A new Retrieval Function for Ontology-Based Complex Case Descriptions

Hans Friedrich Witschel[1], Andreas Martin[1,2],
Sandro Emmenegger[1], Jonas Lutz[1]

[1]University of Applied Sciences Northwestern Switzerland (FHNW),
Institute for Business Information Systems,
Riggenbachstr. 16, 4600 Olten, Switzerland
[2]University of South Africa,
School of Computing,
P O Box 392, Unisa 0003, South Africa
{hansfriedrich.witschel, andreas.martin,
sandro.emmenegger, jonas.lutz}@fhnw.ch

Abstract This work focuses on case-based reasoning in domains where cases have complex structures with relationships to an arbitrary number of other (potentially complex and structured) entities and where case characterisations (queries) are potentially incomplete. We summarise the requirements for such domains in terms of case representation and retrieval functions. We then analyse properties of existing similarity measures used in CBR – above all symmetry – and argue that some of these properties are not desirable. By exploiting analogies with retrieval functions in the area of information retrieval – where similar functions have been replaced by new ones not exhibiting the aforementioned undesired properties – we derive a new asymmetric ranking function for case retrieval. On a generated test-bed, we show that indeed the new function results in different ranking of cases – and use testbed examples to illustrate why this is desirable from a user's perspective.

1 Introduction

Case-based reasoning (CBR) addresses the problem of knowledge re-use by storing solutions to problems in a so-called case base and making such historical knowledge accessible through a retrieval interface. The retrieval is at the heart of any case-based reasoning (CBR) system: here, a knowledge worker who is trying to solve a new case, searches for similar cases from the past in order to re-use the knowledge contained in them. The retrieval of past cases is usually based on a measure of similarity. The problem of deriving suitable similarity measures has been studied in CBR for several decades now and a great wealth of measures has been put forward.

Why is there a need for yet another similarity measure? We believe that there is an important special situation in CBR that has been neglected in prior research but is worth studying: the situation where the characterization of the problem (i.e. the *query*) is underspecified or incomplete. This can happen for a number of reasons. We studied an application scenario (see Section 4.1 below) where a company that offers a document management system (DMS) wants to re-use knowledge of how the DMS can be integrated into different, but sometimes similar environments of their customers. In this scenario, the characterization of the problem – which in the end will include a variety of requirements for the integration solution – is incomplete in the beginning and evolves over time as the situation is further explored together with the customer. That is, an integration project proceeds along certain phases, from an initial specification, for which an offer is made, over exploration of requirements in workshops together with the customer to detailed specification, cost estimation and implementation of a solution. And even in the early stages of such an integration project where the case description is still fuzzy, re-use of knowledge from the case base can be desirable, e.g. to proactively suggest suitable features used in earlier projects. We expect similar situations in other areas. For instance in medicine, a physician using a CBR system will first know the symptoms that the patient is describing – and might want to use them for querying the case base. Later, as e.g. diagnostic checks are executed, more knowledge about the problem becomes available and the case characterization can be extended. Especially in the medical domain, underspecified queries might also be caused simply by a lack of time of the physician.

In other areas, characterization of the problem might be incomplete simply because some details of the problem cannot be figured out. For instance, help desk employees using a "conversational" CBR system [9] for solving customers' problems will not always get satisfactory answers from a customer when trying to clarify the faulty behavior of a product (e.g. a piece of computer hardware). In such cases, the corresponding slots in the case characterization remain empty.

The situation of underspecified queries shares many characteristics with information retrieval (IR) systems where a user is trying to find documents from a collection that satisfy a certain information need. In IR, the query – i.e. the description of the information need – is always much shorter than the documents that should be retrieved. In early IR research [23], queries were treated like ordinary documents and retrieval functions were essentially symmetric similarity measures that could be applied to a pair of documents.

Later, other and more successful retrieval functions were developed within the so-called *language modeling* approach. These functions are a) asymmetric, b) strongly conjunctive (but not in the Boolean sense) and c) have a weaker length penalty for documents than earlier functions. All these characteristics account for the underspecification of queries via one fundamental thought: a document may contain a lot of content unrelated to the query – which might be still relevant because the query is simply incomplete – as long as it fully covers the information specified in the query. It is clear that a function reflecting this thought cannot be symmetric: the roles played by the query and the document cannot be interchanged in this argument. The thought also implies that documents should not be overly penalised for length.

In this work, our goal is "interdisciplinary": we want to explore how current effective IR retrieval functions can be transferred to the CBR domain and whether they will lead to more effective retrieval in the case of underspecified queries.

