plans will be described in Section 4.2. The results of the semantic retrieval are represented as graphical information and the touch interaction is a suitable way of interacting with this kind of information. The architect is able to interact with the graphical information using simple and intuitive gestures to zoom or navigate within the floor plan.

The aim of the a.SCatch project is to implement a system that takes advantage of the a.vista concept [1] and the work environment of the Touch&Write table [20]. A semantic search is realised by sketching a concept of an architectural problem and triggering a search for similar projects from the past.

Several subtasks need to be carried out:

1. Semi-automatic extraction of the semantic structure from older floor plans,
2. Extraction of the semantic structure from the architect's hand drawing,
3. Comparison of the sketch fingerprint's graph structure with graph structures in the floor plan repository,
4. Visualisation of the results and the interaction with the user interface.

A schematic overview of the system is given in Fig. 2. Whenever the architect is searching the repository, he formalises his query as a sketch, similar to the

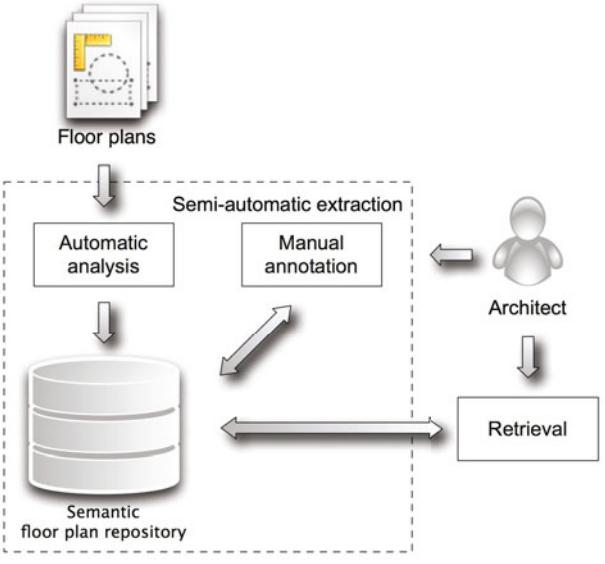


Fig. 2. Overview of the a.SCatch System

fundamental concepts of Spatial-Query-by-Sketch proposed by Egenhofer [22]. The architect sketches an initial floor plan with its associated rooms, zones and units. Afterwards the online data from the pen device is used to detect the geometrical shapes that represent concepts and lines which indicate the connection type. An example for such a visual query is illustrated in Fig. 3, which is a schematic way of drawing a floor plan. It describes an abstraction of geometrical relations and functional coherences. A verbal description of the different shapes is essential to get reasonable search results.

For the purposes of searching the repository and assessing the similarities between the graph extracted from the architect's hand drawing and the graphs from the repository, a similarity measure must be calculated. In graph theory this can be interpreted as subgraph matching. Section 4.3 discusses the basic theory of the search problem.

The results of the semantic search are represented as graphical information and the touch interaction is a suitable way of interacting with this kind of information. The architect is able to interact with the graphical information using simple and intuitive gestures to zoom or navigate within the floor plan. Furthermore, visualisation techniques such as Semantic Zooming[1], could be applied to present more detailed information depending on the zoom level.

[1] In semantic zooming, objects change appearance or shape as they change size. For example a growing dot will become a simple box, then a box with a one-word label, then a box with a longer label, then a rectangle filled with text and pictures. The goal is to give the most meaningful presentation at each size. `http://www.infovis-wiki.net/index.php/Semantic_Zoom`: Last accessed 04/02/2010

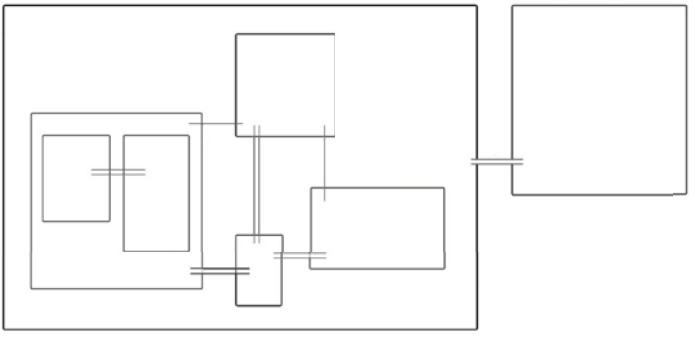


Fig. 3. Example of an abstraction of a sketched query

Further work will involve offering the architect the freedom to choose between the proposed visual query language or to let him sketch an initial floor plan with his own visual language and extract the semantic graph structure directly from this sketch.

4.1 Semi-automatic Extraction from Existing Projects

In this context, semi-automatic extraction is defined as a process whereby the system tries to extract features automatically from a floor plan by using image understanding techniques [23,24,25] and apply machine learning methods that classify the structural information. Another approach is to use standardised data formats, already containing meta information about the floor plan, such as the IFC standard[2].

This semi-automatic extraction process is a seamless procedure which can be divided into four steps:

1. Vectorising the pixel graphics,
2. Interpretation of vector information, such as lines and arcs,
3. Generating room and connection hypotheses,
4. Storing the digital fingerprint represented by a graph structure to enable searching and manual editing.

The extracted structure is presented to the architect who then approves or modifies the results via a pen- and touch-enabled interface. With the support of semi-automatic extraction, an architect is able to formalise knowledge about past projects. This formalisation process comprises two phases. The automatic analysis is the first part of the analysis and is mainly based on the techniques discussed in [23]. Currently the focus is on the detection of single rooms and their interconnections. Future work will involve classifying the type of the room by using symbol recognition [26] and optical character recognition (OCR) for

[2] `http://www.iai-tech.org/products/ifc_specification/ifc-releases/summary`: Last accessed 04/02/2010

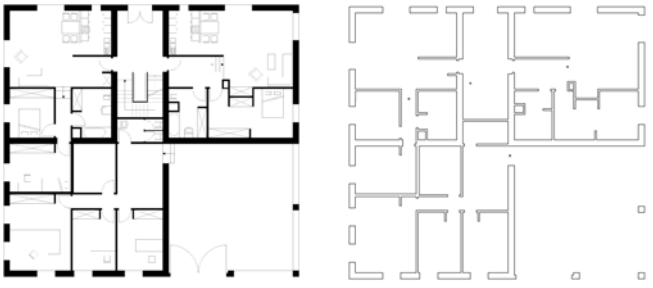


Fig. 4. Example of wall detection

each piece of textual information [27]. Furthermore a rule-based system could be applied in order to group rooms into zones and zones into units. The floor plan with its extracted semantics is stored in a repository and a user interface is provided to manually annotate the floor plan. The first results of the wall detection are illustrated in Fig. 4.

4.2 Sketch-Based Retrieval

As we are dealing with visual information and an exclusively textual description of a floor plan is too fuzzy, we propose that a visual query language be used. Whenever the architect is searching the repository, he formalises his query as a sketch, as in the fundamental concepts of Spatial-Query-by-Sketch proposed by Egenhofer [22]. Initially the architect sketches a floor plan with its associated rooms, zones and units. The corresponding online data from the pen device is used to detect the geometrical shapes representing concepts and lines, which indicate the connection type.

