Applied Intelligence 14, 49–64, 2001

Enhancing the Effectiveness of Interactive Case-Based Reasoning with Clustering and Decision Forests

QIANG YANG
School of Computing Science, Simon Fraser University, Burnaby, BC Canada, V5A 1S6
qyang@cs.sfu.ca

JING WU
AMDOCS Limited, Mount Pleasant Rd, Toronto, Canada
jingw@amdocs.com

Abstract. In interactive case-based reasoning, it is important to present a small number of important cases and problem features to the user at one time. This goal is difficult to achieve when large case bases are commonplace in industrial practice. In this paper we present our solution to the problem by highlighting the interactive user-interface component of the CASEADVISOR system. In CASEADVISOR, decision forests are created in real time to help compress a large case base into several small ones. This is done by merging similar cases together through a clustering algorithm. An important side effect of this operation is that it allows up-to-date maintenance operations to be performed for case base management. During the retrieval process, an information-guided subsystem can then generate decision forests based on users' current answers obtained through an interactive process. Possible questions to the user are carefully analyzed through information theory. An important feature of the system is that case-base maintenance and reasoning are integrated in a seamless whole. In this article we present the system architecture, algorithms as well as empirical evaluations.

Keywords: interactive case-based reasoning, information retrieval, applications of clustering, decision trees

1. Introduction

Case-based reasoning (CBR), representing a new generation of expert system technology, relies on the retrieval, reuse, and revision of stored cases. Detailed descriptions of the technology can be found in books written by Kolodner [1], Leake [2], Watson [3] and Lenz et al. [4]. Research has been conducted in various sub-fields of CBR [5–10]. Industrial success can be found through many fielded applications by companies such as Inference Corporation (http://www.inference.com).

In this paper, we focus on applying interactive CBR to help desk applications. In a case-based reasoning system, each case is a combination of a problem description and a solution. The case base is organized to allow for efficient real-time retrieval. Once retrieved, a user can adapt the case through simple modifications to solve their problem. In help desk applications, for example, a customer provides information about the particular problems they are encountering. The *customer service representatives* (CSR's) who operate the help desk system enter the customer's information into the system. Case-based reasoning is then employed to retrieve similar problem cases, each with a recommended course of action. Using the courses of action suggested for similar problems as a guideline for problem solving, the CSR can make recommendations to the customer. At Simon Fraser University, we have developed a CBR system called CASEADVISOR for problem diagnosis in the help desk domain.

In the application of CASEADVISOR to a cable-TV domain, a particular problem has captured our attention. In a typical application, the case-base size

increases with alarming rate, making it more and more difficult for a CSR to retrieve the right case in a short amount of time. A large case base containing many cases is difficult to maintain. A user may find it difficult to focus on the right case and problem feature in real time.

In this paper, we present our solution to the problem. In CASEADVISOR, decision forests are created in real time to help compress a large case base into several small ones. This is done by merging similar cases together through a novel clustering algorithm. An important side effect of this operation is that it allows up-to-date maintenance operations to be performed for case base management. During the retrieval process, an information-guided subsystem can then generate decision forests based on users' current answers obtained through an interactive process. Action steps are suggested to the user in an incremental manner, and the results of the actions are used to formulate the next questions and suggestions. Possible questions to the user are carefully analyzed through information theory. An important feature of the system is that case-base maintenance and reasoning are integrated in a seamless whole.

The organization of the article is as follows. We first introduce the basic case-based reasoning problem domain and the particular problem we face (Section 2). We then introduce our general solution which consists of an interaction model (Section 2). We present our detailed algorithms next, including the density-based clustering algorithm (Section 3) and a decision-forest creation algorithm (Section 4). We present our experimental results in Section 5 and the system architecture in Section 6. Finally, we conclude the article with a discussion of future work in Section 7.

2. Problem Statement

In this section we present an overview of the diagnosis problem domain through a cable-TV diagnosis example, and present our initial interactive CBR design. We pay special attention to the underlying domain characteristics which motivate the interaction model presented later in the article.

2.1. Cable-TV Diagnosis and Initial CASEADVISOR *Design*

A critical problem in the help desk industry today is to provide high-quality customer support at a reasonable cost. Customer service is a knowledge-intensive task, often provided by CSRs who receive varying levels of training and documentation. For example, a typical cable-TV company call-center help desk, which is capable of receiving calls from across the country, handles tens of thousands of calls per month. Approximate costs to diagnose and resolve a customer's problem over the telephone may be in the neighborhood of several dollars per call, while the cost of dispatching technical service personnel to a customer site may cost in excess of over ten times higher.