In CBR, similarity measures have been developed not only for non-complex case descriptions with elementary attributes, but also for relational (or ontology-based) case descriptions [17]. For instance, cases may be represented as instances of an ontological schema that defines the entities involved in cases and the possible relationships among them (e.g.[14]). Thus, both in the query case as well as in the compared historical case, a theoretically unlimited number of instances of the same type or class can appear to fill a case property. Therefore we are faced with the kind of n:m relations that Bergman [2] calls multivalued relational attributes.

In our work, we also consider cases with relational descriptions – where relations can be of any kind (in particular, not only taxonomic relations are allowed). Furthermore, we require similarities to be computed on the level of instances and not only concepts.

2 Related Work

2.1 Case-based Reasoning and Retrieval Functions

Regarding the retrieval phase of a CBR cycle [1], [21], Perner [19] and Cunningham [6] introduced a taxonomy of similarity mechanisms used in CBR. As they remark, similarity measures in CBR usually make assumptions about how cases are represented.

Some approaches do not assume the structure of the case base to be known in advance. Instead, knowledge can be incrementally re-organized by detecting novelty, adapting parameters and applying prototype-based classification [12]. Perner [20] proposes an algorithm that incrementally learns the organizational structure of a case base via *conceptual clustering*.

When case structures are defined in advance, multiple options exist. For instance, *direct similarity* mechanisms operate on simple feature-value representations of cases. Other, more sophisticated mechanisms are *transformation based*. They "[...] the effort required to transform one object into the other" [6, p.9] such as Levenshtein edit distance or graph edit distance approaches (e.g. [15,4]). Graphs are a natural way to represent any kind of structured objects (including cases). By using attributes on vertices and edges, any structure can be represented; similarity measures can then be defined e.g. by identifying partial *graph isomorphisms* [18].

Many CBR approaches use *description logics* for representation. A number of corresponding similarity measures have been developed – a common thought behind these is to put into relation the amount of information shared between two cases and to contrast it with the amount of non-shared additional content that both cases carry. Often, this is based on the notion of a least common subsumer of two concepts [7], [24] and the idea of so-called *anti-unification*, a symbolic representation of what two objects (e.g. cases) have in common [17]. In [8], similarity is not only defined on the concept level, but also extended to the actual values of properties when comparing instances.

Apart from those approaches that are based on description logics, there exist also some other measures for relational case representations:

– The work of Hefke et al. [10] adapts the cosine measure from information retrieval to ontology-based case descriptions. Because of its similarity to our approach (also taking the field of information retrieval for inspiration), we will later use it as a reference in our experiments.
– Finally, since instances of ontological concepts can be represented as graphs with typed edges and vertices (e.g. via RDF), graph-edit distances [15,4] and approaches based on partial graph isomorphisms [18] are also applicable to ontological case representations.

Case-based reasoning has also been approached via probabilistic reasoning. For instance, [11] suggests a novel probabilistic approach to case-based inference. Case-based inference, however, is more related to the "Reuse" phase of the CBR cycle than to the Retrieval, i.e. similarity measures are used but not studied in detail. In an older work [16], a probabilistic similarity measure based on Bayesian inference is proposed; however, this mechanism assumes a non-structured case representation – i.e. each case is assumed to be represented by a real-valued feature vector.

2.2 Asymmetric similarity measures

All of the above-mentioned approaches result in symmetric similarity measures. Apart from being symmetric, they also introduce severe penalties for non-shared content between two cases, i.e. the similarity score drops rather steeply if at least one of the two cases contains much (additional) information not found in the other case.

Research that proposes asymmetric measures of similarity goes back to Tversky [26] who argues that a statement "a is like b" is directional ("We say 'the portrait resembles the person' rather than 'the person resembles the portrait"') and who was able to prove that we usually regard the variant ('a' / 'the portrait') as more similar to the prototype ('b' / 'the person') than vice versa. Hence, similarity measures should not be symmetric. The same thought has been taken up by researchers in CBR. For instance Jantke [13] states that " less representative examples are often considered to be more similar to more representative examples than the converse".

Tversky [26] presents a family of similarity measures that is based on the intersection (shared characteristics) and relative complements (non-shared characteristics) of the set of properties of two objects. That family generalizes the Jaccard and Dice coefficients. Jantke [13] makes a case for asymmetric similarity and then proposes a new "inductive" similarity measure based on the most specific antiunifier of a set of terms. However, this measure is ironically (as also Bridge [3] remarks) symmetric. Gupta [9] focuses on conversational CBR (as used e.g. in help desks) where cases are sets of question-answer pairs. He concentrates on "abstraction", i.e. the ability of the CBR system to retrieve cases with more general or more specific values when compared to the query, assuming that some case authors may have deeper knowledge and hence ability to specify on greater level of specificity than others. Questions are thus arranged into a taxonomy and an asymmetric measure is proposed to compare them: if the question in the query is an ancestor of (i.e. more abstract than) the question in the case, the similarity is higher than in the opposite case.

