Dynamic Case Bases and the Asymmetrical Weighted One-Mode Projection

Rotem Stram[1,2], Pascal Reuss[1,3], and Klaus-Dieter Althoff[1,3]

[1] Smart Data and Knowledge Services Group, German Research Center for Artificial Intelligence, Kaiserslautern, Germany
{rotem.stram,pascal.reuss,klaus-dieter.althoff}@dfki.de
[2] Department of Computer Science, Technical University of Kaiserslautern
[3] Institute of Computer Science, Intelligent information Systems Lab, University of Hildesheim, Hildesheim, Germany

Abstract. Building a case base for a case-based reasoning (CBR) system is incomplete without similarity measures. For the attribute-value case structure similarity between values of an attribute should logically fit their relationship. Bipartite graphs have been shown to be a good representation of relationships between values of symbolic attributes and the diagnosis of the cases in a technical diagnosis CBR system, while using an asymmetrical weighted one-mode projection on the values to model their similarity.
However, the weighted one-mode projection assumes that the set of symbols is static, which is contradictory to the dynamic nature of case bases as defined by the retain phase of the CBR cycle. In this work we present two methods to update the similarity measure whenever new information is available and compare them. We show that even though updating the similarity measure to exactly reflect the case base had the new information been available a-priori produces better results, an imperfect update is a feasible, less time consuming temporary solution.

Keywords: Dynamic case bases, Bipartite graph, Weighted one-mode projection, Local similarity, Symbolic attributes

1 Introduction

The basic idea behind Case-Based Reasoning (CBR) is that similar problems have similar solutions. It is a paradigm for problem solving by using previously solved problems as the starting point for new solutions. The CBR cycle was formalyzed by Aamodt and Plaza in [1] and is also known as the four R cycle, named after its four steps: Retrieve, Reuse, Revise, and Retain.

The retrieve step of the cycle is one of its crucial parts, as it decides which past experiences the system uses as a basis for a new solution. It takes as input a case description representing a new problem, and outputs either the most similar case from the case base (past experience), or a list of n most similar cases. When considering a diagnosis system, where a solution is a member of a pre-defined set

of possible diagnoses, there is usually almost no processing done on the retrieved cases, making the accuracy of this step even more crucial.

At the heart of the retrieval step is the similarity between two cases. There are two types of similarities in CBR, local and global. If we focus on the attribute-value case structure, local similarity is defined as the similarity between two values of a single attribute. Global similarity is then the similarity of two cases as a whole by aggregation of local similarity values.

For local similarities the type of the attribute defines the similarity function that is used. For numerical attributes, for instance, a distance measure such as Euclidean distance can be used. For strings the edit distance is a good solution [4], while the similarity of symbolic attributes is usually either modeled by experts, defined by taxonomies, or is a combination of both [2], [10].

Recently, graph theory and network analysis methods have been introduced as tools to extract local similarity measures of symbolic attributes, most notably the weighted one-mode projection (WOMP) [15]. In this work from the technical diagnosis domain, textual problem description were transformed into keywords, and each keyword connected to the diagnosis of the case, effectively creating a bipartite graph (BPG) where the weight of each edge is the number of times each keyword appeared under each diagnosis. A novel method for asymmetrical weighted one-mode projection (aWOMP) was introduced, and used as a similarity measure between the keywords. This work, however, assumed that the set of keywords is fixed and did not address the possibility that the case-base will change over time. Here we will addresses the possible updates of the aWOMP as the case base is updated.

This work is structured as follows. In section 2 we will give an introduction to aWOMP. Section 3 will discuss the update options of aWOMP and introduce two methods to update the similarity measure under different conditions. Section 4 will present the premise of our experimentation and their results. Section 5 will list works that are related to ours, while section 6 will conclude this paper and offer possible directions for future work.

