Learning Similarity Measures: A Formal View Based on a Generalized CBR Model

Armin Stahl

German Research Center for Artificial Intelligence DFKI GmbH
Research Group Image Understanding and Pattern Recognition (IUPR)
Erwin-Schrödinger-Str. 57, 67663 Kaiserslautern, Germany
Armin.Stahl@dfki.de

Abstract. Although similarity measures play a crucial role in CBR applications, clear methodologies for defining them have not been developed yet. One approach to simplify the definition of similarity measures involves the use of machine learning techniques. In this paper we investigate important aspects of these approaches in order to support a more goal-directed choice and application of existing approaches and to initiate the development of new techniques. This investigation is based on a novel formal generalization of the classic CBR cycle, which allows a more suitable analysis of the requirements, goals, assumptions and restrictions that are relevant for learning similarity measures.

1 Introduction

The concept of similarity is certainly one of the most important and characteristic aspects of Case-Based Reasoning (CBR). In spite of the importance of similarity measures, clear methodologies for defining them efficiently and accurately are still missing. Instead, similarity measures are often defined in an ad hoc manner or one simply applies quite general distance metrics. When defining more complex measures that take account of domain knowledge, this is often done in an unstructured and not in a goal-directed fashion and often only experienced and skilled knowledge engineers are able to produce satisfactory results. Therefore, different machine learning approaches have been developed in order to facilitate the definition of similarity measures. However, the choice and application of an accurate learning approach is also a difficult task since one often is not aware of the actual requirements, goals, assumptions and restrictions of the application domain, the employed CBR system and the available learning techniques. Hence, learning is often performed in a trial-and-error fashion. Basically, when considering the application of learning techniques, some important questions have to be answered first, for example:

- What is the desired semantics of the similarity measure?
- What kind of training data is suitable and how can it be acquired?
- Which learning techniques are suitable to achieve best results?

Until now little or no work to clarify these questions and to provide a categorization of current learning approaches has been done. Only for learning feature weights in classification tasks such a categorization has been provided [19]. One problem when trying to answer the questions above is, that this requires a deeper understanding of the relationships between CBR functionality, application requirements, training data and available learning algorithms. In order to be able to analyze these relationships, a unified terminology and a certain degree of formality is mandatory. Unfortunately, the common CBR model [2] seems not to be suited to represent a good foundation because it is described rather informally and does not accurately model all important aspects.

Therefore, the goal of this paper is to provide a formal foundation and terminology for analyzing and categorizing approaches to learning similarity measures. Therefore, first a generalization and formalization of the classic CBR cycle is introduced in Section 2. An overview and first categorization of existing learning techniques is presented in Section 3. Finally, in Section 4 we examine some important issues for future research towards improved approaches for learning similarity measures.

2 A Formal Generalized Model for CBR

The classical CBR cycle introduced by Aamodt and Plaza [2], consisting of the four basic steps *retrieve*, *reuse*, *revise* and *retain*, is certainly the most established and accepted model for CBR. The success of this model may be explained by its simplicity and clarity, in particular for CBR novices. However, for describing and analyzing certain current research issues and popular application scenarios we argue, this classical model has some crucial limitations.

2.1 Limitations of the Classical CBR Cycle

In the following, we want to discuss some of the deficiencies of the classical CBR cycle in order to motivate the introduction of a more generalized model capturing also some of the current developments in CBR research.

CBR-Scenarios: Problem-Solving vs. Utility-Oriented Matching. One motivation of CBR was to imitate problem solving strategies of humans in order to enable computers to solve problems more efficiently. Hence, the traditional CBR cycle assumes a typical problem solving situation, i.e. the input—also called *query*—is expected to describe a problem and the output is expected to describe a corresponding solution suggestion. Typical application tasks that fit this assumption are classic problems of artificial intelligence such as classification, diagnosis, configuration or planning.

This assumption was the decisive factor for the structure of today's CBR systems, the underlying concepts, and the central paradigm of CBR: *"Similar problems have similar solutions"*. One quite important consequence of the problem

solving scenario is the traditionally used structure to described case knowledge. Here, a case is supposed to consist of the following two distinct parts:

Problem part: The problem part describes a particular problem situation of the past, e.g. in a diagnosis situation a set of symptoms and other relevant information about the entity under consideration.

Solution part: The solution part describes a corresponding solution successfully applied to solve the past problem, e.g. a correct diagnosis and a corresponding therapy. Although cases are usually supposed to contain only 'good' solutions, the solution part may contain any further information that might be useful when trying to reuse the solution, e.g information about the quality of the solution, justifications, explanations, etc.