Some researchers discuss asymmetry in CBR similarity measures without proposing new measures. For instance, Bridge [3] discusses partial orderings of similarity values, i.e. argues that similarity functions do not need to return numbers, but can return anything on which an ordering is possible and then shows how to combine different similarity functions with different return types – and including both symmetric and asymmetric measures. Burke [5] discusses challenges of efficient computation of asymmetric similarity, presenting a very simple asymmetric measure to discuss its computational complexity.

2.3 Retrieval functions in information retrieval

Information retrieval (IR) systems retrieve and rank textual documents in response to natural language queries formulated by end users. Retrieval functions have been at the heart of IR research for many decades. A big step in that history was the invention of the vector space model [23] where both queries and documents are represented as vectors over keywords and can be compared e.g. via the cosine function. The vector space model became very popular and was used in many areas outside IR, including CBR (see [10]).

However, it was found that the cosine function had several undesirable properties, in particular the strong penalty that it imposes on long documents by normalising all vectors to unit length. Therefore functions with alternative length normalisations were invented, e.g. [25]. Such functions are asymmetric – as opposed to the cosine function. More recently, a new family of retrieval functions has been introduced that are based on so-called language models [22]. These new functions are also asymmetric and have been shown to consistently outperform all other retrieval functions.

2.4 Contribution

All in all, there have been arguments for asymmetric similarity measures in general and in CBR, but the few asymmetric measures that have been put forward do not satisfy the requirements of complex relational case bases: they are either only suitable for feature-value case representations (e.g. the one proposed by Tversky [26]). Or they only work for taxonomic, but not arbitrary relations, such as in the work of Gupta [9].

The contribution of this paper lies in the transfer of principles from IR to CBR: we argue that, when used to rank case characterisations *in response to a potentially incomplete query*, a CBR retrieval function needs to satisfy the same criteria as IR functions. In particular, we will examine the criteria of "length" normalisation and symmetry and explain how advances in IR functions can be transferred to CBR. This leads to the formulation of a new (family) of retrieval functions for CBR that can be applied to arbitrary ontology-based queries and case characterisations.

3 A new retrieval function

We will now develop a new retrieval function for case retrieval that is particularly suitable for situations where queries are incomplete (see Section 1). It thus addresses the shortcomings of the existing ontology-based functions w.r.t. these scenarios.

For better comparison, we follow the notation used in [10]: in order to compare two instances i_1 and i_2, we look at the k relationships $(r_1, ..., r_k)$ that are defined for the class to which i_1 and i_2 belong. And with J_{k1} we denote the set of instances that are linked to i_1 through relationship r_k, and analogously for J_{k2}. For each relation r_k, we will define a function that measures how closely i_1 matches i_2 with respect to all instances linked to both of them via r_k. Later, we will combine the relationship-based scores, together with scores yielded by comparison of atomic attributes (such as labels) into an overall score for comparing i_1 to i_2. This scoring function is applied recursively, and when i_1 is a query and i_2 is a case from the case base, it becomes the retrieval function.

As a running example, we will imagine that we have an instance of a case C_q that is linked to two module instances m_1 and m_2 (i.e. r_1 = *has-module*) and that is used as a query. It will be compared to three other case instances C_1, C_2 and C_3, each with a different set of linked modules, as depicted in Figure 1. We assume that the similarities of modules are known and given in the two matrices in Figure 2 (we have broken the matrices down because e.g. the similarities between m_3 and m_5 are not required for any of the following computations).