2 Asymmetrical Weighted One Mode Projection

One-mode projection (OMP) refers to an action performed on a bipartite graph (BPG) to transform it to another graph depicting the relationship of only some of the original nodes. A BPG, as can be seen in Fig. 1a, is a graph with two groups of nodes generally referred to as left (L) and right (R). Connections are only allowed between groups, but not within them. A OMP is a projection of the graph on either the L nodes or the R nodes, where two nodes are connected if they share a neighbor in the BPG (e.g. Fig. 1b).

In some cases a weighted OMP (WOMP) is needed, and even more so when the BPG itself is weighted. To this end, Stram et al. [15] introduced a novel WOMP method in their work from 2017 that produced asymmetrical weights in the resulting graph (aWOMP). This method was used as a local similarity measure of symbolic attributes in a Case-Based Reasoning (CBR) system, where they showed the superiority of aWOMP over other common methods.

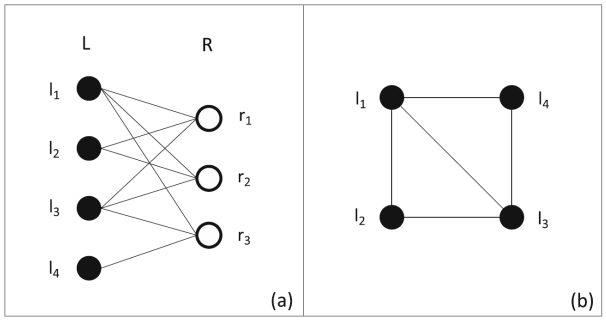

Fig. 1: a. A bipartite graph. b. The one-mode projection of the bipartite graph

The aWOMP relies on resource allocation of nodes in a bipartite graph, and is based on a method introduced by Zhou at al. [16]. It takes a weighted BPG as an input, and produces a new graph quantifying the relationship between nodes in a single group from the BPG by measuring the amount of resources node A allocated to all the neighbors N_i that it shares with node B, together with the resources that nodes N_i allocate to node B.

Let $G = (L, R, E)$ be a BPG where E is a set of edges (l_i, r_j, w_{ij}), where w_{ij} the weight between nodes $l_i \in L$ and $r_j \in R$, and $|L| = n$, $|R| = m$. Then the resources that node l_i accumulates is the sum of its adjacent edges:

$$W_i^L = \sum_{j=1}^{m} w_{ij} \tag{1}$$

The resources that node l_i allocates to node r_j is the weight of the edge between them normalized by the total amount of l_i's resources:

$$w_{ij}^{L \to R} = \frac{w_{ij}}{W_i^L} \tag{2}$$

Which leads to the resources that each node $r_j \in R$ accumulates:

$$W_j^R = \sum_{i=1}^{n} w_{ij}^{L \to R} \tag{3}$$

This process corresponds to the flow of resources seen in Fig. 2a. Now we switch directions and regard the flow from R to L:

$$w_{ij}^{R \to L} = \frac{w_{ij}^{L \to R}}{W_j^R} \tag{4}$$

This is visualized in Fig. 2b. When we look at two nodes $l_a, l_b \in L$, the resources that flow between them are:

$$w_{ab}^{L \to L} = \sum_{j=1}^{m} p_{aj} \cdot p_{bj} \cdot (w_{aj}^{L \to R} + w_{Bj}^{R \to L}) \tag{5}$$

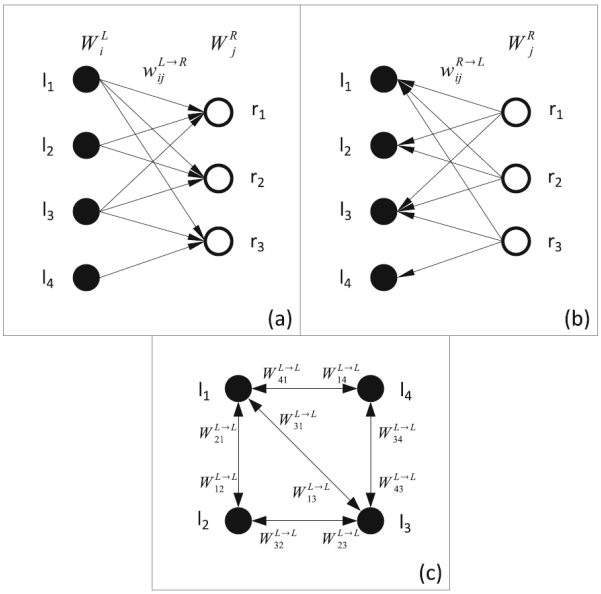

Fig. 2: a. Resource allocation from L to R. b. Resource allocation from R to L. c. Asymmetrically weighted one-mode projection