The high differential between the cost of resolving customer problems over the telephone versus in person provides strong motivation for the development of knowledge-based decision-support systems that can help a service oriented company reduce it's "truck roll rate" as well as improve customer service quality and reduce training costs. By encoding most of the knowledge about the domain in a case-based system, suggestions for solutions to customers' problems can be given more effectively in real time. In addition to providing solutions to problems, the system can also provide computerized training appropriate for new technical service representatives and for new services provided by the industry.

Several domain characteristics motivated the development of the CASEADVISOR system, a case-based reasoning system that has a Web interface. First, the help desk domain is knowledge intensive. Much of the knowledge and experience of a technical service representative is built up over time through the direct experience of working through and trouble-shooting numerous customer problems. A large portion of this knowledge is in the form of electronic user manuals, flow charts and schematic diagrams.

Second, the help desk domain also requires real time interactive response. This task is becoming increasingly difficult because CSRs are usually faced with a high call volume, an increasing variety and complexity of customer problems, and a general need to provide prompt cost-effective service to their customers. This is especially true in light of emerging competition in the help-desk industry and the introduction of new services. In the case of cable-TV companies, services such as Internet access and digital TV are being introduced; likewise, telephone companies are introducing cable-TV services.

Third, the majority of the research in case-based reasoning has concentrated on cases with well-defined features. These cases have a relational structure, where

each feature is more or less a field in a relational database. In reality, however, formulating a case into a structured format requires extensive knowledge engineering. For a given domain, the user has to first determine the important features to use to represent each case. Then a decision has to be made on the type of values for each feature. The process of authoring knowledge in this feature-value format requires extensive maintenance when a new feature is discovered and inserted, or when an existing feature becomes irrelevant.

Based on these observations, we concluded that the target case-based reasoning system must provide highly efficient interactive real-time problem solving capability. The representation of the cases must allow both well-defined attributes and semi-structured problem descriptions. We have observed that a CBR system is similar to an AI real-time planning system. The system first makes an initial judgment on the problem categorization in order to focus on a problem area. The CSR then solves the problem for a customer by repeatedly listening to a user's description of the problem symptoms, and then suggesting action sequences for the user to follow. If, in the course of the interaction, the CSR determines that the problem area is incorrect, a fast recourse must be provided to backtrack to another problem area to work on.

In industrial practice, a majority of the case bases come directly from either unstructured text documents, which are scanned in, or end-users' verbal descriptions. These cases may have generic features such as *problem description* and *problem solution*, each containing natural language texts that can serve as keywords for case indexing. The representation of a case is as follows:

Keywords Keywords are short descriptions of the case which can be used in fuzzy string matching with the users initial free-form English text input.

Questions (Attributes) These are the features in a typical case represented in relational form. The questions with multiple choice answers provide an indexing mechanism for logarithmic time retrieval of the cases. Questions are also referred to as features or attributes in the CBR or database literature. We use "questions" and "attributes" synonymously in this paper.

Case Description This is a more detailed textual description of the case used to confirm the general problem area. A CSR will use this description to confirm that the problem being diagnosed indeed falls into the intended problem category. Multimedia representation is supported for this field, where a hyper-link can be provided using HTML to any HTML compatible image, video or audio data format.

Case Solution The case solution provides a solution to the case in either textual format or any multimedia format.

Tables 1 and 2 show an example case base representation for the cable-TV domain.

Using this case representation, it is now possible to provide a solution guideline in solving a problem. In a typical scenario, a customer representative receives a phone call from a user who reports a problem. In response, the CSR enters a short description of the problem in English. The system replies by returning a set of likely cases associated with questions. The responses to these questions are used to further rank the retrieved cases.

At this point, a user can open a case up to look at its more detailed descriptions. Multimedia presentation provides cues for the user to solve the target problem. The first use of the system is through its function as an advisor. Figure 1 shows an example of problem resolution in action. The arrows in the figure illustrate the workflow of the aforementioned problem resolution process. As can be seen, a CSR first enters a problem description to identify the context of problem solving. A collection of initial cases are then retrieved which match the partial description; these cases serve as the candidate pool for subsequent problem solving. Questions (problem attributes) that are associated with the candidate cases are then presented to the CSR, who in turn asks the customers on the phone to provide answers, in order to further distinguish between the cases. The process iterates until correct cases are eventually identified.