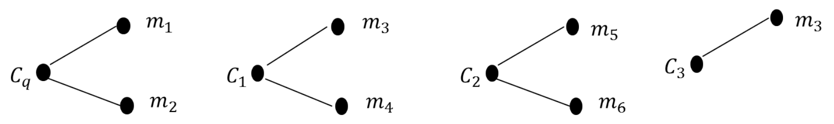

Figure 1. Example instances of cases, each with a set of linked module instances

We now consider the disadvantages of existing CBR functions, as exemplified by the cosine function, for ontology-based CBR with incomplete queries by transferring knowledge from the area of information retrieval (IR):

$$\begin{array}{c|cccc} & m_1 & m_2 & m_3 & m_4 \\ \hline m_1 & 1 & 0 & 0.7 & 0 \\ m_2 & 0 & 1 & 0 & 0 \\ m_3 & 0.7 & 0 & 1 & 0 \\ m_4 & 0 & 0 & 0 & 1 \end{array} \qquad \begin{array}{c|cccc} & m_1 & m_2 & m_5 & m_6 \\ \hline m_1 & 1 & 0 & 0.3 & 0 \\ m_2 & 0 & 1 & 0 & 0.3 \\ m_5 & 0.3 & 0 & 1 & 0 \\ m_6 & 0 & 0.3 & 0 & 1 \end{array}$$

Figure 2. Similarity matrices for the modules involved in the case instances of Figure 1

- **Symmetry** The modern language model-based retrieval functions in IR are asymmetric: they define how to score documents against a query; when the roles of query and document are switched, the score will change. For CBR, we argue for asymmetry, too: when considering the example in Figures 1 and 2, we propose that the value $f(C_q, C_3)$ of a retrieval function f should be smaller than when using C_3 as the query (i.e. $f(C_3, C_q)$ – since, when a user specifies two modules, she cares to have them matched and will be disappointed with a result that does not (such as C_3). On the other hand, if she only specifies one module in the query (as when using C_3 as the query), she will be satisfied with any case that (partially) matches the module (such as C_q).
- **Length normalisation:** In information retrieval, a known problem of the cosine function (related to its symmetry) is the fact that it overly punishes long documents. In CBR terms, it will punish cases having many attribute values that are not specified in the query. In our example, when querying for C_q, cosine will produce a substantially smaller score for C_1 (0.56) than for C_3 (0.73) since C_1 contains – besides the partially matching module m_3 – a module that does not match the query at all (m_4), whereas C_3 contains only m_3. However, from the perspective of a searcher, C_1 is an equally good result, assuming that the searcher is willing to ignore the non-matching module m_4. The length penalty is something that can also be counted as a weakness of many other approaches, including the ones based on description logics.
- **Conjunctiveness:** Although modern retrieval functions in IR do not rely on Boolean operators anymore – and hence do not apply strict conjunction of query terms – the implicit interpretation of queries in web search engines (such as Google or Yahoo!) is nearly conjunctive. In academic IR research, the new family of language model retrieval functions (e.g. [22]) has been introduced with much success – such functions are inherently more conjunctive than e.g. the cosine function (see e.g. [27]). One reason for the success of conjunctive query interpretation is the fact that the implicit semantics of most queries typed into a search box is conjunctive – that is, when a user types a short query such as "case based reasoning", then usually none of these terms is optional, i.e. the desired results contain all three terms. When designing retrieval functions for CBR, there is good reason to assume a similar situation: when several characteristics of a case are specified for retrieval, there

is high probability that the user would like to see all characteristics matched by the highly ranked retrieval results. In terms of our example, this means that – when querying for C_q with its two modules – C_2 should be preferred over C_1 since it has two (although only slightly) matching modules whereas C_1 can match only one. However, when using the cosine-based similarity function presented in [10], C_1 receives a slightly higher score (0.56) than C_2 (0.55).

The new retrieval function shall be derived from the insights gained on language model-based retrieval functions in IR. When using language models, the basic assumption is that the searcher formulates a query by thinking of a typical (or "the") relevant document and selecting the most important terms from this prototypical relevant document (cf. [22]). The score $f(q,d)$ of a real document d w.r.t. q is then computed by working out how probable it is that the query q would be drawn as a sample from the language model M_d of the document where a language model is a probability distribution over terms:

$$f(q,d) = P(q|M_d) = \prod_{t \in q} P(t|M_d) \tag{1}$$

Here, $P(t|M_d)$ is the probability of drawing term t from the document's language model. Different language modeling techniques have been proposed – they differ mainly in the way $P(t|M_d)$ is computed. An important aspect in this regard is *smoothing*, i.e. the way we assign probability to query terms that do not appear in a document [28]. Setting that probability to 0 would result in a strict conjunctive query interpretation (i.e. any document not containing all query terms receives a score of 0) – hence, some probability is reserved for such absent terms.

In order to translate this to CBR, we first concentrate on a single relationship r_k (e.g. *has-module*). We may assume that, when formulating a query C_q, a user thinks of a prototypical historical case that is similar to her current situation in terms of the set E of involved related instances, e.g. modules. For a given case C_i from the case base with a set of linked modules F, and in analogy with the language model approach, we want to estimate the probability that E would be drawn as a sample from F, i.e.

$$P(E|F) = \prod_{e \in E} P(e|F) \tag{2}$$

where $P(e|F)$ is the probability that instance e would be generated from the set of instances F. The word "generating" refers here to a process of remembering: how likely is it that someone would specify element e, assuming that she remembers the historical case with its attached set of elements F?

