Retrieving Relevant Experiences

Armin Stahl

In order to enable the efficient reuse of collected experience knowledge, the identification of the most relevant experiences with respect to the current reuse need is one of the most crucial issues. One approach for supporting this process with AI methods is the application of similarity-based retrieval techniques developed in the area of Case-Based Reasoning. We describe the basic idea of this approach and present a novel open source tool which simplifies the development of knowledge-based retrieval functionality in experience management applications.

1 Introduction

Experience management requires the implementation of several processes in order to be applied successfully in practice. According to [2] the following six specific activities must be considered to enable efficient reuse of experience knowledge: Collecting, Modeling, Storing, Reusing, Evaluating, and Maintaining Experience.

On the one hand, some of these activities typically have to deal with certain social aspects. For example, the successful collection of experiences often depends on approaches to motivate people to provide their own experience knowledge. On the other hand, several activities like the storage and efficient reuse of collected experience knowledge require a lot of technical support using computer science methods.

In this article we focus on the latter aspect, i.e. on the more technical issues related to experience management. In particular, we discuss an approach for implementing one central task of the reuse process, namely retrieving relevant experiences from a given experience base with respect to a current problem situation or information need. This approach is based on *similarity-based retrieval* techniques developed in the area of *Case-Based Reasoning (CBR)*.

One basic precondition for identifying relevant experiences is an accurate representation of the experience knowledge collected. This requires a model about the domain of the specific kind of experiences that have to be managed in a given application scenario. In Section 2 we briefly discuss how to represent experience knowledge. In Section 3 we then describe the idea of using similarity-based techniques for retrieving experience knowledge and in Section 4 we present a novel software tool which facilitates the implementation of these techniques. Finally, we conclude with a summary and an outlook on future improvements of the presented tool.

2 Representing Experiences

Depending on the underlying application scenario, in general, experience knowledge may be represented by using quite different representation formalisms, e.g. free text, attribute value pairs, object-oriented, graph-based, or image-based representations, etc.

The advantage of the free text approach is certainly its flexibility and independence from a fixed domain model. Moreover, today many companies have already stored huge collections of text-based documents which describe a lot of experience knowledge. However, this flexibility is also the major drawback of the pure text-based approach because natural language processing methods are still a long way from being able to process arbitrary text in a reliable way.

Hence, many knowledge management and expert systems still rely—at least partially—on more well-defined domain models, for example, based on attribute value pairs. In order to represent knowledge in a more structured way, attribute value pair representations can easily be extended to object-oriented models which are commonly also used for representing *ontologies*. Due to the increasing interest in ontologies over the last few years, today a lot of tools for modeling ontologies are available. One of the most popular tools is certainly the open source platform *Protégé* [5] which provides powerful graphical user interfaces and algorithms not only for modeling and visualizing ontologies, but also for reasoning from them.

Other approaches to describe experience knowledge like image-based representations (e.g. medical images) are typically special purpose approaches which also require very specific algorithms to process the knowledge.

Hence, in the following we restrict ourselves to attribute value based and non-cyclic object-oriented representations which allow aggregation and specialization relations between defined classes. The classes themselves are described by a set of attributes with simple data types, e.g. strings, numbers, symbols, etc. For the reasoning approach described in Section 3 a clear definition of the attributes' specific data types including allowed value ranges (e.g. for numeric attributes) or enumerations of allowed values (e.g. for symbolic attributes) is also a central part of the domain model. For representing aggregation relations between classes, we assume the existence of specific *relational attributes* with class instances as values. Moreover, classes can be arranged in a specialization hierarchy where sub-classes inherit the attributes of their super-classes.

The main purpose of such a description is the characterization of the most important aspects of an experience in a well-defined way in order to enable the automation of reasoning procedures. This does not necessarily mean that all available knowledge about an experience has to be represented by attribute value pairs. Some attributes may contain large portions of free text or may link to any other available external information (e.g. by specifying URLs of text documents or images) which is also relevant in the scope of the experience. Such information may not be processable by the reasoning algorithms, however, it may provide the user additionally useful knowledge for the reuse of the experience.

