Efficient Retrieval for Case-Based Reasoning

David Patterson, Niall Rooney, Mykola Galushka
Northern Ireland Knowledge Engineering Laboratory (NIKEL)
Faculty of Informatics
University of Ulster
Jordanstown
N. Ireland
wd.patterson, nf.rooney, mg.galushka@ulster.ac.uk

Abstract

In this work two indexing approaches are presented for case-based reasoning. The first is a hybrid technique which uses a combination of a matrix structure and a tree structure to solve problems. The matrix and tree structures index cases by their discretised feature values. The second approach is based solely on the tree structure and never uses the matrix. The two techniques are evaluated in terms of their competency and efficiency with respect to nearest neighbor retrieval. Both approaches provide average efficiency gains of up to 20 fold in comparison to nearest neighbour with only a slight loss in competency, as averaged across all case-bases tested. It is argued that these approaches are appealing due to their simplicity, competency and efficiency.

Introduction

The nearest neighbor (NN) algorithm is a commonly used similarity metric in Case-Based Reasoning (CBR). Its appeal includes its simplicity, its transparency, its robustness in the presence of noise and the fact that it does not require training. Over the years researchers have studied the nearest neighbor algorithm in detail to try and improve upon its competency. For example its noise tolerance has been improved by retrieving k nearest cases and introducing a 'voting' scheme to combine the various predictions [20]. Attribute values have been weighted according to their significance to help deal with the curse of dimensionality [20], cases themselves have been weighted to increase the retrieval probability of more competent cases [3,2,10] and approaches for improving the manner in which symbolic attributes are dealt with proposed [15,2,3]. All of these improvements have focused on improving the competency of the NN algorithm. A major drawback of the algorithm, due to its exhaustive search, remains its efficiency. This is especially poor for large case-bases or those of high dimensionality. A serious consequence of this poor retrieval efficiency in CBR is an exacerbation of the effects of the utility problem [17].

The utility problem, which is seen in many problem solving systems, manifests itself as a reduction in the efficiency (average problem solving time) of a problem solving system as new knowledge is learned. It is classically observed in first principles reasoners such as speed up learners, but has also been observed in CBR systems which do not have a first principles reasoner at their core [16]. In CBR the time taken to solve a problem is composed of the time required to search the case-base to retrieve an appropriately similar case or cases plus the time taken to adapt the retrieved case's solution. The less similar a target problem and retrieved case are, the more processing is required during adaptation and the longer the problem solving process will be [18]. Therefore to improve efficiency, compelling arguments exist which advocate continually adding cases to a case-base over time to try and maximise problem space coverage thus reducing the amount of adaptation required and therefore by default, the time taken to solve problems. Although this may initially seem to be logical unfortunately in reality it has not been shown to be true [16] as in some situations the overall time taken to solve a problem actually increases with the addition of a new case to the case-base. This is known as the utility problem in CBR.

A number of techniques have been developed to reduce the effects of the utility problem in CBR. All can be regarded as knowledge maintenance strategies, which add a considerable overhead to the housekeeping functions of the CBR

system and all focus on the improved management of either similarity (retrieval) knowledge or case knowledge. Most of these techniques focus on ensuring that the basic problem solving ability (competence) of the CBR system is maintained whilst reducing retrieval time. These techniques fall into two main categories, namely, policies which do not limit the size of the case-base in achieving the maximal problem space coverage possible and those which do. The former use indexing structures to improve search efficiency while the latter, which include case addition and deletion policies, rely on the fact that the number of cases present in the case-base will never be allowed to become so large so as to affect the search efficiency of the retrieval process significantly. Overviews of these strategies are now presented.

