Summer 2007
Vol 9, No 2
ISSN 1465-4091
£12.50

The Specialist Group on Artificial Intelligence

EXPERT UPDATE

SGAI

ECCAI

BCS®
THE BRITISH COMPUTER SOCIETY

Special Issue on the 10th UKCBR Workshop

EXPERT UPDATE
(ISSN 1465-4091)

What is Expert Update?
Expert Update (www.comp.rgu.ac.uk/staff/nw/expertUpdate.htm) is the bulletin/magazine of the SGAI, the British Computer Society's Specialist Group on AI (BCS-SGAI: www.bcs-sgai.org). The purpose of Expert Update is to foster the aims and objectives of the group by publishing news, conference reports, book reviews, conference announcements, calls for papers and articles on subjects of interest to the members. Expert Update is generally published 3 times per year by BCS-SGAI. The group's official postal address is: SGAI, The BCS, Davidson Building, 5 Southampton Street, London, WC2E 7HA.

How do I subscribe?
It is free to all SGAI members. Please visit www.bcs-sgai.org/ for details on joining the group.

How do I contribute?
Submissions are welcome and must be made in electronic format sent to the editor or sub-editor.

EDITORIAL

Welcome to Expert Update's Summer 2007 special issue on Case-Based Reasoning (CBR).

The seven papers included here were presented at the 10th UK-CBR workshop organised by Dr Miltos Petridis from the University of Greenwich (also SGAI). They cover both theoretical and application issues in CBR related to framework architectures, case-based recommender systems and case-base maintenance.

SGAI will once again host this year's UK-CBR workshop on 10th Dec 2007 followed by the 2-day SGAI International Conference on Artificial Intelligence (AI) in Cambridge. Further details of this conference appear on the back cover. SGAI have also started a new workshop series on "Basics of AI" targeted at anyone who is new or wishes to have a broader understanding of AI. The first of these took place on 12th July 2007 at the BCS offices near Covent Garden. For information on future workshops and other SGAI events please visit www.bcs-sgai.org.

Nirmalie Wiratunga
Editor Expert Update
nw@comp.rgu.ac.uk

---

jCOLIBRI in a nutshell: A software tool for designing CBR systems*

Juan A. Recio, Antonio Sánchez, Belén Díaz-Agudo, Pedro González-Calero
Dep. Sistemas Informáticos y Programación
Universidad Complutense de Madrid, Spain
email: {antonio.sanchez jareciog}@fdi.ucm.es,{belend,pedro}@sip.ucm.es

Abstract

jCOLIBRI is a framework that aims to formalize Case Based Reasoning and to provide a design and implementation assistance with software engineering tools [1, 2, 3, 4 & 5]. In this paper we describe some design, implementation and use details regarding jCOLIBRI latest release. We try to demonstrate the relevance and usefulness of the jCOLIBRI framework in view of encouraging other CBR researchers to use it.

Keywords: jCOLIBRI, Case-based reasoning, framework, ontologies.

1. Introduction

Case-based reasoning (CBR) has become an established part of artificial intelligence (AI), both as a means for addressing fundamental AI problems and as a basis for fielded AI technology. Many researchers in the field agree about the increasing necessity to formalise this kind of reasoning, define application analysis methodologies, and provide a design and implementation assistance with software engineering tools [9,11,12].

Our work tries to solve all these increasing necessities, providing a tool to help application designers to develop and quickly prototype CBR systems. Furthermore we want to provide a software tool useful for students who have little experience with the development of different types of CBR systems. jCOLIBRI[1] is an object-oriented framework in Java for building CBR systems, and it has been designed as a wide spectrum framework that is able to support several types of CBR systems: from the simple nearest-neighbour approaches, based on flat case structures, to complex knowledge intensive CBR systems. jCOLIBRI also contains textual and conversational extensions. jCOLIBRI offers an easier development process that is based on the reuse of past designs and implementations. Although there are other CBR shells [8,10], our work goes beyond them in terms of reuse, flexibility, scope and usability. jCOLIBRI promotes software reuse integrating the application of well proven Software Engineering techniques with a knowledge level description that separates the Problem Solving Methods (PSMs), which define the reasoning processes, from the domain model, that describes the domain knowledge. In this paper we provide an overview of the main components of jCOLIBRI and describe how to use it to create simple CBR applications. We have documented other specific aspects of jCOLIBRI in other papers [4,5,6].

* Supported by the Spanish Committee of Science & Technology (TIN2006-15140-C03-02)

[1] http://jcolibri-cbr.sourceforge.net http://sourceforge.net/projects/jcolibri-cbr/

Section 2 introduces the sequence of steps afforded by a designer who is using jCOLIBRI to design a simple CBR system. Sections 3 and 4 describe the graphical tools that assist users to develop CBR applications, both to define the data structures and to configure the processes. Section 5 exemplifies the use of jCOLIBRI to create a travel recommendation system from the travel domain case base, a well-known example frequently used in the CBR community. Section 6 concludes the paper.

2. Developing CBR systems using jCOLIBRI Tools

Developing a CBR system is a complex task where many decisions must be made. When using jCOLIBRI some of these decisions are:

- Representation of the cases: we can use simple plain cases, textual cases, or we can define the cases as complex hierarchical structures where attributes are connected.
- Case base: the cases can be stored in relational databases, ASCII text files, XML files, etc. The designer has to define the storage and retrieval mechanism, and how to index the cases in memory to increase the speed of the queries.
- Similarity measures: one of the most important tasks to be solved in a CBR system consists on finding the most similar cases to one given. Usually different similarity functions are used to compare the attributes of the cases.
- Behaviour of the CBR system through tasks and methods: a CBR application can be structured as a sequence of tasks that must be achieved. A CBR designer should choose these tasks and select the methods that will solve them.

jCOLIBRI stores all the configuration data using different XML configuration files. When the application is executed, the framework core reads these files to know how to configure the CBR system. Although you could manage this configuration files by hand, we have developed graphical and easy-to-use interfaces. Next sections describe these interfaces.

3. Data structures in a CBR systems

The most important source of knowledge in a CBR system is its set of cases. jCOLIBRI uses a case representation language based on terminology from CBROnto, an ontology that describes the main terms used in CBR systems [2,3].

3.1 Definition of case structures

jCOLIBRI offers a visual tool to define case structures (see Figure 4). The left panel displays the structure of the case as a tree, and the right panel shows the property values of the selected attributes. A *Case* has a *Description*, a *Solution* and a *Result*. *Description* and *Solution* are sets of simple or compound *Attributes*. A simple attribute is characterized by its *name*, *type*, w*eight* and *local similarity function*. Compound attributes collect other simple or compound attributes allowing complex case structures. The properties of the compound attributes are the name and the global similarity function. Local similarity functions compare simple attribute values. Global similarity functions combine these values in a unique similarity value. The similarity value between the query and each case is computed as the similarity of their descriptions.

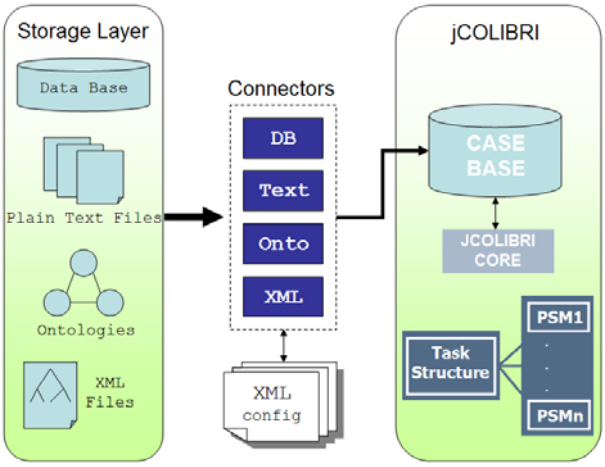


Figure 1: jCOLIBRI connector architecture

This interface does not allow using all the expressiveness of OWL that is the language that we use as the case interchange language [1]. For instance, you cannot restrict the cardinality of attributes. If you need to define more complex case structures you can use external applications, as the PROTEGE [7] editor, to generate the corresponding OWL description. PROTEGE is an advanced editor that let you to use all the description power of OWL, but of course, it has a more complex interface too.

3.2 Persistence of cases: connectors

In jCOLIBRI we propose to separate the case storage from the indexing structure and from the PSMs that reason with cases --like retrieval or adaptation methods. That way, indexes can be built and methods can be configured without knowing how and where the cases are stored. Moreover, different indexes can be defined upon the same set of cases to allow the evaluation of different indexing techniques, or the adequacy of different retrieval or adaptation methods.

We propose an architecture using two layers: persistence mechanism and in-memory organization (see Figure 1). Persistence of cases in jCOLIBRI is built around the concept of Connector. Connectors provide an abstraction mechanism that allows users to load cases from different storage sources in a common way. jCOLIBRI includes connectors that work with plain text files, XML files, relational databases and Description Logics languages like OWL. Other connectors can be included depending on the specific application requirements by means of the *jcolibri.cbrcase.Connector* java interface. jCOLIBRI offers a graphical tool that is used to easily configuring connectors to load existing case bases in different formats (see Section 5).

3.3 Managing the similarity measures

During the definition of the case structure a similarity measure is associated to each attribute. The available similarity measures can be managed using the tool shown in Figure 2. Each similarity measure should be associated to a java class that computes the similarity values when the application is running. These classes should implement the *jcolibri.similarity.SimilarityFunction* java interface.

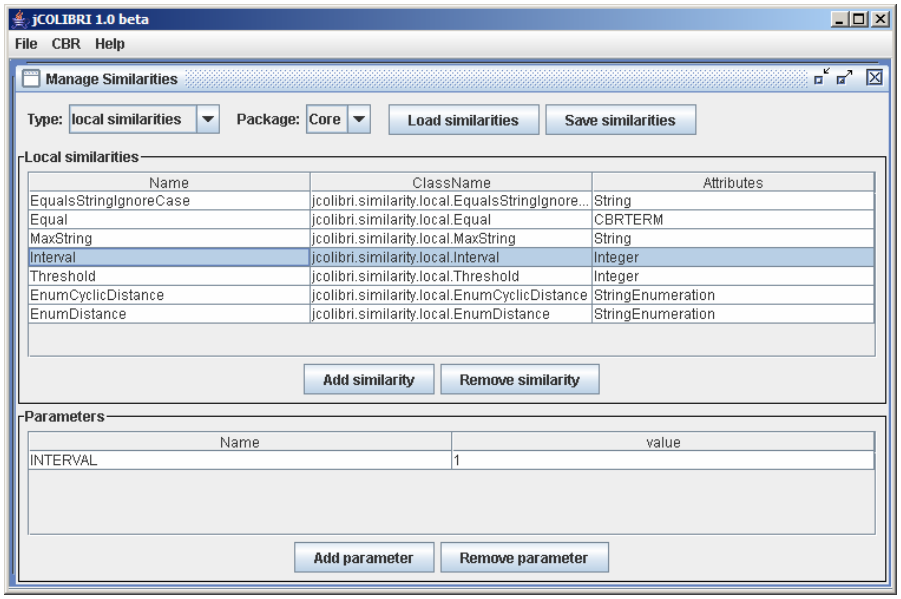


Figure 2: Managing the similarity measures

4. Behaviour of the CBR system

After defining the data structures of the CBR system, the configuration of the connectors to load and store the case base and the similarity measures to compare attributes of the cases; it's time to configure the more dynamic part of the CBR systems: tasks and methods.

The CBR system designer creates a CBR application following an iterative process, where s(he) selects one of the not configured tasks and assigns one method from the library of reusable PSMs. Note that task/method constraints are being tracked during the configuration process so that only applicable methods in the given context are offered to the system designer.

At the beginning of the process the only unsolved task is CBR Task (solve a problem using previous experiences). When we choose the CBR method to solve the CBR Task, we obtain the CBR cycle into a task sequence: retrieve, reuse, revise, retain. While the system is not complete, select one of the not configured tasks and choose and configure a method that resolves it. There are methods to solve tasks either by decomposing a task into subtasks or by solving it directly.

jCOLIBRI offers a semiautomatic configuration tool that guides the instantiation process through a graphical interface (Figure 3). It reasons about applicable methods for each task using a declarative description of each method. This reasoning is done using the DL reasoner. See [4] for a detailed explanation about the reasoning tool.

4.1 Deploy the CBR application

The CBR application is finished when all the tasks have been configured. You can test the system from inside the graphical interface selecting the tasks that are going to be executed in the tasks tree and pushing the  *"Solve to …"* button. The effects of the execution are shown in the *results window*.

This feature provides an easy way to test the system before generating the java code that is needed to run your system as an independent java application.

The first task of the CBR system, *Obtain query task*¸ obtains the query that is going to be used to retrieve the most similar cases. The framework supplies a useful and general method, named *Configure Query Method*, to solve this task. This method reads the case structure from the XML file that stores the case structure, and dynamically generates a visual form that can be used to introduce the query data. This form takes advantage of the best graphical components available according to the case structure and the types of the attributes. Once you have created and tested the application, jCOLIBRI allows you to generate a code template that contains the tasks and methods invocation code and can be modified as needed.

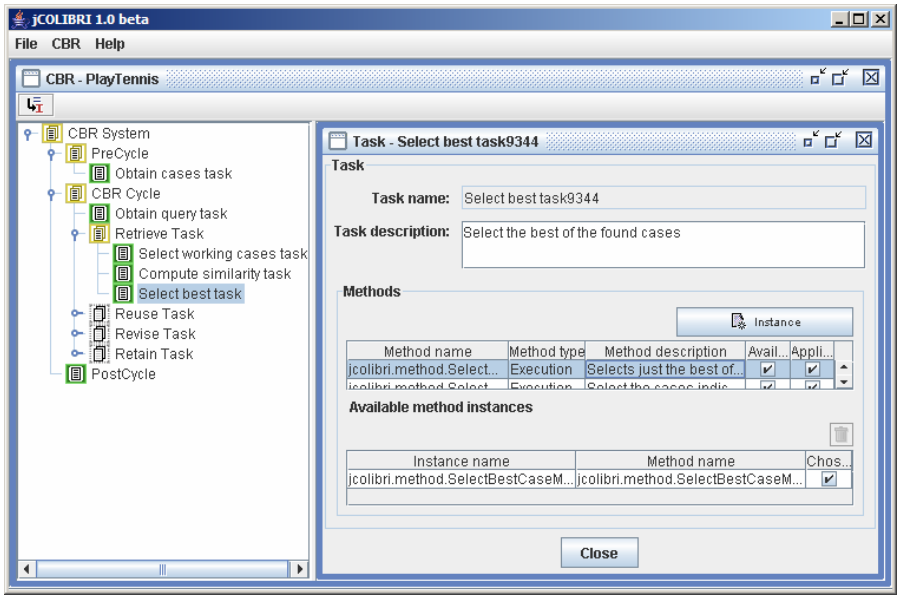


Figure 3: Selecting Tasks/Methods of the CBR application

5. Travel Domain Example

The travel domain case base that is available for download from the AI-CBR website (www-ai-br.org/cases.html) offers a set of 1024 cases that represent travels. We are using jCOLIBRI to design a travel recommendation system based on this case base.

1. Create a new CBR system and name it as desired. Then, create the case structure, i.e, define simple and compound attributes that describe the cases together with their types, weights, similarity measures. jCOLIBRI offers a graphical tool to include new similarity functions (see Figure 2 in Section 3.3). The case structure can be saved/loaded in/from a XML file. We have created a case structure with the 9 simple attributes shown in Figure 4.

2. Once the case structure is defined we could define new cases following this structure. Instead doing this, since we have a previously existing case base, we configure a connector to read in the travel cases from a MySQL database. The graphical interface helps mapping the case structure defined in step 2 with the tables and columns from the database scheme. Like the case structure the connector configuration can be saved/stored in/from a XML file. In this example we configure a connector to work with cases stored in a relational database *travel*. The form (see Figure 5) is divided into 3

sections. The properties panel shows the information required to access to the database. The table structure panel allows managing the structure of the table containing the cases. Last, the mappings panel connects each column of the data base table witch an attribute of the case structure.

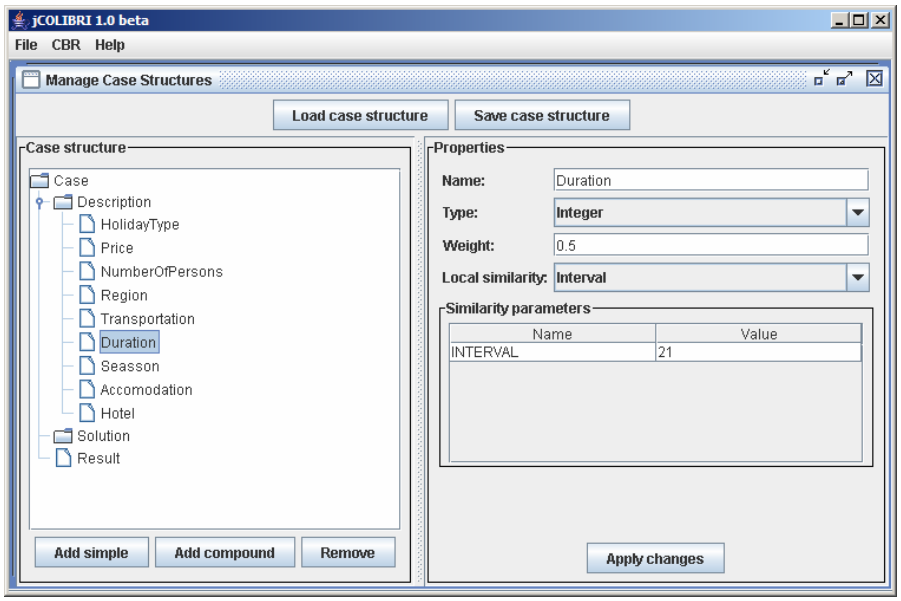


Figure 4. Creating the case structure

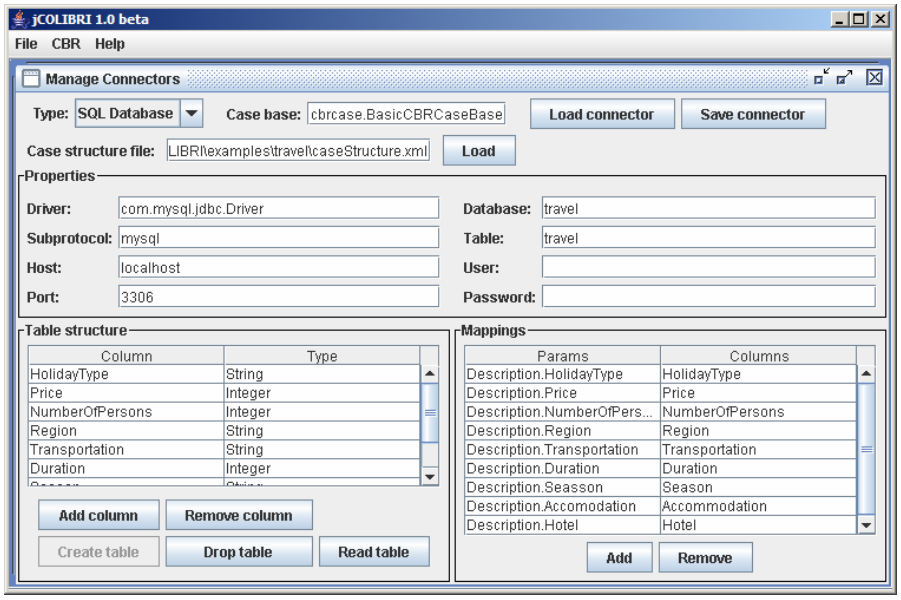


Figure 5. Configuring the connector with the case base

3. Configure tasks and methods. Note that some of the methods could require parameters. For example, to obtain cases we provide with the name of the connector configuration XML file, and to create queries, we use parameterized methods that require the name of the XML file where the system

stored the case structure (see Figure 6). JCOLIBRI automatically defines a graphical interface based on this structure to let the user introducing new queries (Figure 7).

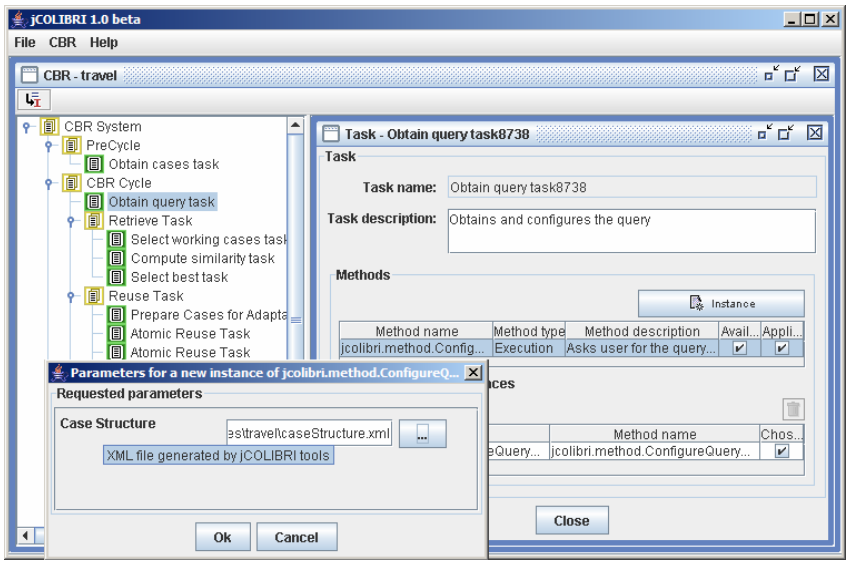


Figure 6. Configuring tasks and methods

4. Once every task is configured by a method that solves it we can run the CBR deployed system where the tasks are solved using the given sequence. You can also use the  (solve to.) button in the GUI to execute a selected set of tasks. The result for the query is shown in the right area of the graphical interface.

