Representing Similarity for CBR in XML

Lorcan Coyle, Dónal Doyle and Pádraig Cunningham

Department of Computer Science
Trinity College Dublin
{Lorcan.Coyle, Donal.Doyle, Padraig.Cunningham}@cs.tcd.ie

Abstract. As Case-Based Reasoning has matured as a discipline; the need for a standard means of representing case-based knowledge has come to the fore. While proposals exist for representing the vocabulary and the case-base knowledge containers, there are still no proposed standards for representing similarity or adaptation knowledge. In this paper we present extensions for representing similarity knowledge to CBML, an XML-based CBR language.

1 Introduction

Kitano and Shimazu have proposed that CBR applications have been too narrowly focused on domain specific problems [12]. They suggested that a CBR system should be viewed as a *medium* to be used in conjunction with the mainstream corporate information system. We share this perspective and anticipate that a standard way of marking up cases will facilitate this. The standard proposed for marking up structured, knowledge-rich data is XML. Our earlier work [3, 9, 10] described an XML-based case representation language called CBML (Case-Based Markup Language). Several other XML-based CBR systems have appeared over the past few years, e.g. [8, 15]. However, as Wilson has pointed out, the benefits that accrue to XML in general will not be fully passed on to the CBR community until a *standard* means of representing case data in XML is developed [18]. We propose that CBML has the capabilities to become such a standard.

Richter has identified four different ways in which knowledge can be represented in a Case-Based Reasoning (CBR) system [14]. He has named these *knowledge containers* and they have met wide acceptance as a natural organisation of knowledge in CBR. Richter's knowledge containers are:

- The vocabulary used,
- The similarity measure,
- The casebase,
- The solution transformation

Given the wide acceptance of this organisation of knowledge in CBR, it is perhaps surprising that attempts to create a representation language for CBR have concentrated for the most part on the vocabulary and casebase containers, e.g. [8, 11]. Our earlier work also focused on the vocabulary and casebase containers. This paper describes our more recent work on the representation of the similarity measure container. This representation is an extension of the CBML standard and allows the CBR developer to make the definition of similarity completely independent from the application code. Section 2 outlines the requirements for such a representation and describes our approach. Section 3 describes some advantages that we have observed in the fields of personalization, distributed CBR, explanation-based CBR and collaborative CBR.

2 Representation of Similarity

This section outlines the representation of similarity in CBML. It begins with a brief description of the feature types, documents the requirements for representing the traditional similarity function, and finishes with a description of how this is achieved in CBML. A number of example CBML fragments are used to illustrate the representation.

2.1 CBML – Cases and Case Structures

CBML was originally developed to facilitate distributed CBR and modular CBR objects. There were two CBR objects in CBML; the case content object and the case structure object. The case structure object defines the hierarchy and cardinality of the features that can appear in a case. Within the case structure, there are a number of feature structures that define the features that can appear in a case. These feature structures defines the feature's type, its value restrictions, and other attributes. The following are a list of the possible feature types and the restrictions that can be imposed on them:

- Symbolic – the feature value must be one of an enumerated list of possible values
- Numeric – can be integer or double type (ranges can be set)
- Boolean – the value can be either true or false
- String – the value can be any string
- Taxonomy – similar to symbolic type except that the possible values are represented with a tree structure

The case content document contains the casebase information in XML format. It must conform to the specifications laid out in the case structure document both in structure and content to be considered valid.

2.2 Similarity Measures

Traditionally, the similarity between a query, *Q* and a case, *C* is defined as the sum of the similarities of its constituent features multiplied by their relevance weights:

$$Sim(Q,C) = \sum_{f \in F} w_f \times \sigma_f(q_f, c_f) \qquad \mathbf{(1)}$$

Where w_f is the feature relevance weight and σ_f is the local similarity measure (i.e. feature specific similarity measure). In order to provide a representation of the similarity measure it is therefore necessary to represent both a relevance weight and a description of the local similarity measure for every feature. Weights are just attributes of features but local similarity measures are more complex. We have defined three types of local similarity measures:

- Exact similarity measures - similarity is 1 if the feature values are equal, otherwise it is 0.
- Difference based similarity measures – Similarity is directly related to the difference, δ between feature values. This measure is only suitable where a difference can be defined between feature values, e.g. with numeric features the difference is the mathematical difference.
- Complex Similarity Measures – Any similarity measure that is different to the above representations falls under this category. It is impossible to provide for a representation scheme that could cover every possible type of complex similarity measure. However, if the number of possible values is finite, it is possible to calculate and store similarity values for every combination of feature value in advance.