In classical CBR applications, one is often interested only in the information contained in the solution part, whereas the problem part is used as an index to find useful solution information. However, in recent years CBR techniques have been applied very successfully to other application tasks that actually do not match this problem solving scenario. One important characteristic of such scenarios is a different case structure, where a clear distinction between a problem and a solution part is impossible. A typical example of such applications is product recommendation[8]. Here, queries represent requirements and wishes of customers with respect to desired products. Cases contain descriptions of available products and the task of the CBR system is to identify particular products that are most suitable to fulfill the given customer demands.

In principle this task could be solved in the traditional case-based manner. Therefore, one would have to store customer queries of the past—representing the problems—together with the description of successfully sold products—representing the solutions. Here, it would be sufficient to store only a product-ID to describe products uniquely. New customer queries then could be compared with customer queries of the past using similarity measures in order to select products that probably will also be bought by current customers.

However, most case-based product recommendation systems follow a different approach. Here, a case typically consists of a detailed description of an available product solely. In order to select suitable products, a customer query is compared with these product descriptions by applying an accurate similarity measure.

The product description can be interpreted as the solution part of traditional cases but the traditional problem part (here this would be a past customer query) is missing completely. Hence, such systems compare problems, namely current customer queries, directly with solutions, namely product descriptions. This procedure does not really comply with the traditional idea of CBR. Instead, it may be characterized as *utility-oriented matching* [3] because one tries to estimate the utility of a solution for a given problem more or less directly. Similar situations also occur in other applications scenarios, for example, in the area of Knowledge Management. Most of those scenarios have in common that they may be seen more as intelligent information retrieval than actual problem solving.

Advanced CBR Techniques. Another limitation of the traditional CBR cycle is that it does not consider some crucial aspects and issues of current CBR systems sufficiently which have come into the focus of research just recently. Some quite important of those issues are for example:

Dialog Strategies: The traditional CBR cycle assumes a formalized query given as input prior to the actual reasoning process without considering how this query can be obtained. However, the efficient acquisition of an accurate query is a crucial issue in diagnosis tasks and has also come into focus of research in the area of product recommendation systems recently [14].

Explanation: A popular topic of current CBR research is explanation [1]. However, the traditional CBR cycle does not explicitly consider the generation of explanations about presented solutions or the underlying reasoning process.

Feedback: An important characteristic of the traditional CBR cycle is the possibility to learn new cases during the retain phase. Although Aamodt and Plaza have mentioned the possibility to learn not only cases but also general knowledge (e.g. refining indexes), the traditional CBR cycle does not explicitly introduce a feedback loop which is required to realize advanced learning approaches.

2.2 A Formal Generalization of the Classical CBR Cycle

In this section we introduce a more general and more formal model for CBR. This model aims to avoid some of the deficiencies of the classical CBR cycle. Although it does not capture all aspects of current CBR research, at least it represents a foundation for analyzing certain CBR functionality in more detail. Our main goal is to introduce a formalism that can be used to examine important aspects to be considered when developing approaches for learning similarity measures. In the future the model may be extended to describe other, still disregarded CBR aspects. An illustration of the model is shown in Fig. 1.

The starting point is a given informal *situation* s in the application environment which triggers some more or less abstract information need. The task of a CBR system is to provide the necessary information by generating a corresponding output o. For example, in the traditional problem solving scenario, s is an unsolved problem for which a solution is required and o may be the description of suitable solution or a solution method, respectively. In a first step the situation s has to be described formally in order to obtain a query q that can be processed by the CBR system:

Definition 1 (Situation-Characterization, Query). *A situation characterization* $sc : \mathcal{S} \rightarrow \mathcal{Q}$ *where* $\mathcal{S}$ *is the situations space and* $\mathcal{Q}$ *is the query space, characterizes the informal situation* s *formally through query* $q = sc(s)$*. The set of all situation characterizations is denoted by* $\mathcal{SC}$*.*

In practice, sc implements certain transactions between the application environment and the CBR system. In the simplest case it might import query data

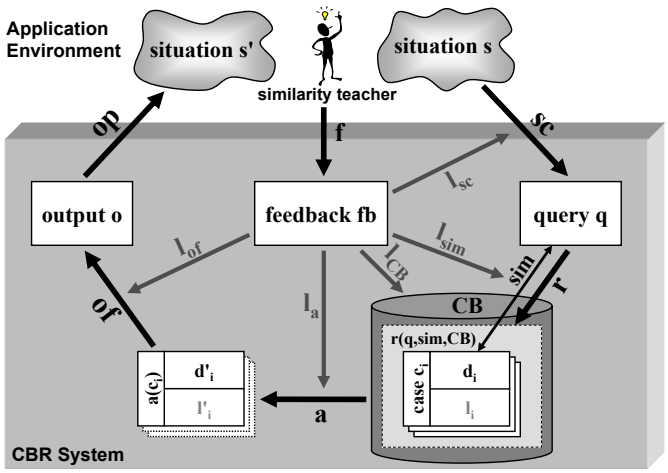

Fig. 1. A Formal Model for CBR

from some data source but usually the query will be acquired from the user, for example, by providing a query form or by performing an elaborate dialog [14].