From the CBR point of view, an experience (in CBR called *case*) is typically described by two central parts. The first part contains a description of a past problem situation and the corresponding context in which the experience was made. Because this part of the description is used to estimate the relevance of an experience for a current problem situation automatically, it has to be described as formal as possible. In the following we denote this part as the *experience characterization*. The second part is the description of a suitable solution for the described problem including any information that may help to solve a similar problem (e.g. information about the quality of the solution, lessons learned when applying the solution, etc.). Depending on the desirable degree of automation of the applied reuse processes, this part of the experience description may be represented more informally, e.g. if it is sufficient that a human user can reproduce the proposed solution.

3 Utility Approximation

After having collected, characterized and stored a certain amount of experience knowledge, the foundation for reusing this knowledge is the capability to identify the most relevant experiences with respect to a given information need. We assume that this information need is described by using the same model as specified for the experience characterization and we denote it in the following as the *query*.

One possibility to estimate the relevance of an particular experience is to measure the *similarity* between its characterization and the current query. This is the basic approach of CBR, where it is assumed, that similar problems usually also have similar solutions, i.e. if the problem descriptions are sufficiently similar, the corresponding case including its solution description is supposed to be relevant for the query.

Depending on the experience characterization various kinds of similarity measures can be applied. In general, the basic task of a similarity measure is to compute a score for each stored experience. These scores, which are typically normalized to the interval $[0, 1]$, then can be used by the retrieval algorithm to select and rank the experiences with respect to their relevance before presenting them to the user or subsequent reasoning steps.

3.1 Knowledge-Poor Similarity Measures

Many CBR applications apply quite traditional similarity or distance metrics. Typical examples are the Hamming Distance for pure binary or symbolic attributes, the Euclidean Distance for pure nummeric attributes, or the Heterogeneous Euclidean Overlap Metric [7] for mixed representations.

All these traditional metrics have in common, that they only measure syntactical differences without considering much specific knowledge about the application domain. When using these measures the only possibility to encode certain domain knowledge is the definition of specific feature weights in order to express the individual relevance of different attributes. However, in particular when dealing with symbolic attributes this is often not sufficient since the relationship between different symbols cannot be modeled with these kinds of metrics. For example, when having an attribute "Color" with the allowed values "green", "blue", "red", and "orange", traditional metrics are not able to express that "red" is more similar to "orange" than it is to "blue".

3.2 Knowledge-Intensive Similarity Measures

The knowledge-poor similarity measures described in the previous section certainly have the advantage that they are easy to apply in many applications. However, their generality is also their major drawback. Since they do not consider available background knowledge about the application domain, they often fail to estimate the relevance of stored experience knowledge accurately, where relevance can be interpreted as the *utility* of the experience with respect to the query.

In general, the utility of available experience knowledge strongly depends on the specific application scenario including the characteristics of the domain, the preferences and background of the users, the context of the query, etc. However, since these influences are typically quite complex, not well understood, or even partially unknown, a calculation of the actual utility is impossible. Instead, similarity measures are used as heuristics for approximating this utility [3].

As typical for heuristics, the quality of this approximation can be improved by incorporating as much domain knowledge as possible. In order to simplify the definition of such *knowledge-intensive similarity measures* [6], a decomposition of the *global similarity measure* for a class C into a set of independent *local similarity measures* sim_i—one for each attribute a_i—and an accurate amalgamation of the resulting similarity values has proven its value. By using a weighted sum as amalgamation function the similarity between two instances I and J of class C may be computed as follows:

$$Sim_C(I, J) = \sum_{i=1}^{n} w_i \cdot sim_i(I_i, J_i) \quad \text{with} \quad \sum_{i=1}^{n} w_i = 1$$

Here, I_i and J_i denote the values and $w_i > 0$ denotes the weight of attribute a_i. In particular the attribute specific local similarity measures sim_i can be used to encode a significant amount of domain knowledge. If a_i is a relational attribute referring to some instance of another class D, the local-global-principle can be applied in a recursive manner, this means sim_i will be the global similarity measure of D.

