Keep It Simple: A Case-Base Maintenance Policy Based on Clustering and Information Theory

Qiang Yang and Jing Wu

[1] School of Computing Science
Simon Fraser University
Burnaby, BC Canada, V5A 1S6
[2] Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Abstract. Today's case based reasoning applications face several challenges. In a typical application, the case bases grow at a very fast rate and their contents become increasingly diverse, making it necessary to partition a large case base into several smaller ones. Their users are overloaded with vast amounts of information during the retrieval process. These problems call for the development of effective case-base maintenance methods. As a result, many researchers have been driven to design sophisticated case-base structures or maintenance methods. In contrast, we hold a different point of view: we maintain that the structure of a case base should be kept as simple as possible, and that the maintenance method should be as transparent as possible.
In this paper we propose a case-base maintenance method that avoids building sophisticated structures around a case base or perform complex operations on a case base. Our method partitions cases into clusters where the cases in the same cluster are more similar than cases in other clusters. In addition to the content of textual cases, the clustering method we propose can also be based on values of attributes that may be attached to the cases. Clusters can be converted to new case bases, which are smaller in size and when stored distributedly, can entail simpler maintenance operations. The contents of the new case bases are more focused and easier to retrieve and update. To support retrieval in this distributed case-base network, we present a method that is based on a decision forest built with the attributes that are obtained through an innovative modification of the ID3 algorithm.

1 Introduction

Case based reasoning [18,9,10] is a technique to reuse past problem solving experiences to solve future problems. The basic idea is based on analogy, whereby similar problems are found and their solutions are retrieved and adapted for solving the new problem. The effectiveness of a CBR system critically depends

© Springer-Verlag Berlin Heidelberg 2000

on the speed and quality of the case base retrieval process. If the retrieved cases are not accurate or the retrieval performance is too low, then a CBR system cannot function as expected. If too many seemingly similar solutions are retrieved, as in the case of some web browsers where thousands of items are returned, a CBR system cannot provide its users with much assistance either. The purpose of this paper is to present a novel case-base maintenance and retrieval system aimed at improving the accuracy and performance of a CBR system when the number of cases gets large.

Our approach is based on two related ideas. The first one is clustering, whereby a large case base is decomposed into groups of closely related cases. Based on the partitioned structure, we then create a collection of distributed case bases across a network on different sites, where each element in the distributed case base structure is one cluster created as a result of the clustering process. From each cluster we build a representative case, which takes a subset of the attributes (or "features" in some literature) in that cluster as its representation. This step builds a two-level hierarchical structure.

Our second idea is to allow a user to retrieve the distributed case bases by incrementally selecting the attributes that are information-rich and can cover the entire distributed case base structure. These attributes are presented to the user in an interactive manner, whereby a user chooses some attributes to provide values with. In each iteration, a subset of the case-base clusters are removed from consideration. The process repeats until a target case base is identified. At this point, a case based reasoning system is used to rank and identify the cases in the case base in order to find the final answer.

Our work is motivated by our experience in case base maintenance research. Case-base maintenance refers to the task of adding, deleting and updating cases, indexes and other knowledge in a case base in order to guarantee the ongoing performance of a CBR system. Case-base maintenance is particularly important when a case based reasoning system becomes a critical problem solving system for an organization. Recently, there has been a significant increase in case-base maintenance research. One branch of research has focused on the ongoing maintenance of case-base indexes through training and case base usage [5,7,2,21]. In [13], a prototype-based incremental network is proposed to accelerate the indexing process in CBR. This network is built on an abstract hierarchy of a given case base, which is a clustering structure for the case base. Researchers have also focused on increasing the overall competence of the case base through case deletion [19,6,1,19,16]. Leake and Wilson [11] provide an excellent survey of this field.