The schematic abstraction in Fig. 3 shows the floor plan translated into Langenhan [1]'s proposed semantic structure. The entities listed in Table 2 are represented by rectangles. Enclosing rectangles are interpreted as a part-of relationship. For instance, if a rectangle R_1 encloses another rectangle R_2 and R_3, it indicates that R_2 and R_3 are part of R_1, such as a sleeping zone which contains two bedrooms. How the units, zones and rooms are connected with each other is indicated by lines connecting the rectangles.

As discussed in Section 3 two different connection types have to be considered, because two entities are either adjacent to one another or directly connected. In the schematic view this is indicated by two parallel lines if the entities are directly connected, or one line if they are only adjacent to one another. Hence, no line between two entities indicates no connection or adjacent relation. By interpreting the sketch, a graph structure can be extracted for our query. For the shape detection we used the Vision Objects shape detection algorithm[3]. But as the shape detection is single stroke detection, we had to add a preprocessing

[3] `http://www.visionobjects.com`: Last accessed 04/02/2010

step which combines single strokes following simple heuristics. To summarise the semantics of the visual query language:

- **Rectangles** represent **structural entities**,
- **Enclosing** implies **part-of relation**,
- **Single lines** indicate an **adjacent connections**,
- **Two parallel lines** indicate a **direct connection**.

4.3 Graph Structure

In this section we briefly discuss the theoretical foundation of the retrieval. The extracted semantics are represented as a graph $G = (V, E)$. The vertexes V have a different type T_{vertex} which reflects a level, unit, zone or room. Each of these types has finite set of subtypes (cf. example of entities depicted in Table 2). As the types are hierarchically ordered $T_{level} > T_{unit} > T_{zone} > T_{room}$, the resulting graph is limited in its "'vertical depth"' to the maximum depth of four.

The vertexes E have also different types T_{vertex} indicating if the vertexes are connected directly or are just adjacent, both of these relations are symmetric. The *part_of* relation indicates which vertex adheres to a vertex of a superior type T_{vertex}, for instance a sleeping room which is part of a sleeping zone. As we are dealing with undirected and directed vertexes the graph is a mixed graph.

The types of the node and vertexes are assigned by labelling functions $\alpha : V \rightarrow T_{vertex}$ and $\beta : E \rightarrow T_{vertex}$.

For the retrieval, the query graph G has be compared with the set of graphs from the database. In order to calculate the similarity of a query graph and a database graph the edit cost could be applied. In graph theory a similar problem is known as *Maximum Common Subgraph-isomorphism (MCS)* [28,29], which is known to be NP-hard. Thus the number of matched vertexes is a potential measure of similarity.

Our current work focuses on researching standard approaches for solving the MCS problem and the subgraph isomorphism respectively. As the resulting graph structure has some limitations, such as the limited 'vertical depth"', and the structure among different floor plan graphs is similar, we are focusing on finding heuristics to overcome the intractable character of the problem.

5 Comparison with Existing Systems

First of all we must state that no significant breakthrough has occurred in Case-Based Design yet. The two studies published by Heylighen [2] and by Richter et al. [3] discovered that the acquisition bottleneck [30] of appropriate planning material has yet to be solved. We propose a semi-automatic extraction of the semantic structure to address this limitation shared by all CBD applications, as the content of a database is actually the most important part for the user.

However, we are not attempting to solve all the problems at once. The a.SCatch concept is simple and well defined, using architectural drawings as its starting

Table 3. a.SCatch features

CBD application / application feature	Data Storage			Input System			Output System						
	Floor plans + text	Abstraction	Topology	Graphic	Verbal	Adaptation	Reference projects	Applying solutions	Graphical Information	Learning	Subproblems	Semantic net	Analogy
a.SCatch	X	X	X	X	X		X		X		X		X

point. 'Room' 'zone' 'unit' and 'level' are organised within a topology enriched with relationships and typologies.

Furthermore we create a fingerprint of the project which can be understood as the equivalent of keywords in today's verbally driven search engines. It is the signature of the planning material and will allow the user to recover the appropriate project. By observing the work process of architects we are able to use experiential knowledge.

Unlike most CBD applications, the application we propose assists architects during the design process rather than creating the design itself. The main focus is on getting the projects into a database and recovering them in the easiest, fastest and most intuitive way possible. Doing things intuitively means creating a sketch-based query on the Touch&Write table. With regards to the way architects work, it is a consistently Cased-Based Design concept. Table 3 summarises the features of the a.SCatch system and Section 6 presents results of an initial evaluation of our implementation.

6 Experiments and Results

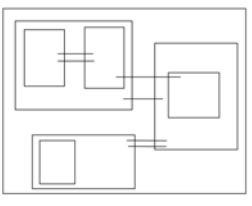
Sketch-based interactions are the essential part of the a.SCatch system. A possible scenario of sketching a floor plan would involve using visual query language, thus the first experiments focus on interpretation of the visual language. As the a.SCatch system generates the query from the architect's hand-drawing, we were able to evaluate the accuracy of the shape detection. We therefore defined ten example queries covering different complexity levels (see Table 4) and asked ten participants to copy each of these sketches, resulting in a total of 100 sketches.

The participants were male and female students aged between 23 and 29 years. All sketches were drawn on the Touch&Write table and the hand drawing was recorded. To assess the pure recognition performance, we did not give a direct feedback of the recognised shapes. In order to measure the accuracy of the detection algorithm we counted the correctly detected quadrangles and connections.

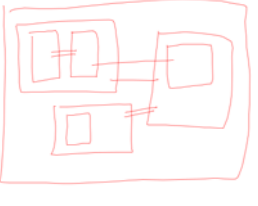
Figure 5a shows a template of Query 1 and Fig. 5b shows the recorded hand-drawing of Participant 1. The shapes detected by the Vision Objects shape detection algorithm are illustrated in Fig. 5c.

Table 4. Complexity and Detection rate for each query

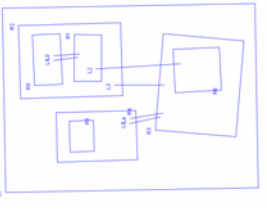
Query	Quadrangles		Adjacent Connections		Direct Connections	
1	#	Corr	#	Corr	#	Corr
1	8	0.96	2	0.95	2	0.8
2	5	0.96	2	09	3	0.9
3	8	0.93	2	0.95	5	0.7
4	6	0.95	3	0.93	1	0.7
5	3	1.0	2	1.0	1	1.0
6	9	0.99	3	0.97	5	0.96
7	6	0.98	2	0.95	4	0.65
8	9	0.99	4	0.93	4	0.9
9	10	0.97	1	0.8	4	0.9
10	3	0.97	1	0.9	2	0.9
Overall		0.97		0.93		0.86



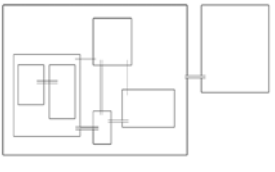
(a) Query 1.



(b) Hand-drawing of the Participant 1.



(c) Detected shapes.



(d) Query 3.

(e) Hand-drawing of the participant 2.

(f) Unrecognised strokes.