Indexing

Case indexing has been widely applied in CBR as a method of improving search efficiency to combat the effects of the utility problem. As only a selective portion of the case-base is made available during retrieval, search time is reduced and the efficiency of identifying a possible solution is increased dramatically. Unfortunately building and maintaining a competent index is not an easy task as it is highly dependent on the retrieval circumstances, which are constantly changing. Therefore the indexing structure (retrieval knowledge) of the case-base must be maintained to reflect this. Additionally the addition of new cases to the case-base over time must be accommodated. This maintenance of retrieval knowledge can be a major burden on efficiency in CBR systems. If the indexing scheme is poor or maintenance is ignored, cases with good solutions to the target problem may be overlooked as they reside in a different part of the case-base inaccessible under the current indexing scheme. This can lead to the complex adaptation of less suited cases, a reduction in competency and in severe situations, problem-solving failures. Therefore, due to poor retrieval knowledge or insufficient maintenance, in an attempt to improve efficiency, competency is often sacrificed [11].

A number of researchers have applied indexing strategies to CBR. Aha [1] presented a methodology of continually refining a case library in the domain of conversational case-base reasoning to improve both competency and efficiency. Deangdej [4] devised a dynamic indexing structure to retrieve cases at run time from a case-base of over two million cases. Fox [5] developed an introspective reasoning technique for dynamically refining case indexes based on previous retrievals. Smyth has devised an indexing scheme based on a case competence model [18], which improves both retrieval competency and efficiency. Nearest neighbour retrieval can be improved by space partitioning techniques such as k-d trees [19]. However such techniques are restricted to continuous attributes.

Addition

The basic theory behind addition policies is to prevent the case-base from ever becoming so large that efficiency is detrimentally affected. The majority of these techniques focus on some concept of problem space coverage provided by the cases in the case-base and aim to build efficient case-bases with the maximal coverage using the minimal number of cases. Space prevents an in depth analysis of these techniques but McSherry [8], Zhu & Yang [21] and Portinale et al [13] have all proposed case addition strategies which limit the size of the case-base.

Deletion

Most case deletion policies are also based on some notion of the problem space coverage provided by cases. Deletion is a very difficult strategy to implement in CBR as some cases are inevitably more expendable than others. This is due to the fact that cases are the basic unit of both *competency* and *efficiency* in a case-base [16], [13]. A number of deletion policies have been proposed by researchers [18], [7].

The main difficulty with both deletion and addition policies is the efficiency overhead they impose on the system. Another drawback is that by limiting the size of the case-base, knowledge will inevitably be lost to the system. It is widely recognized that knowledge can be converted from one knowledge container into another in CBR systems [14]. For example Hanney [6] and McSherry [8] have demonstrated how case-knowledge can be used as a source of adaptation knowledge. Patterson et al. [10] have demonstrated that case-knowledge can also act as a rich source of similarity knowledge. Therefore by excluding some of the cases vital knowledge could be lost to the system.

In this work we report on advances to the Discretised Highest Similarity with Pattern Solution Re-use algorithm, (D-HS+PSR) [12], within the M^2 CBR framework [9], specifically analyzing its improvements over NN in terms of efficiency and its ability to competently index case-bases. We present two algorithms for analysis. The first, D-HS+PSR(II) is a hybrid retrieval technique, which uses a combination of a matrix and a tree structure for retrieval. The tree is used to provide solutions to problems already encountered in the lifetime of the system, whereas the matrix solves new (previously un-encountered) problems from scratch and then updates the tree with this knowledge for reuse in the future. The second technique, D-HS+PSR(III), uses the matrix to build the tree structure initially, but it ignores the matrix during retrieval, converts the tree into a case vector representation and only uses this to solve target

problems. In this technique, if a target problem has not been encountered before a solution is formed from what are judged to be the most similar cases, using Euclidean distance as a measure. Both techniques can be used for classification and regression type problems.