Most existing works on case-base maintenance are based on one of two general approaches: either build an elaborate case-base structure and maintain it continuously, or use a sophisticated algorithm to delete or update a case base when its size reaches a certain threshold. In contrast, we take a different view in this paper. We maintain that to ensure effective case-base maintenance, both the structure of the cases and the maintenance method must be simple. This philosophy is based on the following observations on the changing landscape of computing in general. First, in recent years we have seen that disk space is getting increasingly cheaper and networked databases are increasingly more efficient. This makes it highly feasible to store gigabyte of case data and retrieve the data efficiently. Second, the approach that maintains a highly structured hierarchical CBR system in order to maintain a case base is likely to encounter the recursive problem of maintaining the structure itself on a continuing basis. Third, case-base maintenance algorithms that delete cases are likely to be short-sighted and can easily erase invaluable corporate memory which may prove to be useful in the long run. These algorithms are often sophisticated, thus incurring high computational overhead.

Based on these observations, we have adopted a new strategy whereby we keep all cases and use simple method to maintain the case bases. Our idea is to create multiple, small case bases that are located on different sites. Each small case base contains cases that are closely related to each other. Between different case bases the cases are farther apart from each other. Our approach is to allow the cases to be added and deleted at each small case base without affecting the whole. This distributed concept ensures that each case base is small and it is easier to maintain each one individually. Figure 1 illustrates the architecture.

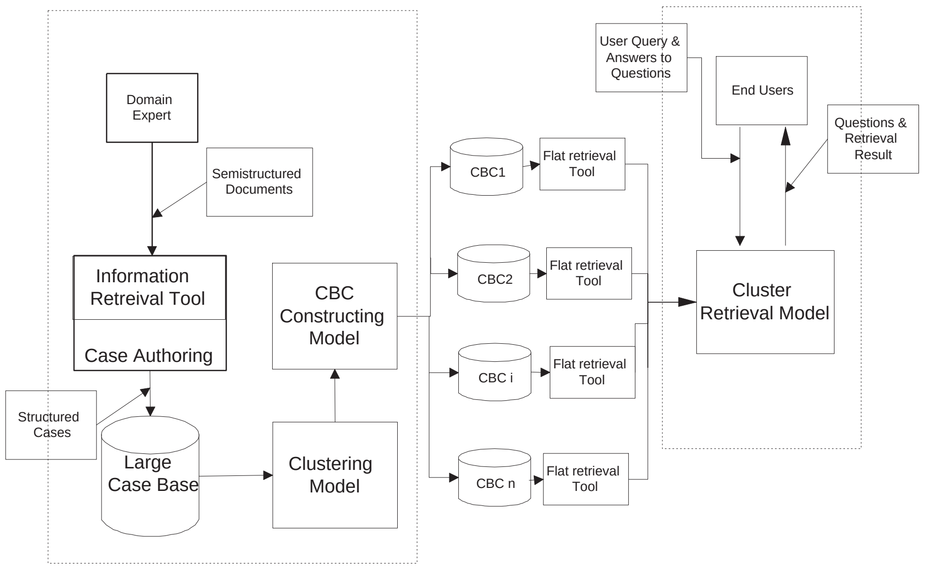

Fig. 1. System Architecture. In the figure "CBC" refers to case base clusters.

2 Case Base Clustering Using CBSCAN

In this section we present an algorithm which uses cluster analysis to build a case base maintainer. The basic idea is to apply clustering analysis to a large case base and efficiently build natural clusters of cases based on the density of attribute values.

2.1 Clustering Techniques in CBR

Clustering technique is applicable to CBR because each element in a case base is represented as an individual, and there is a strong notion of a distance between different cases. In the past, some attempts in inductive learning and neural network have been made in applying clustering techniques to case-based reasoning [12]. The basic idea of the inductive methods is to build a classification tree based on analysis of information gain that are associated with each attribute or question used to represent a case [3,4,12]. A drawback of inductive approaches are that they are expensive to run for large case base [4,14]. When case bases change, it is necessary to rebuild the whole structure. It is also difficult to tackle the problem of missing values; to compute a decision tree, most inductive methods assume that all cases are associated with all attributes. From our experience in applying case based reasoning, this assumption is generally false.

We choose a density-based clustering method as our basis. The density-based method overcomes the defects of inductive clustering methods and many other clustering methods because it is relatively efficient to execute and does not require the user to pre-specify the number of clusters. Density-based methods are based on the idea that it is likely that cases with the same attributes should be grouped into one cluster. Intuitively, a cluster is a region that has a higher density of points than its surrounding region. For a case, the more cases that share the same attributes with it, the larger its density is. The density-based method originates from a method called GDBSCAN [17,20], proposed for data mining. The main feature of the algorithm is that it relies on the density of data, so that it can discover clusters of arbitrary shape which is very important for CBR to group all similar cases together.