Fig. 5. Visualisation of the results

For the evaluation we distinguished between detection rates for quadrangles, adjacent and direct connections. Whenever a quadrangle is not detected it could cause a misinterpreted connection between quadrangles, which is also counted as a false detection. Table 4 shows detection rates for each query and Tab. 5 the detection rates for each participant.

An example of a false detection is given in Fig. 5f. Several participants had different strategies of drawing rectangles, and the current methods seem not to

Table 5. Detection rate for each participant

Participant	Quadrangles	Adjacent Connections	Direct Connections
1	1.0	1.0	0.97
2	0.96	0.82	0.81
3	0.93	0.82	0.77
4	0.91	0.95	0.74
5	0.99	1.0	0.87
6	1.0	0.9	0.97
7	0.97	0.95	0.83
8	0.97	0.95	0.84
9	0.99	0.95	0.94
10	0.97	0.91	0.84

cover all of them. The lines do not merge together, especially if a long pause occurred during the sketching of a rectangle. Figure 5f also shows that none of the rectangle's connections were recognised.

The detection algorithm needs further improvement by using more sophisticated methods such as dynamic programming [31] to achieve results close to 100%. The final system will also be interactive, hence the user will be able to correct his input whenever the shape detection fails.

7 Conclusion and Future Work

In this paper, we presented a retrieval system for searching floor plans using a sketch-based interface. The first evaluation results show that shape detection already produces reasonable detection results, though there is still room for improvement. Dynamic programming approaches or a combination of online and offline detection will lead to results close to 100%.

Another focus is the automatic analysis of existing floor plans. Improvements in the automatic phase of the extractions should help when processing large amounts of data, as it should mean less effort in correcting the results. We used pattern recognition methods to solve the knowledge elicitation bottleneck [30] of existing CBD Systems.

Furthermore, we proposed a graph-based semantic structure to capture the content of floor plans inside the digital fingerprint. The retrieval is based on space configurations and semantic descriptions. This representation of architectural data in semantic models is helpful to the architects using the information further on. It will be possible to use the data within the life cycle of a building.

Combining different technologies will balance the inabilities of each one. Technologies such as *semantic-based query* or even *picture-based query* will help to upgrade the design process. It is the master key to allow architects access to the digital information society. They have not got there yet.

References

1. Langenhan, C.: a.vista Semantische Suche Web 3.0 für die Architektur, Bauhaus. Universität Weimar (2008)
2. Heylighen, A., Neuckermans, H.: A case base of case-based design tools for architecture. Computer-Aided Design 33(14), 1111–1122 (2001)
3. Richter, K., Heylighen, A., Donath, D.: Looking back to the future - an updated case base of case-based design tools for architecture. In: Knowledge Modelling - eCAADe (2007)
4. Hua, K., Faltings, B., Smith, I.: Cadre: case-based geometric design. Artificial Intelligence in Engineering 10, 171–183 (1996)
5. Voss, A.: Case design specialists in fabel. In: Issues and Applications of Case Based Reasoning in Design, pp. 301–335 (1997)
6. Lottaz, C., Stalker, R., Smith, I.: Constraint solving and preference activation for interactive design. Artif. Intell. Eng. Des. Anal. Manuf. 12(1), 13–27 (1998)
7. Flemming, U.: Case-based design in the seed system. Automation in Construction 3(2-3), 123–133 (1994)
8. Lee, J., James, P., Garrett, H., Stephen, P., Lee, R.: Integrating housing design and case-based reasoning (2002)
9. Mubarak, K.: Case based reasoning for design composition in architecture. PhD thesis, School of Architecture. Carnegie Mellon University (2004)
10. Oxman, R., Oxman, R.: Precedents: memory structure in design case libraries. In: CAAD Futures 1993: Proceedings of the Fifth International Conference on Computer-Aided Architectural Design Futures, Amsterdam, The Netherlands, pp. 273–287. North-Holland Publishing Co., Amsterdam (1993)
11. Inanc, B.S.: Casebook. an information retrieval system for housing floor plans. In: CAADRIA 2000, Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia, pp. 389–398 (2000)
12. Heylighen, A., Neuckermans, H.: Dynamo: A dynamic architectural memory online. Educational Technology & Society 3(2) (2000)
13. Taha, D.S.: Moneo, An Architectural Assistant System. PhD thesis, University of Alexandria, Egypt (2006)
14. Domeshek, E.A., Kolodner, J.L., Zimring, C.M.: The design of a tool kit for case-based design aids (1994)
15. Lin, C.-Y., Mao-Lin, C.: Smart semantic query of design information in a case library. digital design - research and practice. In: Proceedings of the 10th International Conference on Computer Aided Architectural Design Futures, pp. 125–135 (2003)
16. Eastman, C., Teichholz, P., Stalker, R., Kolodner, L.: BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors. Wiley, Chichester (2008)
17. Eastman, C.: Building product models: computer environments supporting design and construction. CRC Press, Boca Raton (1999)
18. Ubiquity: Interview with Jerry Laiserin (2005)
19. Interoperability, I.A.F.: Building Smart (2010)
20. Liwicki, M., El-Neklawy, S., Dengel, A.: Touch & Write - A Multi-Touch Table with Pen-Input. In: Proceedings International Workshop on Document Analysis Systems (to appear, 2010)
21. Han, J.Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: UIST 2005: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology, pp. 115–118. ACM, New York (2005)

22. Egenhofer, M.J.: Spatial-query-by-sketch. In: VL 1996: Proceedings of the 1996 IEEE Symposium on Visual Languages, Washington, DC, USA. IEEE Computer Society, Los Alamitos (1996)
23. Dosch, P., Tombre, K., Ah-Soon, C., Masini, G.: A complete system for the analysis of architectural drawings. International Journal on Document Analysis and Recognition 3(2), 102–116 (2000)
24. Lu, T., Yang, H., Yang, R., Cai, S.: Automatic analysis and integration of architectural drawings. International Journal of Document Analysis and Recognition (IJDAR) 9(1), 31–47 (2006)
25. Or, S.h., Wong, K.h., Yu, Y.k., Chang, M.M.y., Kong, H.: Highly Automatic Approach to Architectural Floorplan Image Understanding & Model Generation. Pattern Recognition (2005)
26. Tabbone, S., Wendling, L., Tombre, K.: Matching of graphical symbols in line-drawing images using angular signature information. International Journal on Document Analysis and Recognition 6(2), 115–125 (2003)
27. Tombre, K., Tabbone, S., Pélissier, L., Lamiroy, B., Dosch, P.: Text/graphics separation revisited. LNCS, pp. 200–211. Springer, Heidelberg (2002)
28. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters 18(8), 689–694 (1997)
29. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
30. Watson, I., Farhi, M.: Case-based reasoning: A review. The Knowledge Engineering Review (1994)
31. Feng, G., Sun, Z., Viard-Gaudin, C.: Hand-drawn electric circuit diagram understanding using 2D dynamic programming. In: Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition, Québec, pp. 493–498 (2008)

Runtime Estimation Using the Case-Based Reasoning Approach for Scheduling in a Grid Environment

Edward Xia[2], Igor Jurisica[1], Julie Waterhouse[2], and Valerie Sloan[2]

[1] Department of Computer Science, University of Toronto, and Ontario Cancer Institute, Toronto, Canada
juris@cs.toronto.edu
[2] IBM Canada Lab, IBM Canada Ltd., Toronto, Canada
{xiaeg,juliew,vsloan}@ca.ibm.com

Abstract. Grid scheduling performance is significantly affected by the accuracy of job runtime estimation. Since past performance is a good indicator of future trends, we use a case-based reasoning approach to predict the execution time, or run time, based on past experience. We first define the similarity of jobs and similarity of machines, and then determine which job and machine characteristics affect the run time the most by analyzing information from previous runs. We then create a case base to store historical data, and use the *TA3* case-based reasoning system to fetch all relevant cases from the case base. We apply this approach to schedule Functional Regression Tests for IBM® DB2® Universal Database™ (DB2 UDB) products. The results show that our approach achieves low runtime estimation errors and substantially improves grid scheduling performance.