In the next step, q has to be compared with cases in the case base in order to select cases that are expected to contain information that is useful for satisfying the information need of s.

Definition 2 (Case, Case Characterization, Case Lesson, Case Space). *A case c is a tuple $(d, l) \in \mathcal{D} \times (\mathcal{L} \cup \emptyset)$ where d is called a case characterization and l is called a case lesson. $\mathcal{D}$ and $\mathcal{L}$ are the corresponding spaces of case characterizations and case lessons. $\mathcal{C} = \mathcal{D} \times \mathcal{L}$ is called the case space and the set of available cases $CB = \{c_1, \ldots, c_m \mid c_i \in \mathcal{C}\}$ is called the case base.*

In our model we explicitly allow empty lesson parts, i.e. a case may consist of a characterization only. It is important to note, that case characterizations have not necessarily to represent problem descriptions but any information that is useful to estimate the utility of cases. This means, that cases may also be characterized by using solution information.

Definition 3 (Similarity Measure). *A similarity measure is a function $sim : \mathcal{Q} \times \mathcal{D} \rightarrow [0, 1]$. To simplify the notation we write $sim(q, c)$ instead of $sim(q, d)$ for representing the similarity between a query q and a case $c = (d, l)$. The set of all similarity measures is denoted by $\mathcal{SIM}$.*

Definition 4 (Retrieval Function). *A retrieval function $r : \mathcal{Q} \times \mathcal{SIM} \times \mathcal{P}(CB) \rightarrow \mathcal{P}(CB)$ returns a subset of the case-base CB for a given query q according to a given similarity measure $sim \in \mathcal{SIM}$. The returned cases $c_r \in r(q, sim, CB)$ are assumed to be ordered w.r.t. to their corresponding similarity values $sim(q, c_r)$.*

We do not make any assumptions about the realization of r, e.g. it might simply return the most similar case, i.e. $r(q, sim, CB) = \arg\max_{c_i \in CB} sim(q, c_i)$. After having retrieved a set of cases, the information contained in the retrieved cases may be adapted in order to construct a new, more accurate case:

Definition 5 (Adaptation Function). *An adaptation function* $a : \mathcal{Q} \times \mathcal{P}(CB) \rightarrow \mathcal{C}^k$ *generates a set of new cases* $\{c_{a_1}, \ldots, c_{a_k}\}$ *given a set of input cases* $\{c_1, \ldots, c_n\}$ *with* $c_i \in CB$, $n, k \geq 1$ *and a query* q. *The set of all adaptation functions is denoted by* $\mathcal{A}$.

Typically it holds $k \leq n$. If a single adapted case c_{a_i} is constructed from several input cases, this is called *compositional adaptation*. A simple example are voting policies like those applied in k-nearest-neighbor classification. In systems without adaptation, a is considered to be the identity function with respect to the input cases. The result of the adaptation process is used as source information for generating the final output of the CBR system:

Definition 6 (Output Function, Output Space). *Given a query* q *and set of cases* $\{c_1, \ldots, c_n\}$, *the output function of* $: \mathcal{Q} \times \mathcal{P}(C) \rightarrow \mathcal{O}$ *generates an output* $o = of(q, c_1, \ldots, c_n)$ *where* $\mathcal{O}$ *is the space of outputs.* $\mathcal{OF}$ *denotes the set of all output functions.*

In principle one might put a lot of 'intelligence' into the output function, but in practice the output function typically is used

- to select appropriate cases to be returned to the application environment, e.g. to ensure the right degree of diversity,
- to extract the required information from the given cases, e.g. class labels,
- to generate additional explanations in order to explain the result of the CBR system to the users.

The resulting output then is returned to the application environment in order to satisfy the information need of the initial situation s:

Definition 7 (Output Processing Function). *The output processing function* $op : \mathcal{S} \times \mathcal{O} \rightarrow \mathcal{S}$ *generates a new situation* $s' = op(s, o)$ *by applying the output* o *to situation* s *within the application environment.*

In practice, the output processing function typically is an informal process which is executed within the application environment with little or no support from the CBR system. For example, a suggested therapy in a medical diagnosis situation will be applied by a doctor where the new situation s' will be a modified state of health of the patient. If s' is still associated with an unsatisfied information need, it might be used as a new initial situation for executing the cycle again.