Where $p_{ij} = 1$ if l_i and r_j are neighbors, and $p_{ij} = 0$ otherwise. We then normalize this weight:

$$W_{ab}^{L\to L} = \frac{w_{ab}^{L\to L}}{w_{bb}^{L\to L}} \tag{6}$$

The resulting weights are those used in the aWOMP graph as seen in Fig. 2c.

3 aWOMP and Dynamic Case Bases

The case base of a diagnostic CBR system is comprised of case description and diagnosis pairs. We focus on the attribute-value case description type, and so cases are comprised of a set of attribute values, where a value can on its own be a set of values. As an example take the system described by Reuss et al. [10] in the technical diagnosis of aircraft faults domain. Here, symbolic attributes were extracted from textual fault descriptions and divided into different attributes such as *fault*, *location*, and *time*. For a single case each attribute can hold several values. Focusing on a single attribute, each value representing a keyword is connected to the diagnosis of the case. In the BPG the keywords build the node set L and the diagnoses the node set R. The weight between a keyword and a diagnosis is the number of cases the keyword appeared in that had this

diagnosis. If this keyword appears in several cases with different diagnoses then in the BPG it will be connected to several nodes from R.

In order to use the aWOMP as a local similarity measure, the BPG needs to be known beforehand, and the aWOMP is calculated offline. However, in real-world applications the environment is dynamic and new information is constantly added. In order to overcome this, the aWOMP should be updated whenever new information arises. New information in this context could be one of the follwoing:

- a new node in L
- a new node in R
- a new edge
- a different weight on an existing edge
- removing a node from L
- removing a node from R
- removing an edge

Removing nodes or edges from the graph is a result of removing cases from the case base in a process called *forgetting* [9]. Since *forgetting* is outside the scope of this work we will focus only on changes resulting from adding new cases to the case base. More specifically, we will focus on adding a new node to L, adding a new edge, and changing the weight of an existing edge. Adding a new node to R will not be described here for simplicity reasons, however it is straight forward since the new node will only have one edge when it is added.

In this section we will discuss two possible update method to the BPG so that the similarity measure remains up to date with the case base.

3.1 Perfect Update

A perfect update is an update to the aWOMP such that it is indistinguishable from an aWOMP that would have been calculated had we built the BPG with the new information in advance. There are two ways to obtain a perfect update of an aWOMP: either calculate the new aWOMP from scratch, or making local changes to the BPG only where it is relevant. Let's focus on two possible local changes to the BPG: adding a new node to L and changing the weight of an existing edge.

Adding a new node to L Assume that we have a BPG, and for each node $l_i \in L$ and $r_j \in R$ we know the values of W_i^L and W_j^R respectively. Adding a new node l_x to L, along with an edge to at least one $r_j \in R$ (e.g. Fig. 3) would affect W_j^R (see Eq. 3) and thus ultimately $W_{ab}^{L\to L}$ (Eq. 6) for all l_a and l_b that share r_j as a neighbor.

The required local changes are then:

$$W_{jnew}^R = W_j^R + \frac{w_x}{W_x^L} \tag{7}$$

and from here on every call to $W_{ab}^{L\to L}$ would give the correct weight.