Keywords: Case-based Reasoning, Grid Computing, Job Scheduling, Runtime Estimation.

1 Introduction

Grid scheduling is essential in achieving effective use of resources in a grid environment [4]. Diverse scheduling algorithms have been proposed and developed [2], [5], [6], [20], [22]. These algorithms usually assume that the time a job will take to run on a machine is known *a priori*. However, in the real world, it is difficult to precisely determine the actual run time of a job in advance, even if this job has already been executed in the past. The difficulty of actual runtime estimation is due to the dynamic nature of the resources in the grid [26]. With a large grid, it is unlikely that the same job will be executed on the same machine multiple times. Even if a job does run on the same machine again, the running environment, e.g., the workload of the machine, might be different. Estimating new job run times poses additional challenges. A possible solution is to compute the approximate run time from historical information [7], [23], use estimates from developers who create jobs, or combine the two approaches. Since past performance can be a good indicator of future trends, we can estimate a job run time using a case-based reasoning (CBR) approach [15]. We focus on runtime estimation for long-term applications in heterogeneous systems.

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 525–539, 2010.

To use the CBR approach, we first need to design a case base, which is a repository of historical data. Here, a case is a record that contains job information, machine information, and job run times. Job and machine information is used to match similar cases, and job run times are used to calculate the estimated run time. The similarity of cases depends on both the similarity of jobs and similarity of machines.

Since a job and a machine may have many characteristics, we must decide which characteristics significantly affect the similarity of jobs and similarity of machines, and thus should be included in the case representation. In addition, as job and machine characteristics are prioritized in order to search similar cases effectively, we need to determine the priority of each characteristic. For the priorities of job characteristics, we develop a general approach by using a modified k-nearest neighbor algorithm [21] to check the runtime standard deviation of each characteristic. A characteristic with a low standard deviation has a high priority. For the priorities of machine characteristics, we change the value of each machine characteristic and keep the values of other characteristics unchanged, and check how the run time is affected. A characteristic with a high runtime change has a high priority.

To estimate the run time of new jobs, we find similar jobs, or cases, from the case base and fetch their actual run times. We can apply several machine learning algorithms to match cases, including k-nearest neighbor, Naive Bayes, neural networks, or decision trees [17]. The k-nearest neighbor (KNN) approach finds the *k* closest objects based on distance measure and predicts the output for an input object. KNN is suitable for our application since it has good classification accuracy performance and can scale to large repositories. In order to use KNN effectively, we need to overcome its disadvantages. For example, we use groups of weighted attributes to diminish KNN performance degradation with many irrelevant attributes, and we use relaxation as a surrogate measure of similarity in the absence of clear attribute value hierarchies.

In the KNN approach, two cases may be matched exactly or closely based on available cases in the case base. The distance (e.g., Euclidean distance) between two cases represents their closeness or similarity. Two cases are similar if their distance is small. After finding the similar cases, we fetch their actual run times. As the machines running similar jobs in the case base could be different from the machines running the new job, we then make adjustments to the fetched run times, i.e., we adapt the old case to a new situation. Machine characteristics could be static (such as the machine speed or the memory size) or dynamic (such as the number of slaves[1], or workloads). In adapting run times, we should consider both static and dynamic characteristics. After the running of a job, the new case is created. If the same case (i.e., the same job on the same machine under the same workload) exists in the case base, the new case will be incorporated into the existing case; otherwise, the new case will be stored in the case base.

We performed experiments in a real system: scheduling Functional Regression Tests (FRT) for the IBM® DB2® Universal Database™ product Version 8.2 (DB2 UDB) [25]. FRT needs to run large numbers of test cases (jobs) with different releases

[1] The number of slaves means the number of currently parallel running jobs in a machine. One slave corresponds to one running job.

on different platforms (Linux, AIX, Windows, SUN, HP, etc.) frequently to ensure that a new version of a product functions as designed. In such application, some jobs may run only on specific DB2 mode (serial, DPF, etc.) and/or on specific platform. The time to run a job may take minutes to hours. The target of FRT scheduling is to reduce the overall job run times. Thus, the accuracy of runtime estimation is crucial and challenging. During estimation, we use the *TA3* case-based reasoning (CBR) system [14] to retrieve cases from the case base. The experimental results show that our approach achieves average estimation error within 22%.

The paper is organized as follows. Section 2 discusses the related work. Section 3 talks about the design and structure of the case base. Section 4 describes how to estimate run times of jobs using the *TA3* case-based reasoning system. Section 5 shows the experimental results. Conclusions and future work are presented in Section 6.

2 Related Work

Much effort has been applied to improving the accuracy of performance prediction or runtime estimation. However, there is no general approach that is good for all situations. An estimation strategy can usually be applied with some restrictions.

The Grid Harvest Service (GHS) [24] is a system for long-term performance prediction and task scheduling. It mainly assumes that a job can be divided into independent tasks and the arrival of the jobs follows a Poisson distribution. A job completion time is calculated by mathematical analysis. This work differs from ours in that we calculate run times using historical data, which are stored in the case base during the previous runs.

Performance modeling and prediction based on workload are investigated by Zhang et al. [27] who predict run times for grid tasks based on CPU loads, and evaluate the results using simulation. We also consider the workload change in terms of the number of slaves running in the system. But we also consider other factors, such as job heterogeneity and machine heterogeneity.

Estimating job run times using historical information has been examined in [3], [7], [23]. The idea is to create templates that include a set of job categories to which jobs can be assigned, and then define the similarity of jobs based on job characteristics. New job run times are estimated according to "similar" jobs that have run in the past. Downey [3] categorizes jobs using workload only, and then models the cumulative distribution functions of the run times in each category. Gibbons [7] adopts fixed templates, and uses characteristics such as "executability", user names, number of processes, and memory usage. Smith et al. [23] use more characteristics, such as types, queues, arguments, and network adapters, and also included times estimated by users.

Nassif et al. [19] use CBR to predict job run times. To represent cases, authors separate the problem, solution, and result classes into different tables. The problem class contains attributes that describe a job, mainly including application name and arguments, file size, and job start time. The solution class contains attributes of job prediction information, such as job execution prediction and similarity between the

new case and similar past cases. The result class contains attributes of actual job execution information, such as prediction errors, and workloads during job execution. The three classes are connected using case identification (CasId). To retrieve similar cases, the authors first compare similarity between attributes (local similarity), and then use a geometric distance measure to compare the similarity of cases.