Table 1. Question/answer example in the cable-TV domain.

Question name	Answer1	Answer2	Answer3
Q1: What type of problem?	VCR recording	No sound	Remote control or converter
Q2: Which channels have the problem?	Pay TV	All	

Table 2. Example cable-TV case base, where the questions Q1 and Q2 refer to the same questions in Table 1. Keywords are not shown due to lack of space.

Case ID	Description	Q1	Q2	Solution
1	VCR is not taping the required channels	VCR recording		Check TV/VCR switch, etc.
2	VCR plus problems	VCR recording		Replug VCR-Starlink connection
3	Hong Kong TV will not work with VCR or converter	VCR recording		Re-tune TV set, . . .
4	Pay TV hook-up problem	VCR recording	Pay TV	Display correct hook-ups. . .
5	VCR hook-up problem	VCR recording	Pay TV	Tune VCR hook-ups. . .
6	No stereo sound	No sound		No stereo sound volume audio
7	No sound/poor sound	No sound	All	Connect directly to TV. . .
8	Starlink remote control problems	Remote control or converter		Replace battery. . .
9	Converter hookup problems	Remote control or converter		Try connect directly to TV. . .
10	Remote control skips channels	Remote control or converter		Enter the actual channel number. . .

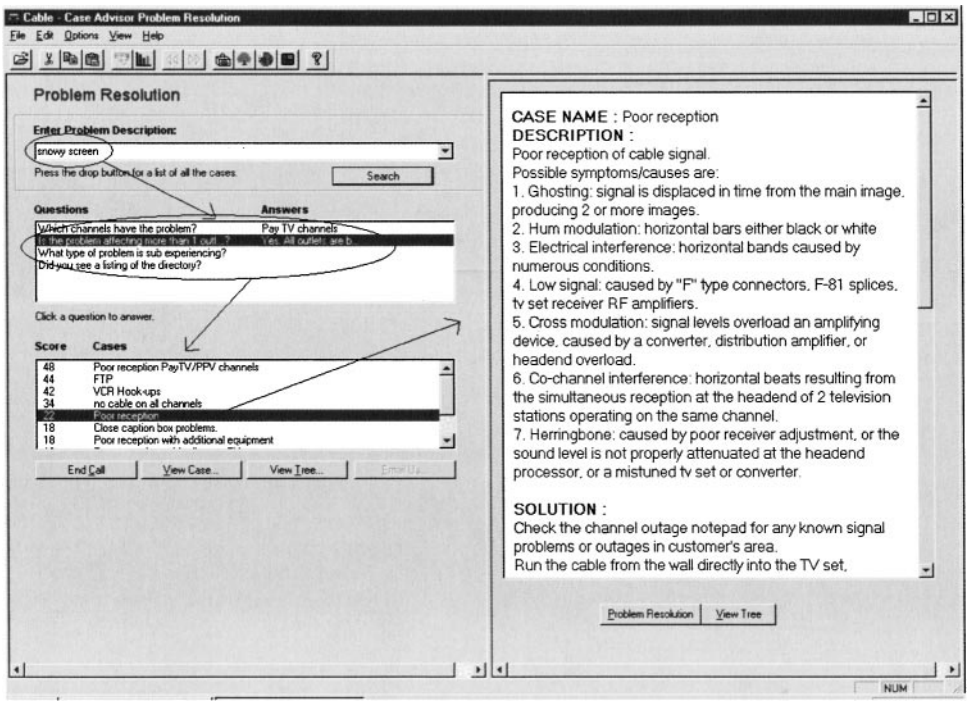

Figure 1. Case retrieval example and workflow in an initial interactive model of CASEADVISOR.

2.2. Motivation: Interactive Problem Solving through a Decision Forest

We applied the CASEADVISOR system to field testing in Rogers Cablesystems Ltd., the largest cable-TV company in Canada. We are especially interested in testing its ability in real-time problem resolution by many CSR's in the cable-TV call-center environment. Initially when the case base is small and focused, the result is encouraging; many CSR's can easily find what they look for in the case base. However, as the application progresses, an important issue of case base maintenance presents itself acutely.

In particular, we have observed that the size of the case base increases at a very rapid rate, caused both by the addition of new cases and cases that are produced

by continual tweaking of existing cases. For example, when a new Internet service is introduced, there is a need to introduce new cases that address this problem. Likewise, cases that represent variations of an approach to solving a particular problem may generate cases that overlap by a large amount but also differ in important ways. When the number of cases is large, the questions that must be presented to the user increase dramatically, making the system unusable as a real-time problem solving tool.