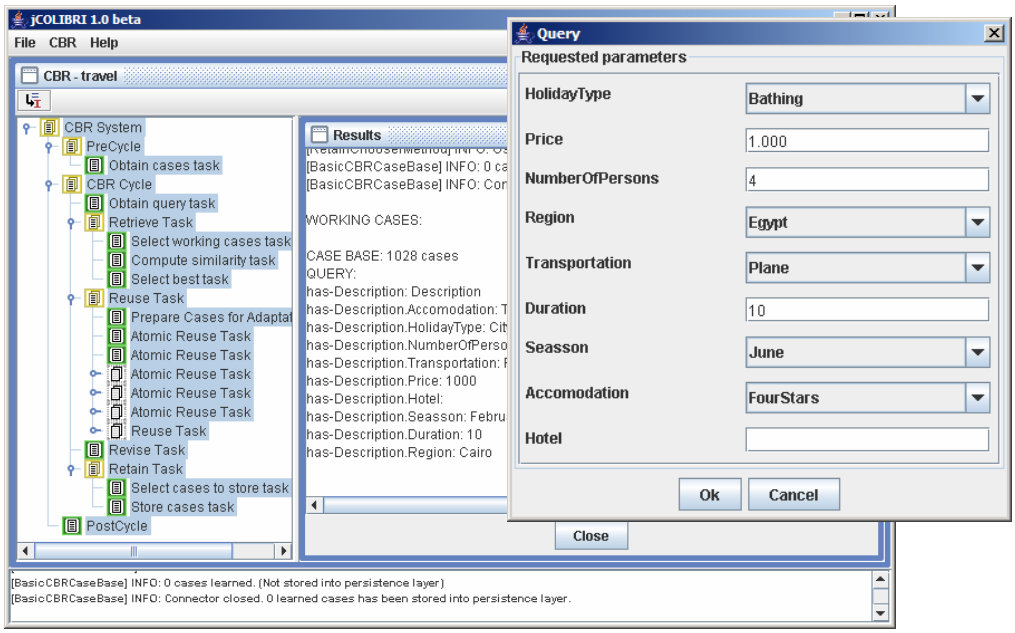


Figure 7. Testing the CBR system

6. Conclusions

jCOLIBRI is a framework that aims to formalize Case Based Reasoning and to provide a design and implementation assistance with software engineering tools. The main advantage of using jCOLIBRI is that it provides an easier development of CBR systems. To reach this goal we propose a design process that is based on reusing existing CBR knowledge (terminology, designs, tasks, methods, implementations), the integration of new components, and the extension of existing components and their collaborations.

jCOLIBRI provides graphical tools to facilitate the CBR systems design. In this paper we have focused on some design, implementation and use details regarding jCOLIBRI latest release. We have demonstrated the relevance an usefulness of the jCOLIBRI framework in view of encouraging other CBR researchers to use it.

References

1. A. Sánchez, J. A. Recio, B. Díaz-Agudo, P. A. González-Calero. Case structures in jCOLIBRI. Procs. of the AI-2005 conference. Cambridge. Springer BSC Series.
2. B. Díaz-Agudo and P. A. González-Calero. An architecture for knowledge intensive CBR systems. In E. Blanzieri and L. Portinale, editors, Advances in Case-Based Reasoning – (EWCBR'00). Springer-Verlag, Berlin Heidelberg New York, 2000.
3. B. Díaz-Agudo and P. A. González-Calero. CBROnto: a task/method ontology for CBR. In S. Haller and G. Simmons, editors, Proc. Of the 15th International FLAIRS'02 Conference. AAAI Press, 2002.
4. J. A. Recio-García and B. Díaz-Agudo. An introductory user guide to jcolibri 0.3. Technical Report 144/2004, Dep. Sistemas Informáticos y Programación, Universidad Complutense de Madrid, Spain, November 2004.
5. P. A González-Calero, B. Díaz-Agudo, J. A Recio-García, J. José Bello-Tomás. Authoring tools in Jcolibri. UK CBR 2004, Universidad Complutense de Madrid, Spain.
6. J. A. Recio, B. Díaz-Agudo, Marco A. Gómez-Martín, Nirmalie Wiratunga. Extending jCOLIBRI for Textual CBR. ICCBR 2005, Universidad Complutense de Madrid, Spain.
7. Protégé at Stanford University. http://protege.stanford.edu/
8. M. Jaczynski and B. Trousse CBR*Tools: an object oriented library for indexing cases with behavioural situation. Research Report n°3215, INRIA, July 1997. in French.
9. R. Bergmann, W. Wilke and J. Schumacher. Using Software Process Modeling for Building a Case-Based Reasoning Methodology: Basic Approach and Case Study. In D. B. Leake and E. Plaza, editors, *Case-based Reasoning Research and Development*, volume 1266 of *Lecture Notes in AI*, pages 509-518, Springer, 1997.
10. K. D. Althoff, E. Auriol, R. Barletta and M. Manago. *A Review of Industrial Case-Based Reasoning Tools*. AI Perspectives Report, AI Intelligence, 1995.
11. Bergmann, R., Wilke, W., Althoff, K.-D., Breen, S., Johnston, R. (1997). Ingredients for Developing a Case-Based Reasoning Methodology. In: R. Bergmann & W. Wilke (eds.), Proceedings of the 5 th German Workshop in Case-Based Reasoning (GWCBR'97), LSA-9701E, University of Kaiserslautern, pp. 49-58.
12. Fensel, D., Motta, E., Benjamins, V. R., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., van Harmelen, F., Musen, M., Plaza, E., Schreiber, G., Studer, R., ten Teije, A. and Wielinga, B. J. (1999). The Unified Problem-Solving Method Development Language, UPML. ESPRIT project number 27169, IBROW3, Deliverable 1.1, Chapter 1.

Fast Case Retrieval Nets for Textual CBR

Sutanu Chakraborti, Nirmalie Wiratunga, Robert Lothian and Stuart Watt

School of Computing,
The Robert Gordon University
Aberdeen AB25 1HG, Scotland, UK
Email: {sc|nw|rml|sw}@comp.rgu.ac.uk

Abstract. Case Retrieval Networks (CRNs) facilitate flexible and efficient retrieval in Case Based Reasoning (CBR) systems. While CRNs scale up well to handle large numbers of cases in the case base, the retrieval efficiency is still critically determined by the number of feature values (referred to as Information Entities) and by the nature of similarity relations defined over the feature space. For textual domains it is typical to perform retrieval over large vocabularies with many similarity interconnections between words. This can have adverse effects on retrieval efficiency for CRNs. This paper proposes an extension to CRN, called the Fast Case Retrieval Network (FCRN) that eliminates redundant computations at run time. Using artificial datasets, it is demonstrated that FCRNs can achieve significant retrieval speedups over CRNs, while maintaining retrieval effectiveness.

Keywords: Textual Case based Reasoning**,** Case Retrieval Networks

1 Introduction

A prominent theme in current text mining research is to build tools to facilitate retrieval and reuse of knowledge implicit within growing volumes of textual documents over the web and corporate repositories. Case Based Reasoning (CBR), with its advantages of supporting lazy learning, incremental and local updates to knowledge and availability of rich competence models, has emerged as a viable paradigm in this context [15]. In Textual CBR (TCBR), documents are usually mapped directly to cases [4]. Thus, a textual case is composed of terms or keywords; the set of distinct terms or keywords in the collection is treated as the feature set. In practical usage scenarios, the size of the feature set and the number of cases could both be extremely large, posing challenges to retrieval strategies and memory requirements.

The Case Retrieval Network (CRN) formalism proposed in [1] offers significant speedups in retrieval compared to a linear search over a case base. Lenz *et al.* [6,7] have successfully deployed CRNs over large case bases containing as many as 200,000 cases. The applicability of CRNs to real world textual CBR problems has been demonstrated by the FALLQ project [10]. Balaraman and Chakraborti [5] have also employed them to search over large volumes of directory records (upwards of 4 million) and more recently spam filtering has benefited from CRN efficiency gains [9].

While CRN scales up well with increasing case base size, its retrieval efficiency is critically determined by the size of the feature set and nature of similarity relations defined on these features. In textual CBR applications, it is not unusual to have thousands of terms, each treated as a feature [10]. The aim of this paper is to improve the retrieval efficiency of CRNs. We achieve this by introducing a precomputation phase that eliminates redundant similarity computations at run time. This new retrieval mechanism is referred to as Fast CRN (FCRN). Our experiments reveal that the proposed architecture can result in significant improvement over CRNs in retrieval time without compromising retrieval effectiveness. The architecture also reduces memory requirements associated with representing large case bases.

Section 2 presents an overview of CRNs in the context of textual CBR. We introduce FCRNs in Section 3 followed by an analysis of computational complexity and memory requirements. Section 4 presents experimental results. Finally conclusions appear in Section 5.

2 Case Retrieval Networks for Textual CBR

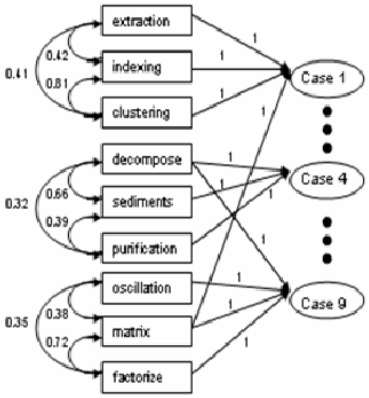
The CRN has been proposed as a representation formalism for CBR in [1]. To illustrate the basic idea we take the example case base of Fig. 1(a) which has nine cases comprising keywords, drawn from three domains, CBR, Chemistry and Linear Algebra. The keywords are along the columns of the matrix. Each case is represented as a row of binary values; a value 1 indicates that a keyword is present and 0 that it is absent in the case. Cases 1, 2 and 3 relate to the CBR topic, cases 4, 5 and 6 to Chemistry and cases 7, 8 and 9 to Linear Algebra.

Fig. 1(b) shows this case base mapped onto a CRN. The keywords are treated as feature values, which are referred to as Information Entities (IEs). The rectangular nodes stand for IEs, the oval ones represent cases. IE nodes are linked to case nodes by relevance arcs which are weighted according to the degree of association between terms and cases. In our example, relevance is 1 if the IE occurs in a case, 0 otherwise. The relevances are directly obtained from the matrix values in Fig. 1(a). IE nodes are related to each other by similarity arcs (circular arrows), which have numeric strengths denoting semantic similarity between two terms. For instance, the word "indexing" is more similar to "clustering" (similarity: 0.81) than to "extraction" (similarity: 0.42). While thesauri like WordNet can be used to estimate similarities between domain-independent terms [2], manual intervention is typically needed to acquire domain-specific similarities.

To perform retrieval, the query is parsed and IEs that appear in the query are activated. A similarity propagation is initiated through similarity arcs, to identify relevant IEs. The next step is relevance propagation, where the IEs in the query as well as those similar to the ones in the query spread activations to the case nodes via relevance arcs. These incoming activations are aggregated to form an activation score for each case node. Cases are accordingly ranked and the top *k* cases are retrieved.

Document #	extraction	indexing	clustering	decompose	sediments	purification	oscillation	matrix	factorize
1	1	1	1	0	0	0	0	1	0
2	1	0	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0
4	0	0	0	1	1	1	0	0	0
5	0	0	0	1	0	1	0	0	0
6	1	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	1	1	1
8	0	0	0	0	0	0	1	0	1
9	0	0	0	1	0	0	1	1	1

(a)



(b)

Fig. 1. Case Retrieval Network in Textual CBR

A CRN facilitates efficient retrieval compared with a linear search through a case base. While detailed time complexity estimates are available in [3], intuitively the speedup is because comparison for establishing similarity between any distinct pair of IEs happens only once. Moreover, only cases with non-zero similarity to the query are taken into account in the retrieval process.

3 Speeding Up Retrieval in Case Retrieval Networks

In this section we present the FCRN. To facilitate further analysis, we formalize the CRN retrieval mechanism described in Section 2. A CRN is defined over a finite set of *s* IE nodes E and a finite set of *m* case nodes C. Following the conventions used by Lenz and Burkhard [1], we define a similarity function σ

$$\sigma: E \times E \rightarrow \Re$$

and a relevance function

$$\rho: E \times C \rightarrow \Re$$

We also have a set of propagation functions Π_n: $\Re^n \rightarrow \Re$ defined for each node in $E \cup C$. The role of the propagation function is to aggregate the effects of incoming activations at any given node. For simplicity, we assume that a summation is used for this purpose, although our analysis applies to any choice of propagation function.

The CRN uses the following steps to retrieve nearest cases:

Step 1: Given a query, initial IE node activations α_0 are determined.
Step 2: *Similarity Propagation*: The activation is propagated to all similar IE nodes.

$$\alpha_1(e) = \sum_{i=1}^{s} \sigma(e_i, e).\alpha_0(e_i) \quad (1)$$

Step 3: *Relevance Propagation*: The resulting IE node activations are propagated to all case nodes

$$\alpha_2(c) = \sum_{i=1}^{s} \rho(e_i, c).\alpha_1(e_i) \quad (2)$$

The cases can then be ranked according to their activations.

We observe that in the face of a large number of IEs, Step 1 accounts for most of the retrieval time. The idea of FCRN stems from the motivation to identify and eliminate redundant computations during this similarity propagation.

3.1 Fast Case Retrieval Network (FCRN)

We now present an adaptation to CRN to facilitate more efficient retrieval. We observe that the final case activation in (2) can be alternately rewritten as

$$\alpha_2(c) = \sum_{j=1}^{s} \rho(e_j, c). \sum_{i=1}^{s} \sigma(e_i, e_j).\alpha_0(e_i) \quad (3)$$

Let us consider the influence of a single IE node e_i on a single case node *c*. This is computed as the aggregation of effects due to all nodes that e_i is similar to, and is given by

$$inf\ (e_i, c) = \sum_{j=1}^{s} \rho(e_j, c)\sigma(e_i, e_j)\alpha_0(e_i). \quad (4)$$

The last term can be extracted out of the summation to yield

$$inf\,(e_i, c) = \left\{ \sum_{j=1}^{s} \rho(e_j, c)\sigma(e_i, e_j). \right\} \alpha_0(e_i) \quad (5)$$

We refer to the term within parenthesis as the "effective relevance" of the term e_i to case c and denote it by $\Lambda(e_i, c)$.

It can be verified that (3) can be alternatively rewritten as

$$\alpha_2(c) = \sum_{j=1}^{s} \Lambda(e_j, c).\alpha_0(e_j) \quad (6)$$

The significance of this redefinition stems from the observation that given an effective relevance function $\Lambda : E \times C \rightarrow \Re$, we can do away with Step 2 in the CRN retrieval process above. We can now construct a Case Retrieval Network that does not use any similarity arcs in the retrieval phase. Instead, the precomputation phase makes use of the similarity as well as relevance knowledge to arrive at effective relevance values. The resulting CRN is called FCRN (for Fast CRN) and its operation is shown in Fig. 2.

Precomputation Phase

The similarity and relevance values are used to precompute the effective relevance values

$$\Lambda(e_i, c) = \left\{ \sum_{j=1}^{s} \rho(e_j, c)\sigma(e_i, e_j). \right\}$$

Retrieval Phase

Step 1: Given a query, initial IE node activations α_0 are determined.

Step 2: The resulting IE node activations are propagated to all case nodes

$$\alpha_2(c) = \sum_{j=1}^{s} \Lambda(e_j, c).\alpha_0(e_j)$$

The cases are then ranked according to their activations, and the top k retrieved

Fig. 2. Precomputation and Retrieval in FCRN

Fig. 3 shows an example CRN depicting a trivial setup with 4 IES and 4 cases, and the corresponding equivalent FCRN. It is readily observed that while the relevance values in the original CRN were sparse, the effective relevance values in the FCRN are relatively dense. This is because an Information Entity is connected to all cases that contain similar Information Entities.

In the example shown, the effective relevance between case C_1 and Information Entity IE_1 is computed as follows:

$\Lambda(IE_1,C_1)=\rho(IE_1,C_1)\sigma(IE_1,IE_1) + \rho(IE_2,C_1)\sigma(IE_1,IE_2) + \rho(IE_3,C_1)\sigma(IE_1,IE_3) + \rho(IE_4,C_1)\sigma(IE_1,IE_4)$

$= (1\times1) + (0\times0) + (0\times0.5) + (1\times0.7) = 1.7$

Other elements of the effective relevance table can be similarly computed. It is interesting to note that the effective relevance of the *i*th Information Entity with the *j*th case is given by the dot product of the *i*th row of the similarity table (σ) with the *j*th row of the relevance table (ρ).

3.2 Time Complexity Analysis

In this section we briefly compare the retrieval time complexity of FCRNs with CRNs. Fig. 4 illustrates the pseudo-codes for retrieval using the CRN and FCRN respectively.

Relevance function ρ

	IE_1	IE_2	IE_3	IE_4
C_1	1	0	0	1
C_2	0	1	1	0
C_3	1	0	1	0

Similarity function σ

	IE_1	IE_2	IE_3	IE_4
IE_1	1	0.0	0.5	0.7
IE_2	0.0	1	0.5	0.0
IE_3	0.5	0.5	1	0.3
IE_4	0.7	0	0.3	1

Effective Relevance function Λ

	IE_1	IE_2	IE_3	IE_4
C_1	1.7	0	0.8	1.7
C_2	0.5	1.5	1.5	0.3
C_3	1.5	0.5	1.5	1.0

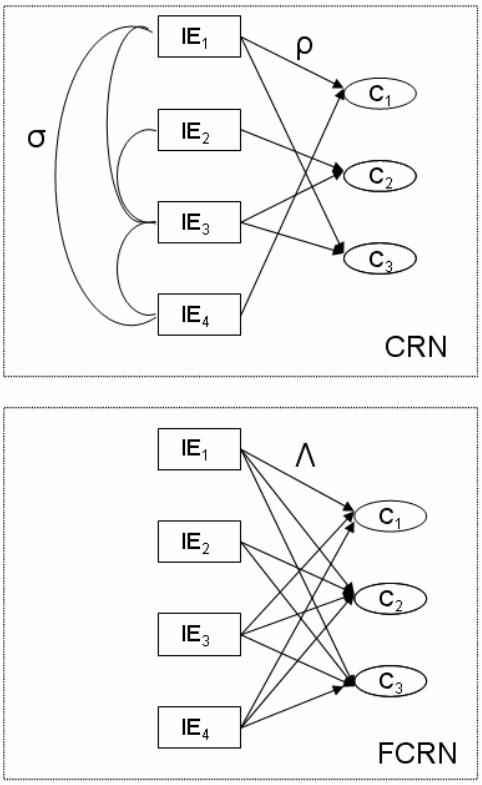


Fig. 3. A CRN over 3 cases and 4 IEs, and an operationally equivalent FCRN

CRNRetrieval

```
FOR each activated query IE (attribute A, value Vq in query)        /* A */
    Determine all related IEs using similarity function σ
    FOR each IE that is found relevant                              /* B */
        Determine all cases relevant to that IE using relevance function ρ
        Increment  scores of relevant cases
    END FOR
END FOR
Rank and display related cases
```

FCRNRetrieval

```
FOR each activated query IE (attribute A, value Vq in query)        /* B */
        Determine all cases relevant to that IE using effective relevance function Λ
        Increment scores of relevant cases
END FOR
Rank and display related cases
```

Fig. 4. Pseudo-codes for retrieval using CRN and FCRN

The retrieval complexity is a function of loops /* A */ and /* B */ in the pseudo-codes :

$$complexity(CRNRetrieval) \propto O(A \times B)$$

and

$$complexity(FCRNRetrieval) \propto O(B)$$

The following two reasons contribute to the speedup in FCRN retrieval:

(a) Step A in the CRNRetrieval pseudo-code involves spreading activation to IE nodes similar to the query IEs based on similarity values. This step is eliminated in FCRN retrieval since the similarity knowledge is transferred to the effective relevance values during the precomputation step. Thus, FCRN retrieval amounts to a simple table lookup for all cases "effectively" relevant to the query IEs and aggregating the scores received by each case from the individual query IEs. Using FCRNs, we can obtain efficiency very similar to inverted files typically used in Information Retrieval applications [8]. However unlike inverted files, FCRNs also integrate similarity knowledge in the retrieval process.

(b) Step B in FCRN retrieval involves a loop over IE nodes activated by the query. In contrast, Step B of the CRN retrieval loops over all IEs *similar to* IE nodes activated by the query. In a situation where most IEs are connected to many others by non-zero similarities, Step B in FCRN would involve much fewer iterations compared to step B of a CRN.

The downside of FCRNs is that incremental and batch maintenance of the case base involves extra precomputations. The effective relevance values need to be re-computed each time new cases are inserted or existing cases deleted or edited. However, the recomputations can be limited to only those effective relevance values that could potentially be affected.

3.3 Memory Requirements

Typically CRNs consume more memory when compared to a flat case base. This difference can be largely attributed to the following two factors: (a) CRNs explicitly record $|E|$ number of values corresponding to Information Entities. (b) Also $|E|^2$ values are required to model similarities between IEs. In addition we have $|Casebase| \times |Attribute|$ relevance values between the IEs and the cases.

The memory requirement of a CRN is approximately given by

$$\begin{aligned} memory\ (CRN) &= |E| + |CaseBase| + |E|^2 + |Casebase| \times |Attribute| \\ &= |E| + |E|^2 + |CaseBase| \times (|Attribute|+1) \\ &\approx |E| + |E|^2 + memory(flat\ case\ base) \end{aligned}$$

In FCRN we do not need to explicitly record the similarities between IEs. The memory requirement of FCRN is given by

$$\begin{aligned} memory\ (FCRN) &= |E| + |CaseBase| \times (|Attribute|+1) \\ &\approx |E| + memory(flat\ case\ base) \end{aligned}$$

In textual CBR applications, the number of IEs could be extremely large, and the saving of $|E|^2$ could mean substantial gains in terms of memory requirements.

It is worth noting that while the in-memory requirement for FCRN retrieval is considerably less than in CRN, we would still need to store the $|E|^2$ similarity values for off-line maintenance. In a situation where a particular IE is deleted, we would need to re-evaluate the effective relevance values to reflect this change. This is possible only when the similarity information is available.

4 Experimental Results

In this section, we present empirical results to illustrate FCRN efficiency in practical applications. The datasets used for our experiments are not real textual case bases. Rather, a large number of IEs and cases were simulated with randomly generated similarity and relevance values. The synthetic nature of the datasets is not a major concern, since we are not really concerned with the actual cases retrieved. Our main intent is to observe how CRNs and FCRN scale up with case base size. A similar experimental strategy was also used in [14].

Table 1 shows the impact of the increase in number of IE nodes on the retrieval time. The case base has 1000 cases and the similarity matrix is optimally dense, that is each similarity node is connected to each other by a non-zero similarity value. Thus this result may be viewed as a worst-case comparison of the CRN performance against FCRN. While CRN performance degrades steeply with growth in the number of IEs, FCRN is stable. This is attributed to the savings in similarity computation, and corresponds closely to our theoretical analysis in Section 3.2.

We also conducted experiments to relax the density of similarity connections and compare FCRN performance against CRN. The results are shown in Table 2. We use 8000 IE nodes and 1000 cases in our experiments. The density refers to the proportion of non-zero similarity values in the similarity matrix. As the density increases from 0 (when no IE node is similar to any other node) to 1 (when all IE nodes are related to all others), the CRN retrieval slows down considerably. Since FCRN does away with the step of similarity propagation across IEs, its performance is not impeded by growth in similarity matrix density.