The motivations for developing these techniques were three-fold. We wanted to develop a means of similarity determination, which was more efficient than the k-NN algorithm (as measured by a comparison of the speed of the technique) whilst maintaining similar competency (as measured by the accuracy of prediction). Secondly we were interested in developing an approach, which could help reduce the effects of the utility problem with a minimal overhead in terms of processing. Finally we wanted to ensure that the new approach was easily maintained in real time

Methodology

Here we describe the theory behind the construction of the matrix and tree structure used in the D-HS+PSR(II) technique and describe how they combine to solve problems. Each case-base was split into training and test cases. Each case was composed of a combination of numeric and/or nominal values. The training cases were processed to create a matrix *M* where each cell *M(i,j)* contains a list of cases whose normalized attribute value x for attribute *i* lies in an interval $1/d_i * (j-1) <= x < 1/d_i * j$. For nominal attributes the value of d_i is simply the number of possible attribute values. For continuous numeric attributes, we chose d_i to have the value of 10 so in essence numeric attributes are discretised into 10 intervals. This discretization parameter was found from previous experiments to give competent results [12]. A target case was said to match a case from the case-base on a particular attribute value if both of their attribute values fell into the same discretised interval as defined in equation 1.

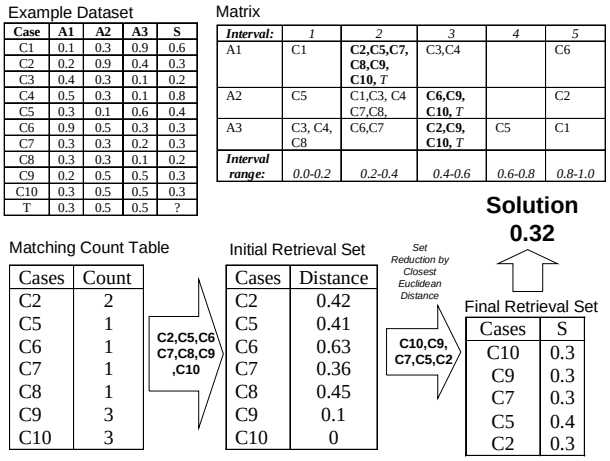
$$similarity(C_1,C_2) = \sum_{n=1}^{d} match(C_1(n),C_2(n)) \quad (1)$$

where
d is the number of attributes
$C_1(n)$ is the case's nth attribute value
and $match(C_1(n),C_2(n)) = 1$ if $C_1(n)$ and $C_2(n)$ lie in the same discretised interval, 0 otherwise.

At least N (N = 10 in our experiments) cases from the matrix, with highest matching count were considered for retrieval as part of an *initial retrieval set* for a Target. This set was reduced down to a *final retrieval set* of 5 entries by taking the 5 closest according to a Euclidean distance metric. This *final* retrieval set was then used to predict an average value in the case of a continuous output field, and a majority class for classification problems.

Figure 1 shows an example of how the matrix was constructed using the example case-base where cases consist of 3 attributes (A1-A3) and a solution field S. It also describes the retrieval process for a target T. For simplicity it shows how each case attribute value was discretised into one of 5 intervals. A count was kept of the number of times the attributes of each case in the case-base fell into the same discretised intervals as the corresponding attribute of the target, T. All cases, which overlap with T, are shown in bold in Figure 1. This means that the similarity function has maximal value if two cases agree on all their attribute values and 0 if they agree on none of their attribute values.

This matrix approach could be speeded up further by creating a tree like representation of the cases in the case-base. It was seen that, especially for larger case-bases, there was a high probability that a number of different target cases shared the same matrix retrieval pattern (as a result of the attribute value discretisation process) as cases in the case-base.



Case	A1	A2	A3	S
C1	0.1	0.3	0.9	0.6
C2	0.2	0.9	0.4	0.3
C3	0.4	0.3	0.1	0.2
C4	0.5	0.3	0.1	0.8
C5	0.3	0.1	0.6	0.4
C6	0.9	0.5	0.3	0.3
C7	0.3	0.3	0.2	0.3
C8	0.3	0.3	0.1	0.2
C9	0.2	0.5	0.5	0.3
C10	0.3	0.5	0.5	0.3
T	0.3	0.5	0.5	?