A second issue is that case-based diagnosis is not a one-shot process. It involves a process in which a CSR incrementally narrows down the cause of a problem and provides solutions. Throughout a problem solving session, a CSR first determines the problem solving context by asking a few general questions. These answers will allow the CSR to retrieve a candidate set of cases. An interactive problem solving session then proceeds in which the CSR suggests a next action to perform and the customer reports back the findings. This repeats until a plausible cause for the problem is determined.

These two observations prompted us to adopt the following approach in improving the CASEADVISOR system. After the CSR determines the problem solving context through initial questioning, a candidate set of cases, a subset of the entire case base, will be determined. Then an efficient clustering algorithm will be applied to partition the candidate cases into case clusters; each cluster corresponds to abstract versions of the cases under it. Common questions are extracted for each cluster, and the "important" questions are selected and presented to the user. Since in our case representation, a question (attribute) may *not* be relevant to all cases, therefore more than one important question may be selected and presented to the user. This produces not one decision tree, but a *decision forest*. The user chooses among these questions to answer, and the case clusters are correspondingly ranked. Then the user may decide to retrieve a case from any cluster using a CBR tool that does not cluster cases.

The above case-retrieval process can be divided into the following interactive model:

- The first phase is a case clustering process. In this phase a partition will be constructed that distinguish the cases into groups. We discuss this phase in Section 3.
- The second phase is a case-cluster ranking process. In this phase a subset of questions are selected and presented to the user in order to quickly narrow down to the best candidate case clusters. At each step of this phase a decision forest is created and the questions associated with the roots are presented to the user for answering. We discuss this phase in Section 4.
- The third phase, the final phase, is a traditional case retrieval process in which the retrieval is performed within a single cluster or the union of a small number of clusters.

Besides this process, the user can always backtrack to a previous level and answer a question in a different way.

To support real-time problem solving, our system must satisfy the following requirements. First, the case clustering algorithm must be efficient enough to permit real-time processing of the cases. Second, the decision forest must be efficient to generate, and the questions (attributes) selected must be able to cover the entire set of case clusters when any particular question (attribute) may not have values for all clusters. Third, the questions selected by the system must have high differential power to allow the CSR to quickly differentiate between different clusters. In what follows, we will explain the detailed algorithm for building case clusters and then building decision forests from these clusters. We will also present empirical tests to support the feasibility of our approach.

2.3. Related Work

Our work is closely related to case base memory organization, and case base maintenance for large-scale case base reasoning. When a case base is large, there is a need to organize cases hierarchically so that only some small subset of cases needs to be considered during retrieval. This subset has to have the best matching cases in it. There are some inductive clustering methods [11–13] and neural network methods [14]. Inductive clustering methods usually look for similar cases and then form groups based on the similarity. Neural networks take advantage of the network model, and uses the feedback from case usage to update the network indexes. Among the criteria for guiding the partitioning, some of them focus on dividing the groups to roughly the same sizes, while others cluster on sets of features that are shared among large number of items.

The above approaches produce various kinds of hierarchical structures for the case base. A major advantage of hierarchical methods is that hierarchical memory makes retrieval more efficient than flat retrieval, given

that the criteria for building the hierarchy and the relation being searched for are tightly coupled together. It does have its disadvantage in that adding cases to a case base required work. When a flat memory is used, cases are added to a list. With a hierarchical case-base structure, cases must be placed in the right place in the network as they are added. In addition, the network requires more space itself. The added costs are outweighed by retrieval benefits, since space often is not expensive.

Memory organization corresponds to one kind of case base maintenance. One branch of research has focused on the ongoing maintenance of case-base indices through training and case base usage [15–18]. Another branch of research has focused on maintaining the overall competence of the case base during case deletion [19–23]. Surveys of this field can be found in [24] and [3]. Compared with the above approaches, we observe that while it is important to maintain structures in a large case base through case base maintenance, it is also important to support the interactive nature of real-time CBR application. The best combination might be to integrate case base maintenance and interactive retrieval in a seamless whole. Indeed, this is the approach taken by our work.

In addition to case base maintenance, many researchers have worked on the speed of CBR retrieval problems through improvements on case-base indexing and optimality of nearest neighbor algorithms [25–27]. Our work differs from theirs in that we focus on how to help a user to identify the right case using an intelligent user interface algorithm, given that cases have been retrieved already.