It may be noted that the retrieval times in Tables 1 and 2 are rounded off to two significant digits. This results in very low FCRN retrieval times being recorded as 0.00 secs.

Table 1. Retrieval time as a function of number of IEs

No. of IE Nodes	CRN Retrieval Time (secs.)	FCRN Retrieval Time (secs.)
1000	0.04	0.00
2000	0.12	0.00
3000	0.22	0.00
4000	0.35	0.00
5000	0.49	0.00
6000	0.66	0.00
7000	1.42	0.01
8000	3.40	0.01
9000	3.86	0.01
10000	4.98	0.02

Table 2. Retrieval time as a function of the density of similarity matrix

Density of the Similarity Matrix	CRN Retrieval Time (secs.)	FCRN Retrieval Time (secs.)
0	0.00	0.00
0.2	0.92	0.00
0.4	1.71	0.00
0.6	2.43	0.00
0.8	2.81	0.00
1.0	3.38	0.01

5 Conclusion

We have presented a Fast Case Retrieval Network formalism that remodels the retrieval mechanism in CRNs to eliminate redundant computations. This has significant implications in reducing retrieval time and memory requirements when operating over case bases indexed over large numbers of Information Entities and cases. A theoretical analysis of computational complexity and memory requirements comparing FCRNs against CRNs is presented. Experimental results over large case bases demonstrate significant speedup with FCRN. While we have used textual CBR as the running theme for presenting our work, FCRN could, in principle, be applied to any large scale CBR application.

References

1. Lenz, M., Burkhard, H.-D.: Case Retrieval Nets: Basic Ideas and Extensions. KI (1996) 227-239
2. Chakraborti, S., Ambati, S., Balaraman, V., Khemani, D.: Integrating Knowledge Sources and Acquiring Vocabulary for Textual CBR. In Proceedings of the 8th UK CBR Workshop (2003) 74-84
3. Lenz, M., Burkhard, H.: Case Retrieval Nets: Foundations, Properties, Implementation, and Results, Technical Report, Humboldt-Universität zu Berlin (1996)
4. Lenz, M.: Knowledge Sources for Textual CBR Applications, Textual Case based Reasoning: Papers from the 1998 Workshop Technical Report WS-98-12 AAAI Press (1998) 24-29
5. Balaraman, V., Chakraborti, S.: Satisfying Varying Retrieval Requirements in Case-Based Intelligent Directory Assistance. Proceedings of the FLAIRS Conference (2004)
6. Lenz,M.: Case Retrieval Nets Applied to Large Case Bases. In: Proc. 4th German Workshop on CBR, Informatik Preprints, Humboldt-Universität zu Berlin (1996)
7. Lenz, M., Auriol, E., Manago, M. : Diagnosis and Decision Support (Chapter 3) In: Lenz,M., Bartsch-Sporl,B.,Burkhard,H.-D.,Wess, S. (eds.) Case Based Reasoning Technology, Lecture Notes in Artificial Intelligence 1400, (1998) 51–90
8. Rijsbergen, C. J.: Information Retrieval. 2nd edition, London, Butterworths (1979)
9. 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, Procs. of AI 2004, Springer (2004) 3-16
10. Lenz,M., Burkhard, H.-D.: CBR for Document Retrieval - *The FAllQ Project.* In: Leake, D., Plaza, E. (eds.) : Case-Based Reasoning Research and Development, Springer Verlag, LNAI 1266 (1997)
11. Chakraborti, S., Watt, S.,Wiratunga, N: Introspective Knowledge Acquisition in Case Retrieval Networks for Textual CBR. Proceedings of the 9th UK CBR Workshop , Cambridge, UK (2004)
12. Wilson,D., Bradshaw,S.: CBR Textuality. In Proceedings of the Fourth UK Case-Based Reasoning Workshop (1999) 67-80
13. Lytinen, S.L., Tomuro,N. : The Use of Question Types to Match Questions in FAQFinder, in Mining Answers From Texts and Knowledge Bases, AAAI Technical Report SS-02-06, AAAI Press (2002) 46-53
14. Lenz, M.: Case Retrieval Nets as a Model for Building Flexible Information Systems, PhD dissertation, Humboldt Uni. Berlin. Faculty of Mathematics and Natural Sciences (1999)
15. Lenz, M., Hubner A, Kunje M.: Textual CBR (Chapter 5) In : Lenz, M., Bartsch-Sporl, B., Burkhard, H.-D, Wess, S. (eds.): Case Based Reasoning Technology, Lecture Notes in Artificial Intelligence 1400, (1998) 115-137

Addressing the 'Vocabulary Gap' in Product Recommenders

Edwin Costello, John Doody, Lorraine McGinty and Barry Smyth

School of Computer Science and Informatics, UCD Dublin, Belfield, Dublin 4
Email: {firstname.surname}@ucd.ie

Abstract. E-Commerce consumers are routinely overwhelmed with the sheer volume of product choices available to them and often find it difficult to relate their subjective preferences to the presented product descriptions. In this paper we describe a work in progress that concentrates on how research ideas from conversational recommender systems and intelligent user interfaces can be combined. More specifically we look at online retail environments where a conflict can exist between preference elicitation and the vocabulary used to describe recommendations. In particular, we are interested in case-based conversational recommenders that rely heavily on explicit feature-level feedback to help narrow down the number of relevant products for a user. The key challenge in these e-Commerce domains is how to avoid lost sales as a consequence of users being unable/unwilling to provide such fine-grained feature-specific feedback. In this paper we introduce the *i*CARE System, which provides intelligent assistance for recommending suitable eyeglasses to individuals. We describe how the *i*CARE System represents products and handles preference elicitation through a merger of conversational recommendation and visualization techniques.

Keywords: Conversational Case-Based Recommendation, Preference Elicitation, Intelligent User Interfaces, E-Commerce Applications.

1 Introduction

A key problem with case-based recommenders is the underlying assumption that users are readily able (and willing) to describe their needs and preferences in terms of the product features that are available to the recommender. Considerable research indicates that this is not always possible for a variety of reasons [e.g., 1, 2, 3, 4, and 5]. Product domain examples such as jewelry, clothing, technology, and art, are especially challenging. For example, in each of these domains, detailed case descriptions of the recommendation items are usually readily available, but users are often unable to understand and map how these relate to their subjective needs. This amounts to a '*vocabulary gap*' in case-based recommenders, often due to limited user expertise of domain characteristics, which renders available case descriptions effectively useless.

Take, for example, an online user seeking to purchase a diamond engagement ring. While a multitude of distinctive features describe every engagement ring, the stone makeup itself is usually the most important aspect. Features that characterize the stone include, *weight*, *color*, *carat*, *clarity*, *cut*, *girdle*, *fluorescence*, *rest*, *polish*, to name but a few. Importantly, these features are readily available and these are the precise characteristics that influence recommendation for experts in the domain (e.g., jewelers)[1].

It is understandable that the majority of users may often be unable to provide a recommender with such precise feature-specific feedback; feedback that it *needs* to influence retrieval. This severely limits the recommender's ability to narrow down the range of suitable alternatives, thus compromising overall efficiency.

[1]In the 1950's, Gemological Institute of America Inc. (GIA) created the International Diamond Grading/Reporting System, and established the standards that revolutionized the diamond industry. By this system the characteristics of every diamond are precisely described by a lengthy set of technical feature-value pairs and a quality grading. The process is involved and very strict but by the end of it quality certified diamonds are the result. This both protects the consumer and the jeweler from fraud and helps determine the price of any particular diamond.

With these points in mind we focus on one such domain, and discuss how we integrate ideas from conversational case-based recommender systems and product visualization through the interface of an intelligent customer assistant. We introduce the *i*CARE system (i.e., Intelligent Customer Assistance for Recommending Eyewear). *i*CARE suggests eyewear (spectacles, sunglasses etc.) to users. A recommendation facility is appropriate in this domain given that the range of alternatives is enormous and often outweighs a user's ability to survey them all. Importantly, structured case descriptions are available for all eyewear products in the form of feature-value representations. However, few of these features are likely to be meaningful/highly influential to the user, and so they may be unable to provide crucial feedback to the recommender system in terms of these specific features. It is also a domain where a user's subjective appreciation of the *effect* brought about by a recommendation has an enormous influence. At the end of the day, how a pair of glasses will look on the user will largely influence their final purchase decision. Thus, our recommender system must include a visualization component such that users may appreciate how a particular pair of glasses will look when worn.

In the following sections we describe underlying architecture and basic operation of the *i*CARE system, highlighting how eyeglasses *cases* are represented within system. Section 3 focuses on the technical details relating to one of the central components of the system namely the recommendation engine. Finally, in section 4, an illustrative walkthrough of the system is provided to illustrate the effect of combining these visualization and recommendation processes.

2 The *i*CARE System

The *i*CARE System is an online system that allows users to shop for suitable glasses (i.e., frames). The overview of its current functionality is as follows: (1) a user can upload their picture to the system, (2) the *i*CARE system processes the image using effective feature-detection algorithms in order to pin-point the precise location and dimensions of the user's eyes, (3) the user enters into a conversational dialogue with the system, where the system recommends frame options over a series of recommendation cycles. The user can see these frames on their uploaded picture and provide feedback, (4) the *i*CARE system uses their feedback to adapt the behavior of the recommendation component and influence subsequent retrievals. In addition, the user can interact with *i*CARE in a variety of other ways.

2.1 Overview of System Architecture

The following subsections provide an overview of the basic *i*CARE system architecture. More precise technical details relating to the functioning of the *i*CARE application will be discussed in later sections. The *i*CARE system (Fig. 1) has three key component layers; (1) the product data representation layer, (2) the application layer, and (3) the user interface layer.

2.1.1 Representing Products as Cases

The *i*CARE system uses a case-based representation to store product information. Each case stores a detailed content description of a pair of glasses, along with a corresponding image of the glasses described. Currently, there are a total of 3061 cases in the *i*CARE case base. A partial eyeglasses *case* is shown at the bottom of Fig 1, with each case description represented by a list of *feature-value* pairs. There are two main types of *feature-value* pairs, nominal and numeric. Examples of nominal and numeric descriptive features common to all cases include *price*, *shape*, *lens size*, *bridge size* and *material*. In total, each case is described by 12 individual feature-value pairs. In addition, a corresponding image of the product option (i.e., a frame) that each case describes is also stored.

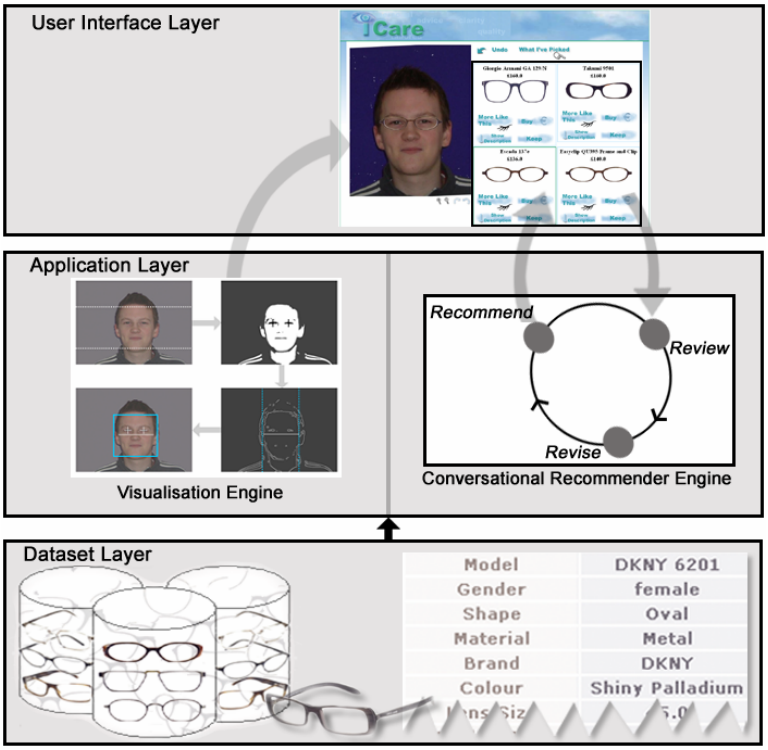


Fig. 1. Overview of the basic *i*CARE system architecture

2.1.2 Application Overview

There are two crucial components central to the functioning of the *i*CARE application: the conversational recommendation engine and the product visualization engine. This section provides details relating to the responsibilities and operational overview of these two components and in Section 3 the recommender engine will be discussed in greater detail. Ultimately, the iCARE system will be accessible to users online through a retailer's website, or in-store through a kiosk/interactive screen. SpecSavers Optical Group Ltd. , for example, already offer their in-store customers a kiosk-based opportunity to try on different frame options, while having their picture taken and displayed at the same time. Importantly, the service they provide does not allow users to interact in any other way, nor do they have access to product data or capability of seeing suggestions; they simply take pictures and display these for the customer's information.

The conversational recommender engine is responsible for retrieving relevant product recommendations in view of user feedback, and routing these to the interfacing layer. Importantly, the recommendation approach supports a conversational interaction between a user and the system, and is based on the comparison-based framework proposed by McGinty and Smyth [6]. The basic algorithm behind the approach we have implemented can be summarized as follows: (1) new items are *recommended* to the user based on the current query; (2) the user *reviews* the recommendations. Specifically they are given the opportunity to 'try on' the options and indicate which option they prefer the most; (3) information about the difference between the selected item and the remaining alternatives is used to *revise* the query for the next cycle. The recommendation process terminates when the user is presented with a suitable recommendation.

The visualization engine is responsible for the image processing functions within the *i*CARE system and provides the product visualization/'Try On' capability at each interaction cycle. Facial feature detection algorithms are used to detect the location of the eyes in the image as well as the width, height and skin tone of the face. Briefly, in detecting the eyes for example, the first step is to convert the image into a monochrome image that emphasizes the eyes, nose and mouth areas, leaving the eyes, nose and mouth as black regions

against a predominantly white face, Fig. 2. A vertical scan is performed through the image and location of the eyes can be estimated based on changes in the number of black and white pixels in a given area. The width and height of the face are also calculated using an edge detection technique[7] . (see [8] for a more detailed explanation). Determining the location of the eyes and facial dimensions using techniques such as these, allows the recommender to help users visualize how a pair of glasses will look when worn

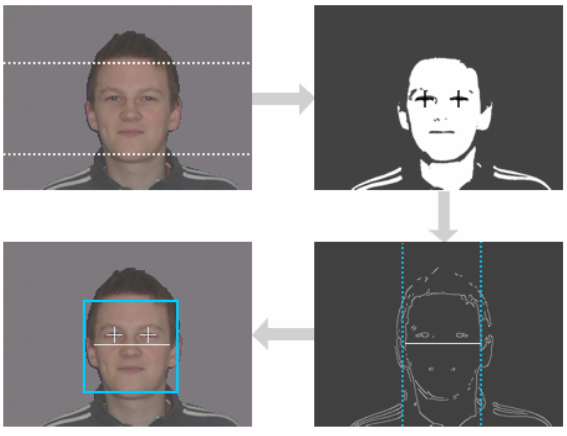


Fig. 2. The key facial detection steps

2.1.3 User Interfacing

The primary roles of the user interface layer are to handle message passing between the user and the application layer and to display the outputs of the recommender and visualization engines. By design, the *i*CARE user interface is clear and intuitive; the left-hand side of the interface is dedicated to visualization interaction and the right-hand side is reserved for the display and review of product recommendations (Fig. 4). Aside from having the opportunity to see the *visual effect* for each recommendation, the user also has the opportunity to directly apply a range of further image adjustments as they feel necessary. Examples include image tilts, zoom-in, zoom-out etc. In addition, at any stage the user may review the technical descriptions that relate to each recommendation alternative, or backtrack to an earlier recommendation cycle.

The provision of product visualization as well as product descriptions is useful as it caters for both novice and expert users. Product visualization is useful at the start of the session where a user can get a good idea of the style of glasses that suits them best without having to provide exact values for specific technical features. Later in the recommendation session the user may indeed wish to consult the recommendation descriptions to better appreciate the trade-offs between *neck and neck* alternatives (e.g., price differences could be influential in their final purchase decision).

Ultimately, the *i*CARE system will be accessible to users online through a retailer's website, or in-store through a kiosk/interactive screen. SpecSavers Optical Group Ltd.[2] , for example, already offer their in-store customers a kiosk-based opportunity to try on different frame options, while having their picture taken and displayed at the same time. Importantly, the service they provide does not allow users to interact in any other way, nor do they have access to product data or capability of seeing suggestions; they simply take pictures and display these for the customer's information.

3 The Recommendation Engine

The *i*CARE conversational recommender engine supports an iterative interaction with the user, providing them with cyclic feedback opportunities to influence retrieval. In terms of preference elicitation the

[2]A well-known European-based optician.

assumptions we make here are: (1) users are capable of recognizing what they like when the see it, (2) users are willing to provide minimal preference information on examples in order to see more suitable recommendation results. As mentioned earlier the algorithm we have implemented is based on the comparison-based framework proposed in [6], and has three key stages: Review, Revise and Retrieve. In this section we describe what happens at each of these stages in the *i*CARE System.

4.1 Review

At the review stage of each recommendation cycle the user is afforded the opportunity to provide feedback over presented alternatives. It is intended that in each cycle the user need only provide high-level preference-based feedback, PBF, (largely based on visual preference). They do this by clicking the 'More Like This' option associated with the recommendation alternative that they feel suits them best as illustrated in Figs. 4 and 5. Importantly, there is a direct mapping between the high-level feedback they provide and the feature-based item descriptions (i.e., cases) available to the system.

4.2 Revise

The *i*CARE recommender responds and revises its understanding of a user's personal requirements (i.e., the evolving query) at the revise stage of each recommendation cycle. Once a user has indicated their preference cases from amongst the recommended alternatives the corresponding feature-based description for that case is used to do this. There are a number of ways that a user's preference-based feedback can be utilized (at the query revise stage) in order to influence retrievals in the next cycle. The most common approach is to use the preferred case description as the query for the next cycle (Fig. 3). Smyth and McGinty describe a number of alternatives to this More-Like-This (MLT) approach [6], such as the idea of weighting certain features according to the number of alternatives presented. The current version of *i*CARE allows for the use of each of these proposed strategies, as well as others that use feedback gathered over a number of cycles to influence retrieval. Investigations are underway to determine the best strategy to use in this domain and a real-user study of the whole system is also planned. For the purposes of this discussion we will be confining ourselves to the basic MLT strategy.

Fig. 3. Illustrating query revision using feature values from the preference case

4.3 Retrieve

Before the recommender can present the user with the *k* most similar cases to their most recent preference, the remaining product cases are ranked in decreasing order of their similarity to the current query, *Q*, according to Equation 1. In this equation all the features are treated equally but weights in the range *0* to *1* can be introduced to give certain features more influence on the final similarity. Accordingly, the final score is always a number between *0* and *1*.

$$\mathrm{Sim}(Q,C) = \frac{\sum_{i=1}^{n} \mathrm{featureSim}(F_{Qi}, F_{Ci})}{n} \quad \textbf{(1)}$$

When calculating similarity at the feature level nominal and numeric values need to be handled differently. For nominal features an exact match comparison is carried out, returning the value *1* when the values match and *0* otherwise. Numeric values, on the other hand, use their relative difference as a basis for similarity calculation. The equation for this is shown in Equation 2 where F_Q and F_C are the values for the numeric features being compared.

$$\mathrm{featureSim}(F_Q, F_C) = 1 - \frac{|F_Q - F_C|}{\max(F_Q, F_C)} \quad \textbf{(2)}$$

5 Sample Session Walk-Through

At the beginning of a recommendation session with the *i*CARE System, a user is required to upload their digital picture to the system and provide some basic query constraints (i.e., *price, shape, gender*) if they wish. The resulting query is made up of the users facial dimensions (following processing), and a list of these feature-value pairs. Once the submits this query they are presented with a set of *k* recommendations and are invited to *"Try These On"* if they wish. Fig. 4 illustrates the effect of a user asking to try on a specific recommendation from a set of four presented alternatives. The user can indicate which option they prefer by clicking the *"More Like This"* link associated with their choice.

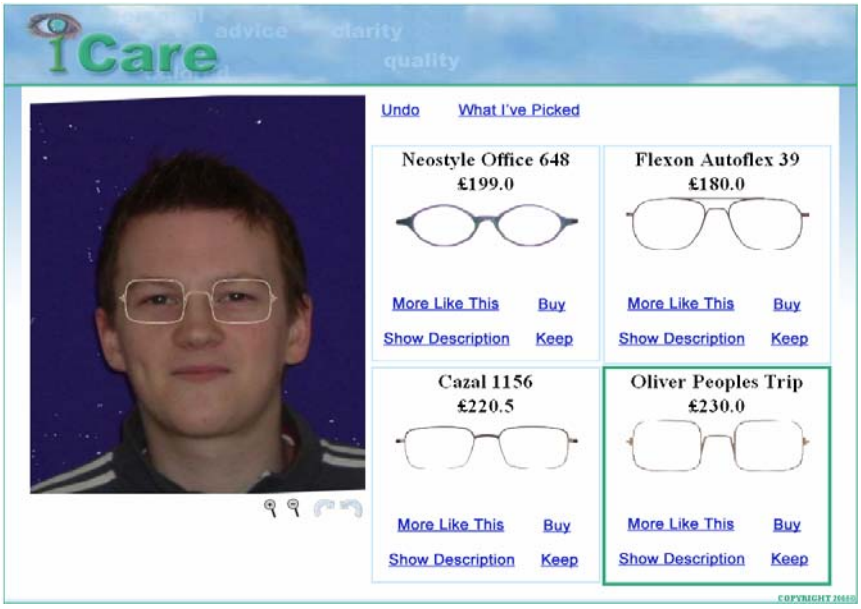


Fig. 4. A set of recommendations is presented to the user. He decides to try in the 'Oliver Peoples Trip' pair of glasses and the frame for that pair is superimposed over his image.

This will bring up a new set of recommendations, i.e., those that are most similar to their previous preference (Fig. 5). The user can decide to fit on these options as before, and/or view their associated feature descriptions by clicking the *"Show Description"* button). Irrespective of how the user arrives at their decision, the recommender will always wait for a user to indicate which product they prefer the most, and retrieve

similar alternatives to this preference. This process continues until the user decides to *"Buy"* a presented option.