For enabling a CBR system to improve its performance by applying learning strategies it must receive feedback from the application domain about the actual usefulness of its output:

Definition 8 (Feedback Function). *The feedback function* $f : \mathcal{S} \times \mathcal{O} \rightarrow \mathcal{F}$ *evaluates the usefulness of output* o *for situation* s *and returns some feedback* $fb = f(s, o)$*, where* $\mathcal{F}$ *is called the feedback space.*

Here, we do not assume a particular form of feedback but in Section 3.2 we will discuss this issue with respect to learning similarity measures in more detail. Feedback may be used by the CBR system to improve its functionality by modifying one or several of its knowledge containers [13]:

Definition 9 (Learning Functions). *The following functions allow to modify the case base* CB*, the similarity measure* sim*, and the adaptation function* a *w.r.t. given feedback:*

$$l_{CB} : \mathcal{Q} \times \mathcal{O} \times \mathcal{F} \times \mathcal{P}(C) \rightarrow \mathcal{P}(C)$$

$$l_{sim} : \mathcal{F} \times \mathcal{SIM} \rightarrow \mathcal{SIM}$$

$$l_a : \mathcal{F} \times \mathcal{A} \rightarrow \mathcal{A}$$

The function l_{CB} realizes the traditional idea of learning in CBR systems, namely a modification of the case base, e.g. by storing new or deleting obsolete cases. While l_{sim} and l_a allow to learn general knowledge already considered in the classical CBR cycle, one might also introduce similar learning functions, e.g. l_{sc} for improving dialog strategies [14] or l_{of} for improving the generation of explanations.

2.3 Advantages of the Generalized CBR Model

In principle, our general model can be divided into the same phases as the classical CBR cycle: The functions sc and r implement the retrieval phase, the functions a and of implement the reuse phase, the functions op and f represent the revise phase and the learning functions l_x implement the retain phase.

However, by abstracting from the traditional problem solving scenario, the model is more suitable to describe popular 'modern' application scenarios such as product recommendation. For example, we do not assume that a case consists of a problem and a solution part. Instead, cases may only consist of case characterizations that may describe arbitrary information. This also means that queries and case characterizations do not necessarily have the same semantics, for example, they do not both represent problem descriptions. We will discuss this issue again in Section 4.

By introducing additional processes in the form of the situation characterization and the output function, our model can also be used to describe new research directions such as dialog strategies and explanations more accurately than possible with the classical CBR cycle. Moreover, by introducing the feedback function and a set of learning functions, it enables a better description of advanced learning approaches beyond storing of new cases.

However, the model in its current version is not intended to capture all aspects of any CBR application. For example, more complex dialog strategies that

involve case retrieval cannot be described exclusively with *sc* but require a repeated execution of the entire cycle. Nevertheless, the model may represent a good foundation to be extended for explaining other functionality of CBR systems.

2.4 The Goal of a CBR System

Before we use the introduced model to analyze the task of learning similarity measures, first we will discuss some important general consequences of it.

A CBR system's goal is to generate an output o that is maximally useful for satisfying the information need of a given situation s, i.e. it should help to reach a new, improved situation s' by exploiting the information contained in o. In a formal view, an optimal CBR system should realize the following goal function:

Definition 10 (Goal Function, Utility Function). *The goal function* $g : S \rightarrow O$ *generates an output* o *that is maximally useful for a given situation* s*, i.e.* $g(s) := \arg\max_{o \in \mathcal{O}} u(s, o)$*, where* $u : \mathcal{S} \times \mathcal{O} \rightarrow \mathbb{R}$ *is the domain specific utility function.*

In practice u is usually only implicitly and informally defined within the application environment. However, during the lifetime of a CBR system certain information about u may be provided by the feedback function f. Depending on the application scenario, u may be influenced in many ways, e.g., by

- the correctness of suggested solutions,
- the outputs' degree of applicability or reusability,
- the satisfaction of the user (e.g. a customer) or
- the output's information gain for the user.

The basic idea of a CBR system is to acquire and encode knowledge about u by using different knowledge containers [13], namely the *vocabulary*, the *case base*, the *similarity measure* and the *adaptation knowledge*[1]. The vocabulary defines the important aspects required to describe situations, cases, outputs and feedback, i.e. it determines Q, C, O and F. Traditionally, the cases represent known points of u corresponding to a maximal or at least high utility and adaptation knowledge defines knowledge about additional points or certain subspaces of u. Finally, the similarity measure should encode knowledge about the relationships between different points of the input space of u. However, due to the difficulty of acquiring this knowledge, the employed similarity measures often only represent quite simple heuristics about the typically expected shape and smoothness of u.

In order to facilitate the acquisition of similarity knowledge and the definition of more accurate similarity measures, several learning approaches have been developed, e.g. see [19, 16]. In the following we investigate important general issues of such learning approaches in more detail on the basis of the previously introduced formal CBR model.

