Relevance Matrix Generation Using Sensitivity Analysis in a Case-Based Reasoning Environment

Rotem Stram[1], Pascal Reuss[1,2], Klaus-Dieter Althoff[1,2], Wolfram Henkel[3], and Daniel Fischer[3]

[1] Knowledge Management Group, German Research Center for Artificial Intelligence, Kaiserslautern, Germany
{rotem.stram,pascal.reuss,klaus-dieter.althoff}@dfki.de
[2] Institute of Computer Science, Intelligent information Systems Lab, University of Hildesheim, Hildesheim, Germany
[3] Airbus Operations GmbH, Kreetslag 10, 21129 Hamburg, Germany
{wolfram.henkel,daniel.fischer}@airbus.com

Abstract. Relevance matrices are a way to formalize the contribution of each attribute in a classification task. Within the CBR paradigm these matrices can be used to improve the global similarity function that outputs the similarity degree of two cases, which helps facilitate retrieval. In this work a sensitivity analysis method was developed to optimize the relevance values of each attribute of a case in a CBR environment, thus allowing an improved comparison of cases. The process begins with a statistical analysis of the values in a given dataset, and continues with an incremental update of the relevance of each attribute.
The method was tested on two datasets and it was shown that the statistical analysis performs better than evenly distributed relevance values, making it a suitable initial setting for the incremental update, and that updating the values over time gives better results than the statistical analysis.

Keywords: Relevance Matrix, Sensitivity Analysis, Case-Based Reasoning, Classification, Retrieval, Similarity

1 Introduction

Relevance matrices are an important tool to represent the contribution of each attribute in a classification task. In this paper we will explore their use in the retrieval task of a case-based reasoning (CBR) system, which is closely related to classification. This work is a contribution to the OMAHA project, with the goal of creating a CBR system for aircraft fault diagnosis for Airbus [14].

The idea behind CBR is that similar problems have similar solutions. The process is made of four steps: retrieve, reuse, revise, and retain [1], and lies heavily on methods humans use to solve problems. When a person is faced with a problem she first compares the current situation to past experience. This action

is parallel to the retrieval phase of CBR. Next, she adapts her previous experience to the current task, much like the reuse stage of CBR, and applies it. Her actions are either successful, or unsuccessful (revise step). If her actions are successful, she will remember this solution for future reference (retain step).

A dataset of past experiences, called cases, each comprising of a problem description and a solution, is the foundation of every CBR system. In our scenario of fault diagnosis each case is built out of a fault description comprising of a set of attributes and their values, a diagnosis, and the actions taken to fix the problem, namely the solution. The main focus of this work is the retrieval phase, where a list of cases from the dataset, which are similar to a new fault description entered by the user, are retrieved.

At the base of the case retrieval is the similarity measure, which comes in two forms: local and global. Local similarity measures how similar two attribute values are to each other, while global similarity measures the similarity between two cases (more specifically, two problem descriptions) and is calculated by amalgamating the local similarities. When calculating the global similarity, weights can be assigned to each attribute in the amalgamation function, and here lies the crux of our task.

Our goal is to find the relevance matrix for the attributes used by the cases in our case base. Each case has a diagnosis, and each attribute in the fault description should have a different weight for different diagnoses. We want to set the weights so that for a retrieval task only cases with a relevant diagnosis will be deemed similar enough to be retrieved. Precision is therefore more important in our case than recall.

Similar work has been done in the past by Wess and Richter [7, 8], [10], under the PATDEX/2 system. In this work, three phases were defined to assign values to the relevance matrix: Initial Phase, where starting values are set according to statistical analysis of the attribute values, Training Phase, where the weight values are optimized for classification, and the Application Phase, where weights are constantly updated according to the changing case base. Since both Wess and Richter only used binary attributes, their work is inapplicable to our system in its current form, but instead is used as the basis of this paper, and further developments, mainly in the second phase, will be discussed in the coming sections.