Interval:	*1*	*2*	*3*	*4*	*5*
A1	C1	**C2,C5,C7, C8,C9, C10,** *T*	C3,C4		C6
A2	C5	C1,C3, C4 C7,C8,	**C6,C9, C10,** *T*		C2
A3	C3, C4, C8	C6,C7	**C2,C9, C10,** *T*	C5	C1
Interval range:	*0.0-0.2*	*0.2-0.4*	*0.4-0.6*	*0.6-0.8*	*0.8-1.0*

Cases	Count
C2	2
C5	1
C6	1
C7	1
C8	1
C9	3
C10	3

Cases	Distance
C2	0.42
C5	0.41
C6	0.63
C7	0.36
C8	0.45
C9	0.1
C10	0

Cases	S
C10	0.3
C9	0.3
C7	0.3
C5	0.4
C2	0.3

Fig 1 Retrieval using the matrix in D-HS+PSR(II)

An auxiliary retrieval tree structure was therefore built, which mirrored the exact matrix intervals each particular case fell into for each of its attributes. Figure 2 shows a partial tree for cases 1, 5, 9 & 10. Attribute decision nodes, A1,A2,A3, discriminate among the matrix intervals a case's attribute falls into. At each leaf node of the tree is stored the actual cases which are indexed via the particular retrieval pattern of its attributes. Utilizing the tree, to exactly match a target case's interval pattern, could negate the need for a matrix lookup when the pattern is recognized by the tree. If the target interval pattern is not recognized, then the matrix can still be used to solve the problem. From Figure 2 it can be seen that each level of the tree corresponds to a specific attribute and each node in the tree can potentially

be split into the number of discretised intervals for that attribute. Each node has child pointers to the next attribute. The tree structure shown in Figure 2 corresponds to the same cases in the case-base as shown in the previous example. The 1st attribute of case 1 falls into interval 1, its 2nd attribute falls into interval 2 and the 3rd attribute falls into interval 5, so the tree is traversed as shown and case 1 stored at the leaf node and can be indexed by a retrieval pattern of {1,2,5}. The diagram demonstrates how the target case T with retrieval pattern {2,3,3} is solved using the tree. The retrieval patterns within the tree can be viewed as new 'case structures', which can be used to solve target problems. They can be seen as a generalization of the case knowledge within the matrix.

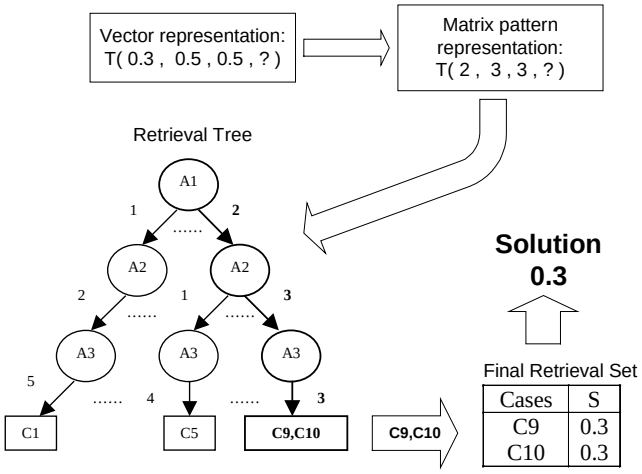


Figure 2 Retrieval using the tree in D-HS+PSR(II)

Figure 3 shows the architecture of this new system and it can be seen that initially retrievals are attempted from the tree (the primary case-base where an exact match is required for retrieval) and the matrix (the secondary case-base) is now only required during retrieval if a target case's pattern has not been encountered before and therefore not recognized by the primary case-base. In this situation the matrix is used to solve the problem and the cases in the initial retrieval set are added to the primary case-base, indexed by the target case's pattern and made available for future retrievals. Therefore the system learns over time increasing re-use and improving efficiency.

It should be noted that the solutions produced by the tree may be different than if the matrix alone were used. This is because the tree only uses cases to form the solution that are indexed by the target's whole pattern, whereas the matrix allows partial overlap. Obviously it is faster to form a solution from the tree than the matrix, as the tree look-up of the initial retrieval set can be done in a time factor proportional to the number of attributes, whereas the matrix formation of the initial retrieval step is dependent on how many cases have some overlap with the target case.