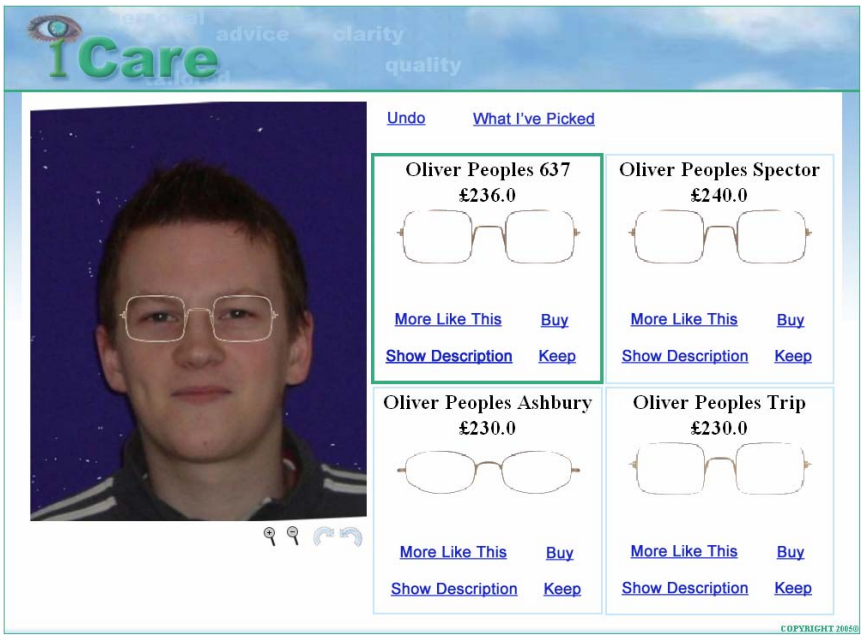


Fig. 5. Having tried on the pair of glasses as shown in Fig. 4, the user selected that pair as his preferred item. Shown here is the set of recommendations returned by the system using that item as a query.

6 Conclusions and Future Work

A key challenge for recommender systems research in the area of e-Commerce is the accurate modelling of user preferences in the course of a *once-off* recommendation session [5, 9, 10]. A closely related topic of interest in the user interfacing community is the notion of providing users with more customized services through intuitive controlled interaction. This project set out to investigate research ideas leading to improved interfacing and recommendation processes in product domains that lend themselves well to case-based techniques and basic preference elicitation modes. These domains tend to have the following key characteristics; (1) detailed (often technical) descriptions are available for recommendation items; (2) other recommendation techniques, such as collaborative recommendation, tend to be unsuitable; (3) a vocabulary problem exists between the user and the system, in the sense that users find it difficult to provide precise feature preference information in relation to the descriptions presented.

In this paper we describe how valuable low-cost, high-level preference-based feedback can be volunteered by the user through visualization measures, and how this low-cost feedback can be translated and utilized by a conversational case-based recommender through the use of an intelligent customer assistant. The *i*CARE system allows users to shop for suitable glasses by providing the facility for users to visualize and appreciate the *effect* of each recommendation option (i.e., see how different frame options look on them). We propose that users are capable of providing high-level preferential feedback in the form of "I like it, or not". Users need not be relied upon to understand or describe precise features as they relate to their preferences. Instead this is implicitly captured by the feature-level item descriptions (i.e., the causes that bring about the effects).

We propose that this mode of user-system interaction is very intuitive and presents only a low-cost to the user. It is very natural that a user may be more interested in the visual effect (how the glasses will look), as opposed to specific technical features (e.g., lens size, weight), particularly at the early stages of the

recommendation process. Similar arguments can be made for other e-Commerce domains (e.g., clothing, hairstyles, cosmetic surgery and artwork). Importantly, the user has the option to review the item feature descriptions at any stage.

We are now at a stage where more advanced recommendation strategies can be employed and evaluated in the *i*CARE system. This could include using the face proportions a user has to perhaps recommend the type of glasses that would best suit their facial shape and/or skin tone. As part of our future work agenda we intend to examine this possibility as well as investigate more sophisticated ways of query revision. Currently, query revision uses only feedback information captured in a single cycle. We have begun to investigate various cumulative revision strategies that could further improve the overall performance, as well as provide justifications for particular recommendations. Finally, we are also in the process of setting up a large-scale real-user trial in order to gain further useful feedback.

Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland.

References

[1] Ardissono, L., Goy, A.: Tailoring the Interaction with Users in Web Stores. In: User Modelling and User-Adapted Interaction. 10(4). Hingham, MA, USA (2000) 251–303

[2] Felix, D., Niederberger, C., Steiger, P., Stolze, M.: Feature-Oriented vs Needs-Oriented Product Access for Non-Expert Online Shoppers. In: Proceedings of the First IFIP Conference on e-Commerce, e-Business and e-Government, Zurich, Switerland (2001) 399–406

[3] Grenci, R.T., Tood, P.: Solutions-Driven Marketing. Communications of the ACM 45(3) (2000) 209–249

[4] Shimazu, H., Shibata, A., Nihei, K.: Expertguide: A Conversational Case-Based Reasoning Tool for Developing Mentors in Knowledge Spaces. Applied Intelligence 14(1) (2001) 33–48

[5] 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), Dublin, Ireland (2003)

[6] McGinty, L., Smyth, B.: Comparison-Based Recommendation. In Craw, S., ed.: Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02), London, UK, Springer (2002) 575-589

[7] Canny, J.: A Computational Approach to Edge Detection. In: IEEE Transactions on Pattern Analysis and Machine Intelligence. **8**(**5**) (1986) 679–698

[8] Costello, E., Doody, J., McGinty, L., Smyth, B.: Using Product Visualization for Preference Elicitation in Case-Based Recommenders. In: Petridis, M. (eds) Proceedings of the 10th UK Workshop on Case-Based Reasoning (UKCBR), Cambridge, UK. (2005) 49-58

[9] McSherry, D.: Explanation of Retrieval Mismatches in Recommender System Dialogues. In: Proceedings of the ICCBR-03 Workshop on Mixed-Initiative Case-Based Reasoning, Trondheim, Norway (2003) 191–199

[10] Ricci, F., del Missier, F.: Supporting Travel Decision Making Through Personalized Recommendation. Designing Personalized User Experiences in eCommerce (2004) 231–251

A Cumulative Approach to Query Revision in Case Based Recommender Systems

John Doody, Edwin Costello, Lorraine McGinty and Barry Smyth

School of Computer Science and Informatics,
UCD Dublin, Belfield, Dublin 4, Ireland.
Email: {firstname.surname}@ucd.ie

Abstract. The growth in e-Commerce and the continued influence it has on our lives has led to the development of supportive technologies that help consumers locate target products easily. As a consequence, conversational case-based recommender systems that narrow the number of relevant products according to user feedback have been the focus of much recent study. One major theme of research here has looked at how user feedback can best inform this process whilst at the same time minimise recommendation session length. In this paper we concentrate on how a user's preference-based feedback over presented product cases can be used to improve the accuracy of case retrievals, and subsequently reduce session lengths. We propose and evaluate two alternative query revision strategies that revise the recommender system's understanding of what the user is looking for based on the *cumulative feedback* they provide in the context of a given recommender session.

Keywords: Query Revision, User Feedback, Recommender Systems

1 Introduction

Product recommender systems help users to navigate towards suitable products of interest when their ability to compare and evaluate the options is over-whelmed by the number of possibilities. The type of recommender system technology to use depends on the level (and type) of product description information that is available [4, 5, 14]. In the absence of detailed feature-rich content information, collaborative filtering techniques are typically used to identify and make recommendations to users with similar tastes [8, 16]. On the other hand, if detailed feature-rich descriptions are available for the recommendation items, case-based recommenders may be more appropriate [6]. The classic CBR recommendation approach assumes that a history of feature-rich prior case experiences (e.g., product transactions, purchases, etc.) is available, and it is expected that these cases will be useful for guiding future searches in other relevant recommendation sessions [7. 15]. Session-independent recommendation refers to recommendation scenarios of this kind, where case-based recommendations are made based on information accumulated independently of the current session [3, 17]. Session-dependent recommendation, on the other hand, refers to recommendation scenarios where only information gathered in the context of the given session can be used to generate recommendations for an individual user [3]. Recommendation tasks where the system has no prior interaction history for a user, or domains where a user's historical record has no influence (e.g., once-off irregular purchases), are well-suited to this kind of recommendation approach [17].

Recent research in the area of case-based recommendation has highlighted the importance of using user feedback to inform retrievals in session-dependent recommendation scenarios. Such conversational CBR recommenders [2, 9] tend to engage the user in a cyclic dialogue where recommendation options are presented to the user and the user provides feedback on these options. Importantly, this user feedback allows a recommender to make better suggestions by adapting its current understanding of the user's requirements. The recommendation task is somewhat straightforward, although hardly trivial, when the user's requirements are clearly specified - a standard similarity-based search technique will produce a ranked list of recommendations based on their similarity to the user's fully specified requirements. Of course in the real world, and especially in consumer application domains where recommender systems are commonly deployed, user requirements are rarely fully specified, leaving the recommender to work with a partial set of

requirements that are unlikely to probe the product space in the region of a suitable product during an initial retrieval. In previous related work, CBR researchers such as Shimazu [18], McSherry [13], and McGinty & Smyth [10, 11, 20], have tried to understand and implement ways of addressing this preference elicitation problem (see Section 2). In all of these works the aim has been to adapt the query (i.e., the system's understanding of the user's product need) by using feedback collected in a given recommendation cycle, on a set of alternative cases. A further objective has been to limit the number of recommendation cycles required before a user is presented with their target case.

In this paper we concentrate on how user feedback (i.e., preference-based feedback) on presented product cases can be used to improve the accuracy of case retrievals and consequently reduce session lengths. We propose and evaluate two alternative query revision strategies that revise the recommender system's understanding of what the user is looking for based on the cumulative feedback they provide solely in the context of a given recommender session. Before describing these strategies in Section 3, we briefly discuss some related research in this area in Section 2. Finally, we report some of the initial evaluation results we have found in relation to our strategies in Section 4.

2 Comparison Based Recommendation

The comparison-based recommendation algorithm [11], provided in Fig 1, defines a three step procedure for each iterative recommendation cycle. Each cycle can be summarized as follows: (1) new items are recommended to the user based on the current query; (2) the user reviews the recommendations and indicates which option they prefer; (3) information about the difference between the selected item and the remaining alternatives is used to revise the query for the next cycle. The recommendation process terminates either when the user is presented with a suitable recommendation or when they give up.

```
1.    define Comparison-Based-Recommend(Q, CB, k)
2.    begin
3.      Do
4.        R ← ItemRecommend(Q, CB, k)
5.        c_p ← UserReview(R, CB)
6.        Q ← QueryRevise(Q, c_p, R)
7.      until UserAccepts(c_p)
8.    end
9.    define ItemRecommend(Q, CB, k)
10.   begin
11.     CB' ← sort cases in CB in decreasing order of their sim to Q
12.     R ← top k cases in CB'
13.     return R
14.   end
15.   define UserReview(R, CB)
16.   begin
17.     c_p ← user selects best case from R
18.     CB ← CB - R
19.   return c_p
20.   define QueryRevise(Q, c_p)
21.   begin
22.     for each f_i ∈ c_p
23.       Q ← add f_i
24.     end for
25.     return Q
26.   end
```

Fig. 1. The Comparison-Based Recommendation Algorithm

2.1 Recommend

At the recommend stage of each comparison recommendation cycle (lines 9 to 14), k cases are presented to the user. These recommendations are typically generated using a similarity algorithm that returns the k most similar cases to the current query. The problem with using a similarity approach is that the recommendations that are returned may be very similar to each other and this limits the coverage of the recommender in each cycle. Unnecessarily long recommendation sessions can be the result of very dense product spaces.

Ways of addressing this problem include introducing diversity or looking at ways of combining similarity and diversity [1, 13, 19]. A number of strategies for improving recommendation diversity have been suggested. Shimazu [18], for example, proposes the recommendation of three items or cases (i_1, i_2 and i_3) per recommendation cycle: i_1 is the most similar case to the query; i_2 is maximally dissimilar from i_1; and i_3 is maximally dissimilar from i_1 and i_2. By design these items are maximally diverse, but similarity to the query may be compromised as there are no guarantees that i_2 and i_3 will be very similar to the query.

Other researchers have proposed alternative diversity enhancing mechanisms with a more flexible balance between item diversity and query similarity [1, 13, 19]. Smyth & McGinty [20] propose a flexible mechanism for introducing recommendation diversity based on the feedback a user provides in each recommendation cycle. They propose the adaptive selection approach where the most recent preference case is carried from cycle-to-cycle, and this is used as a way of assessing the recommendation focus. For instance, a re-selection of the same preference case by the user is seen as evidence that the search is incorrectly focused. Diversity is introduced in order to maximise the coverage of the item space in the next cycle, and so help to refocus the recommender by offering the user contrasting recommendations. In the event that a user selects a new preference case then a pure similarity-based selection method can be used to refine the recommendations and safeguard against passing over the target item.

2.2 Review

After a set of recommendations have been proposed to the user, the *Comparison-Based Recommender* requires that the user reviews these recommendations and provide some form of feedback to the system. A basic form of this feedback is *Preference-Based Feedback*, where the user only has to identify the product which they prefer out of the set of recommendations. The advantage of this type of feedback is that it does not require the user to understand all of the technical features of the product and it also requires minimal effort from the user to interact with the system. Alternative forms of feedback [10] are: 1) Critiquing, which requires the user to apply a critique to a particular feature (e.g. a lower price, a different manufacturer); 2) Ratings Based, which requires that the user apply a rating describing the user's likeness for the particular product or 3) Preference Elicitation, which asks the user to supply exact feature values which that user wants.

2.3 Revise

Lines 20 to 26 of Fig 1 summarize how a typical comparison-based recommender responds and revises its understanding of a user's personal requirements (i.e., the evolving query) at the revise stage of each recommendation cycle using session-dependent feedback. The simplest form of query revision is the More-like-This approach (MLT), where each feature of the preferred case acts as a new query feature (i.e., the most recent preference case becomes the query for the next recommendation cycle).

Line 23 refers to part of the algorithm that updates the query with features from the preferred case c_p. The advantage is that a partial user query can be instantly extended to include additional features (from the preferred case) in the hope of focusing more effectively on the right set of cases during the next recommendation cycle. However, not every feature of the preferred case may be relevant to the current user; for instance, a user may prefer one holiday over others because of its low price and not because of its location. This over-fitting to user feedback can cause the search to wander away to irrelevant parts of the product space resulting in long recommendation cycles and the need to examine more cases than necessary.

Using user feedback to inform the retrieval strategy imposed is one way of addressing this problem. Another idea is to look at alternative ways of adapting the query to more accurately reflect user intention.

McGinty and Smyth [10] propose a number of alternative query revision strategies that try to address this issue. One example, the wMLT strategy, attempts to weight the new query features based on the degree of confidence that we can attribute to them as preferences for the user. The basic idea is that the more alternatives that have been presented to the user, the more confident we can be about learning their preference.

$$\text{weight}(f_i, R) = \frac{\#\ of\ alternatives\ to\ f_i \text{ in } R}{|R|} \qquad (1)$$

While different query revision strategies have been proposed and evaluated against MLT [10, 11], they all share one common characteristic. That is, they all operate using *only* user feedback provided in the context of a single recommendation cycle. Although evaluations have demonstrated that significant reductions in overall session length are achievable (i.e., potential session length reductions of up to 30% over MLT have been recorded [10]), the possibility of following *false-leads* still exists. There are two key reasons for this. First of all, these strategies do not reason about the decisions a user has made previously in the context of a given session. Query revisions are made on the basis of information gained at the cycle level. Thus, aside from the representation characteristics of systems that use these strategies, there is little experience-based reasoning in the conventional CBR sense. Secondly, all the revision strategies that have been proposed work well if the recommendation set in a given cycle is sufficiently diverse in terms of their feature alternatives.

Motivated by the problems associated with cycle-dependent query revision approaches, we have started to look at alternatives that *do* reason based on previous session-dependant decisions made on recommended cases. Specifically, we have started to explore the possibility of adapting the query in each cycle using feedback accumulated over the series of preceding recommendation cycles.

3 Cumulative Query Revision

In this section we describe *two* alternative strategies that focus on using prior experience to revise the query from cycle to cycle, based on the cumulative feedback received by a comparison-based recommender throughout the course of the given session. Both strategies discussed involve revisions to the **QueryRevise** procedure of the comparison-based recommendation algorithm (see Fig 1 and Fig 4). For clarity, the feedback policy we assume is preference-based feedback (i.e., a user simply indicates which recommendation item they prefer from a set of *k* alternatives).

```
20.   define QueryRevise(Q, c_p, R)
21.   begin
22.     R' ← R - {c_p}
23.     Q ← history(c_p, R')
24.     return Q
25.   end
```

```
26.   define history(c, R')
27.   begin
28.     Add c to list of preference cases
29.     Add R' to list of rejected cases
30.     Q ← generate query from lists
31.     return Q
32.   end
```

Fig. 2. Structure of Cumulative Query Update

As users make their way from cycle to cycle they make a preference-based decision and choose one case from a set of recommendations. These choices contain implicit frequency of occurrence information that can provide valuable input for subsequent query revisions and retrievals. Here we propose two approaches, CFO-A and CFO-B, which prioritise feature values that have been repeatedly preferred by a user.

3.2 CFO-A

As users make their way from cycle to cycle they make a preference-based decision and choose one case from a set of recommendations. As the recommendation process continues two sets of cases are built up, one

contains their preference case for each cycle (i.e., *PreferredCase(i)*), and the other contains the cases they rejected (i.e., *RejectedCases(i)*), for each cycle. Ideally, our algorithm will recognize and prioritize feature values that have been repeatedly preferred by a user. A straightforward method of attempting to extract this information is as follows: as a user makes choices throughout their session, a count is kept of how many times a particular feature value, *v*, occurs in each of their preference cases. As a user proceeds, some features will have values that occur in their preference frequently and will build up a larger count than others.

$$\text{Preferred}(i, f, v) = \begin{cases} 1 & \text{if } (f, v) \in \text{PreferredCase}(i) \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

$$\text{Rejected}(i, f, v) = \begin{cases} -1 & \text{if } (f, v) \in \text{RejectedCases}(i) \\ 0 & \text{Otherwise} \end{cases} \quad (3)$$

In addition to this, the values contained in the rejected cases are also monitored such that if a feature value occurs in one of these rejected cases, its count value is decremented. Equations 1 and 2 outline how the count for a feature value is affected if it is contained in either a preference or a rejected case. So the count of a value *v* for the feature *f*, across all cycles *1* to *n* will be as follows;

$$\text{Count}(f, v) = \sum_{i=1}^{n} \text{Preferred}(i, f, v) + \sum_{i=1}^{n} \text{Rejected}(i, f, v) \quad (4)$$

The count kept can become negative if a value is rejected more than it is chosen from cycle to cycle. Importantly, the rejection of a feature value is only counted once in a cycle, even if it occurred in all the rejected cases for that cycle. To penalize a value for being involved in 2 rejected cases for example, would mean that it would need to be in the preference case twice again before breaking even, not taking into account any additional rejected occurrences. So, when revising the query for the next recommendation cycle the feature values that have the largest count are transferred into the new query. In the event of a tie, one of the values with the joint highest count is chosen randomly, and for features where no value has a positive count that feature is left empty so it will not influence similarity calculations and retrieval.

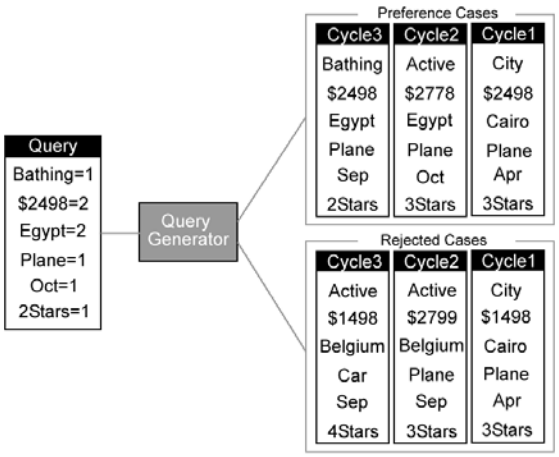


Fig. 3. Example of Cumulative Query Update

Fig 3 shows a simple example of CFO applied to a travel recommendation example. The user has already gone through 3 cycles, in each of which she was presented with 2 cases. From these she chose her preference case and the task is to revise the query so new recommendations can be made. The first feature describes the type of holiday available. From the previous preference and rejected cases “Bathing” is given a score of 1 since it is accepted once while not being rejected. In comparison the “Active” holiday type was given an overall score of -1 and “City” was given a score of 0. Bathing has the largest score and is therefore transferred into the new query. Similarly, the other feature values, “$2498”, “Egypt”, “Plane”, “Oct” and “2Stars” are also transferred.

3.3 CFO-B

Given that this piece of work is in its early stages of maturity, a slight variation of the previously described approach was also implemented for comparison information. The only difference in CFO-B is that counts are not used for numeric value features (e.g. holiday price or duration). Instead, for numeric-valued features, the value transferred to the new query is the mean value of that feature over the values that occur in the preference cases of earlier cycles. This means that, in CFO-B, all numeric-valued features will be included in a query; in CFO-A, on the other hand, if the count for all values of such a feature is negative or zero, the feature is not included in the query and therefore does not influence the next retrieval.

4 Experimental Evaluation

Having presented a straightforward and basic approach to query revision, this section looks to evaluate its performance against the benchmark MLT query revision strategy.

4.1 Setup and Methodology

To carry out the evaluation we used the well-known Travel dataset [1] which contains 1024 holiday case descriptions. In order to generate a set of test queries, each of these cases is selected to serve as a *seed* case. For each of these seed cases random queries of lengths 1, 3 and 5 were generated. A specific *target* case (i.e., the most similar to the seed) was then appointed. All queries were solved using a leave-one-out methodology. For clarity, the preference case in each recommendation cycle was deemed to be that which is most similar to the target. The experiments were conducted for varying values of k (i.e., the number of recommendations returned during each recommendation cycle) from 2 to 10 in increments of 2 with the session terminating when the target case was presented. Three revision approaches were used: the first two being CFO-A and CFO-B as outlined above and also, as a benchmark, the MLT query revision strategy.

4.2 Recommendation Efficiency Results

Previous research has highlighted that a key evaluation criteria for conversational recommenders is the recorded reduction in session length [8, 10, 11, 12, 20]. Thus, in our evaluations we measure the number of recommendation cycles a user must go through before they find their target item. Fig 6(a) shows how all three algorithms compared over a range of query difficulties (i.e., easy queries contain 5 features or more, moderate queries between 3 and 5 features, and difficult queries that only had 1 initial feature).