The remainder of this paper is organized as follows. In the next section an overview of the related work, namely finding and assigning weights to features in the CBR environment, is described. In section 3 we define the keywords and basic formulas, which are important for this work. Section 4 describes the method that was developed and used. Then, section 5 provides information on the experiments that were run and their results. Finally, a short discussion of this work is provided, along with ideas of how it can be further developed and used withing the OMAHA project.

2 Related Work

There were several researches done about adjustment of feature weights in the past years, and it is still an important topic. Wettschereck and Aha compared

different feature weighting methods and developed five dimensions to describe these methods: Model, weight space, representation, generality and knowledge [11]. According to their work, our approach uses a wrapper model to optimize the feature weights iteratively during the training phases. The weight space is continuous, because the features of our problem vary in their relevance for different diagnoses. Our knowledge representation is a structural case structure with attribute-value pairs and this given structure is used for feature weighting. We are using case specific weights to set the weights for each diagnosis individually. This way we are able to gain more precise results during the retrieval. Our approach for feature weighting is knowledge intensive, because we are using domain-specific knowledge to differ between individual diagnoses and setting case specific weights.

The approach from Richter and Wess introduces a so-called relevance matrix to deal with irrelevant symptoms in the PATDEX/2 diagnosis system. These relevances are determined in context of special situations. This means that for every diagnosis individual symptom relevances are set. These initial relevances are computed from a given set of cases and improved during training phases [7]. It is important to note that Richter and Wess used only the binary attribute type, and that our work generalizes the analysis to any attribute type.

Another approach for learning feature weights is from Armin Stahl. He presents a framework for learning of similarity measures, which is able to learn local and global similarity measures as well as feature weights. A so-called similarity teacher rates the utility of case pairs with respect to a given query to define the correct order of retrieved cases to the query based on the utility of a case. Based on this correct case order, a so-called similarity learner is able to adjust the similarity measures or feature weights to minimize the error during the retrieval [9].

Zhang and Quang describe an approach for a maintenance system with weight adjustments. They propose a three-layered architecture for a case structure: attribute-value pair layer, problem layer and solution layer. Between the attribute-value pairs and the problem and between the problem and the solution a set of weights can be defined. The feature weights are adjusted based on the feedback of a user, who selects an appropriate solution for a given problem. For each selected solution and the corresponding problem the weights are adjusted [12, 13].

David Aha developed the Case-based Learning Algorithm 4 (CBL 4) as an approach to learn the importance of features. The algorithm sets initial feature weights and then learns the new feature weights during a training phase. A shortcoming of the CBL 4 algorithm is the missing of context consideration. CBL 4 can learn feature weights only for all cases and not case or context specific [2].

3 Definitions

The retrieval task of CBR is based on comparing a new problem description with descriptions stored in the case base, and retrieving only those cases that are *sufficiently similar*, meaning their similarity score is above a predefined threshold.

We define a case base CB as $CB = \{c_1, c_2, ..., c_n\}$ a set of cases c_i, n number of cases. A case is a tuple $c_i = (desc, sol, diag)$, where: $c_i.desc$ a problem description, $c_i.sol$ the description of how the problem was solved, and $c_i.diag$ the diagnosis of the case, i.e. the problem type, which also functions as its class. The difference between the solution and the diagnosis is that the diagnosis is a cluster or set of very similar problems, which may have been solved in different ways. The problem description is a function that maps a set of attributes $A = a_1, a_2, ..., a_m$, m number of attributes, to their values, so that $c_i.desc(a_j)$ is the value of attribute a_j under the case c_i. The solution $c_i.sol$ is a string attribute, while $c_i.diag$ takes symbolic values.