[1] In our formal model represented through the adaptation function a.

3 Learning Similarity Measures: A Formal Analysis

CBR systems often generate output that is composed of a set of independent output alternatives. This functionality is typically desired when presenting the output to human users, e.g. alternative products to customers. Here, we assume that only single retrieved cases $c_r \in r(q, sim, CB)$ are adapted and used to generate a single output alternative o_r. This means, first we do not consider compositional adaptation. The entire output then is an ordered collection of alternative outputs $o = (o_1, o_2, \ldots, o_k)$, where the order is based on the computed similarities, i.e. it holds $\forall\ 1 \leq i < j \leq k\ sim(q, c_i) \geq sim(q, c_j)$. We assume that the utility of o only depends on the sum of the o_r's utilities and their ranking[2]. According to our formal CBR model the utility of an output alternative o_r is defined as

$$u(s, o_r) = u(s, of(q, a(q, c_r))) = u(s, of(sc(s), a(sc(s), c_r)))$$

Moreover, we assume that $u(s, o_r)$ can be expressed by numbers of the interval $[0, 1]$, i.e. it holds $u : S \times O \rightarrow [0, 1]$ where a value of 1 represents the maximal possible and 0 represents the minimal possible utility. From now on, we assume that a and of are static, i.e. that the adaptation and output function are not modified during the lifetime of the CBR system.

3.1 Semantic of Similarity Measures

In general, the basic task of a similarity measure is to estimate the a-posteriori utility of a given case c_r, i.e. in the best case $sim(q, c_r)$ should approximate the a-priori unknown utility $u(s, of(q, a(q, c_r)))$ as closely as possible. This would obviously require that sim is completely informed about the remaining parts of the CBR system, namely the functions a and of as well as about the external utility function u. In practice this ideal property of sim usually cannot be achieved, and hence sim represents a more or less well informed heuristic only.

Retrieval Requirements. Before defining a similarity measure for a particular CBR application one should be aware of the application specific requirements on the expected output. Basically, a similarity measure should help to realize the goal function g, i.e. to maximize the utility of the output $u(s, o)$. According to our assumptions on o we can deduce different criteria that an optimal similarity measure sim_o should fulfill, namely:

Determining the Most Useful Case: In certain application scenarios, in particular when processing the output within the application environment automatically, only a single output alternative is of interest, i.e. $o = \{o_1\}$. Then it should hold:

$$\arg\max_{c_r \in CB} sim_o(q, c_r) = \arg\max_{c_r \in CB} u(s, o_r)$$

[2] This assumption does not hold in some application scenarios, e.g. if a certain diversity of the output alternatives is desired.

Separating Useful and Useless Cases: Often the utility of output alternatives is of a binary nature, i.e. an o_r may be useful or completely useless. In some application scenarios binary output utility can be achieved by introducing artificial utility thresholds, e.g. in information retrieval, the retrieved documents are simply treated as 'relevant' or 'irrelevant'. In such situations we may demand the following from sim_o: Let $CB^+ = \{c_i \in CB \mid u(s, o_i) \geq \theta\}$ be the set of useful and $CB^- = \{c_i \in CB \mid u(s, o_i) < \theta\}$ be the set of useless cases, then

$$\forall c_i \in CB^+, c_j \in CB^- : sim_o(q, c_i) > sim_o(q, c_j)$$

Ranking the Most Useful Cases: Let $CB^u = \{c_i \in CB \mid u(s, o_i) \geq \sigma\}$ be the set of most useful cases. One may demand that sim_o ranks these cases correctly:

$$\forall c_i, c_j \in CB^u, \forall c \in CB \backslash CB^u : \begin{array}{l} sim_o(q, c_i) > sim_o(q, c_j) \Leftrightarrow u(s, o_i) > u(s, o_j) \\ \wedge\ sim_o(q, c_i) > sim_o(q, c) \end{array}$$

Approximating the Utility of the Most Useful Cases: Although in most present CBR applications a good approximation of the cases' absolute utility is not the main goal when defining sim, such a requirement would help the user to judge the reliability of each presented o_r:

$$\forall c_r \in CB^u : sim_o(q, c_r) \simeq u(s, o_r)$$

The first three criteria only demand that the similarity measure partially reproduces the preference relation induced by the utility function, i.e. one is only interested in an estimate of the cases' *relative utility* with respect to other cases. The last requirement is stronger since it requires an approximation of the cases' *absolute utility*.