We find that both of our cumulative approaches achieve significant efficiency advantages over the stand-alone MLT approach. CFO-B performs the best, recording an overall improvement on MLT of 48%; the CFO-A variation is only marginally behind recording an overall improvement over MLT of 46%. Importantly, the relative benefits (over MLT) enjoyed by the CFO algorithms increase with query difficulty.

Fig 6(a) shows that CFO-B leads to session reductions of between 43% (easy queries) and 52% (difficult queries). CFO-A shows a similar pattern of results with benefits of between 41% and 50% for easy and more difficult queries respectively. The recommendation sessions associated with more difficult queries include more false-leads than the sessions for simpler queries, and so offer the CFO approaches an even greater opportunity for cycle reduction. We have found compatible results for other data-sets (e.g., whiskey, PC).

[1]This dataset is freely downloadable from URL: http://www.ai-cbr.org

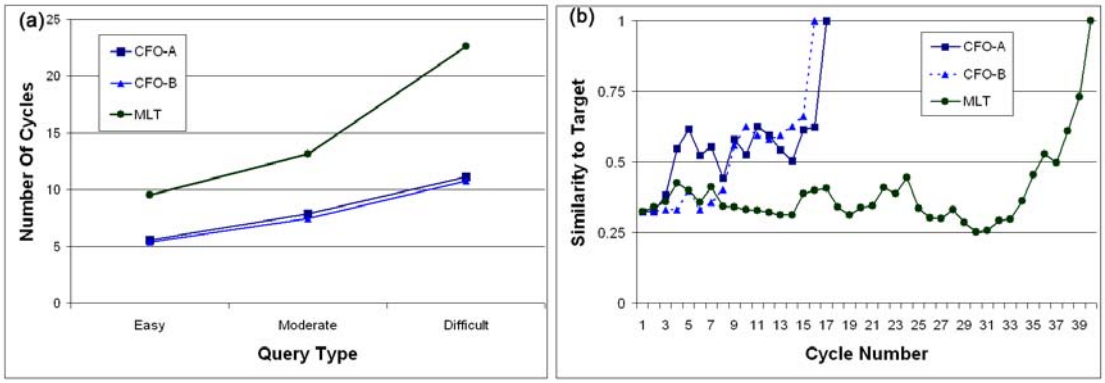


Fig. 4. Evaluation results (using the Travel data-set) looking at recommendation efficiency (a), and illustrating the typical similarity profile of a preference case to the target case in a typical recommendation session (b).

In Section 2.3 we highlighted that the MLT query revision approach tends to *over-fit* to user feedback. Figure 4(b) illustrates the severity of this problem by graphing the similarity of the preference to the target in a sample recommendation dialog. Instead of an increasing similarity profile we find sustained decreases in similarity as MLT follows *false-leads*. Between cycles 5 and 14, target similarity falls from .61 to as low as .5 as the user is forced to accept poor recommendations. In fact, MLT does not manage to present a *better* recommendation than that presented in cycle 4 until cycle 24! Indeed it is only after cycle 36 that MLT finally begins to focus on the target region.

Fig 4(b) also shows that while the cumulative revision methods do not entirely eliminate this over-fitting issue they are far less sensitive to it. For both CFO-A and CFO-B the number of sustained descents are much less than MLT, and both begin to focus in on the target after cycles 13 and 14. This is evidence that using cumulative feedback can help the recommender to better correct its focus and move away from *false-leads*.

5 Conclusions and Future Work

Recent research in the area of case-based recommendation has highlighted the importance of using user feedback to inform retrievals in session-dependent recommendation scenarios. To date, work in this area has focused on implementing query revision strategies that use *only* information collected in an individual recommendation cycle. Importantly, the preference decisions made by the user in the preceding cycles do not influence subsequent case retrievals.

In this paper we propose and evaluate two alternative query revision strategies that revise the recommender system's understanding of what the user is looking for based on the *cumulative feedback* they provide. Our first evaluation results indicate that even very basic approaches to cumulative query revision can lead to very significant reductions in terms of the number of cycles a user must engage in before they find their target.

The research ideas described by this paper simply scratch-the-surface of some of the possibilities when it comes to implementing cumulative query revision polices. We intend to continue this branch of investigation and look at the opportunities that exist to further improve on the ideas discussed in this paper. We also intend to conduct a full comparative evaluation of the wMLT and adaptive selection procedures against our proposed cumulative algorithms. Of course, there is no reason why we could not integrate our cumulative algorithms with the effective adaptive selection retrieval strategy proposed by Smyth and McGinty[2]. It certainly would be interesting to look at the effect of applying reasoning techniques in *both* the retrieval *and* revision stages of the comparison recommendation cycle. We intend to investigate the potential benefits of doing this in the near future with the intent of further demonstrating the advantages of conversational, case-based recommender systems.

6 ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland.

References

[1] D. Bridge. Product Recommendation Systems: A New Direction. In D. Aha and I. Watson, editors, *Workshop on CBR in Electronic Commerce at The International Conference on Case-Based Reasoning (ICCBR-01)*, 2001. Vancouver, Canada.

[2] D. Bridge. Towards conversational recommender systems: A dialogue grammar approach. In D. Aha, editor, *Proceedings of the Mixed-Initiative Workshop on CBR at the European Conference on Case-Based Reasoning (ECCBR-02).*, pages 8–22. The Robert Gordon University, 2002. Aberdeen, Scotland.

[3] D. Bridge. Recommender systems and case-based reasoning. In *Proceedings of the Recommender Systems Workshop at the University of Haifa.*, 2005. Israel.

[4] R. Burke. Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems*, 69(32), 2000.

[5] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modelling and User Adapted Interaction*, 12(4):331–370, November 2002.

[6] R. Burke, K. Hammond, and B. Young. The findme approach to assisted browsing. *Journal of IEEE Expert*, 12(4):32–40, 1997.

[7] L. Coyle and P. Cunningham. Improving recommendation ranking by learning personal feature weights. *Advances in Case-Based Reasoning: 7th European Conference, ECCBR04*, pages 560–572, August 2004.

[8] M. Doyle and P. Cunningham. A Dynamic Approach to Reducing Dialog in On-Line Decision Guides. In E. Blanzieri and L. Portinale, editors, *Proceedings of the Fifth European Workshop on Case-Based Reasoning, (EWCBR-00)*, pages 49–60. Springer, 2000. Trento, Italy.

[9] M. Goker and C. Thompson. Personalized conversational case-based recommendation. In E. Blanzieri and L. Portinale, editors, *Proceedings of the Fifth European Workshop on Case-based Reasoning, (EWCBR-00)*, pages 99–111. Springer-Verlag, 2000.

[10] L. McGinty and B. Smyth. Comparison-based recommendation. In S. Ed. Craw, editor, *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02)*, Aberdeen, Scotland, 2002. Springer.

[11] L. McGinty and B. Smyth. Deep dialogue vs casual conversation in recommender systems. In F. Ricci and B. Smyth, editors, *Proceedings of the Workshop on Personalization in eCommerce at the Second International Conference on Adaptive Hypermedia and Web-Based Systems (AH-02)*, pages 80–89, Malaga, Spain, 2002.

[12] D. McSherry. Minimizing dialog length in interactive case-based reasoning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI01).*, pages 993–998, Seattle, Washington, USA, August 2001.

[13] D. McSherry. Diversity-conscious retrieval. In S. Craw, editor, *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02)*, volume 2416, pages 219–233, Aberdeen, Scotland, September 2002.

[14] S. Middleton, D. D. Roure, and N. Shadbolt. Capturing knowledge of user preference: Ontologies in recommender systems. *Proceedings of the International Conferance on Knowledge Capture (KCap-10)*, ACM Press:100–107, 2001.

[15] F. Ricci and F. del Missier. Supporting travel decision making through personalized recommendation. *Designing Personalized User Experiences in eCommerce*, pages 231–251, 2004.

[16] J. B. Schafer, J. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1-2):115–153, 2001.

[17] D. Shanley and L. McGinty. Using recommendation summaries for retrieval in conversational recommenders. In H. Arabnia, editor, *Proceedings of the International Conference on Artificial Intelligence (IC-AI 2005)*. CSREA Press, 2005. Las Vegas, Nevada.

[18] H. Shimazu, A. Shibata, and K. Nihei. Expertguide: A conversational case-based reasoning tool for developing mentors in knowledge spaces. *Applied Intelligence*, 14(1):33–48, 2002.

[19] B. Smyth and P. McClave. Similarity vs. diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning, ICCBR01*, pages 347–361, London, UK, 2001. Springer-Verlag.

[20] B. Smyth and L. McGinty. The power of suggestion. In *Proceedings of IJCAI*, pages 127–132, 2003.

On the Role of Default Preferences in Compromise-Driven Retrieval

David McSherry

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

Abstract. Compromise-driven retrieval (CDR) is an approach to retrieval in recommender systems in which the products retrieved in response to a given query are guaranteed to provide full *coverage* of the available products. That is, for any product that is not recommended, one that is recommended is at least as similar and involves the same or fewer compromises. We focus in this paper on the compensatory role of similarity in the approach and show that taking account of default preferences with respect to attributes not mentioned in the user's query helps to ensure that otherwise competitive products which fail to satisfy all the user's requirements are not overlooked in the retrieval process. We also examine the role of default preferences in explaining a CDR system's recommendations.

Keywords. Recommender systems, case-based reasoning, user preferences, explanation

1 Introduction

In case-based reasoning (CBR) approaches to product recommendation, descriptions of the available products are stored in a product case base and retrieved in response to user queries. An important advantage of similarity-based retrieval in this context is that if no case exactly matches the user's requirements she can be shown the cases that are most *similar* to her query [1]. However, the most similar cases also tend to be very similar to each other, with the result that the user may be offered a limited choice [2, 3]. It is also not unusual for the recommended cases to involve *compromises* (i.e., unsatisfied requirements) that the user is not prepared to accept, which means that the system has failed to recommend an acceptable product [4].

Compromise-driven retrieval (CDR) has the important benefit of ensuring that the cases recommended in response to a given query cover all possible compromises that the user might be willing to accept [4]. Similarity and compromise play complementary roles in ensuring that for any case not included in the CDR retrieval set, one of the recommended cases is at least as similar and involves the same or fewer compromises. As demonstrated in our CDR recommender system *First Case* [4], another important benefit is that recommendations can easily be explained.

Recent research has highlighted the potential benefits of default (or assumed) preferences in CBR recommender systems, for example as a means of reducing the length of recommendation dialogues in incremental approaches to query elicitation [5-6]. Often in recommender systems it is possible to identify *less-is-better* (LIB) attributes whose values most users would prefer to minimise and *more-is-better* (MIB) attributes whose values most users would prefer to maximise [1, 4-7]. In a recommender system for digital cameras, for example, attributes such as *resolution*, *optical zoom*, *digital zoom*, and *memory* might be treated as MIB attributes and attributes such as *price* and *weight* as LIB attributes. McSherry [5] proposes an approach to the assessment of a case's similarity with respect to such attributes in which there is no need for preferred values to be elicited from the user. Instead, the preferred value of a LIB attribute is assumed to be its minimum value in the case base, while the preferred value of a MIB attribute is assumed to be its maximum value.

As we show in this paper, taking account of default preferences with respect to non-query attributes in CDR has the important benefit of helping to ensure that otherwise competitive products that fail to satisfy all the user's requirements are not overlooked in the retrieval process. We also demonstrate the use of default preferences to explain why a case is more highly recommended than another case that involves additional compromises. Finally, we investigate the effects of taking account of non-query attributes on the size of the CDR retrieval set.

In Section 2, we present a brief overview of CDR and our approach to similarity assessment with respect to default preferences in *First Case 2*, a new version of First Case. The example case base that we use to demonstrate the approach is McCarthy *et al.*'s [8] digital camera (DC) case base, which contains the descriptions of over 200 digital cameras. In Section 3, we present empirical evidence to support the

hypothesis that taking account of default preferences with respect to non-query attributes helps to ensure that otherwise competitive cases that fail to satisfy all the user's requirements are not overlooked. In Section 4, we present an empirical analysis of cognitive load in CDR and how it is affected by taking account of non-query attributes in the retrieval process. Related work is discussed in Section 5 and our conclusions are presented in Section 6.

2 Compromise-Driven Retrieval

In this section, we describe the retrieval process in First Case 2, a new version of our CDR recommender system First Case [4].

2.1 Similarity and Compromise

In First Case 2, the user's query may include *upper limits* for LIB attributes, *lower limits* for MIB attributes, and required/ideal values for other attributes in the domain. In the DC case base, for example, the user's query might be:

$$Q = \{\text{make} = \text{Sony}, \text{price} \leq 350, \text{style} = \text{compact}, \text{optical zoom} \geq 5, \text{weight} \leq 200\}$$

We refer to the number of attributes m in a given query as the *length* of the query. As we shall see, any default preferences with respect to non-query attributes are also used in the retrieval process. A *full-length* user query is one that involves all attributes in the case base.

The user's query is used in two distinct and complementary ways in CDR, namely to assess each case in terms of its *similarity*, and to assess each case in terms of any *compromises* it involves. The *compromises* associated with a given case are those attributes in the user's query with respect to which it fails to satisfy the user's requirements. A case may fail to satisfy the user's requirements in one or more of the following ways:

- Its value for a LIB attribute is more than the upper limit
- Its value for a MIB attribute is less than the lower limit
- Its value for a nominal or numeric attribute for which a required value is specified fails to match the required value

In the DC case base [8], each camera is described in terms of its *make*, *price*, *style*, *resolution*, *optical zoom*, *digital zoom*, *weight*, *storage type*, and *memory*. For the example query Q above, Case 192 in the DC case base involves compromises with respect to *make*, *price*, and *weight*:

Case 192: Hewlett-Packard, 432, compact, 5.1, 8.1, 7, 500, SC, 32

We denote this by writing:

$$\textit{compromises}(\text{Case 192}, Q) = \{\text{make, price, weight}\}$$

We say that a case C exactly matches a given query Q if:

$$\textit{compromises}(C, Q) = \phi \quad (1)$$

In CDR, two cases C_1 and C_2 are said to be *alike* if they involve the same compromises with respect to a given query Q. That is:

$$\textit{compromises}(C_1, Q) = \textit{compromises}(C_2, Q) \quad (2)$$

For any group of *alike* cases, only the most similar case in the group is included in the CDR retrieval set. The query used by First Case 2 in its assessment of similarity is a modified version Q^* of the user's query Q that includes default preferences for any LIB or MIB attributes in the domain whether or not they are mentioned in the user's query. For the example query above, the modified query is:

$$Q^* = \{\text{make} = \text{Sony}, \text{price} = 106, \text{style} = \text{compact}, \text{resolution} = 13.7, \text{optical zoom} = 10, \text{digital zoom} = 8, \text{weight} = 100, \text{memory} = 64\}$$

An important point to note is that for a LIB or MIB attribute, an upper or lower limit specified by the user plays no role in the assessment of a case's similarity. Instead, the preferred value of a LIB attribute is assumed to be the minimum value in the case base. Similarly, the preferred value of a MIB attribute is assumed to be its maximum value in the case base. For example, 106 euro is the minimum *price* in the

DC case base, while 64 Mb is the maximum *memory*. In the case of an attribute for which a default preference is not defined (e.g., *make* or *style*), a value specified by the user, if any, is treated both as a *required* value in the assessment of compromise and as an *ideal* value in the assessment of similarity. A non-query attribute (e.g., *storage type*) for which a default preference is not defined is not included in the modified query.

To assess the similarity (or utility) of a given case with respect to LIB or MIB attributes in the modified query Q^*, we use the measure commonly used for numeric attributes with the attribute's minimum or maximum value as the preferred value. For example, we define the similarity of a given case C with respect to a LIB attribute a to be:

$$sim_a(C, Q^*) = 1 - \frac{|\pi_a(C) - \min(a)|}{\max(a) - \min(a)} = \frac{\max(a) - \pi_a(C)}{\max(a) - \min(a)} \quad (3)$$

where $\pi_a(C)$ is the value of a in C, and max(a) and min(a) are the maximum and minimum values of a in the case base.

2.2 Constructing the CDR Retrieval Set

The first step in the construction of the CDR retrieval set is to rank all cases in the case base in order of decreasing similarity to the modified user query Q^*. Among cases that are *equally* similar to Q^*, any case that involves a subset of the compromises associated with another case is given priority in the ranking process. The algorithm used to select cases to be included in the CDR retrieval set from the resulting list of candidate cases is outlined in Fig. 1. An important point to note is that only compromises with respect to the user's original query Q are considered in the selection process.

```
algorithm CDR(Q, Candidates)
begin
  RetrievalSet ← ϕ
  while |Candidates| > 0 do
  begin
    C1 ← first(Candidates)
    RetrievalSet ← RetrievalSet ∪ {C1}
    Deletions ← {C1}
    for all C2 ∈ rest(Candidates) do
    begin
      if compromises(C1, Q) ⊆ compromises(C2, Q)
      then Deletions ← Deletions ∪ {C2}
    end
    Candidates ← Candidates - Deletions
  end
  return RetrievalSet
end
```

Fig. 1. Algorithm used to construct the retrieval set in compromise-driven retrieval

First, the most similar case is placed in the retrieval set and any cases that include (or equal) the compromises associated with this case are eliminated from the list of candidate cases. The most similar of the remaining cases is now added to the retrieval set and any cases that include (or equal) the compromises associated with that case are eliminated. This process continues until no further cases remain. A detail not shown in Fig. 1 is that for each case that is added to the CDR retrieval set, any cases that involve the same compromises are placed in a separate *reference set* where they kept available for inspection at the user's request [4].

2.3 Recommendation in First Case 2

Fig. 2 shows an example recommendation dialogue in First Case 2 based on the DC case base. Case attributes and weights assigned to them in our experiments are *make* (9), *price* (8), *format* (7), *resolution* (6), *optical zoom* (5), *digital zoom* (1), *weight* (4), *storage type* (2), and *memory* (3). The overall similarity of each recommended case is shown in brackets. In the example dialogue, the user is seeking a compact Sony camera, with at least 5× optical zoom and weighing at most 200 gm, and is prepared to pay up to 350 euro. She is informed that there is no case that exactly matches her query and shown five cases, in order of decreasing similarity to her query, that cover all possible compromises she might be prepared to consider. For example, if not prepared to compromise on *price*, *style*, or *optical zoom*, her best option could be Case 194.

First Case 2: Please enter your query

User: make = sony, price ≤ 350, style = compact, optical zoom ≥ 5, weight ≤ 200

First Case 2: *There is no case that exactly matches your query, but the following cases are recommended*:

Case 29: Sony, 336, compact, 5, 3, 4, 236, MS, 32 (0.76)
Explanation: *This is one of 4 cases that differ from your query in optical zoom and weight. It is better than Case 26 in terms of resolution, optical zoom, digital zoom, and memory.*

Case 26: Sony, 202, compact, 3.9, 0, 3.6, 180, MS, 16 (0.7)
Explanation: *This case differs from your query only in optical zoom*

Case 192: Hewlett-Packard, 432, compact, 5.1, 8.1, 7, 500, SC, 32 (0.6)
Explanation: *This is one of 4 cases that differ from your query in make, price, and weight. It is better than Case 194 in terms of resolution.*

Case 194: Hewlett-Packard, 337, compact, 3.89, 8.1, 7, 489, SC, 32 (0.59)
Explanation: *This case differs from your query only in make and weight*

Case 33: Sony, 875, compact swivel, 8, 7.1, 2, 906, MS, 0 (0.58)
Explanation: *This is one of 2 cases that differ from your query in style, weight, and price*

User: **like** 29

First Case 2: *The following cases also differ from your query only in optical zoom and weight*:
Case 43: Sony, 304, compact, 5, 3, 4, 259, MS, 16 (0.74)
Case 25: Sony, 283, compact, 3.9, 3, 3.6, 243, MS, 16 (0.73)
Case 37: Sony, 237, compact, 3.14, 3, 3.2, 259, MS, 16 (0.72)

Fig. 2. Example recommendation dialogue in First Case 2

As described in Section 2.1, First Case 2 takes account of any default preferences with respect to query or non-query attributes in the retrieval process. It also differs from First Case [4] in that explanations are presented at the system's initiative rather than at the user's request. As shown in the example dialogue, First Case 2 explains why each case is recommended in terms of the compromises it involves. For example, Case 26 is recommended because it differs from the user's query only in *optical zoom*. First Case 2 also explains why Case 29, which involves an additional compromise with respect to *weight*, is more strongly recommended than Case 26 by highlighting the relative benefits that it offers in terms of *resolution*, *optical zoom*, *digital zoom*, and *memory*.

3 The Compensatory Role of Similarity in CDR

The CDR retrieval set cannot fail to include a case that exactly matches the user's query if one or more such cases exist. But just as a human salesperson's strongest recommendation may be for a product that departs slightly from the customer's requirements, an exactly matching case may not be the one that is most strongly recommended in CDR. At least for longer queries, it is not unusual for a case that involves compromises to be more similar than one that does not involve compromises.

It is also not unusual for a single recommended case to provide complete coverage of the available cases because it exactly matches the user's query and is also the most similar case. A potential problem with shorter queries is that while there is more likely to be an exactly matching case, the ability of other cases to *compensate* for their failure to satisfy all the user's requirements may be limited by the number of attributes on which the assessment of their similarity is based. Certainly their ability to compensate will be limited if no account is taken of non-query attributes in the retrieval process. In Section 2, we described how the retrieval process in CDR now takes account of any default preferences with respect to non-query attributes.

Fig. 3 shows how First Case 2 can take account of non-query attributes in recommending a case that would otherwise be eliminated because it narrowly fails to satisfy one of the user's requirements. In the example dialogue, the user is seeking a *Pentax* camera with a *resolution* of at least 4 mega-pixels, and an *optical zoom* of at least 5×. In response to her query, a case that narrowly fails to match on *optical zoom* (Case 186) is more strongly recommended than the only case that exactly matches the user's query (Case 185). By way of explanation, First Case 2 highlights the relative benefits that Case 186 offers in terms of its *price*, *digital zoom*, *weight*, and *memory*. Although none of these attributes is mentioned in the user's query, they all contribute to the overall assessment of similarity.

Particularly in view of its much lower price, Case 186 may be of more interest to the user than Case 185 unless she is seeking a camera with a very high specification, which seems unlikely given her modest requirements with respect to *resolution*. It is worth noting that Case 185 would be the *only* case recommended by First Case 2 if no account were taken of non-query attributes in the retrieval process.