More specifically, GDBSCAN accepts a radius value *Eps* based on a distance measure, and a value $MinPts$ for the number of minimal points. The latter is used to determine when a point is considered dense. Then it iteratively computes the density of points in an N-dimensional space, and groups the points into clusters based on the parameters *Eps* and $MinPts$. The algorithm is found to outperform many well known algorithms such as K-prototype. However, a main problem of GDBSCAN is that a user must input a radius value *Eps* in order to construct the partitions. However, if the user is not a domain expert, it is difficult to choose the best value for *Eps*. In response, we have developed a method for finding a near-optimal *Eps* value through a local search process. We call this new algorithm *CBSCAN*. CBSCAN is based on the observation that the minimum radius value *Eps* is critical in determining the quality of a partition. Thus, a local-search algorithm is used to find a locally optimal *Eps* value that optimizes a certain quality measurement.

Before introducing CBSCAN, we must first discuss how to measure the quality of a given partition of a case base. We use the new Condorcet criteria (NCC) which is based on the idea that for a partition to be good cluster is a good partition has small intra-cluster distances and large inter-cluster distances. More precisely, let Y be a partition; it is a set of clusters. Y can be represented as a matrix such that $y_{ij} = 1$ if and only if cases i and j are in the same cluster. The quality of a partition $NCC(Y)$ can be represented as :

$$\sum_{i=1}^{n} \sum_{j \neq i} (m - 2d_{ij}) y_{ij} = \sum_{i=1}^{n} \sum_{j \neq i} C_{ij} * y_{ij} \tag{1}$$

In this equation, n is the total number of cases, d_{ij} is the distance between two cases i and j, m is the number of attributes or features in a domain, and $C_{ij} = m - 2d_{ij} = (m - d_{ij}) - d_{ij}$ is the difference between the number of agreements between any two elements i, j who are in the same cluster and the number of disagreements between them. The NCC measures the quality of a partition by ensuring that it will not favor large number of clusters. Therefore it can be used as a criterion to optimize the clustering result. For numerical attributes, the distance measures can be modified to be categorical by discretization.

2.2 The CBSCAN Algorithm for Clustering Case Bases

We now introduce the case-base clustering algorithm CBSCAN. In our tests (not shown in this paper due to space limitations) we know that the parameter *MinPts* is not critical in the definition of density, so we arbitrarily set the *MinPts* as a constant such as 2. To find a value for *Eps* in order to get a good partition, we modify *Eps* by the NCC of the partition. In CBSCAN, *Eps* is always moved toward the trend that leads to a larger NCC value. When NCC first increases and then decreases with the change in *Eps* value, we know that we have passed by a locally maximal NCC point. We then let *Eps* to oscillate around this point until an approximate *Eps* value that produces the locally optimal NCC value is found.

Algorithm 2.1 NCC-based clustering algorithm CBSCAN

```
Input: A case base D
Output: A set of clusters to partition the case base D
Method
    1. Initiate parameters:
           MinPts = 2,
           Eps = maximal distance between any two cases
           BestQuality is set a very small number
           BestEps = Eps
           Direction = +1;
           K is a constant larger than 1
```

```
2. While( iteration times $\leq$ MaxIterNumber)
      GDBSCAN(Eps, MinPts);
      CurrentQuality= ComputeQuality( Eps );
      If( CurrentQuality $\geq$ BestQuality )
            BestQuality = CurrentQuality;
            BestEps = Eps;
      Else
            Direction = - Direction ;
            $EpsInterval = EpsInterval/K$;
      End if
      If( Direction $\geq$ 0 )
            Eps = Eps - Epsinterval;
      Else
            Eps = Eps + EpsInterval;
      End if
      iteratenumber++;
   EndWhile
```

The function ComputeQuality() returns the NCC value for the current partition. We will show the test results of this algorithm in Section 4. For each case in the case base, we need to check whether it is dense. This process is repeated m times, so the total time for this algorithm is, in fact, $O(m * n * \log n)$.