In order for a good representation scheme to work we need to define the above similarity measures formally. The exact similarity function is trivial; it depends on value matching. The complex similarity measure is impossible to define completely due to the infinity of possible measures. It is impossible to define a single difference function for string types, and trivial to define one for Boolean types so we will confine ourselves to the definition of numeric, symbolic and taxonomic differences.

Table 1. Table 1 shows the definitions for the difference functions for numeric (integer and double), symbolic and taxonomic feature types.

Feature Type	*Difference (δ) definition between Value1 and Value2*
Numeric	Value1 – Value2
Symbolic	Relative difference in the positions of Value1 and Value2 in the list of possible values.
Taxonomy	Number of branches between the Value1 and Value2's nodes in the taxonomy

With a definition of difference in place it remains to come up with an adequate definition of the relationship between difference and similarity. In the next section we show how a graph may be defined relating them.

2.3 Representation of Similarity in CBML

All CBML is represented using the XML language. A discussion of CBML case content and structure definitions is beyond the scope of this paper; it is documented more fully on the CBML website[1]. The CBML Schema is tightly defined by an XML-Schema document. We have developed a number of parsers that will read valid CBML and convert them into Java objects. Only similarity documents that conform to the CBML Schema will be considered valid CBML.

```
<feature name="Gender" weight="0.25">
  <exact/>
</feature>
```

Fig. 1. *Gender* is a symbolic feature used in the breathalyzer domain [4]. This figure shows the CBML definition of *Gender*'s similarity measure. It defines it as having a relevance weight of 0.25, and that it uses an exact similarity function.

The CBML representation of a similarity measure is a composite of feature specific similarity definitions. These feature similarity definitions have mandatory attributes defining the feature name (*name*) and relevance weight (*weight*). The relevance weight is simply an attribute called weight that can have any normalized double value. The remainder of this section describes each of the similarity types and illustrates each one with a CBML example. Fig 1 shows the representation of a feature that uses an exact similarity measure.

Fig 2 shows the representation of a difference function similarity measure. The parser looks at the feature type from the case structure to determine which type of difference function to use, i.e. numeric, symbolic or taxonomic. The similarity graph is defined here too. This graph may be symmetrical (the default) or asymmetrical. A symmetrical graph only deals with absolute difference values. By adding more points to the graph any piece-wise linear relationship between similarity and difference can be represented.

[1] http://www.cs.tcd.ie/research_groups/mlg/CBML/

```
<feature name="price" weight="0.2">
  <graph type="asymmetrical">
    <point difference="-Infinity" similarity="1"/>
    <point difference="0" similarity="1"/>
    <point difference="200" similarity="0"/>
  </graph>
</feature>
```

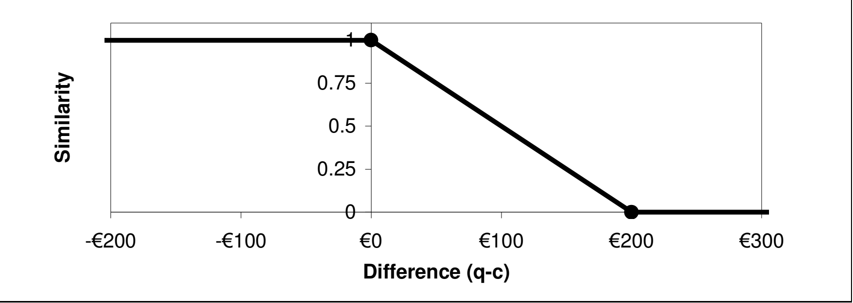

Fig. 2. *Price* is a numeric (double) feature used in the Personal Travel Assistant domain [1]. This figure shows the CBML definition of *Price*'s similarity measure. It defines it as having a relevance weight of 0.2. It then defines an asymmetrical graph of similarity versus difference by defining a number of points. For demonstrative purposes, this graph is also plotted. From the graph, the calculated similarity between two prices with a difference of €100 is 0.5.