Li et al. [16] propose an instance-based learning technique to predict the job run time and wait time. This approach is also based on historical observations. The database includes job attributes (e.g., the user name, the executable name) and resources' state attributes (e.g., the number of free CPUs). Only job attributes are used to predict the job run time. One of the resources' state attributes is chosen to be a policy attribute to reflect the local scheduling policy. The distance between jobs represents their similarity.

The approaches described in [16], [19], [23] are similar to ours in terms of estimating run times using historical data, with the following important differences:

- They consider only the job similarity, while we also consider machine similarity. When the machine running the new job is different from the one running the old job, the run time needs to be adapted. Thus, approaches from [16], [19], [23] are appropriate for cluster systems where all machines are homogeneous. Our approach can be used in the heterogeneous grid environment.
- Existing approaches treat all job attributes equally; we group job and machine characteristics into categories with priorities [9]. Characteristics with high priorities will be considered first during case matching, and low priority attributes will be relaxed first during context relaxation.
- Nassif et al. [19] use the KNN algorithm to retrieve cases, while we use a variable-context similarity assessment approach (a modified KNN algorithm), which is more flexible and efficient, and can handle irrelevant attributes [10]. Compared with [19], we store all job, machine, and runtime information in one table, which results in improved scalability and speed to retrieve relevant information.

In addition, we are studying the domain of jobs that have internal characteristics or attributes, while all other approaches focus on outer characteristics, such as the application name and the user name.

3 Case Base Design

The case base stores records of all previous successful or clean job runs (i.e., properly and fully executed FRT jobs). Each record includes information about a job, machine, and time statistics. Job and machine information describes job and machine characteristics. Time statistics information contains both the actual and estimated run times for a job. Thus, each record represents an individual instance (case) of a run time for a job on a specific machine. A job run time depends on both job and machine characteristics. As the number of job and machine characteristics could be large, we may first decide which of them should be included in the case base, and the priorities of selected characteristics.

```
for each attribute{
  partition all records into a set of groups
  for each group {
    calculate the standard deviation using the formulas
        (3.1) - (3.3) for this group
  }
  calculate the Average Standard Deviation (ASD) from
       all groups for this attribute
  if (ASD < threshold){
        include the attribute in the case base component set
  }
}
sort the attributes in ascending order in term of ASD
set priorities for attributes (attributes with low ASD
    have the high priorities)
```

Fig. 1. An algorithm to select job characteristics and determine their priorities

3.1 Job Similarity

Job similarity represents the closeness of run times of two jobs on the same machine. If three jobs A, B, and C are run on the same machine, and the difference in run time between A and B is smaller than that between A and C, we say that A is more similar to B than to C. Job similarity may be determined by internal job characteristics, which depend on actual applications. For example, in our FRT application, each test case is a job. Some job characteristics include the platform the test case runs, the DB2 release the test case tests, the mode the test case runs on (e.g., whether a test case is tested in DB2 DPF -- Data Partitioning Features mode), the tool the tester uses to implement the test case. Usually a job may contain many characteristics. However, not all of them should be included in the case base for two reasons: 1) to reduce the size of the case base and save the time of case processing, e.g., retrieving, updating, etc.; 2) to remove those characteristics that do not affect the similarity significantly.

In order to determine which characteristics will affect job similarity most significantly, we can analyze statistics from previous runs. The statistics should contain only jobs running on the same machine, so the influence of run times from machine factors can be ignored. For each characteristic, we partition all jobs into different groups, then calculate the average values, variances, and standard deviations using formulas (3.1) – (3.3), under the assumption that the probability of running each job is equal. The values of job characteristics could be alphabetic or numeric. For the former, each value represents a group; for the latter, we can set a data range for each group. Values that are in the same range belong to the same group. Lower standard deviation means run times are closer, i.e., jobs are more similar. Therefore, characteristics with the low standard deviations have a high priority to determine the job similarity. We can set a threshold and select characteristics whose standard deviations are less than the threshold into the case base. The algorithm in Fig. 1 determines the priorities of job characteristics for job similarity. The results are used in Section 4.2.

$$Average = \frac{1}{k} \sum_{i=1}^{k} runTime_i \tag{3.1}$$

$$Variance = \frac{1}{k} \sum_{i=1}^{k} (Average - runTime_i)^2 \tag{3.2}$$

$$Standard\ Deviation = \sqrt{Variance} \tag{3.3}$$

3.2 Machine Similarity

Machine similarity represents the closeness in the run times of the same job when run on different machines. If three machines X, Y, and Z run the same job, and the difference of the run time between X and Y is smaller than that between X and Z, we say that X is more similar to Y than to Z. Machine similarity is determined by machine characteristics, which could be static, such as the disk I/O, the machine speed, the memory size, the cache size, and the number of CPUs, or dynamic, such as the number of slaves and the disk I/O workloads. However, those characteristics do not equally affect the run time.

Static Machine Characteristics
The disk system greatly influences the machine properties, especially for I/O-bound processes. Performance of the disk system depends on both hardware (e.g., the disk physical structure) and software (e.g., disk scheduling algorithms). Usually, the disk speed is the most significant factor that affects performance.

Table 1. Specification of machines for testing the order of importance of machine similarities (in each group, only the bold values are different)

Group	Speed (MHz)	Memory (MB)	# CPU	Cache (KB)	Characteristic Ratio
1	**800**	1024	1	512	1.6
	500	1024	1	512	
2	2192	**1024**	1	512	2
	2192	**512**	1	512	
3	1994	1024	**2**	512	2
	1994	1024	**1**	512	
4	1994	1024	1	**1024**	2
	1994	1024	1	**512**	

Table 2. Total runtime ratios vs. machine characteristics

Machine Characteristics	Total Runtime Ratios
Machine Speed	1.33
Memory Size	1.22
Number of CPUs	1.10
Cache Size	1.03

To determine the order of importance of other machine characteristics, we choose four different pairs of machines; in each pair, we fix the other characteristics, while multiplying only one of the memory size, the number of CPUs, or the cache size by

two, or multiplying the machine speed by 1.6 (see Table 1). All jobs tested have the same or the most similar characteristics, so the results mainly depend on the machine characteristics. We calculated the ratios of the total run times for each pair of machines, as shown in Table 2. The results demonstrate that machine speed affects run time the most, followed by memory size, number of CPUs, and cache size, in decreasing order. We must point out the lack of effect of the number of CPUs on run times in our results. This is because in order for each machine to run under the best dynamic characteristic (discussed more in the next section), the number of slaves running in a dual-CPU machine is more than on the single-CPU machine. Thus, the average number of CPUs for each slave is the same (the value is 1/2), for both single- and dual-CPU machines.