First Case 2: Please enter your query:

User: make = pentax, resolution ≥ 4, optical zoom ≥ 5

First Case 2: The following cases are recommended:

Case 186: pentax, 447, compact, 5, 4.9, 4, 250, SC, 16 (0.72)

Explanation: *This is one of 5 cases that differ from your query in optical zoom. It is better than Case 185 in terms of price, digital zoom, weight, and memory.*

Case 185: pentax, 1389, SLR, 6.1, 10, 0, 650, CF, 0 (0.71)

Explanation: *This case is recommended because it matches your query exactly.*

Fig. 3. Explaining why a case involving compromises is more highly recommended than an exactly matching case

We now present empirical evidence to support the hypothesis that taking account of default preferences with respect to non-query attributes in CDR may help to ensure that otherwise competitive cases that narrowly fail to satisfy one or more of the user's requirements are brought to the user's attention when there is another case that exactly matches the user's query. If this hypothesis is correct, then we would expect to see an increase in the number of occasions when there is an exact match for the user's query and yet the case that is most highly recommended, because of its higher overall similarity, involves one or more compromises.

Our experiment thus examines the effects of taking account of non-query attributes on the number of exactly matched queries on the DC case base for which the most similar case involves compromises. Using *leave-one-out* cross validation, we temporarily remove each of the 210 cases in the DC case base

and generate all possible queries of length $m = 3$, $m = 6$, and $m = 9$ from the description of the left-out case. That is, we use each left-out case to create ${}^{9}C_{3} = 84$ queries of length $m = 3$, ${}^{9}C_{6} = 84$ queries of length $m = 6$, and one query of length $m = 9$.

For a LIB attribute like *price*, the value in the left-out case is treated as a required upper limit in each of the simulated queries in which it appears (e.g., *price* ≤ 375). Similarly, the case value for a MIB attribute like *optical zoom* is treated as a required lower limit (e.g., *optical zoom* ≥ 4). We present each simulated query to First Case 2, observe whether or not there is an exactly matching case and, if so, whether or not the most similar case involves any compromises. We also record the number of cases recommended for each query. We apply the same procedure to each of the 210 left-out cases. Finally, we repeat the whole process but now with no account being taken of non-query attributes by First Case 2 in its overall assessment of a case's similarity.

Table 1 summarises the results of the experiment in terms of the percentages of exactly matched queries for which the most similar case involves compromises. Taking account of default preferences (DPs) with respect to non-query attributes can be seen to have greater impact for short queries ($m = 3$) than for longer queries ($m = 6$), with increases of 20% and 10% respectively in the percentages of exactly matched queries for which the most similar case involves compromises. For short queries in particular, these findings represent a substantial increase in the number of occasions when an exactly matching case is recommended but the user's attention is also drawn to an otherwise competitive case that fails to satisfy all her requirements.

Table 1. Exactly matched queries on the DC case base for which the most similar case involves compromises

	Short Queries ($m = 3$)	Longer Queries ($m = 6$)
Without DPs for non-query attributes:	39%	62%
With DPs for non-query attributes:	59%	72%

A detail not shown in Table 1 is that the most similar case involves compromises for 63% of full-length queries ($m = 9$) for which there is an exactly matching case. It is also worth noting that the percentage of queries for which there is an exactly matching case decreases from 91% for short queries ($m = 3$) to 52% for longer queries ($m = 6$) and again to 19% for full-length queries ($m = 9$).

4 Cognitive Load in CDR

Balancing the trade-off between user satisfaction and cognitive load is an important and challenging problem in recommender systems [9-11]. While the size of the CDR retrieval set cannot be determined in advance, the approach has been shown to offer a good compromise between the strength of the *preference criteria* on which retrieval is based and the average size of the retrieval set [11]. However, an expected trade-off associated with taking account of non-query attributes in the retrieval process is an overall increase in the average size of the CDR retrieval set. In this section, we examine the extent of this trade-off as well as the cognitive load associated with *full-length* queries on the DC case base. The experiment on which our analysis is based is described in Section 3.

Table 2 shows the average size of CDR retrieval sets for queries of length $m = 3$ and $m = 6$ on the DC case base with and without account being taken of any default preferences with respect to non-query attributes. A striking feature of the results is the small size of the CDR retrieval set in all four experimental categories. Even when account is taken of default preferences with respect to non-query attributes, only 2 cases are required on average to provide full coverage of the available cases for short queries ($m = 3$) and less than 3.5 cases for longer queries ($m = 6$). The results also suggest that taking account of non-query attributes has only a minor effect on cognitive load.

Table 2. Average size of CDR retrieval sets for queries on the DC case base with and without account being taken of any default preferences (DPs) with respect to non-query attributes

	Short Queries (m = 3)	Longer Queries (m = 6)
Without DPs for non-query attributes:	1.5	3.0
With DPs for non-query attributes:	1.9	3.3

Maximum, average, and minimum lengths of the CDR retrieval set for short queries (m = 3), longer queries (m = 6) and full-length queries (m = 9) on the DC case base, with account being taken of any non-query attributes for which default preferences are defined, are shown in Fig. 4. The results show clearly the tendency for retrieval-set size to increase as query length increases, with an average retrieval-set size of 5.6 for full-length queries.

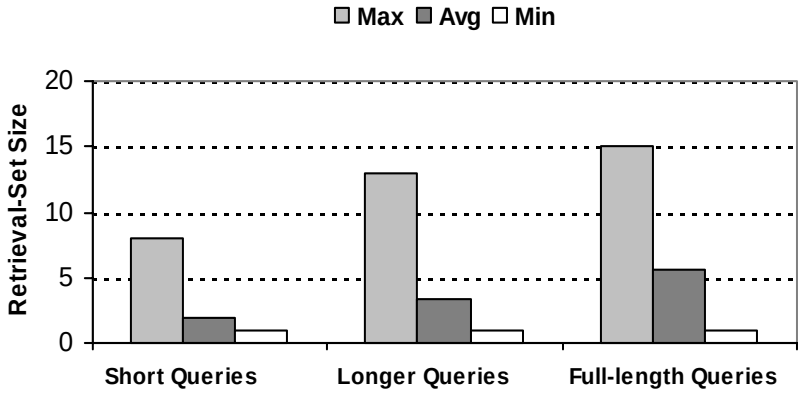


Fig. 4. Retrieval-set sizes in CDR for queries of increasing length on the DC case base

5 Related Work

In any recommender system, more than a single recommendation cycle may be needed before a product that is acceptable to the user is retrieved [4-5, 8, 10, 12-16]. Approaches to extending recommendation dialogues beyond an initially unsuccessful recommendation include *critiquing* approaches to the elicitation of user feedback [8, 10, 12-14], *referral* of the user's query to other recommender agents [15], and the *recommendation engineering* technique of providing the user with access to other cases that involve the same compromises as a recommended case [4, 16]. The latter technique is used in CDR to address the needs of users who are not just seeking a single item, but would like to be informed of all items (e.g., jobs or rental apartments) that are likely to be of interest.

The importance of recommender systems having the ability to explain their recommendations is well recognised [17-19]. Recent work by Chen and Pu [17] focuses on the potential role of explanation as a means of building user *trust* in recommender agents. Their *organisation-based* explanation technique bears some resemblance to CDR in that recommendations are grouped according to the trade-offs they involve. In contrast to CDR, however, the notion of *coverage* does not appear to be addressed in their approach to the organisation of recommended products.

6 Conclusions

As we have demonstrated in First Case 2, default preferences play an important role in a CDR recommender system's ability to explain the benefits of a recommended case relative to another case that

is less strongly recommended. We have also investigated the use of default preferences with respect to attributes not mentioned in the user's query to guide the retrieval process in CDR. As shown by our results on the DC case base, an important benefit is that highly competitive cases which might otherwise be overlooked can more easily *compensate* for their failure to satisfy all the user's requirements.

Our results on the DC case base also show that taking account of non-query attributes for which default preferences are defined only slightly increases the average size of the CDR retrieval set. Even when account is taken of default preferences for non-query attributes, only 2 cases are required on average to provide full coverage of the available cases for short queries ($m = 3$) and less than 3.5 cases for longer queries ($m = 6$). Cognitive load is also reasonably low for full-length queries ($m = 9$) on the DC case base, with an average retrieval-set size of less than 6 for such queries.

References

1. Wilke, W., Lenz, M., Wess, S.: Intelligent Sales Support with CBR. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.): Case-Based Reasoning Technology. Springer-Verlag, Berlin Heidelberg New York (1998) 91-113
2. McSherry, D.: Diversity-Conscious Retrieval. In: Craw, S., Preece, A. (eds.): Advances in Case-Based Reasoning. LNAI, Vol. 2416. Springer-Verlag, Berlin Heidelberg New York (2002) 219-233
3. Smyth, B., McClave, P.: Similarity vs. Diversity. In: Aha, D.W., Watson, I. (eds.): Case-Based Reasoning Research and Development. LNAI, Vol. 2080. Springer-Verlag, Berlin Heidelberg New York (2001) 347-361
4. McSherry, D.: Similarity and Compromise. In: Ashley, K.D., Bridge, D.G. (eds.): Case-Based Reasoning Research and Development. LNAI, Vol. 2689. Springer-Verlag, Berlin Heidelberg New York (2003) 291-305
5. McSherry, D.: Incremental Nearest Neighbour with Default Preferences. Proceedings of the Sixteenth Irish Conference on Artificial Intelligence and Cognitive Science (2005) 9-18
6. McSherry, D., Stretch, C.: Recommendation Knowledge Discovery. In: Bramer, M., Coenen, F., Allen, T. (eds.): Research and Development in Intelligent Systems XXII Springer-Verlag, London (2005)
7. Bergmann, R., Breen, S., Göker, M., Manago, M., Wess, S.: Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology. Springer-Verlag, Berlin Heidelberg New York (1999)
8. McCarthy, K., Reilly, J., McGinty, L., Smyth, B.: Experiments in Dynamic Critiquing. Proceedings of the International Conference on Intelligent User Interfaces (2005) 175-182
9. Branting, L.K..: Acquiring Customer Preferences from Return-Set Selections. In: Aha, D.W., Watson, I. (eds.): Case-Based Reasoning Research and Development. LNAI, Vol. 2080. Springer-Verlag, Berlin Heidelberg New York (2001) 59-73
10. McCarthy, K., McGinty, L., Smyth, B.: Dynamic Critiquing: An Analysis of Cognitive Load. Proceedings of the Sixteenth Irish Conference on Artificial Intelligence and Cognitive Science (2005) 19-28
11. McSherry, D.: Balancing User Satisfaction and Cognitive Load in Coverage-Optimised Retrieval. Knowledge-Based Systems, **17** (2004) 113-119
12. Bridge, D., Ferguson, A.: An Expressive Query Language for Product Recommender Systems. Artificial Intelligence Review, **18** (2002) 269-307
13. Burke, R.: Interactive Critiquing for Catalog Navigation in E-Commerce. Artificial Intelligence Review, **18** (2002) 245-267
14. Linden, G., Hanks, S., Lesh, N.: Interactive Assessment of User Preference Models: The Automated Travel Assistant. Proceedings of the Sixth International Conference on User Modeling (1997) 67-78
15. McSherry, D.: Conversational CBR in Multi-Agent Recommendation. IJCAI-05 Workshop on Multi-Agent Information Retrieval and Recommender Systems (2005) 331-345
16. McSherry, D.: Recommendation Engineering. Proceedings of the Fifteenth European Conference on Artificial Intelligence. IOS Press, Amsterdam (2002) 86-90
17. Chen., L., Pu, P.: Trust Building in Recommender Agents. In: Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces, Second International Conference on E-Business and Telecommunication Networks (2005)
18. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining Collaborative Filtering Recommendations. Proceedings of the ACM Conference on Computer Supported Cooperative Work (2000) 241-250
19. McSherry, D.: Explanation in Recommender Systems. Artificial Intelligence Review, **24** (2005) 179-197

Using Early-Stopping to Avoid Overfitting in Wrapper-Based Feature Selection Employing Stochastic Search

John Loughrey and Pádraig Cunningham

Trinity College Dublin, College Green, Dublin 2, Ireland
{John.Loughrey, Padraig.Cunningham}@cs.tcd.ie

Abstract. Feature selection is often an important pre-processing stage in the development of Case Based Reasoning systems. The wrapper approach is the dominant technique used here but it is acknowledged overfitting can be a problem when there is a limited amount of training data available. It has also been shown that the severity of overfitting is related to the intensity of the search algorithm used during this process. In this paper we show that two stochastic search techniques (Simulated
Annealing and Genetic Algorithms) that can be used for wrapper-based feature selection are susceptible to overfitting in this way. However, because of their stochastic nature, these algorithms can be stopped early to prevent overfitting. We present a framework that implements early-stopping for both of these stochastic search techniques and we show that this is successful in reducing the effects of overfitting and in increasing generalisation accuracy in most cases.

Keywords. Wrapper-based Feature Selection, Overfitting.

1 Introduction

Case Based Reasoning (CBR) normally uses nearest neighbour techniques for case retrieval. Since these techniques are particularly sensitive to irrelevant or noisy features, feature selection is an important pre-processing step in CBR. The benefits of wrapper-based techniques for feature selection are well established [1]. However, it has recently been recognised that wrapper-based techniques have the potential to overfit the training data [2]. That is, feature subsets that perform well on the training data may not perform as well on data not used in the training process. Furthermore, the extent of the overfitting is related to the depth of the search. Reunanen [2] shows that, whereas Sequential Forward Floating Selection (SFFS) beats Sequential Forward Selection (SFS) on the data used in the training process, the reverse is true on hold-out data. He argues that this is because SFFS is a more intensive search process i.e. it explores more states.

In this paper we show that this tendency to overfit can be quite acute in stochastic search algorithms such as Genetic Algorithms (GA) and Simulated Annealing (SA) as these algorithms are able to intensively explore the search space. We show that early-stopping is an effective strategy for preventing overfitting in feature selection using SA or GA. It is worth noting that the applicability of early-stopping depends on the stochastic nature of the search. This idea would not be readily applicable for directed search strategies such as the SFFS and SFS strategies evaluated by Reunanen or the standard Backward Elimination strategy that is popular in wrapper-based feature selection.

In [3] we approach this problem using a modified genetic algorithm (GA) that stops the search early in order to avoid overfitting, and we find that the results we get are favourable. In this paper we show that SA is amenable to a neat form of early-stopping. Optimisation using SA is analogous to the cooling of metals and we show how the SA can be *quenched* so that the search *freeze* before overfitting can occur. In section 3.2 we show how SA can be speeded up to avoid overfitting and in section 3.3 we show how to *calibrate* this process using cross-validation.

The paper is organised as follows. We begin in section 2 with a discussion of the wrapper-based approach to feature selection and an illustration of the potential overfitting problem. The early-stopping solution to overfitting is described in section 3. The approach is evaluated on SA and GA in section 4 and the paper concludes with some suggestions for future work in section 5.

2 Feature Selection

Feature selection is defined as the selection of a subset of features to describe a phenomenon from a larger set that may contain irrelevant or redundant features. Feature selection attempts to identify and eliminate unnecessary features, thereby reducing the dimensionality of the data and reducing the model variance [4]. Improving classifier performance and accuracy are usually the motivating factors. The accuracy of Nearest Neighbour classifiers (*k*-NN) is particularly degraded by the presence of these irrelevant features. The evaluations presented in this paper are on *k*-NN classifiers.

The two alternative approaches to feature selection are the use of filters and the wrapper-based method. Filtering techniques attempt to identify features that are related to or predictive of the outcome of interest and operate independently of the learning algorithm. An example is Information Gain, which was originally introduced to Machine Learning research by Quinlan as a criterion for building concise decision trees [5] but it is now widely used for feature selection in general.

The wrapper approach differs in that it evaluates subsets based upon the accuracy estimates provided by a classifier built with that feature subset. Thus wrappers are much more computationally expensive than filters but can produce better results because they take the *bias* of the classifier into account and evaluate features in context. The wrapper search then uses some heuristic such as Backward Elimination or Forward Selection to traverse the feature subset space in search of an optimal subset. The big issue with the wrapper approach is the computational cost since the search is directed by an assessment of the accuracy attributable to the feature mask (feature subset). This assessment needs to be as accurate as possible so it is common to use cross-validation as is described in section 3.3.3

2.1 Overfitting in Wrapper-Based Feature Selection

In the original publications on wrapper-based feature selection [15] mentioned the problem of overfitting but illustrated that it was not a problem on the datasets they examined. As with all machine learning algorithms this is true if the data available adequately covers the phenomenon. The problem is that sample size is often limited in many real world applications, especially in medical and financial applications, in these situations overfitting in wrapper-based feature selection is a real problem.

Overfitting in feature selection appears to be exacerbated by the intensity of the search since the more feature subsets that are examined the more likely the search is to find a subset that overfits. In [1] [2] this problem is described, although little is said on how it can be addressed. However, we believe that limiting the extent of search will help combat overfitting. [14] describes the feature weighting algorithm DIET, in which the set of possible feature weights can be restricted. Their experiments show that when DIET is restricted to two non-zero weights the resultant models perform better than when the algorithm allows for a larger set of feature weights, in situations when the training data is limited. This restriction on the possible set of values in turn restricts the extent to which the algorithm can search, and therefore constrains the representational power of the model.

There are many examples documented where constraining the representational power of an algorithm can lead to an increase in performance; the addition of noise to the case base during training restricts the models representational power on the underlying data [6], while limiting the number of hidden units in a neural network will have a similar effect. However, in feature selection we only have two possible weights, a feature can only have a value of `1' or `0' i.e. be turned `on' or `off', so we cannot restrict this aspect any further. In a stochastic search we can constrain the intensity of the search through early-stopping.

In Figure 1 we illustrate the idea that the more feature subsets examined in the search the more likely the search is to overfit to the training data. The right hand axis shows the number of nodes visited during the search, while the left hand axis shows the accuracies obtained. The best generalisation accuracy is achieved by the Backward Elimination (be) search while the GA search overfits more and more as the number of generations increases from 100 to 200 to 400. Forward Selection (fs) is also economic in the number of nodes it searches but still manages to overfit the data. The relatively poor performance of Forward Selection compared to Backward Elimination has been documented previously [7].

2.2 Overfitting and the Bias-Variance Decomposition of Error

The bias-variance decomposition [8] makes available the components of the error measure and this enables us to see if the errors we are getting are due to model variance or due to model bias. Errors due to overfitting should show up as model variance error. Thus measures to combat overfitting should reduce the variance component of error if effective. A high bias suggests that the learning method

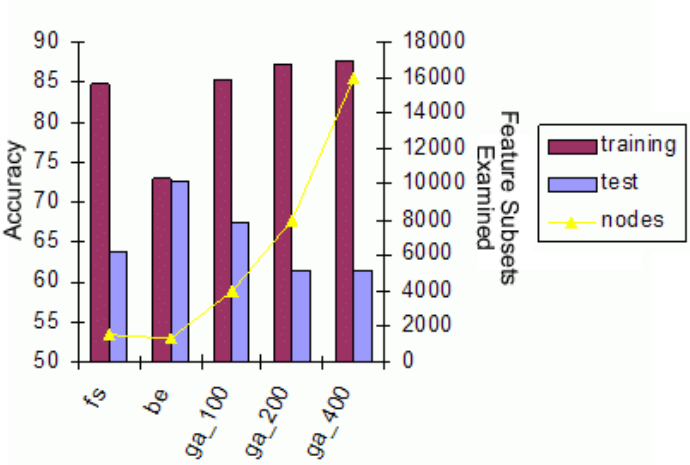


Fig. 1. The Figure shows the effect of the depth/intensity of the wrapper search on the 'spectf' data set, where the generalization accuracy is reduced as more nodes are evaluated.

is not correct for the domain, and represents the shortcomings of the learning method in modelling the data and is generally not related to a lack of data or overfitting. Successful feature selection should result in a reduction in model variance, but we expect that this measure will increase once again when we overfit during the wrapper search.

3 Early-Stopping in Stochastic Search

The motivation behind early-stopping is fairly straightforward - stop the search t the point that overfitting starts to happen. This is achieved by using a cross-validation analysis on the training data to determine when early-stopping starts to occur. Then a model is built with all the training data and the search is stopped at the appropriate point. While the idea is straightforward, it is awkward to evaluate the effectiveness of the process. This requires two nested levels of cross-validation (see section 3), an outer level to assess generalisation accuracy and an inner level to determine the early-stopping point.

As was emphasised in the Introduction, this early-stopping strategy is only meaningful for wrapper-based feature selection where the search strategy is stochastic. It would not be sensible to stop a Forward Selection or Backward Elimination strategy as it would simply exclude some features from consideration.

However it does make sense to stop a GA or SA earlier on in the search process.

3.1 Genetic Algorithms

Genetic Algorithms have been inspired by the biological process of evolution and attempt to capture the concept of the `survival of the fittest'. In the GA search we maintain a fixed population of possible solutions and these individuals evolve as the search progresses in an attempt to find an optimal solution. Evolutionary strategies such as crossover and mutation are used to maintain quality and diversity of the population. A GA is an attractive search policy for wrapper-based feature selection as crossover and mutation are straightforward. Each solution in the population is represented as a feature mask and crossover is achieved by splicing two masks. Mutation simply involves flipping bits on or off in the masks. The results reported here use a basic GA algorithm that uses Roulette Wheel

selection based on the cross-validation accuracy of the feature masks. In fact the log of the accuracy is used to slow down the convergence. We used a two-point crossover and a mutation rate of 0.05. The population size for each of the data sets was fixed to 40, and it was allowed to search for 120 generations. After the inner cross-validation layer of the framework we estimate at which generation overfitting was likely to begin.

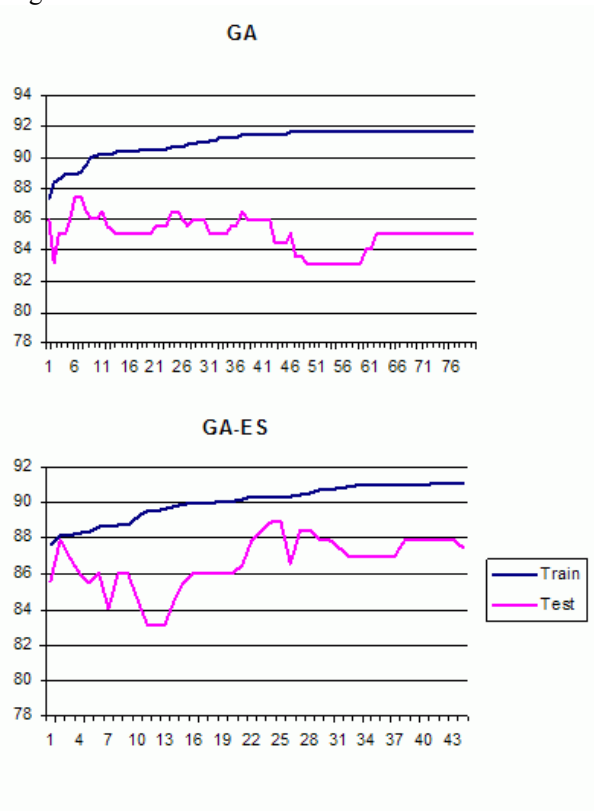


Fig. 2. Analysis of the GA and GA-ES search progression

The GA-ES is modified in that we stop the search at an earlier generation, the generation at which overfitting occurred, as shown in Figure 2. We have already shown that this is a reasonably effective strategy for combatting overfitting in GAs [3].