A local similarity is defined as the similarity function of two attribute values

$$sim_{local}(c_i.desc(a_j), c_k.desc(a_j)) = sim_{local}(a_{ij}, a_{kj}) \tag{1}$$

The global similarity function $sim_{global}(c_i, c_k)$ compares two cases, and is defined as the amalgamation of the local similarities:

$$\begin{aligned} sim_{global}(c_i, c_k) = \\ amal(sim_{local}(a_{i1}, a_{k1}), sim_{local}(a_{i2}, a_{k2}), \\ ..., sim_{local}(a_{im}, a_{km})) \end{aligned} \tag{2}$$

Since some attributes are more important than others to determine the similarity of two cases, a weight for each local similarity can be assigned:

$$\begin{aligned} sim_{global}(c_i, c_k) = \\ amal(w_1 \cdot sim_{local}(a_{i1}, a_{k1}), w_2 \cdot sim_{local}(a_{i2}, a_{k2}), \\ ..., w_m \cdot sim_{local}(a_{im}, a_{km})) \end{aligned} \tag{3}$$

Here w_j is the weight of attribute a_j. Another improvement on the similarity function is to give the same attributes different weights under different diagnoses, so that if $c_k.diag = d$, $d \in D$ we have:

$$\begin{aligned} sim_{global}(c_i, c_k) = \\ amal(w_{d1} \cdot sim_{local}(a_{i1}, a_{k1}), w_{d2} \cdot sim_{local}(a_{i2}, a_{k2}), \\ ..., w_{dm} \cdot sim_{local}(a_{im}, a_{km})) \end{aligned} \tag{4}$$

Where w_{dj} is the weight of attribute a_j under diagnosis d. This means that different weights are used for the amalgamation when comparing against cases from different diagnoses. The reasoning behind this is that the same attributes may be differently important to determine membership of different diagnosis sets. From here we come to the relevance matrix (Table 1), which is nothing more than the weights assigned to each attribute under the different diagnoses.

Table 1: The relevance matrix

Diagnosis \ Attribute	a_1	...	a_m
d_1	w_{11}	...	w_{1m}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
d_s	w_{s1}	...	w_{sm}

Our goal is to find the optimal relevance matrix, so that for each new retrieval task only the relevant cases will be retrieved. In our case, relevant cases are those who share the same diagnosis.

4 Method

Following Wess' three phases as first defined for the PATDEX/2 system and discussed in the introduction, we will focus on the first two: the initial phase and the training phase. His methods were updated to accommodate different types of attributes, and the similarity functions in use.

4.1 The Initial Phase

The purpose of the initial phase is to determine the starting weight values for the optimization that is taking place in the training phase. The initial weights are calculated using a statistical analysis of the attribute values.

Symbolic Attributes We begin by looking at symbolic attributes, where there is a finite set of possible values. We define a diagnosis set $C_j \subseteq CB$ as the set of all cases c_i with $c_i.diag = j$. Let $|D| = s$ be the number of possible diagnoses, $a \in A$ an attribute, and $B = \{b_1, b_2, ..., b_t\}$ the set of *possible* values for attribute a. We also set b_{ji} to be the number of appearances of value b_i in C_j under the attribute a.

The relative weight of value b_i in the diagnosis set C_j is calculated as:

$$w_{ji} = \frac{b_{ji}}{\sum_{x=1}^{s} b_{xi}} \tag{5}$$

The impact of the value on the diagnosis set is then $V_{ji} = \frac{b_{ji}}{|C_j|}$. To calculate the weight of attribute a under the diagnosis set C_j, the following formula is used:

$$W_{ja} = \sum_{x=1}^{|B|} V_{jx} \cdot w_{jx} \tag{6}$$

General Attributes Since possible attribute values in general is not limited to a finite set, the formula to calculate the attributes' weights needs to be adjusted. Let $a \in A$ be defined as before, $B = \{b_1, b_2, ..., b_t\}$ the set of *used* values under a in all diagnosis sets, $B_j \subseteq B$ is the set of values that appear in C_j, $b_{ji} \in B_j$, $r \in \mathbb{R}$ a similarity threshold. The relative weight of b_{ji} is then

$$w_{ji} = \frac{|\{b_{jx} | b_{jx} \in B_j \wedge sim_{local}(b_{ji}, b_{jx}) \geq r\}|}{|\{b_x | b_x \in B \wedge sim_{local}(b_{ji}, b_x) \geq r\}|} \tag{7}$$