2.3 Building Distributed Case Bases

After a large case base is partitioned, a domain expert can build some smaller case bases on the basis of clustering result. We call the large case base the *OLCB* (original large case base), and the smaller case base built with a cluster the *CBC* (case base cluster). Each *CBC* has a case base name and a list of keywords. The case name is the description of the case base. It is a set of the most frequently used words by the cases in the case base. There is a set of attributes associated with the case base. They are all the attributes that are associated with the cases in the case base. The weight of the cluster with that attribute-value pair is the average weight of all the cases in the cluster.

Consider an example case base for a Cable-TV diagnosis application. Suppose the that the initial case base consists of 8 cases in which some are for diagnosing VCR problems and others for TV problems. As shown in Figure 2, after clustering, Case1, ..., Case4 are grouped into Cluster1 which is about VCR, and Case 5, Case 6 and Case7 are grouped into Cluster2, which is about some TV problems, while Case 8 is the noise. The domain expert builds a case base "VCR" for Cluster 1 under a directory called "VCRdomain" and "TV" for cluster 2 under a directory called "TVdomain".

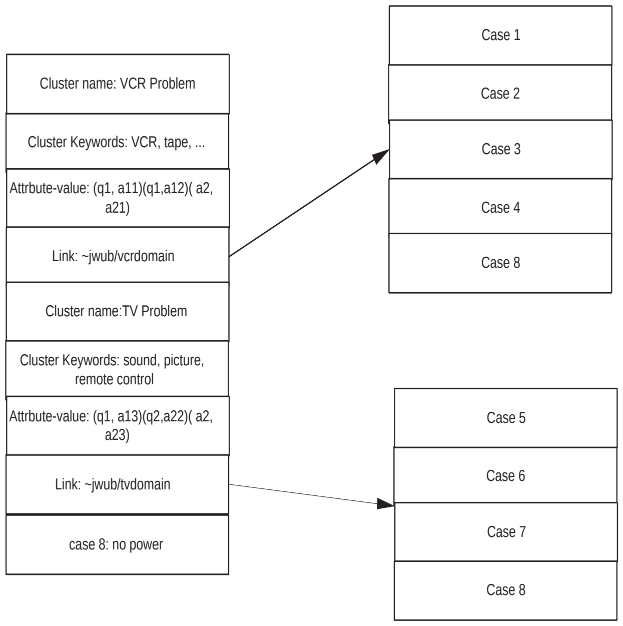
Cluster name: VCR Problem
Cluster Keywords: VCR, tape, ...
Attrbute-value: (q1, a11)(q1,a12)(a2, a21)
Link: ~jwub/vcrdomain
Cluster name:TV Problem
Cluster Keywords: sound, picture, remote control
Attrbute-value: (q1, a13)(q2,a22)(a2, a23)
Link: ~jwub/tvdomain
case 8: no power
Case 1
Case 2
Case 3
Case 4
Case 8
Case 5
Case 6
Case 7
Case 8

Fig. 2. Clustering a Cable-TV case base

3 Information-Guided Cluster-Retrieval Algorithm

In this section we present our second step: retrieve the most similar case-base cluster (CBC) by analysis of information gains of the attributes. Our method is summarized as the follows:

- combine information theory with CBC retrieval to find the attribute that can mostly distinguish the CBCs built with the algorithms introduced in Section 2.
- deal with CBCs that are irrelevant with a selected attribute. To do this a decision forest instead of a decision tree is built. The roots of the decision trees in the forest cover all the CBC's under them.
- allow the users to interactively browse the CBCs to find the most similar CBC instead of traditional searching the pre-built structure.
- create the decision forest dynamically based on information theory as the users narrow down their search.

Given a collection of case bases, we want to select a subset of the attributes to present to the user. A user can choose among this set of attributes a subset to provide answers or values with. For example, in a retrieval process, a user might be given attributes a_1, a_2, and a_4. The user might choose to answer a_1 and a_4. These answers will eliminate a subset of clusters from consideration and promote another subset as possible candidates. The system ranks those case base clusters that are highly likely to contain the final result based on these answers and allow the user to continue browsing into any chosen subsets. The process continues until a final cluster is identified. At this point, a simple CBR system can be used on this case base for case retrieval.