We further assume that $p(e|F)$ is given by the best match between e and any $f \in F$. That is: we ask for the probability that e is specified when f is meant for each $f \in F$ and then assume that only the maximum probability is relevant for the whole set F:

$$P(e|F) = \begin{cases} \max_{f \in F} p(e|f) & \text{if } \exists f \in F : p(e|f) > 0 \\ 0.01 & \text{otherwise} \end{cases} \tag{3}$$

As an example of such computation, consider the query case C_q given in Figure 1 with its set of modules $\{m_1, m_2\}$. When we compare it to case C_1 with modules $\{m_3, m_4\}$, we get $P(C_q|C_1) = P(m_1|\{m_3, m_4\}) \cdot P(m_2|\{m_3, m_4\}) = \max_{i\in\{3,4\}} p(m_1|m_i) \cdot \max_{i\in\{3,4\}} p(m_2|m_i) = 0.7 \cdot 0.01 = 0.007$. On the other hand, $P(C_q|C_2) = 0.3 \cdot 0.3 = 0.09$. Finally, $P(C_q|C_3) = 0.7 \cdot 0.01 = 0.007$, just as $P(C_q|C_1)$.

We see that, by using a product for combining the probabilities of query elements, we end up with a rather conjunctive query interpretation, rather strongly preferring C_2 because it is able to partially match both modules specified in C_q. To avoid strict conjunction, we have introduced the constant smoothing factor of 0.01 in equation 3, such that C_1 does not end up with a score of 0. Later, we could modify the smoothing factor to introduce a (weak) length penalty for cases or to tune the conjunctiveness of the function. We also note that $P(E|F)$ is not symmetric. We finally observe that "length" of the case does not play a role: both C_1 and C_3 receive the same score because both match m_1. The non-matching module m_4 in C_1 does not decrease its score.

We assumed the "atomic" probabilities $p(e|f)$ as given (by the matrices in Figure 2). Actually, in case that e and f also have relational properties, the above equations are applied recursively until basic similarity operations (such as string similarities or equality checks) can be applied.

We finally need to combine the scores across multiple relationships r_k. In [10], such combination is done via a simple average. A slight generalisation of that approach would be to use a weighted average:

$$\sum_k \alpha_k \cdot sim_{set}(J_{k1}, J_{k2}), \text{with } \sum_k \alpha_k = 1 \tag{4}$$

where J_{kl} are again the set of instances that are linked to instance i_l and the weights α_k reflect the relative influence of relationship (or attribute) r_k for determining the overall similarity. In the beginning, such weights may be chosen based on some kind of "business intuition" – and may later be tuned based on relevance feedback.

However, in line with the way we have defined $P(E|F)$ above, we will opt for a more conjunctive version and replace the weighted sum with a weighted product:

$$P(i_1|i_2) = \prod_{k=1}^{n} P(J_{k1}|J_{k2})^{\alpha_k} \tag{5}$$

4 Experimental exemplification

In the following, we will illustrate the behaviour of our proposed new retrieval function by applying it to a case base and comparing its results to the results of the cosine function. The case base is synthetic, but it was generated to reflect a real case that is described in the next subsection.

4.1 Application scenario

We consider the case of ELO Digital Office CH AG[1]. The company develops and sells standard software for enterprise content management (ECM) and document management (DMS). Within their sales processes, ELO experts have to analyse a large set of requirements and answer questionnaires provided by their customers. Based on these analyses an offer is made. The time consuming tasks of collecting information from different project documentations and experts shall be simplified by storing knowledge from previous projects in a case base and reusing it to create new project offers.

A typical task during the installation of the ELO standard software in a company is its integration with any number of existing legacy IT-systems of the customer. Examples of such systems are enterprise resource planning systems (ERP), email servers or existing archiving solutions. Hence, case characterisations comprise a set of such IT systems ("application services") via the relation "hasSystem" (see Figure 3).

ELO's standard software products are modularized. This means that the solution can be offered as a combination of modules, each solving a certain problem. Depending on the requirements of the customer, the finally offered solution will be a configuration with selected modules. In case characterisations, this is reflected by a relation "hasModule" that can be filled by a set of ELO modules required by the customer in a given case. The ontology additionally defines experts for modules with a certain role and level of expertise assigned. Figure 3 shows the complete ontological case model that forms the schema of the case base for this scenario.