Probabilistic Similarity Measures. Up to now we have implicitly assumed, that it is possible, at least in principle, to compute the utility $u(s, o_r)$ given only q and c_r. However, in practice this often does not apply because one is confronted with incomplete and/or noisy data or non-deterministic domains and hence with uncertainty. For example, queries as well as case characterizations often do not contain all information required to describe the underlying situations and cases sufficiently. In such situations a probabilistic interpretation of similarity values seems to be more accurate, i.e. the value $sim(q, c_r)$ then may be interpreted as the probability that the resulting output o_r is maximally useful given q and c_r, i.e. $sim(q, c_r) := P(u(s, o_r) = 1 \mid q, c_r)$. Nevertheless, this interpretation is consistent with the previously discussed demands on sim_o as well.

3.2 Training Data

When thinking about developing or applying an approach for learning similarity measures, one of the most crucial issues is the quality and the amount of

available training data. When being confronted with little and noisy training data, many learning techniques tend to overfit the training data resulting in poor generalization performance.

In principle, the training data must contain some implicit or explicit knowledge about the a-posteriori utility of certain cases. This means for a case c_r and a given query q certain information about $u(s, o_r)$ is required. According to our formal CBR model we assume that such information can be obtained via the feedback function f either *offline* during a particular training phase or *online* during the application of the CBR system. In the following we discuss different types of such *utility feedback*.

Utility Feedback. Basically, information about the a-posteriori utility $u(s, o_r)$ of a case c_r given a query q may be provided in different ways:

Absolute Case Utility Feedback (ACUF): One possibility is to provide information about the absolute value of $u(s, o_r)$. Here, the feedback space F is defined as $(Q \times C \times [0, 1])^n$. This means feedback fb consists of a collection of *training examples* $fb = (te_{11}, \ldots, te_{lk})$ where a single training example $te_{ij} = (q_i, c_j, u(s, o_j))$ represents utility feedback for a particular case c_j w.r.t. a given query q_i.

Absolute Utility Feedback (AUF): When allowing compositional adaptation, i.e. $o = of(q, a(q, c_1, \ldots, c_n))$, a special kind of absolute utility feedback can be acquired. In this situation, the utility of o cannot simply be traced back onto the utility of individual cases. Then F is defined as $\mathcal{Q} \times \mathcal{O} \times [0, 1]$ where corresponding training examples $te = (q, o, u(s, o))$ represent information about the performance of the entire CBR system for a given query q.

Relative Case Utility Feedback (RCUF): Another possibility is to provide information about $u(s, o_r)$ only in a relative manner with respect to other output alternatives. By defining F as $(\mathcal{Q} \times \mathcal{C} \times \mathcal{C} \times \mathcal{UR})^n$ where $\mathcal{UR}$ represents a set of relation symbols (e.g. $\mathcal{UR} = \{<, \leq, =, \geq, >, \neq\}$) a training example can be represented as a tuple $te = (q, c_i, c_j, R)$ where $u(s, o_i) R u(s, o_j)$ for some $R \in \mathcal{UR}$.

Absolute feedback (ACUF/AUF) is mandatory for learning similarity measures that are intended to approximate absolute utility values. When only focusing on a reproduction of the induced preference relation, RCUF feedback is sufficient. However, depending on the desired semantic one should acquire feedback for different cases.

Acquisition of Training Data. Now we describe how the introduced kinds of feedback can be acquired in practice, i.e. how to implement the feedback function f. Basically, two different approaches are possible.

The first approach is *self-optimization*. In traditional problem solving scenarios, i.e. if case characterizations d_i describe past (problem) situations s_i and

case lessons l_i represent corresponding outputs (typically solutions) with high utility, a CBR system is able to extract training data from its case base CB. On the one hand, cases themselves can be interpreted as ACUF where each case c_i represents a training example $te = (d_i, c_i, u(s_i, o_i))$. Information about $u(s_i, o_i)$ may be contained in case lessons or $u(s_i, o_i) = 1$ is assumed.

On the other hand, additional feedback can be obtained by performing a *leave-one-out-crossvalidation*, i.e. single cases c_i are temporarily removed from CB and d_i is used as query q_i. The resulting output o (or o_r) then has to be compared with an output l_i known to have high utility (mostly $u(s_i, l_i) = 1$ is assumed). Depending on the implementation of a the corresponding feedback is typically of the kind ACUF or AUF.

Self optimization is applied by most existing approaches for learning similarity measures, typically for feature weight learning in classification scenarios (see Section 3.3). Here, training examples are simply defined as $te_{ACUF} = (d_i, c_r, u(s_i, o_r))$ or $te_{AUF} = (d_i, o, u(s_i, o))$, respectively, where $u(s_i, o_{(r)}) = 1 \Leftrightarrow o_{(r)} = l_i$ (i.e. if the classification is correct) and $u(s_i, o_{(r)}) = 0$ otherwise. In [18] we have proposed a generalization of this approach where we set $u(s_i, o_{(r)}) = sim_S(o_{(r)}, l_i)$, i.e. we employ a domain specific *solution similarity measure* $sim_S : \mathcal{O} \times \mathcal{L} \rightarrow [0, 1]$ in order to estimate the utility of the generated output in non-classification domains or when misclassification costs [20] have to be considered.