There are several requirements for this process. First, to ensure coverage, the attributes selected by the retrieval algorithm must cover all case base clusters. For most case base applications, not all cases are associated with all attributes. Therefore, no single attribute may cover all case base clusters. Thus, we must select more than one attribute so that the whole subset will cover the case bases. This induces a decision forest instead of a decision tree. Second, we still wish the attributes we present to the user will have the maximal information value.

Our attribute-selection algorithm is informally described as follows. First, for all attributes that are associated with the case bases, we calculate their information-gain ratios based on Quinlan's algorithm [15]. Then, we iteratively select a collection of attributes so that all case bases in a current "candidate set" are covered. Then, we present those attributes in the form of questions to the user, and obtain values as answers to the questions.

We now illustrate this algorithm through an example. Suppose there are ten cases in a case base. Five CBCs were built, each holding two cases. The descriptions of cases are represented by four attributes. Each case is associated with some attribute-value-weight tuples. The case base is shown in Table 3.

Case Name	**Attr 1**	**Attr 2**	**Attr 3**	**Attr 4**	**CBC No.**
Case 1	(a,100)		(a,100)	(a,100)	1
Case 2	(a,100)		(a,100)	(b,100)	1
Case 3	(a,100)		(b,100)	(c,100)	2
Case 4	(a,100)		(b,100)	(d,100)	2
Case 5		(a,50)	(d,100)	(a,100)	3
Case 6		(a,50)	(a,100)	(b,100)	3
Case 7		(a,100)	(c,100)	(a,100)	4
Case 8		(b,100)	(d,100)	(a,100)	4
Case 9		(b,100)		(b,100)	5
Case 10		(b,100)		(c,100)	5

Table 1. Example Case Base For CBC Retrieval

The attribute with the largest information-gain ratio for all the CBC's is located as the root of a decision tree. The CB retrieval system sorts all the attributes by their information gain ratio. The information gain ratio for the attributes are listed in Table 3. Attribute 2 has the largest information gain ratio. It is set as the root for the first decision tree in the decision forest. Before Attribute 2 is returned to the user, it is checked whether Attribute 2 covers all the CBCs. We discover that Attribute 2 only covers CBC 3, CBC 4 and CBC 5. Thus, we continue to calculate the information-gain ratio of Attribute 1, 3

Attribute	**Information Gain Ratio**
Attribute 2	8.72
Attribute 4	6.99
Attribute 3	4.15
Attribute 1	0

Table 2. Sorted Attributes Information-gain ratio for all CBCs

and 4 for CBC 1 and CBC 2. Attribute 3 has the largest gain and it covers the remaining two CBCs. Since Attribute 2 and Attribute 3 cover all the CBCs, these two are returned to the user. The decision forest is illustrated in Figure 3.

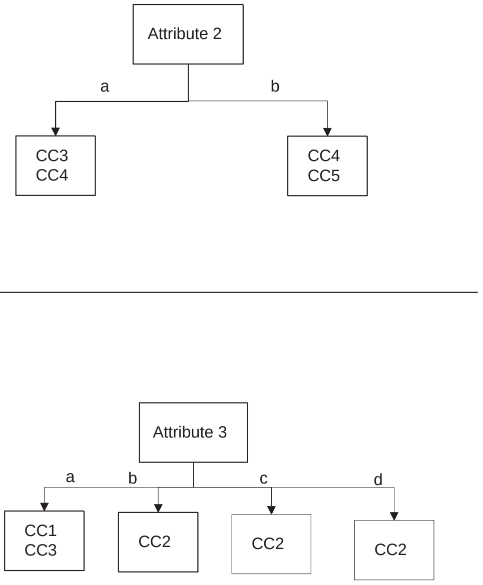

Fig. 3. Decision Forest Example

4 Empirical Tests for Clustering

So far, we have introduced a novel clustering method CBSCAN and a information-gain guided cluster-retrieval method into the framework of CBR. This system is implemented in a case-based reasoning system. In this section, we demonstrate the effectiveness of the algorithms through experiments. All the algorithms are implemented in Microsoft Visual C++^{TM}. All experiments have been run on Pentium PC with 166 MHz and 96 MB memory.