Since we regard each value as unique, the impact of each value on C_j is $V_{ji} = \frac{1}{|B_j|}$. The total weight of attribute a under C_j is

$$W_{ja} = \frac{1}{|B_j|} \cdot \sum_{i=1}^{|B_j|} w_{ji} \tag{8}$$

4.2 The Training Phase

After determining the initial weights our job is not done. In order to optimize the relevance matrix we need to train the system. For a given query a list of cases is retrieved, and sorted according to their similarity to the query, as given by the similarity function sim_{global}. We choose a threshold ξ, such that if $sim_{global}(query, case) \geq \xi$ the diagnosis of both *query* and *case* should be the same. In case $query.diag \neq case.diag$ and $sim_{global}(query, case) = \xi + \Delta$, we have a false positive results, which can be attributed to one of the following reasons [10]:

1. Both *query* and *case* contain the same problem description, such that $query.desc = case.desc$
2. The threshold ξ is too low
3. The weight of the similar attributes of *query* and *case* is too high
4. The weight of the dissimilar attributes of *query* and *case* is too low

If reason 1 is the source of the false positive result and the problem descriptions are identical then we either have an inconsistency problem and *case* should be removed from the case base, or *case* is incomplete and should be updated. Neither of these scenarios are within the scope of this work. We will ignore reason 2 since we want to keep a fixed threshold, and will so focus on reasons 3 and 4. Our goal is then to strengthen the differences between *query* and *case*, and weaken their similarities.

Consider the following scenario: We have a query case q and a retrieved case c with $c.diag = d$, $q.desc \neq c.desc$ and

$$\begin{aligned} sim_{global}(q, c) = \xi + \Delta = \\ w_{d1} \cdot sim_{local}(a_{q1}, a_{c1}) + w_{d2} \cdot sim_{local}(a_{q2}, a_{c2}) + \\ ... + w_{dm} \cdot sim_{local}(a_{qm}, a_{cm}) \end{aligned} \tag{9}$$

We define a local similarity threshold ξ' for the local similarity measure, and the following sets: $S = \{i|sim_{local}(a_{qi}, a_{ci}) \geq \xi'\}$ and $NS = \{j|sim_{local}(a_{qj}, a_{cj}) < \xi'\}$ the sets of those attributes which are similar and those that are not similar, respectively. We want to update the similarity measure so that $sim_{global}(q, c) \leq \xi$. In order to do so, the similarity formula is updated in the following way:

$$sim_{global}(q, c) =$$
$$\sum_{i \in S}(w_{di} - z_i) \cdot sim_{local}(a_{qi}, a_{ci}) + \sum_{j \in NS}(w_{dj} + y_j) \cdot sim_{local}(a_{qj}, a_{cj}) = \xi \quad (10)$$

This way the weight of similar attributes is reduced, while the weight of dissimilar attributes is increased for the diagnosis set of the retrieved case, thus increasing the dissimilarity between the two cases. In order to update the weights we need to find the values of z_i and y_i. For simplicity reasons we set $\forall_{z_i, z_j} z_i = z_j, \forall_{y_i, y_j} y_i = y_j$ and $\sum_{i \in S} z_i = \sum_{j \in NS} y_j = Z$. This means that we need to find a value Z, such that

$$z_i = \frac{Z}{|S|}, y_i = \frac{Z}{|NS|} \quad (11)$$

This value is calculated with the help of the following formula:

$$Z = \frac{\Delta \cdot |S| \cdot |NS|}{|NS| \cdot \sum_{i \in S} sim_{local}(a_{qi}, a_{ci}) - |S| \cdot \sum_{j \in NS} sim_{local}(a_{qj}, a_{cj})} \quad (12)$$