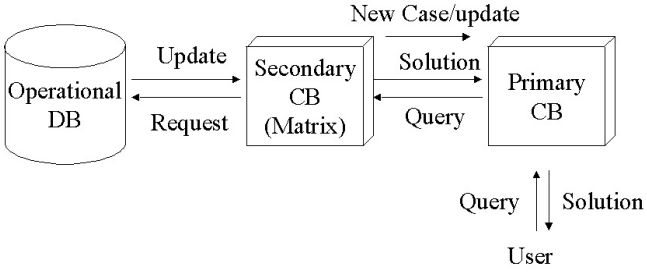


Figure 3 Architecture of D-HS+PSR(II) System

This technique has an initial overhead in setting up the primary case-base but for large data sets with good problem space coverage this should be compensated for by the improved speed by which problems can be solved by using the primary case-base as opposed to the matrix. A major advantage of this technique is that updating the system with new cases can be achieved in real-time, as neither the matrix or tree structure need to be re-structured to allow for the addition of new cases.

DHS+SR(III) was developed to investigate the efficiency and competency of the technique when using the primary case-base alone to solve problems. Here the tree structure was converted into a case vector representation, where each attribute of the case corresponded to the nodes in the tree, e.g. case 1 has a case representation of {1, 2, 5}. Euclidean distance was used to determine similarity. Obviously this may decrease efficiency compared to D-HS+PSR(II) but as the number of cases are less than in the original case-bases (see discussion on compression later), the search space is reduced and it may be more efficient than k-NN on the original case-bases.

Experimental Technique and Results

Five case-bases, obtained from the UCI machine learning repository, were used in the following experiments. In each of the experiments 10-fold cross validation was used to build a model using the training cases. The test cases were used as targets to test the efficiency and competency of the model.

Table 1 shows the % accuracies (or MAE in the case of abalone) for 5 case-bases using the D-HS+PSR(II) retrieval approach compared to k-NN[2]. It also shows the percentage of retrievals which used the primary case-base (% P). Additionally the speedup ratio of retrieval compared to k-NN retrieval is shown.

[2] 4 case-bases are classification problems while 1, Abalone, is a prediction problem. This was included to highlight the flexibility of the approach to solving both types of problems.

Case-Base	DHS+PSR(II) Competency	kNN Competency	% P	Efficiency Ratio
Adult	82.0	82.6	44.1	14.4
Abalone	1.66	1.6	89.5	54.6
Letter	91.5	95.5	20.1	14.5
Mushroom	100	100	0	9.4
Waveform	74.2	79.5	0	8.3

Table 1 Results for the D-HS+PSR(II) retrieval technique

Note that these ratios take into consideration the time required to build the matrix structure, create the primary case-base and then retrieve a solution from either the primary case-base or matrix. From this it can be seen that the competency of retrieval is similar to k-NN (a drop of 3.13% averaged over all case-bases) but with an improvement in efficiency. Efficiency varies from 8.3 times more efficient with Waveform to almost 55 times more efficient with Abalone and taken as an average over all case-bases the approach is seen to be 20.2 times faster. Also shown are the percentage of times a solution is reused from the primary case-base. This varies from 0% with Waveform and Mushroom, indicating that a pattern is never reused (i.e. a case's attributes never exactly overlaps with a target's attributes), to 89.5% with Abalone. These results demonstrate that, the more often retrievals are made using the primary case-base the greater the efficiency gain over k-NN will be. This is because despite there being an initial overhead in setting up the primary case-base, this is compensated for by the savings in time achieved by solving numerous problems using the tree as opposed to the matrix. Abalone, which has a high percentage of reuse, provides the greatest time savings compared to k-NN, whereas Waveform and Mushroom with no reuse provide the least improvement in time. It would be possible to analyze the case-base in advance to determine whether to use the auxiliary tree store. As both Waveform and Mushroom consist of entirely unique case patterns, we could have predicted that there would be no advantage to having an auxiliary tree store. The fact that some efficiency improvement is noted with Waveform and Mushroom is solely due to the speed of the matrix technique, compared to k-NN, which compensates for the extra time required to build the tree even though it is never used in retrieval

Table 2 shows results for D-HS+PSR(III) in terms of competency, efficiency ratio over k-NN and percentage reduction in the size of the search space compared to the original case-base. The percentage reduction in search space obtained varies from case-base to case-base and is expressed as the number of cases in the primary case-base as a percentage of the total number of original cases. The amount of 'compression' obtained is determined primarily by the number of distinct case patterns in the case-base in comparison to the size of the original case-base. The higher the percentage reduction, the fewer the number of distinct patterns and the more efficient the technique will be.