Fig 3 shows the representation of a complex similarity measure. This is used when neither of the above representations is appropriate. It is used for similarity measures that cannot be represented easily in XML. Our parser uses this attribute to refer to a predefined Java class for the similarity measure definition, but other parsers could use it to refer to something else, e.g. a MathML [19] document. We have implemented an interface that all similarity measures must implement (this contains one function that takes in two features and returns the similarity value). This ensures a level of interoperability, but since these objects are not as portable as XML documents the use of this similarity definition is discouraged.

```
<feature name="sepal-length" weight="0.25">
  <measure name="iris.similarity.SepalLength"/>
</feature>
```

Fig. 3. *Sepal Length* is a numeric (double) feature used in Fisher's iris domain [7]. This code is the CBML definition of *Sepal Length*'s similarity measure. It defines it as having a relevance weight of 0.25. It also tells the CBML parser to use the class *iris.similarity.SepalLength* to calculate similarity.

The array similarity measure is the final way to represent similarity in CBML. It is useful for features with a finite number of possible values, but requires the user to pre-calculate the similarities in advance. If a value cannot be found in the array of similarities the exact similarity measure is used. Fig 4 shows a representation for the array similarity measure.

```
<feature name="meal" weight="0.05">
  <array>
    <primary name="none">
      <secondary name="snack" value="0.8"/>
      <secondary name="lunch" value="0.4"/>
    </primary>
    <primary name="snack">
      <secondary name="none" value="0.8"/>
      <secondary name="lunch" value="0.8"/>
      <secondary name="full" value="0.4"/>
    </primary>
    <primary name="lunch">
      <secondary name="none" value="0.4"/>
      <secondary name="lunch" value="0.8"/>
      <secondary name="full" value="0.8"/>
    </primary>
    <primary name="full">
      <secondary name="snack" value="0.4"/>
      <secondary name="lunch" value="0.8"/>
    </primary>
  </array>
</feature>
```

	none	*snack*	*lunch*	*full*
none	1	0.8	0.4	0
snack	0.8	1	0.8	0.4
lunch	0.4	0.8	1	0.8
full	0	0.4	0.8	1

Fig. 4. *Meal* is a symbolic feature used in the breathalyzer domain [4]. This code is the CBML definition of the similarity measure for *Meal*. It defines it as having a relevance weight of 0.05. It also contains an array of every possible feature value combination with a similarity value for each, e.g. the similarity between "none" and "snack" is 0.8. The array as defined is also shown.

3 Applications of CBML Similarity Measures

We have used CBML similarity measures in a number of research areas in the Machine Learning Group in Trinity College, most notably in our Personal Travel Assistant (PTA) application [1, 2] and in the explanation domain [4]. We have also developed a workbench that provides a set of utilities for CBR systems called Fionn[2] [5]. Fionn uses CBML as its internal representation format. This section describes some of

[2] http://www.cs.tcd.ie/research_groups/mlg/Fionn

the advantages we have observed from using CBML represented similarity measures in these applications.

The Personal Travel Assistant

The Personal Travel Assistant is an application that assists users in the booking and selection of flights. The main task for the PTA is the recommendation of suitable flights to the user using CBR. Fig 5 shows a diagram showing the flow of information through the PTA. The PTA recommendation engine uses Fionn components to make recommendations. The CBR functionality is all implemented as Fionn components, and because these components understand CBML we can switch them with ease.

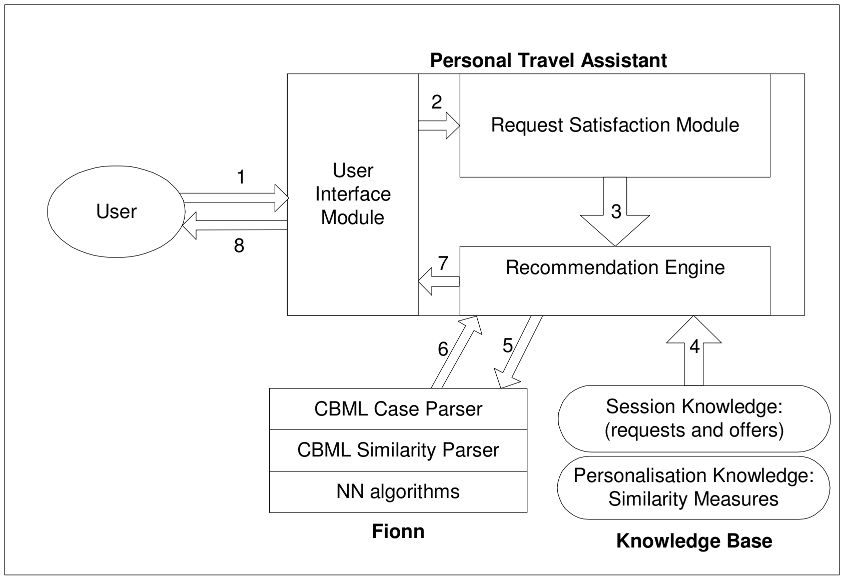
Personal Travel Assistant
User
1
8
User Interface Module
2
Request Satisfaction Module
3
7
Recommendation Engine
6
5
4
CBML Case Parser
CBML Similarity Parser
NN algorithms
Fionn
Session Knowledge: (requests and offers)
Personalisation Knowledge: Similarity Measures
Knowledge Base