The derivation of formula 12 is: let $t = sim_{global}(q, c) - \Delta$, $x_i = sim_{local}(a_{qi}, a_{ci})$, then:

$$\sum_{i \in S} x_i \cdot w_{di} + \sum_{j \in NS} x_j \cdot w_{dj} = t + \Delta \implies$$
$$\sum_{i \in S} x_i \cdot (w_{di} - z_i) + \sum_{j \in NS} x_j \cdot (w_{dj} + y_j) = t \implies$$
$$\sum_{i \in S} x_i \cdot (w_{di} - \frac{Z}{|S|}) + \sum_{j \in NS} x_j \cdot (w_{dj} + \frac{Z}{|NS|}) = t \implies$$
$$\sum_{i \in S}(x_i \cdot w_{di} - \frac{x_i \cdot Z}{|S|}) + \sum_{j \in NS}(x_j \cdot w_{dj} + \frac{x_j \cdot Z}{|NS|}) = t \implies$$
$$\sum_{i \in S} x_i \cdot w_{di} + \sum_{j \in NS} x_j \cdot w_{dj} - \frac{Z}{|S|} \cdot \sum_{i \in S} x_i + \frac{Z}{|NS|} \cdot \sum_{j \in NS} x_j = t \implies$$
$$t + \Delta - Z \cdot (\frac{\sum_{i \in S} x_i}{|S|} - \frac{\sum_{j \in NS} x_j}{|NS|}) = t \implies$$
$$\Delta = Z \cdot \frac{|NS| \cdot \sum_{i \in S} x_i - |S| \cdot \sum_{j \in NS} x_j}{|S| \cdot |NS|} \implies$$
$$Z = \frac{\Delta \cdot |S| \cdot |NS|}{|NS| \cdot \sum_{i \in S} x_i - |S| \cdot \sum_{j \in NS} x_j} \quad Q.E.D$$

For a falsely retrieved case, the diagnosis set's weights are updated as follows:

1. Find the sets S and NS
2. Calculate Z according to formula (12)
3. Calculate z_i, y_j according to formula (11)
4. $\forall_{i \in S} w'_i = w_i - z_i$, $\forall_{j \in NS} w'_j = w_j + y_j$
5. Set all w'_i, w'_j as the new weights of the respective attributes for the diagnosis set of the retrieved case

One epoch of the training procedure would then go as follows:

- For each case c in the case base
  - Build a query q from the problem description of c
  - Use q to retrieve similar cases
  - For each retrieved case c_i, $sim_{global}(q, c_i) > \xi$, $c_i.desc \neq c.desc$
    - Update the weights of the diagnosis set of $c_i.desc$

There are two main problems with this procedure; First, the weight update sequence depends both on the order of the queries and on the order of retrieved cases as the weights are updated. Another problem is that after the first weight update for a query q, the second update for a second falsely retrieved case may not be relevant any longer, as the weights of that diagnosis set may have already changed. A renewed retrieval for the same query after each weight change is computationally too expensive and might even produce an endless loop. In order to overcome these problems the training phase is changed so that weights are updated only once per query.

In the new version, an average Z value, $aveZ$, is calculated for each diagnosis set as the mean Z from all falsely retrieved items of this set within an entire epoch. The local update values are then: $z_i = \frac{aveZ}{|S|}$, $y_j = \frac{aveZ}{|NS|}$, where $|S|$ and $|NS|$ are the number of attributes that appeared more times in S or in NS respectively.

The attribute weights for each set are then updated only once per epoch: For each attribute a_i, let $|S_i|$, $|NS_i|$ be the number of retrieved cases of the diagnosis set where a_i appeared under S or NS respectively. The weight of a_i in the diagnosis set is then updated as follows:

$$w'_i = \begin{cases} w_i - z_i & \text{if } |NS_i| = 0 \\ w_i + y_i & \text{if } |S_i| = 0 \\ w_i - (1 - \frac{|S_i|}{|NS_i|}) \cdot z_i & \text{if } |S_i| < |NS_i| \\ w_i - (1 - \frac{|NS_i|}{|S_i|}) \cdot y_i & \text{if } |NS_i| < |S_i| \end{cases} \tag{13}$$

This phase is closely related to the back-propagation training of an artificial neural network (ANN). In each retrieval or classification task the contribution of each attribute, much like each neuron, to the error is assessed, and its weight, similarly to activation weights, is updated accordingly.