Since queries and an experience characterizations do not necessarily have to be described by an instance of the same class, the following approach to calculate the similarity between two arbitrary instances I and J based on the predefined object-oriented model has been proposed [4]:

$$Sim(I, J) = Sim_{inter-class}(I, J) \cdot Sim_{<I,J>}(I, J)$$

Here, $< I, J >$ denotes the most specific common superclass of I and J, i.e. $Sim_{<I,J>}$ is the global similarity measure of class $< I, J >$ which computes a similarity score based on the common attributes of I and J. $Sim_{inter-class}(I, J)$ computes an additional similarity score that is based on the structure of the inheritance hierarchy in order to express the maximal possible similarity between instances of different classes.

4 myCBR

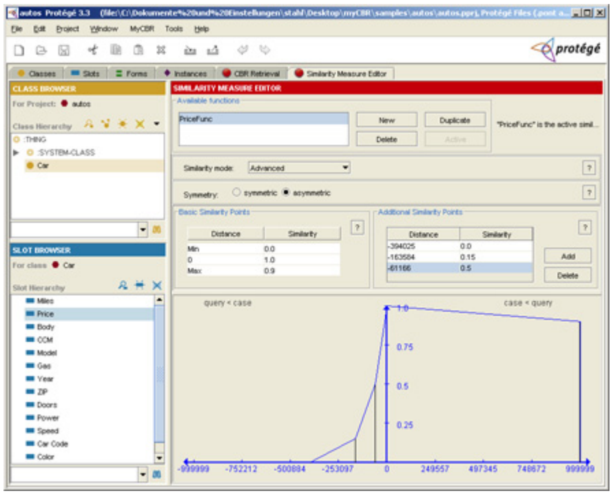
In order to simplify the implementation of experience management applications commercial software tools such as the empolis:Information Access Suite (e:IAS) have been developed. On the one hand, such tools provide very powerful functionality for developing large-scale industrial applications. However, on the other hand, they are often not suited for smaller projects, research development, or teaching support due to their complexity and financial issues.

A first step to provide CBR researchers a freely available platform for rapid prototyping is the open source jColibri system[1]. In the following we want to present another open source tool recently developed at the German Research Center for Artificial Intelligence (DFKI) called *myCBR*[2]. In contrast to the jColibri system which provides a framework for the most CBR issues, myCBR focuses on advanced support of the similarity-based retrieval step. It is designed as a plug-in for the popular ontology editor Protégé, i.e. the experience characterization can be modeled by using Protégé functionality. However, myCBR also supports the automatic generation of characterization models from available raw data given in the csv[3] format.

4.1 Similarity Editors

Once a characterization model has been generated, myCBR provides comfortable graphical user interfaces for modeling and managing corresponding knowledge-intensive similarity measures. In particular the definition of domain specific local similarity measures is supported by various graphical editors.

When dealing with numeric data, the similarity between two values is usually based on their difference. This has led to the use of so-called *difference-based similarity functions* which map difference values to similarity values. A trivial mapping is defined by $sim(x, y) = 1 - \frac{|x-y|}{maxd}$ where $maxd$ is the maximal possible difference with respect to the attributes' value range. However, knowledge-intensive similarity measures typically require very specific mappings in order to consider the domain specific utility function. myCBR provides the possibility to choose and parameterize some standard functions or to define nearly arbitrary functions by applying an interpolation approach (see Figure 1).



"PriceFunc" is the active siml...

Similarity mode: Advanced

Symmetry: ○ symmetric ● asymmetric

Distance	Similarity
Min	0.0
0	1.0
Max	0.9

Distance	Similarity
-394025	0.0
-163584	0.15
-61166	0.5

Abbildung 1: Editor for Difference-based Similarity Functions

When dealing with symbolic data, a straight forward way to model the similarity between different symbols is the definition of a *similarity table* which allows to enumerate the similarity between all possible pairs of symbols explicitly. However, when dealing with larger symbol sets, this approach requires a lot of modeling effort. For example, in the worst case n allowed symbols would require the complete definition of a $n \times n$ matrix of similarity values. In order to reduce the modeling effort, different approaches can be applied. The first possibility is the definition of an order on the symbols and a corresponding mapping to some integer numbers. This allows to apply difference-based similarity functions also on symbolic data.