Fig. 5. Fig 5 shows a diagram of the architecture of the PTA application. The user makes a travel request (1) and this request is forwarded to the *Request Satisfaction Module* (2). This module sends the set of new offers to the Recommendation Engine (3). The recommendation engine gets the relevant user profile information from the knowledge base, i.e. the user's personal casebase and similarity measure. It then uses the Fionn module to generate a sub-set of recommended offers (5, 6) and sends these to the User (7, 8).

To generate accurate recommendations we need to develop similarity measures that reflect the users travel preferences. Part of the personalization process involves updating these measures based on user feedback (this process is documented in more detail in [1]. By using CBML similarity measures we can do this easily and store personal similarity measures for each user (in the Personalisation Knowledge-base).

There are two ways in which we can learn a similarity measure. The first is by altering feature weights and the second by altering the local similarity measure in the manner proposed by Stahl [17]. By altering the feature weights, we adjust the relative importance of features for the user. We have experimented with altering the feature weights [1] and achieved positive results. We intend to assess the usefulness of altering the local measure for certain features in this domain over the coming months. In Fig. 2 we described the price similarity measure used in the PTA. By altering the position of the third point we can change this user's sensitivity to a difference in price, e.g. by changing the point from {200, 0} to {100, 0} we would focus the price similarity measure on offers with differences of less than €100.

Each user of the PTA system has a personalized set of similarity measure and casebase describing their travel preferences. Much work has been done in the area of collaborative CBR [13] and we intend to implement some of these techniques to address bootstrapping problems and problems with case competence [16] in general. To address these we will need to be able to modularize the components we intend to share. CBML facilitates this modularization.

The original motivation for the PTA was as a distributed CBR application along the lines of Gardingen & Watson's HVAC system [8] where the user would access the PTA via a fat client browser. One advantage of using CBML similarity measures in this context is that the expensive personalization calculations on the similarity measure could be calculated and stored in a central server and the relatively cheap recommendation process could be done on the client side. With such a distributed CBR system we could implement different recommender systems depending on the target platform. As long as these heterogeneous CBR systems understand CBML there will not be any problems with transferring the CBR objects from server to client.

Explanation in CBR

Previous research has shown that case-based explanation is more convincing than other types of explanation in classification problems [4]. However it also showed that there is scope for improvement in the quality of our explanations. Our current research uses a two stage process to generate explanations; the first performs a standard CBR classification, and the second uses an explanation function specific to this classification to determine the best case for explanation [6].

As the structure and requirements for our explanation utility functions are the same as those for the similarity measure so it was logical to reuse the definitions and so we use CBML similarity measures as the basis for our explanation functions. Due to their modularity the domain expert can update the different explanation measures individually and without needing to write any code. For most features, our explanation functions are closely based on the similarity functions used in the classification task.

4 Conclusions

Richter's knowledge containers have received wide acceptance in the CBR community. Our earlier work concentrated on defining a formal representation schema for the casebase and vocabulary containers called CBML. This work is concerned with the definition of a schema for representing similarity knowledge. This paper outlines the requirements for such a schema and describes our implementation. We have incorporated this schema into CBML.

Before outlining a representation for similarity measures it is important to first review what is possible to represent. The most common similarity measures are based on exact matching or on numeric differences so our focus has been to develop a compact, intuitive way of representing these. We have also catered for more complex difference-based measures, i.e. taxonomic and symbolic difference functions. Finally we made it possible to define similarity arrays and to refer to user-defined similarity measures. Section 2.3 shows examples of representations of each of these similarity types.

Our experiences with CBML in the Machine Learning Group have been positive. There are currently six researchers in our group using CBML similarity measures in several applications. In Section 3 we outlined some of the advantages we have observed from using CBML similarity measures in personalization, distributed CBR, collaborative CBR and explanation-based CBR.

As the current implementation of CBML can represent three of the four knowledge containers, it is clear that our next step should be the creation of a representation for the fourth container – the solution transform.