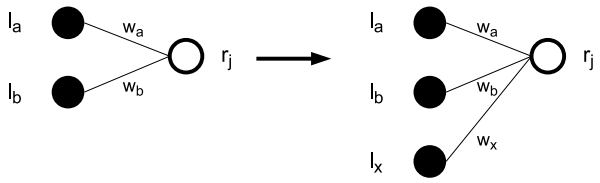

Fig. 3: Adding a new node to a bipartite graph

Changing the weight of an existing edge Assuming again that for a given BPG and for each node $l_i \in L$ and $r_j \in R$ we know the values of W_i^L and W_j^R respectively (e.g. Fig. 4). We now change the weight of w_b. The effect here is much greater than adding a new node, due to its effect on all the neighbors of l_b, since the sum of all its resources, which is used in Eq. 2, is now different. Subsequently, W_j^R of all $r_j \in R$, r_j is a neighbor of l_b needs to be updated.

Fig. 4: Changing the weight of an existing edge in a bipartite graph

The following changes need to be made:

$$W_{bnew}^L = W_b^L - w_b + w_{bnew} \tag{8}$$

And then for each $r_j \in R$, r_j is a neighbor of l_b:

$$W_{jnew}^R = W_j^R - \frac{w_b}{W_b^L} + \frac{w_{bnew}^L}{W_{bnew}^L} \tag{9}$$

From here on every call to $W_{ab}^{L \to L}$ would give the correct weight. With these example, perfectly updating the aWOMP for the remaining scenarios can be easily extrapolated.

Discussion Remaining in the CBR domain where aWOMP is used as a similarity measure, adding or removing a single case may result in a massive chain reaction of changes to the BPG, depending on the density of the graph. A case can be seen as a BPG on its own, where $|R| = 1$ containing the diagnosis of the case (see example in Fig. 5b). Adding this graph representation of the case to the case-base BPG (Fig. 5a) results in a new weight for the shared edges, and possible new nodes and edges.

A change in the weight of just one edge adjacent to a node $l_i \in L$ would require an update to the resources l_i allocates to all its neighbors (see Eq. 2)

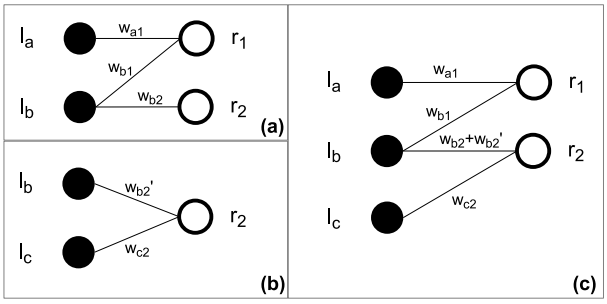

Fig. 5: a. An existing BPG representing a case base. b. A BPG representation of a new case. c. The new BPG resulting from adding the new case to the case base.

and subsequently an update to the resources allocated to all of the neighbors of $r_j \in R$, r_j a neighbor of l_i. Now imagine the new case is comprised of several keywords. This makes the scalability of the perfect update only slightly more feasible than calculating the aWOMP from scratch, if at all, especially when the BPG is dense.

Dynamic real-time systems with large case bases and tens of thousands of keywords in just one attribute may not have the resources to perfectly update the aWOMP for every new case, requiring a temporary scalable solution that is feasible in the short run. A solution can be temporarily imperfect, allowing the BPG to be perfectly updated (or recalculated) in the background for future use.

3.2 Imperfect Update

In order to better understand the imperfect update of a weighted BPG we first define the relation $\leq^c$ between the similarity functions sim_1 and sim_2 under a context c. If the aWOMP of two BPGs is used as local similarity measures sim_1 and sim_2 between symbolic attributes of a case base domain d (the context), then $sim_1 \leq^d sim_2$ if sim_2 leads to better diagnosis accuracy than sim_1 for the same test set.