Another approach is the arrangement of symbols in a taxonomy. In principle, a taxonomy encodes certain knowledge about the relations between symbols which can be used to infer appropriate similarity values automatically. By providing some additional information about the actual semantics of the taxonomy in the underlying application scenario, even more specific similarity measures can be modeled with fairly little modeling effort (see [1] for details). The corresponding editor of myCBR is shown in Figure 2.

The idea to infer similarity knowledge from taxonomic relations can also be applied to model the inter-class similarities for more complex object-oriented experience characterizations. To model these similarities, myCBR provides a corresponding interface which operates on the class inheritance hierarchy.

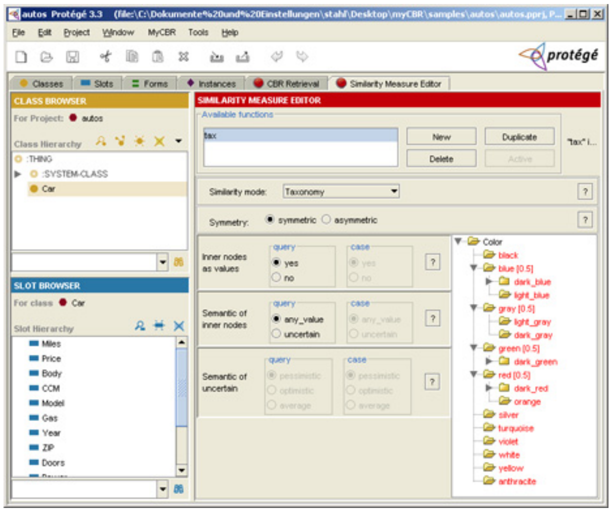
4.2 Similarity-Based Retrieval

After having modeled accurate similarity measures, an initial test of their capability to identify relevant experience

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

[2] http://mycbr-project.net

[3] The *comma separated values* format is also supported by common database and spreadsheet software.



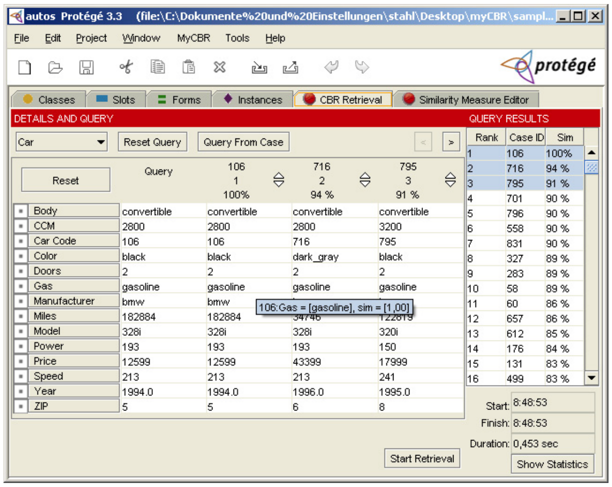
Similarity mode: Taxonomy

Symmetry: ● symmetric ○ asymmetric

	query	case
Inner nodes as values	● yes ○ no	● yes ○ no
Semantic of inner nodes	● any_value ○ uncertain	● any_value ○ uncertain
Semantic of uncertain	● pessimistic ○ optimistic ○ average	● pessimistic ○ optimistic ○ average

Abbildung 2: Editor for Taxonomy Similarities

knowledge from a given experience base has to be performed. Therefore, some experiences have to be entered manually by using Protégé or they may be imported by using the csv interface of myCBR. myCBR then provides a special retrieval interface where the user can define queries and start similarity-based retrievals (see Figure 3) for testing purposes. This interface includes comfortable functionality for browsing through the retrieval result and for analyzing the computed similarity scores.


autos Protégé 3.3 (file:\C:\Dokumente%20und%20Einstellungen\stahl\Desktop\myCBR\sampl...
File Edit Project Window MyCBR Tools Help
protégé
Classes Slots Forms Instances CBR Retrieval Similarity Measure Editor
DETAILS AND QUERY
QUERY RESULTS
Car
Reset Query
Query From Case
<
>
Reset
Start Retrieval
Show Statistics