3.2 Simulated Annealing

[13] have shown that the models that describe the annealing of metals can be used to guide stochastic search. Simulated Annealing is similar to hill climbing search in that there is only one solution at a time under consideration. This solution is perturbed and the new solution is kept if it represents an improvement. The special feature of SA is that the new solution may still be kept even if it is poorer than the existing one. The probability of this is:

$$P(AcceptNewMask) \; \alpha \; e^{\Delta L/T} \qquad (1)$$

In feature selection, ΔL would refer to the difference in accuracy between the old and new feature masks and T would be an artificial variable describing the `temperature' of the system. The effect of this policy is that large deteriorations in accuracy are less likely to be accepted and any deterioration is less likely to be accepted as the temperature drops. The core of an SA algorithm for feature subset selection is described in Figure 3. Initially, the system starts off at a high temperature and the search is allowed to proceed in a fairly random manner. The system cools in stages with the search staying at a given temperature until a number of perturbations have been explored or a number of successes have been achieved. Thus the rate of progress of the SA is determined by the cooling rate (0.9 in this example) and the factor K that determines how long is spent at each temperature level. For

instance if K is halved the cooling will proceed twice as quickly. In terms of the original inspiration for SA, this might be described as *quenching* the system. So our early-stopping policy for SA still allows the system to *freeze*, it just spends less time at each temperature level.

```
T = T x 0.9 /* Reduce the temperature */
NTries = 0; NSucc = 0;
while(NTries < TryLim x 10 x K) and (NSucc < SuccLim x K)
      M' = PerturbMask (M)
      if AcceptNewMask? (M',M)
            NSucc = NSucc + 1
            M = M'
      endif
      NTries = NTries + 1
endwhile
```

Fig. 3. The core of the SA algorithm

Figure 4 describes the idea behind SA-ES. In the normal run, we identify the number of nodes that we evaluate in the iterations before overfitting starts to occur, then in the modified algorithm we stretch these attempts out so that they cover the entire cooling phase of the modified search. This is achieved by reducing K by the appropriate proportion. In the example in Figure 4: $K \leftarrow K$ x 20/65.

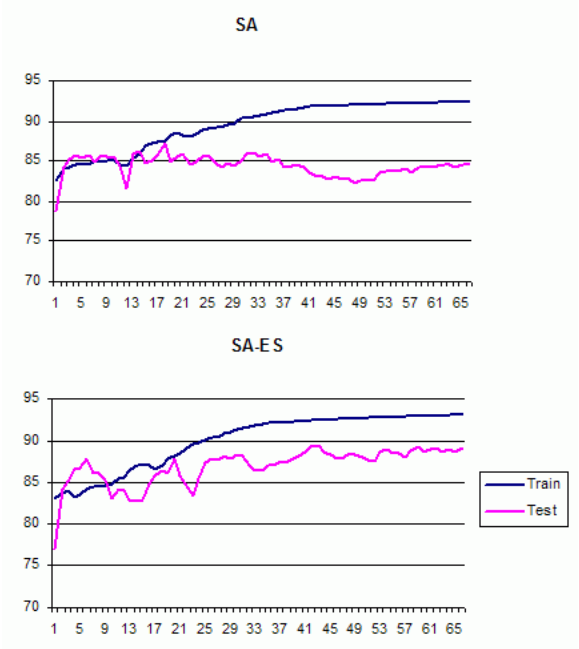


Fig. 4. Analysis of the SA and SA-ES search progression

3.3 Stochastic Search with Early-Stopping - (SS-ES)

So the basic principle for both SA and GA's is to modify the search algorithm so that it will reduce its intensity/depth of search depending on when overfitting was judged likely to occur. Figure 5 shows the SS-ES Framework in which we evaluate the overfitting in the Wrapper-based subset selection process - this follows the principles outlined by [12]. In each *fold* of the outer cross-validation, the original data source is divided into two in a 90:10 split. This 10% will be the outer test set that will be used to evaluate the generalisation accuracy of the resulting feature set. The 90% goes into our inner cross-validation which attempts to identify at which stage overfitting occurs in the wrapper search. The inner cross-validation divides the data again into a 90:10 split. 90% of this data is used to build a classifier and the 10% is used to estimate the validation accuracy. This is repeated 10 times. Therefore in the inner cross-validation we have 81% of the original data to train the classifier in each fold and 9% for estimating the validation accuracy of that classifier (repeated over 10 folds).

4 Evaluation

As [1] point out, overfitting is often not a problem in wrapper-based feature subset selection, thus it will not show up in many feature selection tasks. In the evaluation we present here we work with six datasets from the UCI collection [9]

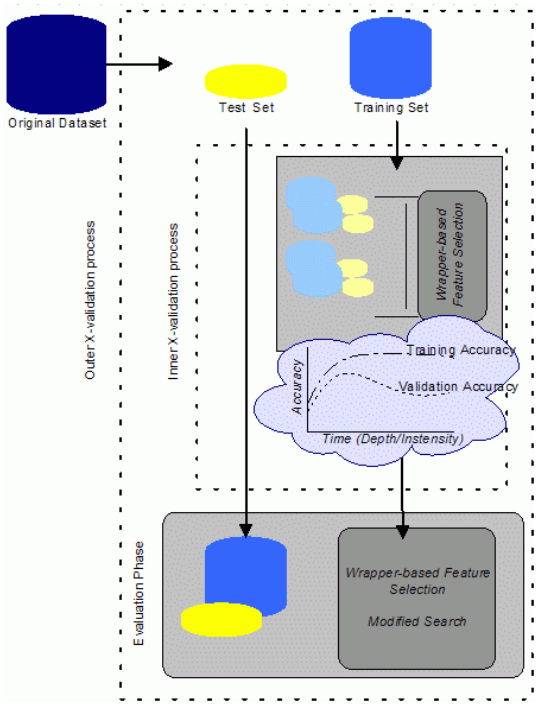


Fig. 5. SS-ES Framework

and two other datasets; the Colon dataset described in [10] and the Bronchiolitis dataset described in [11]. These are data-sets that proved to exhibit overfitting in our preliminary analysis.

The results on the GA-ES are shown in Table 1 and the results on the SA-ES are shown in Table 2. The GA-ES results are not very encouraging with the GA-ES winning in 4, loosing in 3 and drawing in one. This is due to difficulties with automatically identifying the early-stopping point in the cross-validation process. This could be improved by making this a manual (interactive) process; however, the SA-ES strategy shows more promise. The SA-ES improves on the simple SA in 7 of the 8 datasets. This is probably due to the fact that the SA-ES strategy is more robust than the GA-ES strategy. The practice of quenching the cooling process more rapidly so that the SA freezes before overfitting is more robust than the GA-ES strategy. We attribute this to the manner in which SA runs, where it is able to complete its search cycle whereas the GA terminates early. We believe that this makes the GA subject to increased variability because it is being stopped before convergence.

	GA		GA-ES	
	Train	Test	Train	Test
wpbc	83.33	**77.39**	82.3	75.76
spectf	81.25	72.5	73.06	72.5
sonar	91.78	86.55	91.72	**90.36**
ionosphere	94.21	**90.59**	93.89	89.44
glass	80.23	74.19	80.32	**76.71**
diabetes	74.96	70.17	74.81	**71.49**
bronc	67.16	**59.89**	64.07	58.86
colon	93.67	80.08	92.36	**84.21**

Table 1. Comparison of results across eight data sets using GA and GA-ES

	SA		SA-ES	
	Train	Test	Train	Test
wpbc	81.5	73.29	81.1	**73.82**
spectf	83.75	65	80	**73.75**
sonar	92.52	85.07	93.27	**88.43**
ionosphere	94.43	**92.86**	94.14	92
Glass	80.11	73.79	80.06	**76.6**
Diabetes	74.44	69.01	73.71	**72.4**
Bronc	80.32	56.62	77.38	**59.04**
Colon	93.79	83.17	87.88	**83.65**

Table 2. Comparison of results across eight data sets using SA and SA-ES

When perform multiple runs of the feature selection process using the SA and SA-ES strategies and decompose the error on the resulting feature sets into their bias-variance components we have a better understanding of what effect the early-stopping has. The results of this are shown in Figure 6 and it is clear that SA-ES generally reduces the variance component of error. This supports the hypothesis that SA-ES reduces overfitting.

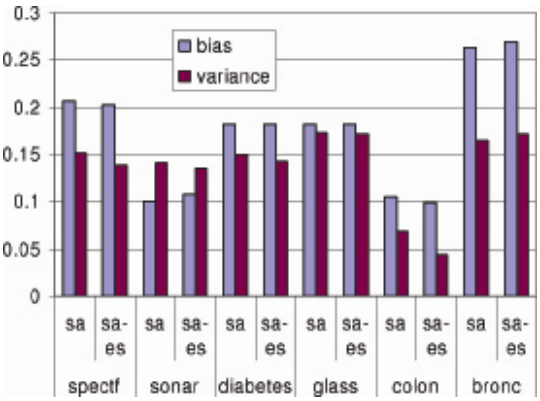


Fig. 6. A comparison of the Bias-Variance decomposition of error for the SA and SAES strategies

5 Conclusions and Future Work

In this paper we have proposed early-stopping as a policy for preventing overfitting in wrapper-based feature subset selection that uses stochastic search. We have described implementations of this idea for Genetic Algorithms and Simulated Annealing. Our evaluation shows that the strategy is effective in general being most successful with Simulated Annealing.

The main problem with this approach is the instability of the stochastic search which can lead to the early-stopping strategy returning poor feature subsets from time to time. So far we have tried to fully automate the early-stopping process, our next objective is to develop an interactive workbench where the user will be shown the overfitting graphs and will be allowed to evaluate a range of early-stopping alternatives.

References

1. Kohavi, R., Sommerfield, D.: Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In: KDD. (1995)
2. Reunanen, J.: Overfitting in making comparisons between variable selection methods. J. Mach. Learn. Res. 3 (2003
3. Loughrey, J., Cunningham, P.: Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets. In Bramer, M., Coenen, F.T.A., eds.: 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI-2004), Springer (2004)
4. van der Putten, P., van Someren, M.: A bias-variance analysis of a real world learning problem: The coil challenge 2000. Machine Learning 57 (2004)
5. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
6. Koistinen, P., HolmstrÄom, L.: Kernel regression and backpropagation training with noise. In: NIPS. (1991)
7. Aha, D., Bankert, R.: Feature selection for case-based classification of cloud types: An empirical comparison. In Aha, D., ed.: AAAI 1994 Workshop on Case-Based Reasoning, AAAI Press (1994)
8. Kohavi, R., Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. In Saitta, L., ed.: Machine Learning: Proceedings of the Thirteenth International Conference, Morgan Kaufmann (1996)
9. Blake, C., Merz, C.: UCI repository of machine learning databases. Technical report, University of California at Irvine, Department of Information and Computer Science (1998)
10. Alon, U., Barai, N., Notterman, D., Gish, K., Ybarra, S., M.D., Levine, A.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc. Natl. Acad. Sci. 96 (1999)
11. Walsh, P., Cunningham, P., Rothenberg, S.J., O'Doherty, S., Hoey, H., Healy, R.: An artificial neural network ensemble to predict disposition and length of stay in children presenting with bronchiolitis. European Journal of Emergency Medicine
12. Weiss, S., & Kulikowski, C. Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems. Morgan Kaufmann Publishers Inc. (1991)
13. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, Number 4598.
14. Kohavi, R., Langley, P., & Yun, Y. (1997). The utility of feature weighting in nearest-neighbor algorithms. Proceedings of the Ninth European Conference on Machine Learning.

15. Kohavi, R., & John, G. (1997). Wrappers for feature subset selection. Artificial Intelligence

Assessing Case Base Quality

Rahul Premraj[1] and Martin Shepperd[2]

[1] Saarland University

[2] Brunel University

premraj@cs.uni-sb.de[1], martin.shepperd@brunel.ac.uk[2]

Abstract. In Case-Based Prediction systems, where the solution is a continuous value, there is heavy reliance upon problem-solution regularity. Often, it is hard to measure, judge or even compare accuracy of such solutions. Our aim is to assess case base regularity before usage to increase the degree of confidence in the proposed prediction. We propose the use of Mantel's Randomisation test to judge overall case base regularity. Thereafter, we focus upon techniques that concentrate on individual constituent cases to single out those that contribute towards overall poor performance. We then present a case discrimination system that ignores poor cases to enhance prediction quality. Our results shed light on the quality of the case base and partially explain poor solution quality. We also identified problematic cases and successfully learnt to avoid their reuse to enhance prediction quality in our problem domain.

Keywords. case-based prediction, software effort prediction, case base regularity, case base visualisation.

1 Introduction

In this paper, we explore techniques to assess case base quality for Case-Based Prediction (CBP). To do so we consider the domain of software project effort where the objective is to accurately predict the effort of new projects based on similar existing and completed projects in the case base. Whilst this particular application of CBR has attracted a substantial amount of research interest, a problem has been the somewhat erratic results in terms of prediction quality.

One explanation for this state of affairs is varying case base quality. Consequently this study explores how we can gain insight into this problem and potentially identify the state of the case base and situations where prediction is not meaningful, that is no better than a random technique. This is analogous to regression modelling where no independent variable has a beta coefficient significantly different from zero. In addition, it may be possible to identify specific problematic cases and quarantine them from the case base.

The remainder of the paper is organised as follows. The next section briefly reviews the current state of CBP for software project effort. We then introduce the Mantel's Randomisation test that confirms irregularity in the case base — a cause for poor prediction quality. Candidate profiles that are used to assess individual cases as potential candidates are presented in Section 5. Lastly, in Section 6, we demonstrate the usage and effect of the quality measures on solution accuracy in our problem domain. The paper is drawn to a close with a summary of results, tentative conclusions and directions for future work.

2 Software Project Prediction

One class of prediction problem that has had some success applying case-based reasoners is software project effort prediction. This is commercially important — since effort is generally the dominant component of cost — but in many respects an extremely challenging problem domain. Problems include small, noisy, heterogeneous and incomplete data sets coupled with large sets of categorical and continuous features that typically exhibit complex interactions. In addition, the solution feature is a continuous value which makes it hard to measure, judge or even compare accuracy. Nonetheless

early work, e.g. [1, 2] produced encouraging results and outperformed traditional methods such as stepwise regression analysis.

Several more recent studies, however, failed to replicate these results, for instance [3]. Closer investigation revealed that this later work used relatively large case bases with more than 40 features. Unfortunately, this prevented them from using an effective feature subset selection approach, instead applying a simple filter method based on a t-test. This then initiated research on the use of meta-heuristic search techniques and, subsequently, we have successfully used greedy search methods, such as forward selection search, to yield good results from large case-bases [4].

Despite this progress, results are not consistent between research groups or even between different random holdout sets. Elsewhere [5] we have conducted a systematic review of published empirical studies using case-based prediction for project effort. We identified 20 distinct studies that compared CBR and some form of regression analysis. Of these 9 supported case-based prediction, 7 regression analysis and 4 were inconclusive. Further analysis reveals that one source of variation is the data sets that are used to form the case bases. For this reason we decided to investigate further and in particular into problem–solution irregularity, where for example, projects that are close neighbours in the feature space but possess strongly divergent solutions (which in this analysis is effort).

There has been existing research in the CBR community on analysing and enhancing case base quality and competence. However, these techniques (e.g. [6, 7]) have been developed largely to enhance solution quality for analytic tasks [8] (e.g. classification, diagnosis, decision support). In such tasks, the solution may involve classifying objects or situations, or imputing missing data or human intervention in each step. Also, in the past techniques that claim to be relatively generic (e.g. [9, 10]) have been demonstrated on analytic problems. This is rather unsurprising given their nature such as well-defined or bounded problem and solution spaces, ease of identifying incorrect solutions and extrapolating performance statistics to trigger corrective measures. In synthetic tasks (e.g. prediction, design and planning, configuration), case base maintenance is more challenging to implement since the concept of solution accuracy is rather subjective.

The most related work is by Keung et al. [11] who originally conceived of the idea of using Mantel's Randomisation to assess the strength of association between the problem and solution spaces. In addition they use a randomisation procedure as set out by Manly [12] to assess the likelihood of this association occurring by chance. We do not adopt the latter approach since we are uncertain concerning the utilisation of this approach in an inferential capacity. Specifically we are unclear as to the population that the inference would relate to.

We decided to use the Desharnais data set [13], which is medium sized ($n = 77$ after 4 incomplete cases are removed), and collected by a Canadian software house from projects distributed amongst 11 different organisations. Apart from total effort (the solution feature) each project is characterised by 10 features including one categorical feature. The data set was randomly split in a $2 : 1$ ratio to form the *TrainingSet* and *TestingSet* respectively. To minimise sample bias, our experiments detailed below were conducted on 30 random samples (without replacement) of the data set.

3 Distance and Residual Rank Ratios Matrices

We first describe the two building blocks of our research, which we refer to at *meta-data*. To ease understanding, we exemplify the methodology using a random sample of 51 cases (simulating the case base) from the Desharnais data set. Our meta-data are concrete instances of *PDist* and *RDist*, which were introduced by Leake and Wilson [14]. The former refers to inter-case distances in the problem space, and the latter refers to distances between cases in the solution space. To cater for our problem domain (having a continuous solution value), we formulated two meta-data as follows:

Distance Rank Ratio ($DRR_{C,T}$). This is the ratio of the order of a candidate's distance from the target (with respect to the other candidates in the case base) to the other total number of cases in the case base.

$$DRR_{C,T} = \frac{DistanceRank}{n}$$

Table 1. Rank Ratio Example

Target	Candidate	Distance	DRR	Residual	RRR
1	2	0.3842	0.42	1498	0.56
1	3	0.4506	0.78	−973	0.36
1	4	0.5250	0.96	−301	0.14
⋮	⋮	⋮	⋮	⋮	⋮
1	10	0.3437	0.20	−7714	0.96
1	11	0.4762	0.86	483	0.24
1	12	0.4954	0.92	2247	0.68
1	13	0.3945	0.46	−1505	0.58
1	14	0.2215	0.12	21	0.02
1	15	0.2330	0.14	2030	0.62
1	16	0.5063	0.94	−5453	0.90

The above fraction computes the $DRR_{C,T}$ of candidate case C for target case T. *DistanceRank* is the candidate's position with respect to other candidates when sorted in increasing order of distance from the target, and n is the number of cases in the case base. Lower values of $DRR_{C,T}$ denote higher similarity of the candidate to the target. To exemplify, assume that a case base CB comprises 11 cases. If *case 1* is the target, then we compute the Euclidean distance of *case 1* from the remaining 10 cases ($n = 10$). Once ranked by increasing order of distance, the closest candidate has $DRR_{C,T}$ $1/10 = 0.1$. The next closest candidate has $DRR_{C,T}$ $2/10 = 0.2$ and so on, until the most distant candidate will have $DRR_{C,T}$ $10/10 = 1$.

Residual Rank Ratio ($RRR_{C,T}$). Likewise, $RRR_{C,T}$ is the ratio of the order of a candidate's residual from the target case (with respect to the other candidate cases in the case base) to the other total number of cases in the case base. Note that residuals are the difference between the actual and predicted values.

$$RRR_{C,T} = \frac{ResidualRank}{n}$$

The above fraction computes the $RRR_{C,T}$ of candidate case C for target case T. *ResidualRank* is the candidate's position with respect to other candidates when sorted in increasing order of residuals from the target, and n is the total number of cases in the case base. Again, lower values of $RRR_{C,T}$ denote better prediction accuracy using the candidate in question. To exemplify, for *case 1* in CB, the 10 candidates are ordered in increasing order of their absolute difference from the target. The closest candidate has $RRR_{C,T}$ $1/10 = 0.1$. The next closest candidate has $RRR_{C,T}$ $2/10 = 0.2$ and so on, until the most distant candidate will have $RRR_{C,T}$ $10/10 = 1$.

The meta-data was generated by jackknifing the *TrainingSet* for every sample. For each *TrainingSet*, we generated 2550 instances (i.e. 51 targets × 50 candidates) of distances, residuals and corresponding $DRR_{C,T}$ and $RRR_{C,T}$ for each of the 30 independent samples. Table 1 is an extract from the meta-data for one sample using *Case 1* as the target. It exhibits the irregularities in the case base that we hoped to perceive using the proposed meta-data. For example, *Case 14* is fairly similar to *Case 1* ($DRR_{C,T} = 0.12$) and expected, its $RRR_{C,T}$ is very low too, i.e., 0.02. Similarly, *Case 16* with very high $DRR_{C,T}$ also has a very high $RRR_{C,T}$. Such cases are examples of *reliable cases* since their behaviour is predictable, i.e. their problem and solution features are proportionally distant from the target's features. This is the behaviour on grounds of which CBR is based - "*Similar problems have similar solutions*".

However, cases with unexpected patterns of behaviour need to be cautiously treated. Examples include *Case 10* that has fairly low $DRR_{C,T}$, but would be a poor choice as a candidate case as reflected by its high $RRR_{C,T}$. Similarly, *Case 4* is very distant from the target case ($DRR_{C,T} = 0.96$) but makes an excellent candidate case given that $RRR_{C,T} = 0.14$. Such cases are labelled *unreliable cases* since their problem and solution features are disproportionably distant from the target case.