An approach to utilizing self optimization in the utility-oriented matching scenario by generating RCUF is described in [17, 16]. Here, the influence of the adaptation function a on the target similarity measure sim_T is estimated by evaluating the utility of adapted cases with a given utility measure represented by an additional similarity measure sim_U that can be defined more easily than sim_T.

The second approach for acquiring training data is to *ask some similarity teacher*. In the utility-oriented matching scenario an extraction of training data from the case base is usually impossible because here the cases do not contain information about u. For example, pure descriptions of technical products contain no explicit knowledge about their suitability for particular customer demands[3]. Therefore, utility feedback has to be provided by an external *similarity teacher* who possesses certain knowledge about u. In principle, the previously mentioned measures sim_S and sim_U as well as external simulation procedures might be interpreted as artificial similarity teachers. However, often only human domain experts or the system's users are able to provide the required feedback, but only a few learning approaches consider human similarity teachers [6, 9, 21].

3.3 Learning Techniques

In this section we give an overview on techniques that have been applied for learning similarity measures in CBR. The following aspects may be used to categorize the techniques:

[3] In current CBR applications this knowledge is often inferred by applying simple distance metrics.

- the desired semantic of the target similarity measure (cf. Section 3.1)
- the type of the training data and the corresponding approach to acquisition (cf. Section 3.2)
- the representation of the similarity measure to be learned
- the applied learning algorithm
- whether background knowledge is used to improve the learning process

Basically, the representations used to model similarity measures determine the hypothesis space $\mathcal{SIM}$. Here, we can distinguish the following commonly applied approaches:

Feature Weights: Because in many CBR systems only simple weighted distance metrics are employed, modifying the weights assigned to features in feature-value based case representations is often the only possibility to influence the similarity measure [19]. Here, one also distinguishes between global and local (e.g. case specific) weighting methods.

Local Similarity Measures: Most commercial CBR tools allow us to define local similarity measures for each feature in order to be able to incorporate more domain specific knowledge. Suitable learning techniques must be able to learn the particular parameters used to describe such local similarity measures [16, 17].

Probabilistic Similarity Models (PSM): Another possibility to represent similarity measures are probabilistic models. Here, the similarity function is encoded using probability distributions which have to be determined by using appropriate techniques (e.g. frequency counts, kernel estimation techniques, neural networks, etc.) [7, 4].

For characterizing learning techniques, Wettschereck and Aha [19] have introduced the following categorization:

Incremental Hill-climbers: Here, single training examples (typically based on ACUF or AUF) trigger the modification of the similarity measure after each pass through the CBR cycle. Existing approaches [5] increase or decrease feature weights in classification scenarios, where success driven ($te = (q, c_r, 1)$) and failure driven ($te = (q, c_r, 0)$) policies can be distinguished.

Continuous Optimizers: The idea of continuous optimizers is to collect a sufficiently large training data set first and to apply optimization approaches afterwards in order to generate a similarity measure that shows optimal results on this training data.

Typically, this is realized by minimizing a particular error function which compares generated outputs with corresponding utility feedback contained in the training data. For learning feature weights, gradient descent approaches have shown good results [15, 19, 20]. While most existing approaches apply ACUF or AUF, we have proposed an approach that utilizes RCUF in order to enable learning in the utility-oriented matching scenario [16]. For more

complex local similarity measures we have developed a corresponding evolutionary algorithm [17, 16].

PSM are usually also learnt by applying continuous optimizers which either optimize probabilistic error functions [12] or estimate underlying probability distributions by applying statistical and Bayesian methods [7].

Ignorant Methods: These methods do not exploit explicit feedback, but only perform a statistical analysis of the ACUF contained in CB, for example, to determine accurate feature weights based on class distributions [4].

Concerning the incorporation of background knowledge into the learning process, few approaches have been developed so far. Approaches that use background knowledge in order to improve the performance of an evolutionary algorithm have been presented in [11, 10].

4 Conclusions and Future Work

In the first part of this paper we have presented a novel formal generalization of the classical CBR cycle. The advantages of this model are its generality, allowing us to describe recent developments in CBR research more accurately, and its formality, allowing more detailed analyses of important research issues. In the second part we have used the novel model to analyze crucial questions concerning the development of approaches for learning similarity measures. On the one hand, this analysis allows us to categorize existing learning techniques in order to simplify the choice of accurate techniques in particular applications. On the other hand, it represent a good foundation for future research.