Case-Base	DHS-PSR(III) Competency	Efficiency Ratio	% Reduction in cases
Adult	80.6	6.5	32
Abalone	1.7	73.7	79
Letter	93.2	2.3	12.5
Mushroom	100.0	0.9	0
Waveform	72.3	0.9	0

Table 2 Results for the D-HS+PSR(III) retrieval technique

From this it can be seen that, with Mushrooms, competency is the same as D-HS+PSR(II), it is better than D-HS+PSR(II) with Letter, and worse with the other 3 case-bases. A decrease in competency of 4.8 % averaged over all case-bases was observed with D-HS+PSR(III) compared to k-NN. This is 1.67% on average less competent than D-HS+PSR(II). As for the efficiency of the algorithm, all case-bases gave better efficiencies than k-NN apart from Waveform and Mushroom, which were slightly less efficient. This was because creating the tree did not reduce the search space at all (see Table 2, the reduction in cases = 0% for Waveform and Mushroom) but there is the extra overhead of creating the tree in the first place. All case-bases were slower than D-HS+PSR(II) with the exception of Abalone. This indicates that the efficiency of D-HS+PSR(II) may be better for case-bases with less compression whereas whenever there is a large reduction in the search space, as with Abalone, an exhaustive NN search may be more efficient using the primary case-base alone as opposed to using the matrix and tree retrieval approach of D-HS+PSR(II). The average ratio of efficiency gain over k-NN was 17.9, which compares to a ratio of 20.2 for D-HS+PSR(II). From this we can conclude that D-HS+PSR(II) is more efficient and on average more competent than D-HS+PSR(III).

Conclusions

In this paper we present two related approaches to improve the efficiency of retrieval in CBR. As such they could be used to reduce the effects of the utility problem in CBR due to the fact they would enable the construction of a much larger case-base before the effects of the utility problem became apparent. Both are considerably faster than k-NN but this efficiency improvement is at the expense of a slight drop in competency. D-HS+PSR(II) is more efficient and on average more competent than D-HS+PSR(III). The only proviso with this is, if there is a large amount of compression achieved by creating the primary case-base, then D-HS+PSR(III) may be more efficient. In general therefore D-HS+PSR(II) proved to be the more desirable technique.

The matrix component of D-HS+PSR(II) can also be used to highlight the areas of the problems space, where the system presently lacks knowledge to solve problems, thus drawing attention to these areas for future case knowledge acquisition. One approach to discovering any missing matrix knowledge would be to rely on the expert to actively acquire it but a more appealing and elegant approach would be to use machine-learning techniques to automatically discover the missing case knowledge. The matrix would be used in this instance as a knowledge acquisition tool. This will be the focus of future work within this system. Other issues for further study include the relationship between case knowledge and adaptation knowledge and an improvement to both techniques so they can more efficiently deal with nominal attributes. In addition, a study of variable sized matrix intervals and the weighting of attributes and retrieval cases will be carried out.

References

[1] Aha, D. W. and Breslow, L. Refining conversational case libraries. In Proceedings of the 2nd International Conference on Case-based Reasoning,-ICCBR-97, pp 267-276, Providence RI, USA, 1997.

[2] Anand, SS; Patterson, DW and Hughes, JG. Knowledge Intensive Exception Spaces, AAAI-98, pp 574-579, 1998.

[3] Cost, S.; and Salzberg, S. 1993. A Weighted Nearest Neighbour Algorithm for Learning with Symbolic Features. *Machine Learning* 10: 57-78.