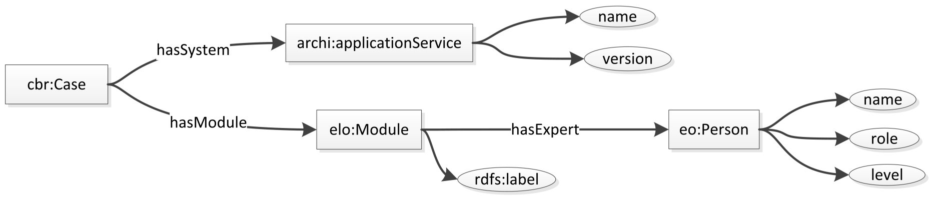

Figure 3. Case characterisation space

We observe that a relational representation is needed in this scenario – the relationships between cases and legacy systems cannot be modeled with simple feature-value pairs because their number is not known in advance and new systems or new versions of existing systems may appear with every new case.

[1] http://www.elo.ch. The company is our partner in the Swiss government funded research project [sic!].

4.2 Experimental Setup

In a first step, we generated a synthetic case base according to the schema we just described (see Figure 3) and based on a few case instances orally described by our project partner.

The basic idea of the generation procedure is that many CBR queries are formulated with a certain case in mind, i.e. the searcher knows of the existence of a certain historical case that is similar to the current problem and would like to retrieve the details of that historical case. We assume that if we are good in retrieving the "meant" case in such situations – and in the presence of "noise", i.e. cases that were not meant but are also somewhat similar – we have a generally good retrieval algorithm.

We therefore adopted an approach which, for each of a set of 10 queries, generates cases that match the given query very closely, less closely and loosely. More precisely, we implemented the following generation strategy:

1. based on an initial set of data objects representing persons, generate an extended set P of person data objects such that a set of persons with balanced number of roles and levels of expertise (beginner, expert) is created
2. based on an initial set of manually created modules (these are real modules present in the application domain), create an extended set M of modules. New modules were created with labels where we added 1-2 nouns from the available module descriptions to an existing label or generated new labels from those descriptions. To each module $m \in M$, a set of 1-5 randomly chosen experts $p \in P$ is assigned.
3. generate a set A of application services (AS): we start from an initial, manually created taxonomy of software containing the types of software at the top level (namely "database", "ERP" or "DMS"), and the actual name of the softwares on the next level. For each of the leaves of this high-level taxonomy, different artificial version names and version numbers of each software were generated (e.g. "Oracle AB 3.4.1")
4. create a set C_0 of 10 "meant" cases, i.e. cases for the case base for which a search is simulated: each meant case consists of at least one module $m \in M$ and one application service $a \in A$. One additional module or AS can be inserted at random and with certain probability.
5. For each "meant" case $c \in C_0$, create a corresponding query q_c that has the same module m and AS a. To simulate the imperfection of human memory, the query is then slightly changed: the version name of a is changed with 30% probability, its version number is slightly changed or omitted with 50% probability. Slightly changing means here that a digit in the version number is changed on the second or third level (for instance version number "8.2.5" is changed to "8.2.3". In addition, the label of the module is slightly changed with 30% probability, by adding or dropping a noun. This results in a set Q of 10 queries.
6. Create noise in the form of a set C_w of n weak mutations (we chose $n = 5$ in our experiments) of each "meant" case c by applying – at random – 2-4 of the following changes to it: a) construct a completely new version of the

AS, b) replace an AS with another one, but of the same type, c) replace a module with another one that has the same types of experts assigned to it or d) add a randomly chosen AS or module. We denote the ith weak mutation of case c with c_w^i.

7. Create more noise in the form of a set C_s of n strong mutations (again, we chose $n = 5$) by applying several of the following operations to each "meant" case: a) remove one of the modules that appear in the query belonging to the "meant" case, b) replace it with a completely different one or d) do deletion or replacement of an AS appearing in the query. We denote the ith strong mutation of case c with c_s^i.

Thus, with the 10 cases in C_0 and the 5 weak and strong mutations per each $c \in C_0$, the case base comprises 110 cases.

4.3 Exemplary retrieval results

In order to illustrate the difference between our new similarity measure and existing ones, we ran the 10 queries against the case base, using both our probabilistic retrieval function and the cosine function [10]. We then selected a query (q_6) for which the difference between the results of cosine and probabilistic function was considerable: query q_6 consists of

– a module named "Replication documents" with attached expert of type *manager, expert*
– the application service "WebERP A 5.3.5"

Table 1 shows the top 3 retrieved cases of both rankings and their characteristics. Those elements of a retrieved case that match the query (partially) are highlighted in bold.