References

[1] Coyle, L., Cunningham, P (2004). Improving Recommendation Ranking by Learning Personal Feature Weights. To appear in the proceedings of the 7th European Conference on Case Based Reasoning, ECCBR 2004.

[2] Coyle, L., Cunningham, P. & Hayes, C. (2002). A Case-Based Personal Travel Assistant for Elaborating User Requirements and Assessing Offers. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (eds.). LNAI Vol. 2416 pp. 505-518, Springer-Verlag

[3] Coyle, L., Cunningham, P. & Hayes, C. (2002). Representing Cases for CBR in XML. Proceedings of 7th UKCBR Workshop, Peterhouse, Cambridge, UK.

[4] Cunningham, P., Doyle, D., Loughrey, J., An Evaluation of the Usefulness of Case-Based Explanation, 5th International Conference on Case-Based Reasoning. K. D. Ashley & D. G. Bridge (Eds.). LNAI 2689, pp122-130, Springer Verlag, 2003.

[5] Doyle, D., Loughrey, L., Nugent, C., Coyle, L., Cunningham, P., FIONN: A Framework for Developing CBR Systems. To appear in Expert Update 2004

[6] Doyle, D., Cunningham, P, Bridge, D., Rahman, Y. (2004) Explanation Orientated Retrieval. To appear in the proceedings of the 7th European Conference on Case Based Reasoning, ECCBR 2004.

[7] Fisher, R. A. (1936). "The Use of Multiple Measurements in Axonomic Problems," Annals of Eugenics 7, 179-188.

[8] Gardigen D., Watson I. (1998). A Web based Case-Based Reasoning System for HVAC Sales Support. Proceedings of British Expert Systems conference 1998

[9] Hayes, C. & Cunningham, P. (1999) Shaping a CBR View with XML. Proceedings of the Third International Conference on Case-Based Reasoning, ICCBR'99, Seeon Monastery, Germany. LNCS Vol. 1650. Althoff, K.-D., Bergmann, R., Branting, L.K. (Eds.) Springer-Verlag Berlin/Heidelberg 1999, pp.468-481

[10] Hayes, C., Cunningham, P., & Doyle, M.. Distributed CBR using XML. In Proceedings of the KI-98 Workshop on Intelligent Systems and Electronic Commerce, number LSA-98-03E. University of Kaiserslauten Computer Science Department, 1998

[11] INRECA consortium (1994). Casuel: A Common Case Representation Language, available at http://wwwagr.informatik.uni-kl.de/~bergmann/casuel/CASUEL_toc2.04.fm.html

[12] Kitano, H. & Shimazu, H. (1996) The Experience Sharing Architecture: A Case Study in Corporate-Wide Case-Based Software Quality Control. In Case-Based Reasoning: Experiences, Lessons & Future Directions. Leake, D.B. (Ed.) pp235-268. AAAI Press/ The MIT Press Menlo Park, Ca, US

[13] McGinty, L. & Smyth, B. (2001). Collaborative CBR for Real-World Route Planning. Proceedings of the 2001 International Conference on Artificial Intelligence (IC-AI'2001) Las Vegas, Nevada

[14] Richter, M. (1998) Introduction – the basic concepts of CBR. In M. Bartsch-Sporl, H. D. B., and Wess, S., eds., Case-Based Reasoning Technology: From Foundations to Applications, LNAI Vol. 1400, Springer-Verlag

[15] Shimazu, H. A Textual Case-Based Reasoning System Using XML on the World-Wide Web. Proceedings 4th European Workshop, EWCBR-98. Barry Smyth, Pádraig Cunningham (Eds.) LNCS 1488 pp274-285, Springer 1998

[16] B. Smyth and E. McKenna. Modelling the competence of case-bases. In B. Smyth and P. Cunningham, editors, Advances in Case-Based Reasoning:Proceedings of the Fourth European Workshop on Case-Based Reasoning, pages 196--207. Springer-Verlag, Berlin, Germany, Sept. 1998.

[17] Stahl, A., Gabel, T. (2003). Using Evolution Programs to Learn Local Similarity Measures. Proceedings of the 5th International Conference on Case-Based Reasoning, ICCBR 2003, Trondheim, Norway, June 2003. LNCS Vol. 2689, pp 537-551, Springer 2003.

[18] Wilson, D. (2001) Case-Base Maintenance: The Husbandry of Experience. PhD dissertation, Indiana University, 2001

[19] Mathematical Markup Language (MathML™) 1.01 Specification. Available online at http://www.w3.org/TR/REC-MathML/