Dynamic Machine Characteristics

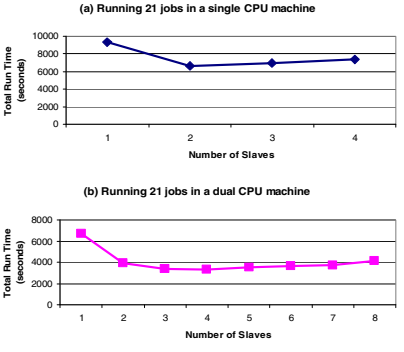


Fig. 2. Total run time vs. number of slaves

The number of running slaves could also affect the run time. If the number of slaves is small, the CPU resources may not be optimally used, while a large number of slaves decreases overall performance because of increased CPU context switching. To evaluate the significance of this effect, we conducted experiments on two machines with different numbers of CPUs. We selected 21 jobs to run in a single-CPU and then in a dual-CPU machine, and varied the number of slaves from 1 to 4, and from 1 to 8, respectively. For each number of slaves, we recorded the average run time over three independent runs. The results are shown in Fig. 2 and demonstrate that there exists an optimal number of slaves, and the optimal value is mostly affected by the number of CPUs. For our two experimental machines, the optimal values are 2 and 4 for single- and dual-CPU machines, respectively. For machines with more CPUs, this value would be proportionally higher.

3.3 Components of the Case Base

As mentioned above, there are three components in the case base: job information, machine information, and time information.

Job information specifies job characteristics and is used to decide the similarity between a new job and the jobs in the case base. As explained in Section 3.1, the relative importance of job characteristics is different; therefore, we set different priorities for different job characteristics in Section 4.2.

Machine information contains the machine characteristics, and is used to adapt the new job run time if the characteristics of the machine running the new job are different from those of machines associated with jobs in the case base. The machine characteristics should include not only static characteristics such as the disk speed, the machine speed, the number of CPUs, the memory size, and the cache size, but also the

dynamic characteristics such as the number of slaves, and the disk I/O workload. Machine characteristics are also assigned different priorities in Section 4.2.

Time information includes both actual and estimated statistical data, e.g., the average actual run time, the average estimated run time, the actual last run time, the estimated last run time, the average estimation error, and the last estimation error.

4 Runtime Estimation

In this section, we introduce the *TA3* case-based reasoning (CBR) system [9], [10], [14] and discuss how we use it to estimate job run times.

4.1 The *TA3* Case-Based Reasoning System

TA3 is a case-based reasoning system that uses a variable-context similarity assessment approach [14]. Cases in the case base describe problem-solution pairs. Reasoning is done by retrieving problems with similar descriptions, and applying analogy-based reasoning to adapt the solutions from retrieved cases to solve the new problem. It has been successfully applied in diverse applications [1], [11], [12], [13].

There are two types of cases: an input case and a retrieved case. The former describes the problem and does not contain a solution. The latter contains both a problem and a solution, and is saved in the case base. A case can contain many attributes grouped into different categories with different priorities (0 indicates the highest priority). Attributes with the highest priority are the most important and should be matched and processed first. The retrieval strategies used in *TA3* are not predefined; rather they are dynamically changed for a particular domain and specific application. The number of the matched cases can be reduced and expanded flexibly. *TA3* determines similarity using the closeness of attribute values in the context, which can be seen as a view, or an interpretation of a case on a relevant subset of attributes. *TA3* uses two methods for adaptation: the distance-weighted KNN algorithm and the average of attribute values.

4.2 Estimation Method

The estimation process includes three steps: case representation and description, case retrieval, and case adaptation, which are described in the next sections.

Case Representation and Description

A case is represented as a record. It includes all job characteristics discussed in Section 3.1 and all machine characteristics discussed in Section 3.2.

To support flexible, scalable, and efficient similarity matching, we organize individual attributes into classes, which contain one or more categories, which in turn contain one or more attributes. We created two classes: problem and solution. The problem class includes different categories with different priorities. Table 3 is a sample problem class in which the first $n+1$ are job categories, each of which contains the job attribute set $0, 1, \ldots n$, respectively, and the rest are the machine categories. Each job attribute set may have more than one attribute. The solution class (not listed in Table 3) contains only one category and one time attribute with priority 0, the highest priority.

Table 3. Categories, attributes, and priorities of problem class

Category	Priority	Attribute Name
P_0	0	job attribute set 0
P_1	1	job attribute set 1
…	…	…
P_n	n	job attributes set n
P_{n+1}	n+1	Hostname Number of slaves
P_{n+2}	n+2	disk speed
P_{n+3}	n+3	machine speed
P_{n+4}	n+4	memory size
P_{n+5}	n***+5***	number of CPUs
P_{n+6}	n ***+6***	cache size

Case Retrieval

TA3 uses a modified KNN algorithm to match cases. It refers to conditions for matching cases as context, and the similarity between cases is defined with respect to the context. *TA3* can control the number of cases to be retrieved. If too many or too few cases are matched in the original context, the system can automatically modify the context[2]. This process is called context transformation, which is implemented as relaxation or restriction. Relaxation makes more cases to be retrieved, and restriction reduces the number of matched cases. *TA3* uses an incremental context transformation algorithm to improve the efficiency of query processing [10]. The *TA3* system retrieves cases using the priorities assigned in Table 3. Attributes with high priorities will be matched before those with low priorities. If multiple cases are found, they are ordered by similarities. The most similar cases have the highest weight. Equal cases have the equal weight. The runtime estimate is the weighted average of values from all retrieved cases. More information about relaxation and restriction can be found in [10].

Case Adaptation

Adaptation makes solutions of the old cases better fit the new case. The machines and their states in the input and retrieved cases are usually not the same. For example, in Fig. 3, an old job (retrieved case) has run on machine X with only 1 slave (point A), while a new job (input case) runs on machine Y with 3 slaves (point D). Thus, we need to adapt the run time from point A in machine X to point D in machine Y. In such situations, both static and dynamic characteristics are different. The adjustment must consider both types of changes. Generally, we need to adjust the retrieved run times using the following three steps:

1. Dynamic adjustment of run times in the retrieved machine (from point A to point B). As in the retrieved case, the old job might not run under the optimal number of slaves. We need to adjust the time to correspond to the case in which the job runs under the optimal number of slaves.

[2] In our FRT application, only case relaxation is needed, and is applied only when the original query retrieves 0 cases.

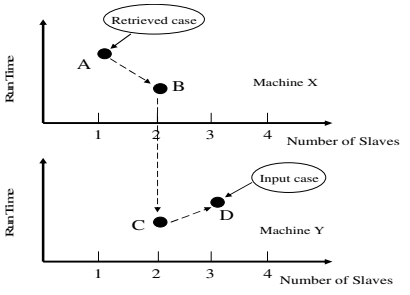


Fig. 3. Case adaptation. The retrieved case is the point A running in machine X with 1 slave. The input case is the point D running on machine Y with 3 slaves. Point B and C run under the optimal number of slaves for machine X and Y, respectively.

2. Static adjustment of run times from the case in the retrieved machine to the case in the input machine with both under the optimal numbers of slaves (from point B to point C).
3. Dynamic adjustment of run times in the input machine (from point C to point D). From step 2, we obtain the time it takes for the input machine to run the job under the optimal number of slaves. We need to further adjust the time to the current number of running slaves.