5 Evaluation and Results

To test the newly developed method we used two datasets from the literature, since an aircraft-related one is not yet readily available. Both datasets were taken from the classification discipline, as each item is assigned to a class, something that is lacking in available CBR data.

The method was implemented in the Java programming language and relied heavily on the myCBR tool [3] both in the creation of the case bases, and for retrieval. The datasets which were used are the *Car Evaluation Data Set* [4] and the *Yeast Data Set* [5], both obtained from the UC Irvine Machine Learning Repository[1], and from here on referred to as Cars and Yeast respectively. These datasets were chosen thanks to the ease of converting them into a case base, and their size, which allowed testing in a timely manner.

The method's thresholds were set to the following values: $\xi = \xi' = 0.75$. The settings for the local similarities were set with the distance function "DIFFERENCE" and type "POLYNOMIAL_WITH".

Since both datasets have a relatively low number of data points, it was necessary to perform cross validation for the training phase. Each dataset was randomly divided into four subsets, and the initialization and training phases were performed four times on the training set, each time using a different subset as the test set.

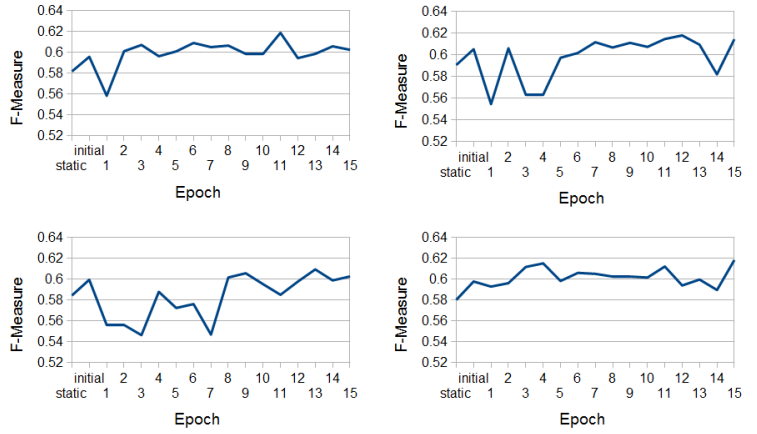
Before the training phase two tests were performed: first, with a static and uniform weight for all attributes, and second using the weights from the initial phase. Following this, the weights were trained for 15 epochs on the training set. The recall and precision results of each epoch was documented, and the F-measure was calculated with $\beta = 0.3$, giving a heavier weight to the precision. The results can be seen in figure 1, and show the F-measure values as calculated from the test set.

As can be seen, the F-measure value of the Cars dataset was higher for the initial weights than the static ones. When the training phases begins the retrieval performance decreases drastically, and then slowly picks up and reaches values higher than those of the initial phase. The situation of the Yeast dataset is different, as the peak of the F-measure is reached after the first training epoch, and values then slowly decrease and approaches what seams to be a saturation point that is above both the static and the initial phase. What can be learned from these results is that the statistical analysis improves performance over the evenly distributed weights, while training improves performance over the statistical analysis.

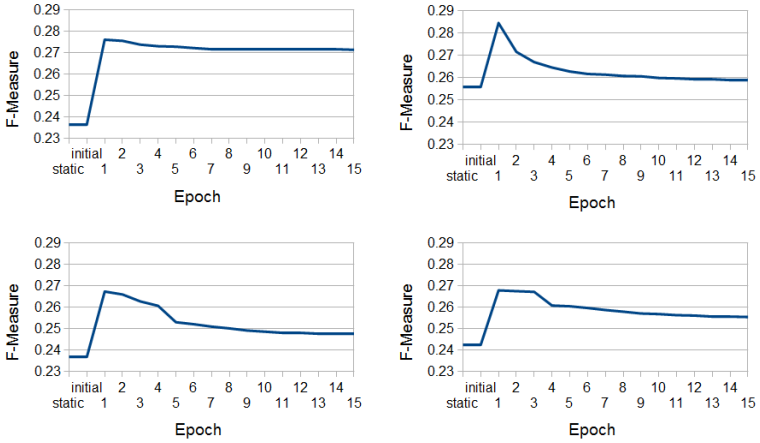
6 Discussion and Future Work

Sensitivity analysis was used in this work to optimize the relevance matrix, which represents the importance of each attribute in a global similarity. The optimization method was developed as part of an aircraft fault diagnosis CBR