[4] Deangdej, J., Lukose, D., Tsui, E., Beinat, P. and Prophet, L. Dynamically creating indices for two million cases: A real world problem. In Smith, I. And Faltings, B. eds., *Advances in Case-Based Reasoning, Lecture Notes in AI*, pp 105-119, Springer Verlag 1996.

[5] Fox, S. and Leake, D.B. Using Introspective reasoning to refine indexing. In roceedings of the 14th International Joint Conference on Artificial Intelligence. Montreal, Canada, August , pp 391-387, 1995.

[6] Hanney, K. and Keane M. Learning Adaptation Rules from a Case-Base, Proc. Advances in Case-Based Reasoning, 3rd European Workshop, EWCBR-96, pp179-192, Lausanne, Switzerland, November 1996.

[7] Hunt, J.E., Cooke, D.E. and Holstein, H. Case-memory and retrieval based on the immune system. 1st International Conference on Case-Based reasoning, pp 205-216, 1995.

[8] McSherry, D. Automating case selection in the construction of a case library. Proceedings of ES99, the19th SGES International Conference on Knowledge-Based Systems and Applied Artificial Intelligence, Cambridge, pp 163-177, December 1999

[9] Patterson, D., Anand, S.S., Dubitzky, D. and Hughes, J.G. Towards Automated Case Knowledge Discovery in the M^2 Case-Based Reasoning System, *Knowledge and Information Systems*:An International Journal, (1), pp 61-82, Springer Verlag, 1999.

[10] Patterson, D; Anand, SS; Dubitzky, D and Hughes, JG. A Knowledge Light Approach to Similarity Maintenance for Improving Case-Based Competence. Workshop on Flexible Strategies for Maintaining Knowledge Containers 14th European Conference on Artificial Intelligence, ECAI 2000. PP 65 - 77. 2000

[11] Patterson, D., Rooney, N. & Galushka, M. Towards Dynamic Maintenance of Retrieval Knowledge in CBR. Proceedings of the 15th International FLAIRS Conference. AAAI Press. 2002

[12] Patterson, D., Rooney, N. & Galushka, M. Efficient Similarity Determination and Case Construction Techniques For Case-Based Reasoning. 4th European Conference on CBR, pp 292-305, 2002.

[13] Portinale, L., Torasso, P. and Magro, D. Dynamic Case Memory Management, Proc. ECAI 98, pp. 73-78, John Wiley and Sons, Brighton, 1998.

[14] Richter, M. The Knowledge Contained in Similarity Measures. Invited Talk, The 1st International Conference in Case-Based Reasoning, Sesimbra, Portugal, 1995.

[15] Stanfill, C.; and Waltz, D. Towards Memory-based Reasoning. *Communications of the ACM*. 29(12): 1213-1228, 1986.

[16] Smyth, B. and Keane, M. Remembering to Forget.: A Competence-Preserving case Deletion Policy for Case-Based Reasoning Systems. Proceedings of 14th IJCAI, pp377-382. 1995.

[17] Smyth, B. Constructing Competent Case-Based Reasoners: Theories, Tools and techniques. Workshop On Automating the Construction of Case-Based Reasoners, Sixteenth International Joint Conference on Artificial Intelligence, Stockholm, Sweden, pp 17-23, 1999

[18] Smyth, B. and McKenna, E. Footprint-based retrieval. Proceedings of the 3rd International Conference on Case-Based Reasoning , pp 343-357. July 1999.

[19] Wess, S., Althoff, K., Richter, M., Using k-d trees to improve the retrieval step in case-based reasoning, *Topics in Case-based Reasoning, First European Workshop (EWCBR-93)*, Springer-Verlag, pp. 67-81, 1993.

[20] Wettschereck et al. Wettschereck, D.; Aha, D.; and Mohri, T. A Review of Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms, *Artificial Intelligence Review Journal*, 1997.

[21] Zhu, J. and Yang, Q. Remembering to Add: Competence-preserving Case Addition Policies for Case-base Maintenance. In International Joint Conference in Artificial Intelligence 1999 (IJCAI-99), pp. 234-239, 1999.