	Query	106 1 100%	716 2 94 %	795 3 91 %
Body	convertible	convertible	convertible	convertible
CCM	2800	2800	2800	3200
Car Code	106	106	716	795
Color	black	black	dark_gray	black
Doors	2	2	2	2
Gas	gasoline	gasoline	gasoline	gasoline
Manufacturer	bmw	bmw	[illegible]	[illegible]
Miles	182884	182884	34746	122819
Model	328i	328i	328i	320i
Power	193	193	193	150
Price	12599	12599	43399	17999
Speed	213	213	213	241
Year	1994.0	1994.0	1996.0	1995.0
ZIP	5	5	6	8

106:Gas = [gasoline], sim = [1,00]

Rank	Case ID	Sim
1	106	100%
2	716	94 %
3	795	91 %
4	701	90 %
5	796	90 %
6	558	90 %
7	831	90 %
8	327	89 %
9	283	89 %
10	58	89 %
11	60	86 %
12	657	86 %
13	612	85 %
14	176	84 %
15	131	83 %
16	499	83 %

Start: 8:48:53
Finish: 8:48:53
Duration: 0,453 sec

Abbildung 3: The Retrieval Interface

Once the modeled similarity measures lead to satisfactory results, one may integrate them in some external experience management application in order to provide similarity-based retrieval functionality there. myCBR allows to export the similarity measures together with the domain model in XML and also provides an additional stand-alone retrieval engine[4] which can easily be integrated in external software, for example, to implement web-based experience management applications. Currently, the retrieval engine still supports only sequential retrieval which sets some limits with respect to the manageable size of experience bases.

5 Summary

The retrieval of relevant experience knowledge with respect to a given query is a central issue in every experience management application. In this article we discussed the application of similarity-based retrieval techniques for this task. We described that the definition and application of knowledge-intensive similarity measures enables an approximation of the experiences' actual utility in order to identify relevant experiences more accurately than it is possible with more traditional, knowledge-poor similarity measures. Moreover, we have presented the open source tool myCBR which simplifies the implementation of similarity-based retrieval functionality in experience management applications dramatically.

For the future, we plan to further extend the functionality of myCBR. This includes the implementation of database interfaces, advanced retrieval algorithms, advanced explanation functionality, as well as the integration of recently developed techniques for learning similarity measures automatically [6]. Since myCBR is an open source project, contributions from other researchers for the improvement and extension of myCBR are of course also highly welcome.

Literatur

[1] R. Bergmann. On the Use of Taxonomies for Representing Case Features and Local Similarity Measures. In *Proceedings of the 6th German Workshop on Case-Based Reasoning (GWCBR'98)*, 1998.

[2] R. Bergmann. *Experience Management*. Springer, 2002.

[3] R. Bergmann, M. Michael Richter, S. Schmitt, A. Stahl, and I. Vollrath. Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning. In *Professionelles Wissensmanagement: Erfahrungen und Visionen. Proceedings of the 1st Conference on Professional Knowledge Management*. Shaker, 2001.

[4] R. Bergmann and A. Stahl. Similarity Measures for Object-Oriented Case Representations. In *Proceedings of the 4th European Workshop on Case-Based Reasoning (EWCBR'98)*. Springer, 1998.

[5] John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubézy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. The evolution of Protégé an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, 2003.

[6] Stahl, A. *Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning*, volume 986. dissertation.de, 2004.

[4] written in Java

[7] D. Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

Kontakt

Dr. Armin Stahl
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Trippstadter Straße 122, 67663 Kaiserslautern
e-mail: Armin.Stahl@dfki.de

Bild

Armin Stahl studied Computer Science at the University of Kaiserslautern with focus on AI. From 2000 to 2003 he worked as a scientific assistant in the research group "Artificial Intelligence - Knowledge-Based Systems" directed by Prof. Michael M. Richter where he finished his Ph.D. in the area of Case-Based Reasoning in 2003. Since 2004 he is the deputy director of the Image Understanding and Pattern Recognition Department at the German Research Center for Artificial Intelligence (DFKI).