Let us assume that given a domain d, G_1 is a BPG representation of the case base CB_1, sim_1 the similarity measure, C a set of new cases, $CB_2 = CB_1 \cup C$ the case base after adding C, G_2 the BPG of CB_2 resulting from the perfect update of G_1, and sim_2 the similarity measure of G_2. We also assume that $sim_1 \leq^d sim_2$. An imperfect update of G_1 is a transformation of sim_1 into a new similarity measure sim_u that maintains the following inequality for some ranked results:

$$sim_1 \leq^d sim_u <^d sim_2 \tag{10}$$

In this work we propose an imperfect update method that approximates the inequality in Eq. 10. Let $G = (L, R, E)$ be a BPG representation of case base CB, sim the local similarity between symbols of a symbolic attribute as derived from the aWOMP of G, and let C a set of new cases. We follow these steps:

1. create a new BPG $G_c = (L_c, R_c, E_c)$ from C and derive the similarity measure sim_c from the aWOMP of C.
2. for each symbol pair $l_a, l_b \in L_c$
   (a) find the similarity value $s = sim(l_a, l_b)$. If $l_a \notin L$ or $l_b \notin L$ then $s = 0$.
   (b) find the similarity value $s_c = sim_c(l_a, l_b)$
   (c) $s_{new}(l_a, l_b) = s + (1 - s) * s_c$

For the case $l_a = l_b$ aWOMP ensures that $sim(l_a, l_b) = 1$, however even if the value is not in G then $sim_c(l_a, l_b) = 1$ and thanks to the equation in step 2c $s_{new}(l_a, l_b) = 1$. We expect that $sim \leq^d sim_{new} <^d sim_{perfect}$ where $sim_{perfect}$ the similarity measure of the perfectly updated BPG.

Discussion The proposed imperfect update method provides a solution for local update of similarity between keywords that appeared in the new cases added to the case base. The advantage here is the quick update for only a subset of keywords instead of all keywords, meaning only those keywords that appeared in the new cases. This suggests that a simultaneous update for several cases at once would give better results than an update for one case at a time, meaning that batch updates would be more beneficial. On the other hand, this update requires calculating an aWOMP for the new cases, which is computationally intensive, especially considering that our goal is to avoid recomputing the aWOMP for the entire case base. This leads to the conclusion that the batches need to be small enough for the update to be worth while.

4 Experimental Results

The dataset used for testing in this work was taken from [15]. This dataset is based on IMDB[1] data and contains 8,000 movies with a list of descriptive keywords for each. The movies are tagged with one of the following genres: action, comedy, horror, and romance. The movies are divided equally between the genres. This dataset is split into training and test set with 6,000 items (1,500 items per genre) and 2,000 items in the test set (500 items per genre). We keep each dataset balanced since accuracy is our chosen method of evaluation.

In order to test if the proposed imperfect update method is viable for our purpose we divide the training set further into two sets: one with 1,300 items per genre, which we called set 1300, and two repository sets with 100 items per genre each. From set 1300 we built a BPG where $L = keywords$ and $R = genres$ and a keyword is connected to a genre if it is listed under a movie that is part of the genre. The edge weights of the BPG is the number of times each keyword appeared under each genre.

We performed both an imperfect and a perfect update on set 1300 with the first repository, the result of which we call 1300 imperfect and 1400 respectively. This was done again on 1400 with the second repository, resulting in BPGs 1400

[1] www.imdb.com

imperfect and 1500. It is clear that 1500 is equivalent to the BPG of the full training set. At the end of this process we had five similarity functions.

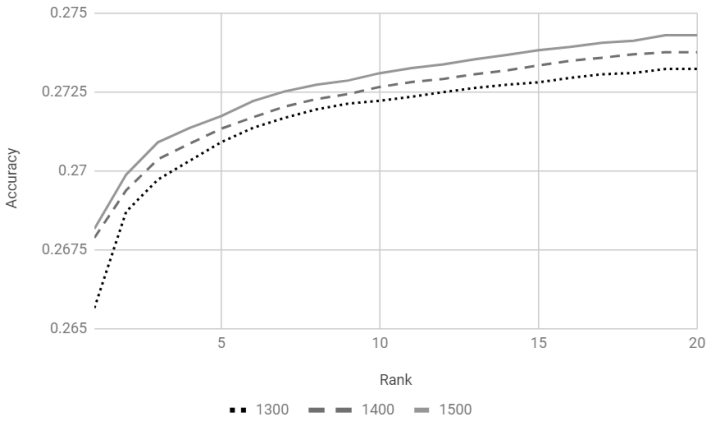

Fig. 6: The retrieval accuracy of the similarity functions derived from 1300 and the two perfect updates