In this section, we test a number of hypotheses which are raised throughout this paper.

- We show through an experiment that the choice in *Eps* value which we mentioned in Section 2 has a dramatic effect on the quality of the resultant partition generated using the density based methods. Our experiment shows that Algorithm CBSCAN can indeed find an Eps point where a near optimal clustering result is defined.
- We show that our clustering method CBSCAN can indeed scale up to large case bases, and can in fact outperform well known algorithms such as the K-prototype method [8] or ID3 [15].

In this experiment, we used a case base with 45 cases in text format from a realistic Cable-TV case base. The text file has the structure as shown below,where each case has a case name, a problem description and a solution.

1. Name: no picture; white screen; faulty TV, descrambler, converter
2. Reason: Faulty Cable-TV set but the descrambler may be the problem.
3. Solution: Connect cable directly to TV; if there is a picture, the descrambler is the problem.

Figure 4 illustrates how *Eps* varies with the NCC quality measurement. We can tell that after eight repetitions in Algorithm CBSCAN 2.1, the *Eps* found is almost optimal. In particular, the *Eps* is first set to 0.92, the maximum distance between cases. When *Eps* = 0.53, the clustering algorithm will get the optimal clustering result where the quality defined by NCC is 13.67. The sequence of the *Eps* is $0.92 \rightarrow 0.77 \rightarrow 0.62 \rightarrow 0.47 \rightarrow 0.50 \rightarrow 0.53 \rightarrow 0.53$. It takes 7 times to get the best result. In the last two steps, the *Eps* values are all 0.53, because the increase at this point is less than 0.01 which is the precision defined by the system.

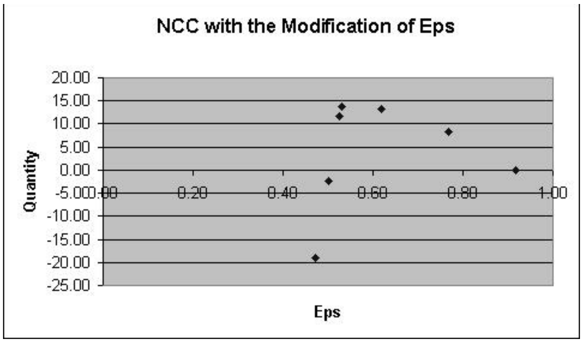

Fig. 4. NCC as a function of Eps

The total CPU time spent in finding the clusters and the final clustering quality values computed using NCC (Equation 1) are shown in Table 3. The

Clustering Method	Time(seconds)	New Cordecet Criteria
CBSCAN	0.7	13.67
K-ProtoType	12	-1075
ID3	66	-120

Table 3. CPU Time and NCC Results on a Cable-TV Case Base

ID	#Cases	#Attributes	CBSCAN	K-Pro	ID3
1	1118	12	49872	-383764	95585
2	2076	20	39645	-17878	36794
3	3804	25	-249271	-98863	-1432227
4	7000	30	13397	-1330	5698

Table 4. Experiment Results on NCC

extended clustering algorithm CBSCAN (Algorithm 2.1) is compared with well known clustering algorithms k-Prototype [8] where the parameter k is set as 5, and with the ID3 algorithm [15].

Next, we used data from UCI repository of machine learning databases and domain theories (http://www.ics.uci.edu/ mlearn/MLRepository.html). The attributes in the case bases are categorical. In case based reasoning, an attribute is not associated with every case especially when a case base is large, because it is common that some attributes may not be relevant to a case. This missing-value phenomenon was not common in the data from the UCI repository, because the number of missing values in the case base is very small. To simulate real case bases where there may be lots of missing values, We merged several case bases together. If an attribute appear in two case bases, these two attributes will be merged as one. This modification created many missing values in the resultant case brae (the total size is 8,000 cases).