We can observe mainly two things: Firstly, the probabilistic function retrieves two weak variations of case 6 (6_w^1 and 6_w^2) and the "meant" case 6 itself within the top 3 ranks, all with the same score of 0.863. The two variations are scored highly because they have a rather perfectly matching module (in terms of both label and experts) and AS (only version number shows a slight deviation). However, they both additionally contain two "non-fitting" application services and one "non-fitting" module. This results in very low scores with cosine (they appear at rank 21 and 35 in the cosine ranking, not shown here) – i.e. they are "punished" for their "length". Considering that a query might be incomplete, we believe that the asymmetry and associated length penalty of the cosine are suboptimal: we expect that a searcher will be able to draw the required knowledge from the cases 6_w^1 and 6_w^2 and will simply ignore the non-relevant additional information that they contain. Hence, it is good to rank them highly.

Secondly, the cosine function does not retrieve any variations of the "meant case" within the first three ranks. Instead, the cases at the first two ranks of the cosine ranking (case 5 and 5_s^2) both have an application service that does not

Rank	Cosine			Probabilistic		
	case id (*score*)	modules (expert types)	application services	case id (*score*)	modules (expert types)	application services
1	5 (*0.938*)	– **Replication** (**manager, expert**) – Signature (**manager, expert**)	– **WebERP** D 9.0	6^2_w (*0.863*)	– **Replication** (**manager, expert**) – Cold (programmer, beginner)	– **WebERP A 5.3.4** – SQLServer ABCDE 5.0 – Documentum ABC 6.3.6
2	5^2_s (*0.929*)	– **Replication** (**manager, expert**) – Signature (**manager, expert**) – teams (TechConsultant, **expert**; Admin, beginner)	– **WebERP** D 8.6	6^1_w (*0.863*)	– **Replication** (**manager, expert**) – DocXtractor trains processing (programmer, beginner)	– **WebERP A 5.3.4** – SQLServer ABCDE 5.0 – Documentum ABC 6.3.6
3	9^1_s (*0.864*)	– system (**manager, expert; manager**, beginner)	– **WebERP A 5.9.8**	6 (*0.863*)	– **Replication** (**manager, expert**)	– **WebERP A 5.3.4** – MySQL CDE 2.0

Table 1. Top 3 results for q_6 with cosine (left) and probabilistic ranking (right)

match very well in terms of version – however, they "compensate" for this mismatch because they have not only *one* matching module, but each also contains another module that matches in terms of the expert signature (i.e. has an expert manager as responsible person). This shows how the disjunctive nature of the cosine function allows to increase the score of a case by matching the same query element (in this case the "Replication" module) several times – something which has no influence with our probabilistic function because only the best-matching element is considered by using the max function (see Equation 3). We believe that results like 5 and 5^2_s are not very useful to a searcher since none of the application services fits closely – hence, they require a bigger adaptation effort when re-using the knowledge than e.g. cases 6^1_w and 6^2_w.

All in all, this analysis has shown that both the symmetry and associated length penalty and the disjunctive nature of cosine lead to – in our opinion – undesired effects when ranking cases in response to potentially incomplete queries. The first of these effects – namely strongly down-ranking cases with additional non-query-related content – will be found also with other symmetric retrieval functions that are widely used in CBR (e.g. those based on description logics). Both effects can be avoided by using our new probabilistic retrieval function.

5 Conclusions and Future Work

This paper has transferred research results from the domain of information retrieval (IR) to case-based reasoning (CBR): properties of recent, successful IR retrieval functions have been used to design a new CBR retrieval function, designed for ontololgy-based, structured case characterisations. Such properties include strongly conjunctive query interpretation, asymmetry and weak (or no) penalties for case “length”. Our experimental exemplification has shown how and to what extent its properties contribute to the differences in ranking w.r.t. our reference, the cosine function.

Future work may focus on the influence of the weights in the query function (which we assumed to be uniform in our experiments) and tune them to correctly meet human requirements.