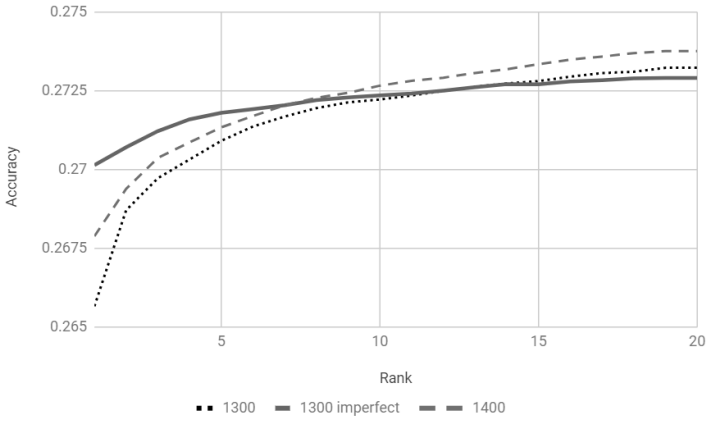

Fig. 7: The retrieval accuracy of the similarity functions derived from one imperfect and one perfect update of 1300

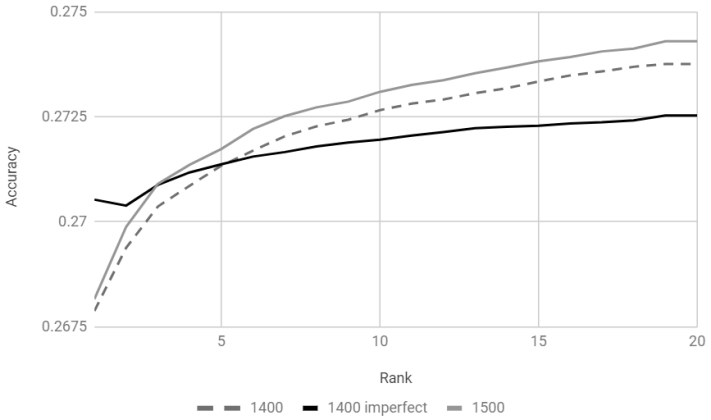

Fig. 8: The retrieval accuracy of the similarity functions derived from one imperfect and one perfect update of 1400

From the test set we built five case bases, one for each similarity function, using the myCBR tool [3], where each case represents a movie and contains a single symbolic attribute with multiple values for the keywords. If a keyword or a link between two keywords did not exist in the aWOMP or the updates, the equality function was used (i.e. $sim(a, b) = 0$ if $a \neq b$ and $sim(a, b) = 1$ if $a = b$). To quantify how well each similarity function performed a retrieval test was done for each case from the test set. A case was deemed as correctly retrieved if it belonged to the same genre of the query case. Using this definition allows us to simulate a diagnostic problem with a dataset that is usually used for recommendation. The aggregated accuracy values were then calculated by retrieval rank (i.e. 1st rank is the case retrieved with the highest similarity value, the 20st rank the case retrieved with the 20st highest similarity value).

We first compare the accuracy results of the two perfect updates. As can be seen in Fig. 6, each update increased the accuracy of the similarity function. It is clear that $1300 < 1400 < 1500$ at least for the first 20 retrieved cases. The first imperfect update, however, does not allow a clear-cut assumption that $1300 < 1300\,imperfect < 1400$. Fig. 7 shows a higher accuracy for the imperfect over the perfect update for the first 7 ranks, and a higher rank over 1300 for the first 13 rank, afterwards its performance is worse than both 1300 and 1400. Similar ratios can be seen for the second imperfect and perfect updates (Fig. 8), and here accuracy becomes worse than the perfect update in even lower ranks.