There are 4 groups of data in this test. The clustering result is shown in Table 4 where the last three columns show the quality measure of the computed clustering according to NCC. The CPU time result is shown in Table 5 with size of a case base. From these results, we can see that, with a few exceptions, our CBSCAN algorithm easily outperforms both K-prototype and ID3 algorithms in both CPU time and result quality.

ID	#Cases	#Attributes	CBSCAN	K-prototype	ID3
1	1118	12	8	16	29
2	2076	20	21	53	40
3	3804	25	29	79	126
4	7000	30	99	147	223

Table 5. Experiment Results on CPU Time

5 Conclusions and Future Work

This work has made two linked contributions. First, a new case-base clustering algorithm is presented that is both efficient and effective in dealing with large case bases. Second, the clustered case bases are organized into a distributed structure so that an interactive process can proceed for a user to identify the target cases. Since the size of a cluster is relatively small, any simple CBR retrieval method can be used once a target case base found in the second step. These two algorithms support our initial philosophy of maintaining large case bases by keeping all cases around, and maintain only the simplest structure.

References

1. D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
2. D.W. Aha and L. Breslow. Refining conversational case libraries. In *Proceedings of the Second International Conference on Case-Based Reasoning, ICCBR-97*, pages 267–276, Provedence RI, USA, 1997.
3. E. Auriol, S. Wess, M. Manago, Althoff. K.D., and R." Traph. Inreca: A seamlessly integrated system based on inductive inference and case-based reasoning. In *Case-Based Reasoning Research and Development*, pages 371–380, 1995.
4. S.K. Bamberger and K. Goos. Integration of case-based reasoning and inductive learning methods. In *First European Workshop on CBR*, 1993.
5. P. Cunningham, A. Bonzano, and B. Smyth. Using introspective learning to improve retrieval in car: A case study in air traffic control. In *Proceedings of the Second International Conference on Case-Based Reasoning, ICCBR-97*, pages 291–302, Providence RI, USA, 1997.
6. P. Domingos. Rule induction and instance-based learning. In *Proceedings of the Thriteenth International Joint Conference on Artificial Intelligence*, pages 1226–1232, San Francisco, CA, USA, 1995. Morgan Kaufmann.
7. S. Fox and D. B. Leake. Learning to refine indexing by introspective reasoning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 1995.
8. Z.X. Huang. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the first Pacific-Asia Conference on Knowledge-Discovery and Data Mining*, 1997.
9. Janet Kolodner. *Case-based Reasoning.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993.
10. David Leake. *Case-based Reasoning – Expriences, Lessons and Future Directions.* AAAI Press/ The MIT Press, 1996.
11. D.B. Leake and D. Wilson. Categorizing case-base maintenance: Dimensions and directions. In *Advances in Case-Based Reasoning: Proceedings of EWCBR-98.* Springer-Verlag, Berlin, 1985.
12. M. Malek. A connectionist indexing approach for cbr systems. In *Case-Based Reasoning Research and Development*, 1995.
13. M. Malek. A connectionist indexing system for cbr systems. In *Proceedings of the First International Conference on Case-Based Reasoning, ICCBR-95*, Sesimbra, Portugal, 1995. Springer-Verlag.

14. M. Manago, K. Althodff, E Auriol, R. Traphoner, S. Wess, N. Conruyt, and F. Maurer. Induction and reasoning from cases. In *First European Workshop on CBR*, 1993.
15. J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
16. Kirsti Racine and Qiang Yang. Maintaining unstructured case bases. In *Proceedings of the Second International Conference on Case-Based Reasoning, ICCBR-97*, pages 553–564, Providence RI, USA, 1997.
17. J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2), 1988.
18. R.C. Schank. *Dynamic Memory: A Theory of Learning incomputers and People.* Cambridge University Press, 1982.
19. B. Smyth and M. Keane. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *International Joint Conference on Artificial Intelligence*, volume 1, pages 377–382, 1995.
20. X. Xu, Ester M., K. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the 14th International Conference on Data Engineering*, 1998.
21. Zhong Zhang and Qiang Yang. Towards lifetime maintenance of case based indexes for continual case based reasoning. In *Proceedings of the 8th International Conference on Artificial Intelligence: Methodology, Systems, applications*, Sozopol, Bulgaria, 1998.