[1] http://archive.ics.uci.edu/ml



(a) Cars



(b) Yeast

Fig. 1: F-measure as a function of each setting and epoch. Each graph shows results of one cycle of cross-validation

tool, with the goal of improving the retrieval of past cases similar to a query case.

With the help of two datasets from the literature, the method was tested on three different settings: static and evenly distributed relevance values, initial statistically analyzed values, and 15 epochs of optimization. Performance of the different settings was measured with the help of the F-Measure, giving higher importance to precision than to recall .

It was shown that the initial values, obtained by statistical analysis, allowed the retrieval to perform better than the evenly distributed values. However, when the relevance matrix was updated with the help of the sensitivity analysis, the performance was improved even more for both datasets.

In the future, we intend to test the optimization method on a real-world aircraft fault dataset. This was not done yet since no such dataset is readily available, but this will change in the near future, as one is currently in the making. Once the test is performed we will better know how the system behaves, and will be able to adjust the termination criteria of the optimization cycles. The sensitivity analysis will be instated as part of the toolchain of the OMAHA project, which is a framework to transform a semi-structured dataset into cases, and to retrieve cases relevant to a query [6].

References

1. Aamodt, A., & Plaza, E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications, 7(1), 39-59 (1994)
2. Aha, D. W. Case-based learning algorithms. In Proceedings of the 1991 DARPA Case-Based Reasoning Workshop Vol. 1, pp. 147-158 (1991)
3. Bach, K., Sauer, C. S., Althoff, K. D., & Roth-Berghofer, T. Knowledge Modeling with the Open Source Tool myCBR. In KESE@ ECAI (2014)
4. Bohanec, M., & Rajkovic, V. Knowledge acquisition and explanation for multi-attribute decision making. In 8th Intl Workshop on Expert Systems and their Applications pp. 59-78 (1988)
5. Horton, P., & Nakai, K. A probabilistic classification system for predicting the cellular localization sites of proteins. In Ismb Vol. 4, pp. 109-115 (1996)
6. Reuss, P., Althoff, K. D., Henkel, W., Pfeiffer, M., Hankel, O., & Pick, R. Semi-automatic Knowledge Extraction from Semi-structured and Unstructured Data Within the OMAHA Project. In Case-Based Reasoning Research and Development pp. 336-350. Springer International Publishing (2015)
7. Richter, M. M., & Wess, S. Similarity, uncertainty and case-based reasoning in PATDEX. In Automated Reasoning pp. 249-265. Springer Netherlands (1991)
8. Richter, Michael M. Classification and learning of similarity measures. In Information and Classification pp. 323-334. Springer Berlin Heidelberg (1993)
9. Stahl, A. Learning feature weights from case order feedback. In Case-Based Reasoning Research and Development pp. 502-516. Springer Berlin Heidelberg (2001)
10. Wess, S. Fallbasiertes Problemlösen in Wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik. Ph.D thesis. TU Kaiserslautern (1995)
11. Wettschereck, D., & Aha, D. W. Weighting features. In Case-based reasoning research and development pp. 347-358. Springer Berlin Heidelberg (1995)

12. Zhang, Z., & Yang, Q. Towards lifetime maintenance of case base indexes for continual case based reasoning. In Artificial Intelligence: Methodology, Systems, and Applications pp. 489-500. Springer Berlin Heidelberg (1998)
13. Zhang, Z., & Yang, Q. Dynamic refinement of feature weights using quantitative introspective learning. In IJCAI pp. 228-233 (1999)
14. German Aerospace Center - DLR, LuFo-Projekt OMAHA gestartet, http://www.dlr.de/lk/desktopdefault.aspx/tabid-4472/15942_read-45359/