In order to make the adjustments in steps 1 and 3, we need to obtain curves similar to ones shown in Fig. 2. For the adjustment in step 2, we run the same set of jobs in each machine under the optimal number of slaves. We create a table in which rows represent input machines, and columns represent retrieved machines. The entry *(i, j)* of the table is the ratio of total run times of machine *i* and machine *j*, and is used to adjust the time. If the input and retrieved cases both run under the optimal number of slaves, only the static adjustment in step 2 is required. If multiple cases are found, we calculate the weighted average of all retrieved times.

Current practice is to use theoretical formulas to adapt run times. However, the real system is complicated, and thus it is difficult to accurately and faithfully model the real situation. We think that statistics from the real system may be more direct and accurate. We applied our approach to the FRT scheduler and achieved significant improvement in scheduling performance (p=0.036).

4.3 Estimation Errors

We use the relative estimation error (REE) to evaluate the accuracy of the estimation. It is defined as follows:

$$REE = \frac{estRunTime - actRunTime}{actRunTime} \times 100\%,$$ where *estRunTime* is the estimated run time of a job and *actRunTime* is the actual run time of a job.

5 Experimental Evaluation

We use a real scenario, scheduling FRT for IBM DB2 UDB products, to evaluate our approach to estimating run times. We next discuss the experimental environment and experimental evaluation.

5.1 Experimental Environment

We performed experiments in the CAS Grid[3], which comprises an IBM AIX® machine (AIX#1) as a server, and seven Linux® machines (Linux#2 - Linux#8) as clients. The configuration of these machines is shown in Table 4. The system structure of the CAS dispatcher, or scheduler, is shown in Fig. 4. The functionality of each component is described below:

- The CAS Grid comprises both AIX and Linux machines. The machine running the dispatcher is called the server, while machines running jobs are called clients. Each client contains one or more slaves (randomly chosen).
- JobMonitor monitors the status of active jobs in the grid, and records this information in the jobs database (JobsDB).
- MachineMonitor monitors the status of machines. It records how many and which slaves are running, and updates information in the SlavesDB.
- ResultsDB stores all test results.
- JobsDB stores all jobs and their status.
- SlavesDB contains information about slaves.
- CaseBase stores historical data for the estimation of job run times. It contains job information, machine information, and actual and estimated runtime information. The CaseBase is initialized using historical data of previous runs from the FRT Team and dynamically expanded over time. Results of an executed job are saved in the ResultsDB; and when the run is successful, a case is created and saved in the CaseBase.
- RunTimeEstimator estimates a job run time and wait time on a slave. The estimated run time is calculated based on historical data from the CaseBase. The estimated wait time is calculated using the estimated run time, and other information retrieved from the JobsDB and the SlavesDB.
- Dispatcher assigns jobs to slaves. The dispatcher checks available jobs from the JobsDB, and available slaves from the SlavesDB, then gets estimated run times and waiting times from the RunTimeEstimator, and finally makes schedules using the scheduling algorithm.

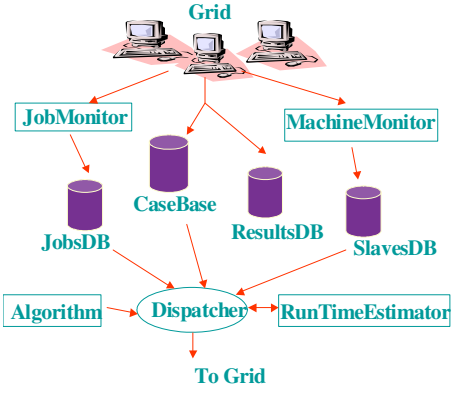


Fig. 4. System structure of the dispatcher

[3] The grid we used was located at the IBM Centre for Advanced Studies (CAS), and hence we refer to it as the "CAS Grid".

Table 4. Specification of Machines in the CAS grid

Machine	Speed (MHz)	MEMORY (MB)	# CPU	CACHE (KB)
AIX #1	375	1024	1	512
Linux #2	498	1024	1	512
Linux #3	794	1024	1	256
Linux #4	2992	1530	2	512
Linux #5	3192	1536	2	1024
Linux #6	2192	512	1	512
Linux #7	2192	512	1	512
Linux #8	1994	512	1	256

5.2 Experimental Evaluation

In this section, we evaluate the accuracy of runtime estimation using our CBR approach. We performed experiments in the real system as discussed in the previous section. We created and initialized the CaseBase by selecting 500 previously run jobs as retrieved cases from the ResultsDB. Then we also randomly selected 700 jobs, which may or may not be the same as the retrieved cases, as input cases to run in the CAS Grid. As job run times vary largely, those retrieved and input cases were sampled to replicate their distribution in the full data set. For each job, we recorded the estimated run time and the actual run time, and calculated the relative estimation error. Fig. 5 shows the histogram of estimation errors using the CBR approach. In the figure, the x-axis is the estimation errors, while the y-axis is the percentage of the jobs whose estimation error is within two adjacent values in the x-axis. For example, 4% of jobs have zero estimation errors; about 16% of jobs have estimation errors between 5% and 10%, and so on. Thus, by using our CBR approach, we can achieve 90% of 700 jobs with estimation errors less than 45%; only 4% of jobs have estimation errors larger than 65%; and the average estimation error is 22% or less.

As a given estimation method may only work under certain conditions, it is difficult to give a precise comparison among different approaches in general. Our approach is different from others in that it is the first to study the domain of jobs with internal characteristics, and the method can apply to real heterogeneous systems. We compared three approaches [16], [19], [23] that also use historical data. We used the template approach [23] to run the same jobs in our experiments; the average estimation error is 45%. Li. et al. [16] report that their estimation errors are between 35% and 70%. Although Nassif et al. [19] also use the CBR approach, their method is more suited for the homogeneous

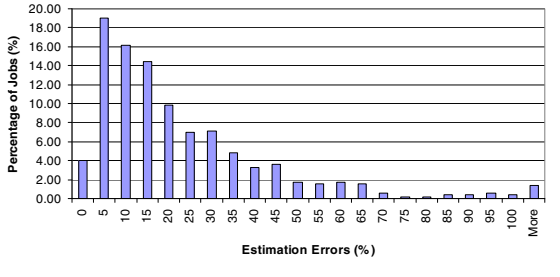


Fig. 5. Histogram of estimation errors using the CBR approach. The x-axis is the estimation errors. The y-axis is the percentage of jobs whose estimation errors are within two adjacent values in x-axis. The total number of testing jobs is 700.

system, since they do not consider the heterogeneity of machines. If we do not adapt the run time for the difference of machine characteristics, the estimation error will increase to ~32%.

6 Conclusion and Future Work

In this paper, we presented an approach to estimate a job run time using a case-based reasoning system. A job run time can be affected by a large number of both job and machine characteristics. Therefore, we first need to determine which characteristics affect a job run time the most and thus should be included in the case base. Job and machine characteristics are also prioritized, to improve KNN algorithm performance. We developed a generic algorithm to determine the priorities of job characteristics. We analyzed the static machine characteristics, such as the disk I/O system, the machine speed, the memory size, the number of CPUs, and the cache size; we also analyzed the dynamic machine characteristics, such as the number of slaves, and disk I/O workloads. From the experimental results, we compared the importance of those machine characteristics and determined their priorities.