Acknowledgements

This work was supported by the Commission for Technology and Innovation CTI of the Swiss Confederation under Grant 14575.1 PFES-ES and the ELO Digital Office CH AG.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
2. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications, LNCS, vol. 2432. Springer, Berlin, Heidelberg (2002)
3. Bridge, D.G.: Defining and Combining Symmetric and Asymmetric Similarity Measures. In: Proc. of 4th European Workshop on Case-Based Reasoning. pp. 52–63 (1998)
4. Bunke, H., Messmer, B.T.: Similarity Measures for Structured Representations. In: Proc. of EWCBR-93. pp. 106–118 (1993)
5. Burke, R.D.: Ranking Algorithms for Costly Similarity Measures. In: Proceedings of the 4th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development. pp. 105–117. ICCBR ’01 (2001)
6. Cunningham, P.: A Taxonomy of Similarity Mechanisms for Case-Based Reasoning. IEEE Transactions on Knowledge and Data Engineering 21(11), 1532–1543 (2009)
7. D’Amato, C., Staab, S., Fanizzi, N.: On the Influence of Description Logics Ontologies on Conceptual Similarity. In: Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns. pp. 48–63. EKAW ’08 (2008)
8. González-Calero, P.A., Díaz-Agudo, B., Gómez-Albarrán, M., et al.: Applying dls for retrieval in case-based reasoning. In: In Procs. of the 1999 Description Logics Workshop (1999)
9. Gupta, K.M.: Taxonomic conversational case-based reasoning. In: Proceedings of the Fourth International Conference on CBR. pp. 219–233 (2001)
10. Hefke, M., Zacharias, V., Abecker, A., Wang, Q., Biesalski, E., Breiter, M.: An Extendable Java Framework for Instance Similarities in Ontologies. In: Proceedings of ICEIS 2006. pp. 263–269 (2006)

11. Hüllermeier, E.: Toward a Probabilistic Formalization of Case-Based Inference. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. pp. 248–253. IJCAI '99 (1999)
12. Jänichen, S., Perner, P.: Conceptual Clustering and Case Generalization of Two-dimensional Forms. Computational Intelligence 22(3-4), 177–193 (2006)
13. Jantke, K.P.: Nonstandard concepts of similarity in case-based reasoning. In: Information Systems and Data Analysis, pp. 28–43 (1994)
14. Martin, A., Emmenegger, S., Wilke, G.: Integrating an enterprise architecture ontology in a case-based reasoning approach for project knowledge. In: Proceedings of the First International Conference on Enterprise Systems: ES 2013. pp. 1–12 (2013)
15. Minor, M., Tartakovski, A., Schmalen, D., Bergmann, R.: Agile Workflow Technology for Long-Term Processes: Enhanced by Case-Based Change Reuse. In: Sugumaran, V. (ed.) Methodological Advancements in Intelligent Information Technologies, vol. 4, pp. 279–298. IGI Global (2010)
16. Myllymäki, P., Tirri, H.: Massively Parallel Case-Based Reasoning with Probabilistic Similarity Metrics. In: Selected Papers from the First European Workshop on Topics in Case-Based Reasoning. pp. 144–154. EWCBR '93 (1994)
17. nn, S.O., Plaza, E.: Similarity Measures over Refinement Graphs. Machine Learning 87(1), 57–92 (2012)
18. Perner, P.: Content-Based Image Indexing and Retrieval in an Image Database for Technical Domains. In: Proc. of Multimedia Information Analysis and Retrieval, IAPR International Workshop, MINAR'98 (1998)
19. Perner, P.: Why Case-Based Reasoning is Attractive for Image Interpretation, pp. 27–44. LNAI 2080, Springer Verlag (2001)
20. Perner, P.: Case-base Maintenance by Conceptual Clustering of Graphs. Engineering Applications of Artificial Intelligence 19(4), 381–393 (2006)
21. Perner, P.: Mining Sparse and Big Data by Case-based Reasoning. Procedia Computer Science 35, 19 – 33 (2014), knowledge-Based and Intelligent Information & Engineering Systems 18th Annual Conference, KES-2014 Gdynia, Poland, September 2014 Proceedings
22. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proceedings of SIGIR '98. pp. 275–281 (1998)
23. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM 18(11), 613–620 (1975)
24. Sánchez-Ruiz-Granados, A.A., Ontañón, S., González-Calero, P.A., Plaza, E.: Refinement-Based Similarity Measure over DL Conjunctive Queries. In: Proc. of ICCBR-13. pp. 270–284 (2013)
25. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: Proceedings of SIGIR '96. pp. 21–29 (1996)
26. Tversky, A.: Features of Similarity. Psychological Review 84(4), 327–352 (1977)
27. Witschel, H.F.: Carrot and stick: combining information retrieval models. In: Proceedings of DocEng 2006. p. 32 (2006)
28. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to Ad Hoc information retrieval. In: Proceedings of SIGIR '01. pp. 334–342 (2001)