While traditional approaches towards learning similarity measures in CBR mainly focus on learning of feature weights by employing ACUF/AUF, recently developed approaches also allow to employ RCUF which can be acquired in non-classification scenarios more easily than ACUF/AUF. Moreover, these approaches also enable learning of complex local similarity measures.

For future research we intend to develop new approaches towards the application of PSM. In our view, PSM have some advantages compared with explicit models (e.g. feature weights, local similarity measures). On the one hand, they may allow to weaken the hard attribute independence assumptions underlying common representations. Moreover, they would allow the definition of similarity measures in utility-oriented matching scenarios where it might hold: $\mathcal{Q} \neq \mathcal{D}$. For example, this would allow to compute 'similarities' between abstract queries (e.g. "I want a PC suited for video processing") and precise product descriptions (e.g. $\mathsf{HD\text{-}Size} = \mathsf{200GB}$). However, existing learning approaches for PSM are only applicable in classification scenarios. To employ PSM in other scenarios we plan to develop techniques to learn PSM from RCUF. Moreover, we want to investigate how to incorporate background knowledge efficiently. Last but not least we want to develop new techniques that aim to learn similarity measures that approximate the absolute utility values as closely as possible. This would allow to build more dependable CBR systems because the user would get information about the reliability of the presented output.

References

1. Proceedings of the ECCBR-2004 Workshop on Explanation, 2004.
2. Aamodt, A., Plaza, E. Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994.
3. Bergmann, R., Richter, M.M., Schmitt, S., Stahl, A., Vollrath, I. Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning. In *Proceedings of the 1st Conference on Professional Knowledge Management*. Shaker, 2001.
4. Blanzieri, E., Ricci, F. Probability Based Metrics for Nearest Neighbor Classification and Case-Based Reasoning. In *Proceedings of the 3rd International Conference on Case-Based Reasoning*. Springer, 1999.
5. Bonzano, A., Cunningham, P., Smyth, B. Using Introspective Learning to Improve Retrieval in CBR: A Case Study in Air Traffic Control. In *Proceedings of the 2nd International Conference on Case-Based Reasoning*. Springer, 1997.
6. Branting, K. Acquiring Customer Preferences from Return-Set Selections. In *Proceedings of the 4th International Conference on CBR*. Springer, 2001.
7. Breuel, T. Character Recognition by Adaptive Statistical Similarity. In *Proceedings of the 7th Int. Conf. on Document Analysis and Recognition*. Springer, 2003.
8. Burke, R. The Wasabi Personal Shopper: A Case-Based Recommender System. In *Proceedings of the 11th International Conference on Innovative Applications of Artificial Intelligence*, 1999.
9. Coyle, L., Cunningham, P. Exploiting Re-ranking Information in a Case-Based Personal Travel Assistent. In *Workshop on Mixed-Initiative Case-Based Reasoning at the 5th International Conference on Case-Based Reasoning*. Springer, 2003.
10. Gabel, T. On the Use of Vocabulary Knowledge for Learning Similarity Measures. In *Proceedings of the 3rd German Workshop on Experience Management*. Springer, 2005.
11. Gabel, T., Stahl, A. Exploiting Background Knowledge when Learning Similarity Measures. In *Proceedings of the 7th European Conference on Case-Based Reasoning*. Springer, 2004.
12. Lowe, D. Similarity Metric Learning for a Variable-Kernel Classifier. *Neural Computation*, 7, 1993.
13. Richter, M. M. The Knowledge Contained in Similarity Measures. Invited Talk at ICCBR'95, 1995.
14. Schmitt, S. *Dialog Tailoring for Similarity-Based Electronic Commerce Systems*. dissertation.de, 2003.
15. Stahl, A. Learning Feature Weights from Case Order Feedback. In *Proceedings of the 4th International Conference on Case-Based Reasoning*. Springer, 2001.
16. Stahl, A. *Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning*, volume 986. dissertation.de, 2004.
17. Stahl, A., Gabel, T. Using Evolution Programs to Learn Local Similarity Measures. In *Proceedings of the 5th International Conference on CBR*. Springer, 2003.
18. Stahl, A., Schmitt, S. Optimizing Retrieval in CBR by Introducing Solution Similarity. In *Proceedings of the Int. Conf. on AI*. CSREA Press, 2002.
19. Wettschereck, D., Aha, D. W. Weighting Features. In *Proceeding of the 1st International Conference on Case-Based Reasoning*. Springer, 1995.
20. Wilke, W., Bergmann, R. Considering Decision Cost During Learning of Feature Weights. In *Proceedings of the 3rd European Workshop on CBR*. Springer, 1996.
21. Zhang, Z., Yang, Q. Dynamic Refinement of Feature Weights Using Quantitative Introspective Learning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1999.