We then discussed the structure of the case base, and the steps for the runtime estimation, which includes case representation, case retrieval and case adaptation. A case is represented as a record that includes job information, machine information, and runtime information. Cases are retrieved based on the priorities of job and machine characteristics. Both static and dynamic machine characteristics are considered for the case adaptation. Our experimental results show that for more than 90% of jobs, the estimation error is 45% or less, and the average estimation error is 22% or less, which is a substantially better performance compared to existing approaches.

To further improve the estimation accuracy using the CBR approach, we can use clustering algorithms [8] to cluster cases into small groups so that cases within the same group will be more similar than those in other groups. As different attributes may have different effects on the run times, we can use feature-weighting approaches [18] to add a weight for each attribute. Then the priorities of attributes can be determined dynamically instead of statically. In the CBR approach, we used Euclidean distance to measure the closeness of cases. There are other methods for distance measure, such as Manhattan distance, maximum distance, Canberra distance, Bray-Curtis (Sorensen) distance, and angular separation. Their impact on overall performance will need to be tested in the future. In addition, in a general grid environment, applications may be transmitted through a network. Thus, the transmission time should be considered and estimated.

Acknowledgments. We would like to thank IBM DBT Regression Test Team for help with setting up the regression test environment on the CAS grid, and the IBM Development Support Services (DSS) team for supporting the CAS grid.

The views expressed in this paper are those of the authors and not necessarily of IBM Canada Ltd. or the University of Toronto.

product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References

1. Arshadi, N., Jurisica, I.: Integrating case-based reasoning systems with data mining techniques for discovering and using disease biomarkers. IEEE Transactions on Knowledge and Data Engineering. Special Issue-Mining Biological Data 17(8), 1127–1137 (2005)
2. Braun, T.D., Siegel, H.J., Beck, N., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. of Parallel and Distributed Computing 61(6), 810–837 (2001)
3. Downey, A.B.: Predicting queue times on space-sharing parallel computers. In: 11th Intl. Parallel Processing Symposium, pp. 209–218 (1997)
4. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure, 2nd edn. Morgan Kaufmann, San Francisco (2004), ISBN: 1-55860-933-4
5. Fujimoto, N., Hagihara, K.A.: Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks. In: SAINT 2004 Workshop on High Performance Grid Computing and Networking, Tokyo, Japan, pp. 674–680. IEEE Press, Los Alamitos (2004)
6. Garg, S., Buyya, R., Siegel, H.J.: Scheduling Parallel Applications on Utility Grids: Time and Cost Trade-off Management. In: Proceedings of the 32nd Australasian Computer Science Conference (ACSC 2009). Australian Computer Society, Wellington (2009), ISBN 978-1-920682-72-9
7. Gibbons, R.A.: Historical Application Profiler for Use by Parallel Schedulers. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1997 and JSSPP 1997. LNCS, vol. 1291, pp. 58–77. Springer, Heidelberg (1997)
8. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann/Elsevier (2006)
9. Jurisica, I., Glasgow, J.: Improving performance of case-based classification using context-based relevance. In: International Journal of Artificial Intelligence Tools. Special Issue of IEEE International Conf. on Tools with AI (ICTAI 1996) Best Papers, vol. 6(4), pp. 511–536 (1997)
10. Jurisica, I., Glasgow, J., Mylopoulos, J.: Incremental iterative retrieval and browsing for efficient conversational CBR systems. International Journal of Applied Intelligence 12(3), 251–268 (2000)
11. Jurisica, I., Glasgow, J.: Application of case-based reasoning in molecular biology. Artificial Intelligence Magazine, Special issue on Bioinformatics 25(1), 85–95 (2004)
12. Jurisica, I., Mylopoulos, J., Glasgow, J., Shapiro, H., Casper, R.: Case-based reasoning in IVF: Prediction and knowledge mining. Artificial Intelligence in Medicine 12(1), 1–24 (1998)

13. Jurisica, I., Rogers, P., Glasgow, J., Fortier, S., Luft, J., Wolfley, J., Bianca, M., Weeks, D., DeTitta, G.T.: Intelligent decision support for protein crystal growth. IBM Systems Journal, Special issue on Deep Computing for Life Sciences 40(2), 394–409 (2001)
14. Jurisica, I.: TA3: Theory, Implementation, and Applications of Similarity-Based Retrieval for Case-Based Reasoning. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario (1998)
15. Leake, D.: Case-Based Reasoning: Experiences, Lessons, & Future Directions. AAAI Press, Menlo Park (1996)
16. Li, H., Groep, D., Wolters, L.: Efficient Response Time Predictions by Exploiting Application and Resource State Similarities. In: Proceedings of 6th IEEE/ACM International Workshop on Grid Computing (Grid 2005), in conjunction with SC 2005, Seattle, Washington, USA, pp. 234–241. IEEE Computer Society Press, Los Alamitos (2005)
17. Mitchell, T.: Machine Learning. McGraw Hill, New York (1997), ISBN: 0-07-042807-7
18. Modha, D., Scott-Spangler, W.: Feature Weighting in K-means Clustering. Machine Learning 52(3), 217–237 (2003)
19. Nassif, L., Nogueira, J., Karmouch, A., Ahmed, M., Andrade, F.: Job Completion Prediction Using Case-based Reasoning for Grid Computing Environments. In: Concurrency and Computation: Practice & Experience, vol. 19, pp. 1–12. Wiley InterScience, Hoboken (2006)
20. Rao, I., Huh, E.: A Probabilistic and Adaptive Scheduling Algorithm Using System-generated Predictions for Inter-Grid Resource Sharing. Journal of Supercomputing 45(2), 185–204 (2008)
21. Shakhnarovich, G., Darrell, T., Indyk, P.: Nearest-Neighbor Methods in Learning and Vision. MIT Press, Cambridge (2006), ISBN 0-262-19547-X
22. Sakellariou, R., Yarmolenko, V.: Job Scheduling on the Grid: Towards SLA-Based Scheduling. In: Grandinetti, L. (ed.) High Performance Computing and Grids in Action, pp. 207–222. IOS Press, Amsterdam (2008)
23. Smith, W., Foster, I., Taylor, V.: Predicting application run times using historical information. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1998, SPDP-WS 1998, and JSSPP 1998. LNCS, vol. 1459, pp. 122–142. Springer, Heidelberg (1998)
24. Sun, X.H., Wu, M.: Grid Harvest Service: A System for Long-Term, Application-Level Task Scheduling. In: Proceedings of the 2003 IEEE International Parallel and Distributed Processing Symposium (IPDPS 2003), Nice, France (2003)
25. Xia, E., Jurisica, I., Waterhouse, J., Sloan, V.: Scheduling functional regression tests for IBM DB2 products. In: Proceedings of 2005 IBM Center for Advanced Studies Conference (CASCON 2005), Toronto, Canada, pp. 290–302 (2005)
26. Yuan, Y., Wu, Y., Yang, G., Zheng, W.: Adaptive hybrid model for long term load prediction in computational grid. In: CCGRID 2008, pp. 340–347 (2008)
27. Zhang, Y., Sun, W., Inoguchi, Y.: Predict task running time in grid environments based on CPU load predictions. Future Generation Computer Systems 24(6), 489–497 (2008)

Author Index