The questions arise, why does the imprefect update perform better in higher ranks than both perfect measures, and why does it perform worse in lower ranks? In order to answer these questions we first need to look into the properties of the evaluation method. Accuracy by rank can also be seen as accuracy for k-nearest neighbor, i.e. k=rank. The imperfect update, in its nature, is local and therefore

impacts only nodes in its vicinity. This creates a strong positive impact on lower k's, however this impact is diminished when increasing the scope of influence.

These results show that the proposed imperfect update method is not as reliable as perfect updates, however it is still usable as a temporary solution with the assumption that a recalibration with a perfect update is performed in the future.

Discussion The question is asked, when should the perfect and imperfect updates be used? As stated before, adding new information to a dense BPG can cause a wave of changes across the graph, making the perfect update almost as computationally intensive as recalculating the aWOMP from scratch. In this scenario it would be more beneficial to use the imperfect update when new information becomes available, as it has been shown to be reliable *enough*, while a new aWOMP can be recalibrated either periodically or when a predefined number of new cases have been added to the case base. On the other hand, if the graph is sparse then a perfect update may be the best course of action.

5 Related Work

Several works combined complex network analysis (CNA) methods with CBR. Cunnigham et al. in [5] described a system that transforms textual case descriptions into graphs where terms are connected according to the sequence of their appearance in the text. The similarity measure on these cases was then defined as the maximum common subgraph. Although experiments showed promise, the time complexity of the similarity measure is polynomial, making its feasibility for real-world applications difficult.

Another notable work that combined CNA and CBR is Sizov's Text Reasoning Graph (TRG) [13, 12]. Here detailed descriptions of how each case was solved are transformed into a graph representing causal relationships with textual entailments and paraphrase relations. In their first attempt this graph was built only for the solution of the case, while case similarity was based on the vector space model with TF-IDF weights. In their followup work the TRG was incorporated into the case description and the so-call longest common paraphrase was used as the similarity function. In order for this method to be applicable, the solution process needs to be available beforehand for each case, and this is unfortunately hard to obtain for most domains.

There has been some discussion on how symbolic attributes should be compared. A common way to define symbolic attribute similarity is by using a taxonomy [6]. Recently, Bach et al. [2] described a technical fault diagnosis system where fault descriptions were given in textual form. Keywords were extracted using NLP methods and a taxonomy was manually constructed by experts. Reuss et al. described a similar scenario in [10], where keywords were extracted from textual fault descriptions in the technical fault diagnosis domain. Again, taxonomies were manually defined by experts, however they were also automatically supplemented with further information from tools such as wordnet[2].

[2] wordnet.princeton.edu

More complex automated similarity measure have also been discussed in the past. in 1995 Ricci et al. described a reinforcement method for learning a similarity function for symbolic attributes [11], and in 2003, Stahl et al. described an evolutionary algorithm for the same purpose [14]. Both these methods require an arduous learning phase with intensive time complexities. For a static case base and set of attribute values this can be feasibly done offline, however if a new case is added to the case base containing a new keyword retraining the similarity function is required.

Simply weighted one-mode projection (sWOMP) was used by Jimenes-Diaz in a CBR recommendation system for link prediction. The system recommended students with programming tasks based on previously solved task. A BPG was built between students and tasks and the sWOMP was derived on the tasks, where the weights between them were the number of common neighbors in the BPG. In a followup work, Gómez-Martin et al. [8] performed the sWOMP on the users, and a user was recommended a task they hadn't completed yet if a similar user completed it.

6 Conclusions and Future Work

In this work we discussed the aWOMP method in its role as a local similarity method in a diagnosis CBR system. In previous works the aWOMP was seen as a single-pass similarity measure of static symbolic attributes, however, real-world application are dynamic and ever-changing. We presented two update options for aWOMP, the perfect and the imperfect update, which can be used when new cases are added to the case base. The perfect update is equivalent to recalculating the aWOMP from scratch with the updated case base and is computationally intensive for dense BPGs, while the imperfect update allows for local changes on-the-fly that should be usable until a recalibration of the aWOMP is possible.