4 Mantel's Randomisation Test

The Mantel's Randomisation test (Mantel's test) was primarily developed to compare two distance matrices (generated using a distance measure, e.g. Euclidean), and has so far been used across a wide range of disciplines such as ecology and biology [12]. Fundamentally, the test measures the association between corresponding elements of two distance matrices by using a suitable statistic (usually correlation). Consider two distance matrices, A (predictor variables) and B (response variable)

$$A = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}, \text{ and } B = \begin{bmatrix} 0 & b_{12} & \cdots & b_{1n} \\ b_{21} & 0 & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & 0 \end{bmatrix}$$

Here, matrix A records the inter-case distances between the independent or predictor variables, while matrix B records inter-case distances between the dependent or response variables. In both matrices, elements a_{ij} and b_{ij} respectively record problem space and solution space distances between the i^{th} and j^{th} case in the case base. The diagonal elements in A and B are zero because in these cases, the candidates are same as the targets (i.e., $i = j$).

$$R = \frac{\sum_{i,j=1}^{n} a_{ij}b_{ij} - \sum_{i,j=1}^{n} a_{ij} \sum_{i,j=1}^{n} b_{ij}/n}{\sqrt{\left[\left\{\sum_{i,j=1}^{n} a_{ij}^2 - (\sum_{i,j=1}^{n} a_{ij})^2/n\right\}\left\{\sum_{i,j=1}^{n} b_{ij}^2 - (\sum_{i,j=1}^{n} b_{ij})^2/n\right\}\right]}} \quad (1)$$

Eqn. 1 calculates Mantel's test statistic (correlation in this case) between A and B, where n is the square of the dimension of the matrices (both being square and of the same size) less the number of diagonal elements. The correlation (R_1) of the two distance matrices A and B is calculated by measuring across the pairs of corresponding elements excluding the diagonal elements. Thus, R_1 is an indicator of the degree of regularity in the case base. Higher values of R_1 suggest corresponding cases are proportionally spaced out in the problem and solution spaces and vice versa.

Thereafter, the indices of one matrix, say A, are randomised and the correlation (R_2) is computed between the original distance matrix B and the randomised distance matrix A. Now, if $R_2 > R_1$, it suggests that there exists no relationship between the predictor and response variables since random pairs are more strongly correlated. This is analogous to regression modelling where no independent variable has a beta coefficient significantly different from zero. To ascertain the likelihood of a relationship, Mantel's test statistic is calculated between matrix B and 4999 randomisations of matrix A to test for statistical significance.

The Mantel's Randomisation test's contribution lies in measuring the degree of association in the problem and solution spaces of cases with respect to each other. Such a test is crucial to CBP systems since they operate by assuming a strong relationship between predictor and response variables. Results would expose the degree of underlying irregularity in the case base and indicate if there is a need to incorporate additional pre-processing or functionality to enhance prediction accuracy or even disregard the case base from use.

We conducted the test on the 30 *TrainingSet*s of the Desharnais data set. The results unfolded several interesting characteristics of the data set. Firstly, the correlation coefficient for each sample between the original distance matrices A and B were positive. This suggests that from a virtual reference point, corresponding data points in the problem and solution space tend to move in the same direction, thus warranting the use of this data set for CBP.

Secondly, although positive, the value of correlation from each sample is low. The highest recorded value was 0.37, while the lowest was 0.15. Weak correlation suggests the existence of many outliers that contribute towards overall irregularity in the case base. This was verified by the range and low variance of the correlation values, which imply that every random sample contained at least few

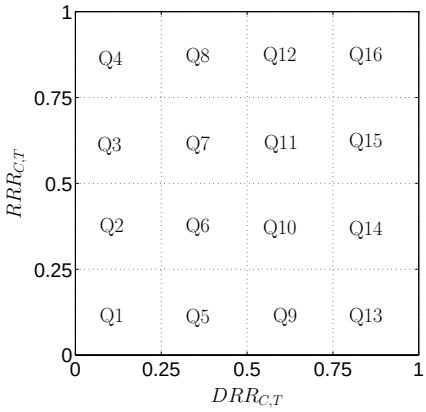


Fig. 1. Case Profile

unreliable cases that distorted overall irregularity. This augments the need to supplement inter-case distance with more information prior to selecting the case for reuse.

Lastly, for each random sample, the correlation coefficient between the original distance matrices was the highest amongst all 5000 computed coefficients. Thus, each sample passed the statistical significance test ($p < 0.001$). This is an important observation since it ascertains existence of a pattern between the predictor and response variables or strongly indicates a problem–solution relationship. Thus, any prediction model or method would derive the best possible results using the original pairs of predictor and response variables.

Though the Mantel's Randomisation test uncovers overall irregularity in the case base , it lacks the provision to identify the cases that substantially contribute towards the disorder. This is indeed a crucial task to improve prediction accuracy. Identified unreliable cases must be reused with caution or even quarantined to instead, favour more reliable cases. The next two sections proposes a framework to accomplish the same.

5 Case Quality

The rank ratios in Section 3 can be consolidated to build what we term as a *candidate profile* for every case. A profile by definition highlights characteristics of the object in question. Hence, with candidate profiles, we aim to build a unified view that reflects the candidates's performance previously.

To build individual candidate profiles, we divide the range of $DRR_{C,T}$ and $RRR_{C,T}$ ($[0-1]$) into 4 equal intervals of size 0.25. This results in a matrix as Fig. 1. Each time a candidate is reused, we compute its $DRR_{C,T}$ and $RRR_{C,T}$ and increment the count of the candidate profile's cross-section quartile within which the $DRR_{C,T}$ and $RRR_{C,T}$ lie. For example, for any retrieval instance, if a candidate has $DRR_{C,T} = 0.12$ and $RRR_{C,T} = 0.3$, we increment $Q2$ by one or if $RRR_{C,T} = 0.8$, we increment $Q4$ by one.

A case with high density of data points in blocks $Q1$, $Q2$, $Q5$ and $Q6$ is desirable since its distance in the problem and solution space are proportional. Also, cases with higher density of data points in blocks $Q11$, $Q12$, $Q15$ and $Q16$ are equally desirable for the same characteristic as above. Importantly, given high values of $DRR_{C,T}$, such cases may be seldom retrieved. Hence, an ideal candidate may be one whose profile vastly covers the eight blocks discussed yet to form a cigar shaped distribution.

Case profiles with data points lying in blocks $Q9$, $Q10$, $Q13$ and $Q14$ reflect large distances in the problem space but nearness in the solution space between the candidate and targets. Though these cases may be seldom be reused due to large distance in the problem space, they pose little risk since the likelihood of a good solution may be high. Conversely, case profiles with data points concentrated in blocks $Q3$, $Q4$, $Q7$ and $Q8$ signify nearness to the target case in the problem space but large distances in the solution space. Such cases are most important to be recognised due to the high probability of their reuse and delivery of a poor solution.

6 Enhancing Case–Based Prediction

Our approach's larger goal is to improve prediction accuracy. We now show how one can exploit the candidate profiles by using them in tandem with the retrieval algorithm to reuse only reliable candidates and rejecting those that are identified as unreliable. The proposed system simply leverages the candidate profile to compute the likelihood of being delivered a good solution upon reuse. If the likelihood equals or exceeds a set threshold value (T), the candidate is reused. Else, it is rejected and the next nearest candidate that meets the threshold is reused. This modifies the *Retrieval* stage in CBR into a two-step process. First, for a given target, all candidates are ranked in order of increasing distance. Second, starting from the closest candidate, the first k candidates that meet the threshold likelihood level are selected to be reused for prediction.

The candidate profile in Fig. 1 is populated with the frequency of good and poor solutions delivered when it has been reused. With access to this tabulated frequency matrix, likelihood of a good solution can be easily computed as

$$\text{Probability}_{DRR_{C,T}} = \frac{\text{Frequency of Good Solutions}_{DRR_{C,T}}}{\text{Frequency of Use}_{DRR_{C,T}}}.$$

The above fraction results in a probability value with range $[0-1]$. However, the concept of a good solution remains to be defined. For our case, a good solution is characterised by the $RRR_{C,T}$ of the candidate. Candidates with a higher frequency in the lower halves of their profiles should be preferred over other candidates that have a higher density in the upper halves of their profiles. Based on this reasoning, for $DRR_{C,T} = 0.2$, the above fraction can be transformed into:

$$P = \frac{|Q1| + |Q2|}{|Q1| + |Q2| + |Q3| + |Q4|}$$

Likewise, for $DRR_{C,T} = 0.4$, $Q1$, $Q2$, $Q3$, and $Q4$ are to be replaced by $Q5$, $Q6$, $Q7$, and $Q8$ respectively. A candidate can only be reused, iff $P \geq T$, where T is a probability threshold value.

Candidate selection for reuse can be influenced by varying the value of T. As T has range $[0-1]$, setting its value to 0 is equivalent to using simply k-NN for prediction. However, when the value of T is increased, the system becomes proportionally more selective, i.e. increasingly discriminatory against potentially poor candidates. Thus, for $T = 1$, all data points on the candidate profile would need to lie in the lower quartiles for it to get selected for reuse. Hence, an in-between value is preferred so that the system is has a balance between being too permissive or too discriminatory .

We now consider the results of coupling candidate profiles with distance on prediction accuracy using the Desharnais data set samples. In Fig. 2, we plot a comparison of median sum of absolute residuals ($Sum|Res|$) of the 30 random *TestingSet*s for every combination of $k([1-5])$ and probability threshold ($[0-0.8]$). We limited the range for T since for $T > 0.8$, predictions repeatedly failed as no candidates with consistently good performance were to be found. Also note that when $T = 0$, the system reduces to simply k-NN. We use the results from the latter to serve as a benchmark for our approach.

We first examine the results using k-NN exclusively for selecting candidates for reuse, i.e., $T = 0$. Using only the nearest neighbour ($k = 1$) results in the largest $Sum|Res|$. However, $Sum|Res|$ continues to decline as k increases to 5. During the experiments, we found that k-NN often retrieved cases that lay far away on opposite sides of the target case in the solution space and thus, provided a ballpark solution by averaging the extreme values. Consequently, larger values of k dampened the effect of extreme values on the proposed solution. Though the solution may potentially be close to the true value, this technique is likely to reduce the confidence of users in the system.

However, once coupled with T set even as low as 0.1, we observe an improvement in performance for every value of k. The trend of improved performance continues until $T = 0.6$ and thereafter, the $Sum|Res|$ begins to increase. This is because the system became more discriminating and overlooked

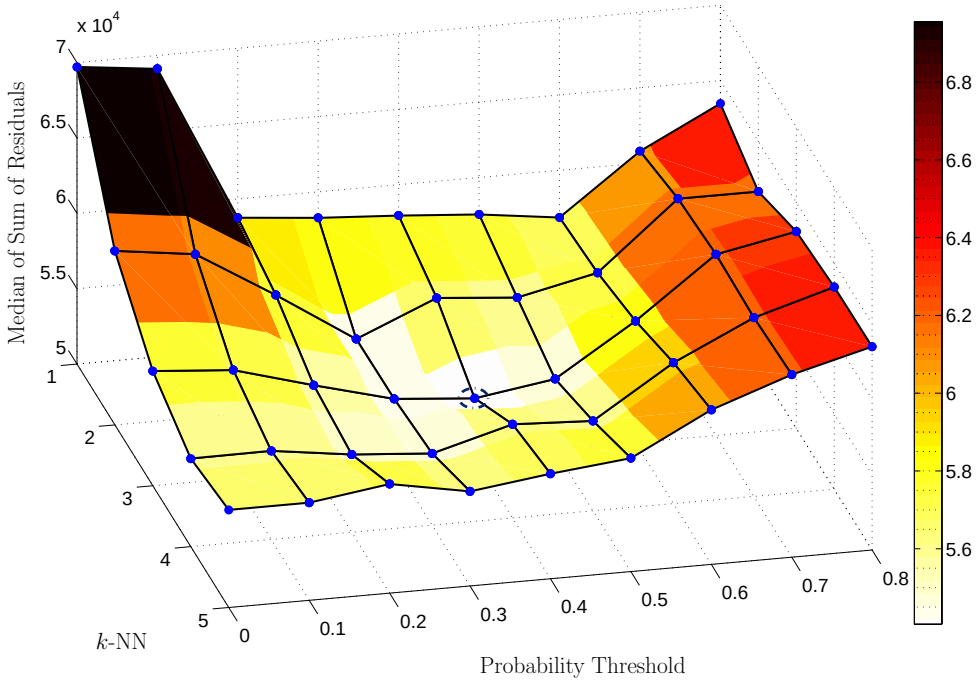


Fig. 2. Comparison of Performance by Coupling k-NN and Probability Threshold for the Desharnais Dataset

many similar cases in search of high quality cases. As a result, very distant cases got reused and this decreased prediction accuracy. For this data set, we found the optimum combination of k and T to be 3 and 0.4 respectively. A total gain of 22% improvement in prediction accuracy was made in comparison to using more than one nearest neighbour and probability threshold in comparison to using the nearest neighbour only.

Also note that by using $k = 1$, the lowest mean of $Sum|Res|$ is obtained by using T as high as 0.5. This is indicative of the number of unreliable nearest neighbours in the case base that may have been reused for values of $T < 0.5$. This decrease in the residuals using the sole neighbour confirms that the system successfully learnt to favour relatively more distant yet quality cases for reuse instead of unreliable nearer cases.

7 Summary

In this paper, we proposed methods to assess case base quality to measure inherent problem–solution irregularity, and to exploit case bases better to increase solution accuracy. The proposed methods targeted CBP systems with continuous value solutions. The methods ascertained that cases in the Desharnais data set do possess a relationship in the problem and solution spaces. This was confirmed by the Mantel's Randomisation test to check case base regularity and its suitability for CBR. It further uncovered that albeit positive and statistically significant, the relationship was weak due to irregular cases. To measure individual case reliability objectively, we created individual candidate profiles that updated the frequency of good and poor solutions delivered with respect to the a candidate's distance from the target case.

Thereafter, we demonstrated the applicability of such crucial information about cases by using their case profiles in conjunction with target–candidate case distance. Only selective candidate cases identified as reliable were reused to make a prediction. Their degree of reliability was measured by their likelihood to propose an acceptable solution. We found that reuse by reflection upon case quality provided better results than using only the k nearest neighbours. Though the optimum combination for k and T was found to be 3 and 0.4, we expect this to vary across case bases and their sizes.

The significance of this work lies in providing possibilities of improving performance of CBR systems which deal with imperfect and noisy data. Dealing with such case bases is all the more chal-

lenging when solutions are continuous values. We expect inherent irregularity to be a cause for erratic prediction quality and hence, it needs to be effectively dealt with to increase prediction accuracy. Importantly, we believe this technique to be generic and can be applied to a variety of CBR domains with little or no adaptation. This paper also contributes to the body of knowledge of case base maintenance that has so far largely focussed upon classification domains. Our results warrant further validation using more real world software engineering data sets to comment on broad affectivity of the proposed techniques. Another direction for future work involves developing more sophisticated decision making techniques that gauge case reliability to consider them for reuse.

References

1. Prietula, M.J., Vicinanza, S., Mukhopadhyay, T.: Software-Effort Estimation with a Case–Based Reasoner. Journal of Experimental and Theoretical Artificial Intelligence **8** (1996) 341–363
2. Shepperd, M., Schofield, C.: Estimating Software Project Effort Using Analogies. IEEE Transactions on Software Engineering **23** (1997) 736–743
3. Briand, L., T.L., Wieczorek., I.: Using the European Space Agency Data Set: A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. In: Procs. of the 22nd International Conference on Software Engineering, Limerick, Ireland, Computer Press Society (2000) 377–386
4. Kirsopp, C., Shepperd, M.J., Hart, J.: Search Heuristics, Case–Based Reasoning and Software Project Effort Prediction. In: Proc. of Genetic and Evolutionary Computation Conference, GECCO 2002, New York, USA (2002)
5. Mair, C., Shepperd, M.: The Consistency of Empirical Comparisons of Regression and Analogy-based Software Project Cost Prediction. In: In Proc. of the 4th IEEE International Symposium on Empirical Software Engineering, Noosa Heads, Australia (2005)
6. Smyth, B., Cunningham, P.: The Utility Problem Analysed - a Case Based Reasoning Perspective. In Smith, I.F.C., Faltings, B., eds.: Advances in Case–Based Reasoning: 3rd European Workshop, EWCBR–96. Volume 1168 of LNCS., Lausanne, Switzerland, Springer (1996) 392–399
7. Smyth, B., McKenna, E.: Modelling the Competence of Case-Bases. In Smyth, B., Cunningham, P., eds.: Advances in Case-Based Reasoning: 4th European Workshop, EWCBR–98. Volume 1488 of LNCS., Dublin, Ireland, Springer-Verlag (1998) 208–220
8. Voss, A., Oxman, R.: A Study of Case Adaptation Systems. In Gero, J.S., Sudweeks, F., eds.: Artificial Intelligence in Design, Stanford, USA, Kluwer Academic Publishers (1996) 173–189
9. Portinale, L., Torasso, P., Tavano, P.: Speed-up, Quality and Competence in Multi-Modal Case-Based Reasoning. In Althoff, K.D., Bergmann, R., Branting, L.K., eds.: Case-Based Reasoning and Development, 3rd International Conference, ICCBR–99. Volume 1650 of LNCS., Seeon Monastery, Germany, Springer-Verlag (1999) 303–317
10. McKenna, E., Smyth, B.: Competence-Guided Case-Base Editing Techniques. In Blanzieri, E., Portinale, L., eds.: Advances in Case–Based Reasoning: 5th European Workshop, EWCBR–00. Volume 1898 of LNCS., Trento, Italy, Springer (2000) 186–197
11. Keung, J., Kitchenham, B., Jeffery, R.: 'Analogy-X' - An Extension to Cost Estimation Using Analogy. Technical Report Unpublished Report, NICTA, Sydney, Australia (2005)
12. Manly, B.F.J.: Randomisation, Bootstrap and Monte Carlo Methods in Biology. Second edn. Texts in Statistical Science Series. Chapham & Hall/CRC (2001)
13. Desharnais, J.M.: Analyse Statistique de la Productivitié des Projets Informatique à Partie de la Technique des Point des Fonction. Master's thesis, University of Montreal, Canada (1989)
14. Leake, D., Wilson, D.C.: When Experience is Wrong: Examining CBR for Changing Tasks and Environments. In Althoff, K.D., Bergmann, R., Branting, K., eds.: Case-Based Reasoning Research and Development: 3rd International Conference, ICCBR–99. Volume 1650 of Lecture Notes in Computer Science., Seeon Monastery, Germany, Springer (1999) 218–232

The Group has links with many outside bodies. It is a specialist group of the British Computer Society and a member of ECCAI, the European Co-ordinating Committee for Artificial Intelligence.

Since its inception the group has enjoyed a good working relationship with government departments involved in the AI field (beginning with the Alvey Programme in the 1980s). A succession of Department of Trade and Industry (DTI) representatives, have been co-opted as committee members. The Group acted as co-organiser of the annual DTI Manufacturing Intelligence awards and has included sessions presenting the results of the DTI Intelligent Systems Integration Programme (ISIP) in its annual conferences.

The group also has a good relationship with the Institution of Electrical Engineers (IEE), with which it has co-sponsored colloquia over many years, and with NCAF, the Natural Computing Applications Forum. We also host the annual UK-CBR (Case-Based Reasoning) workshops at our annual conferences. This year, the group will be co-hosting IJCAI-05 in Edinburgh.

Benefits of Membership

- Preferential rates for the Group's prestigious international conference on Artificial Intelligence, which has run annually since 1981.
- Discounted rates at other SGAI-sponsored events.
- Discounted rates for ECCAI organised events. The Group has been a member of the European Co-ordinating Committee for Artificial Intelligence since 1992.
- Discounts on international journals, and occasional special offers on books.
- Advance information on the SGAI Evening Lectures, which are held on a regular basis in central London.
- Free subscription to the *Expert Update* journal, containing reviews, technical articles, conference reports, comment from industry gurus and product news.
- The SGAI website at www.bcs-sgai.org and the AI-SGES list server to facilitate communication on all aspects of AI.
- A substantial proportion of the Group's membership is from industry. Providing a valuable forum where both academic and industrial AI communities can meet.

How to Join BCS-SGAI?

To join BCS-SGAI you do not need to be a member of the BCS. For further information please visit our website at www.bcs-sgai.org.

Subscription Rates

Annual subscription rates for Individual and Corporate Members are:

INDIVIDUALS

Standard Members (UK addresses)	£31.00
Standard Members (Overseas addresses)	£41.00
BCS Members (UK addresses)	£22.00
BCS Members (Overseas addresses)	£31.00
Students (UK addresses)	£11.00
Students (Overseas addresses)	£21.00
Proof of student status is required	
Retired (UK addresses)	£11.00
Retired (Overseas addresses)	£21.00

CORPORATE

UK addresses	£150.00
Overseas addresses	£190.00

Add £5 to all these rates if not paying by standing order.

27th BCS-SGAI International Conference on

Innovative Techniques and Applications of Artificial Intelligence

10-12th December 2007, Cambridge, UK

http://www.bcs-sgai.org/ai2007

AI-2007

AI-2007 is the 27th Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGAI). The scope of the conference comprises the whole range of AI technologies and application areas. Its principal aims are to review recent technical advances in AI technologies and to show how these advances have been applied to solve business problems.

CONFERENCE VENUE

Peterhouse College, founded in 1284, and its hall, built between 1286 and 1290, was the first collegiate building in Cambridge. Peterhouse Gardens achieved fame between 1830 and 1930 as the smallest deer park in England. Located in the centre of Cambridge, the college combines historic buildings with modern conference facilities, within easy reach of the shopping and entertainment of Cambridge.

CONFERENCE OUTLINE

TECHNICAL STREAM – Areas of interest include (but not restricted to): knowledge based systems; semantic web; constraint satisfaction; intelligent agents; machine learning; model based reasoning; natural language understanding; case based reasoning; neural networks; genetic algorithms; data mining and knowledge discovery in databases; robotics and pervasive computing.

APPLICATION STREAM – Papers in recent years have covered all application domains, including commerce, manufacturing and defence. Papers are selected to highlight critical areas of success (and failure) and to present the benefits and lessons learned.

WORKSHOPS – This year's workshop day on 10th Dec. will include 4 half day workshops and the full-day UK Case-Based Reasoning workshop.

Fuzzy Logic and Modelling of Uncertainty Chair: Prof Bob John, De Montfort University.

Serious Games Chair: Prof. David Brown, Nottingham Trent University.

Artificial Intelligence in Education Chair: Dr Maria Fasli, University of Essex.

Intelligent Systems in Accounting, Finance and Management Chair: Dr Bob Berry, University of Nottingham.

UKCBR – The 12th UK CBR workshop will be collocated with AI-2007. It concerns all aspects of case-based reasoning and practical applications.

POSTER SESSION – There is also a poster session for presenting work in progress.

PRIZES – There are sponsored prizes (a trophy plus £500) for the best paper submitted in each stream. There is also a trophy and a cash prize for the best-presented poster.

THE BCS MACHINE INTELLIGENCE PRIZE – Trophy plus £1,000 cash prize for the best live demonstration of progress towards machine intelligence. Deadline October 1st, 2007.

REGISTRATION

Online registration is open from the end of July 2007. Accommodation is available on a first-come, first-served basis.

ENQUIRIES

sgai-conference@bcs.org.uk