In our experimentation we showed that a perfect update of the BPG improves the retrieval accuracy, and this is consistent with the notion that more information allows for a better modeling of an environment. The proposed imperfect update method performed better than the perfect update for higher retrieval ranks, however it did not keep its superiority for lower ranks. In the future the imperfect update method can be refined in order to improve retrieval results.

As for the perfect update, in order to allow faster updates that make this solution feasible to be used over many changes to the case base, a map-reduce [7] type system could be used. In an initial architecture, for instance, one could use a sequence of mappers and reducers where the first mapper phase could collect the resources of each keyword while the reducer calculates Eq. 1, and the second mapper would then accumulate the resources that nodes l_i allocate to their neighbors, with the reducer performing a sum function that would result in Eq. 3. Once we know the values of W_i^L and W_j^R, finding the aWOMP is straightforward.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations and system approaches. AI Communications 7(1), 39–59 (1994)
2. Bach, K., Althoff, K.D., Newo, R., Stahl, A.: A case-based reasoning approach for providing machine diagnosis from service reports. In: International Conference on Case-Based Reasoning. Springer Berlin Heidelberg (2011)
3. Bach, K., Sauer, C., Althoff, K.D., Roth-Berghofer, T.: Knowledge modelling with the open source tool mycbr. In: CEUR Workshop Proceedings (2014)
4. Ceausu, V., Desprès, S.: A semantic case-based reasoning framework for text categorization. The Semantic Web pp. 736–749 (2007)
5. Cunningham, C., Weber, R., Proctor, J.M., Fowler, C., Murphy, M.: Investigating graphs in textual case-based reasoning. In: European Conference on Case-Based Reasoning. Springer Berlin Heidelberg (2004)
6. Cunningham, P.: A taxonomy of similarity mechanisms for case-based reasoning. In: IEEE Transactions on Knowledge and Data Engineering 21.11 (2009)
7. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM 51.1 (2008)
8. Gómez-Martin, P.P., Gómez-Martin, M.A.: Case-based recommendation for online judges using learning itineraries. Case-Based Reasoning Research and Development: 25th International Conference, ICCBR (2017)
9. Markovitch, S., Scott, P.D.: The role of forgetting in learning. In: Machine Learning Proceedings (1988)
10. Reuss, P., Stram, R., Juckenack, C., Althoff, K.D., Henkel, W., Fischer, D.: Feature-tak - framework for extraction, analysis, and transformation of unstructured textual aircraft knowledge. In: International Conference on Case-Based Reasoning. Springer (2016)
11. Ricci, F., Avesani, P.: Learning a local similarity metric for case-based reasoning. International Conference on Case-Based Reasoning pp. 301–312 (1995)
12. Sizov, G., Öztürk, P., Aamodt, A.: Evidence-driven retrieval in textual cbr: bridging the gap between retrieval and reuse. In: International Conference on Case-Based Reasoning. Springer International Publishing (2015)
13. Sizov, G., Öztürk, P., Štyrák, J.: Acquisition and reuse of reasoning knowledge from textual cases for automated analysis. In: International Conference on Case-Based Reasoning. Springer International Publishing (2014)
14. Stahl, A., Gabel, T.: Using evolution programs to learn local similarity measures. In: International Conference on Case-Based Reasoning, pp. 537-551. Springer Berlin Heidelberg (2003)
15. Stram, R., Reuss, P., Althoff, K.D.: Weighted one mode projection of a bipartite graph as a local similarity measure. In: International Conference on Case-Based Reasoning. Springer (2017)
16. Zhou, T., Ren, J., Medo, M., Zhang, T.C.: Bipartite network projection and personal recommendation. Physical Review E 76.4, 046115 (2007)