The interactive issue we attack here is a difficult one, because there are often *too many* cases that are similar to the one given in the initial problem description. Our system provides a mechanism to help a user answer fewer questions and reach the target case sooner. In this aspect, our work is also related to conversational CBR of Aha and Breslow [17]. One difference is that while Aha and Breslow compute a single decision tree to interact with a user, we compute a decision forest. The reason is that the case base data we deal with often come with missing values for attributes. Thus, it is natural to use multiple decision trees to cover a large sparse case base rather than using a single one. Also, we have found that using more than one decision tree provides the user with more flexibility, because at any given time during an interactive CBR process, the user will have more than one question to select from.

3. Phase 1: Case Clustering using CBSCAN

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

In the past, many clustering techniques have been explored by various researchers [11, 28]. We decided to adopt a clustering algorithm that is well-known in the literature, rather than choosing a best clustering algorithm. In this section, we discuss one such algorithm, GDBSCAN (see below), which we adapted for our purpose.

3.1. Clustering Techniques in CBR

Clustering techniques are 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 networks have been made in applying clustering techniques to case-based reasoning [14]. 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 [14, 29–31].

In constructing a case clustering algorithm, the method must be incremental and efficient. By incremental we mean to be able to accommodate the arrival of new cases without making major changes to the hierarchical structure. We also require that the method has a complexity of less than $O(n^2)$ in time to be able to handle large case bases of size n.

We choose a density-based clustering method as our basis. The density-based method is relatively efficient to execute and does not require the user to prespecify 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 [32, 33], 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 shapes that are unions of spheres in order to group similar cases together.

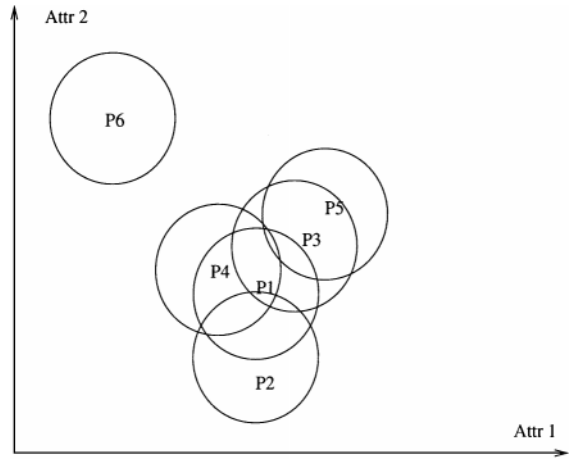

Figure 2. An example for GDBSCAN.

More specifically, GDBSCAN accepts a radius value *Eps* (which stands for ϵ) based on a distance measure, and a value *MinPts* for the number of minimal points. The latter is used to determine when a cluster 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*. A main problem of GDBSCAN is that a user must input a radius value *Eps* in order to construct the partitions. If the user is not a domain expert, it is difficult to choose the best value for *Eps*.

We illustrate the GDBSCAN algorithm with an example. Suppose there are six cases in a case base, represented by two attributes. The position of a point P_i in a two-dimensional space represents the values of the attributes, as illustrated in Fig. 2. The distance between two cases is the dissimilarity between them. The more similar any two cases are, the smaller their distance.

Definition 1. Given a set of cases D, the **Eps-neighbourhood** of a point p is defined as $N_{Eps(p)} = \{q \in D \mid dist(p, q) \leq Eps\}$

In the example, if Eps is equal to the radius, then the Eps-neighbourhood of a point P_i is the area with a circle centered at point P_i.

If two points are connected by a "dense" region then they are called "directly density reachable"; in this case they should belong to the same cluster.

Definition 2. A point p is **directly density-reachable** from a point q if

1. $p \in N_{Eps}(q)$
2. $|N_{Eps}(q)| \geq MinPts$

In the example, suppose MinPts $= 2$, then P_4 is directly density-reachable from P_1, but P_1 is not directly density-reachable from P_2 because $|N_{Eps}(P_2)| = 1$. So the density-reachable relation is not symmetric. We see also that P_1 and P_5 are directly density-reachable from P_3.

Definition 3. A point p is **density reachable** from a point q if there is a chain of points $p_1, p_2, \ldots, p_n$, where $p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

In the example, P_4 is density reachable from P_3 because P_4 is directly density-reachable from P_1 and P_1 is directly density-reachable from P_3.

Definition 4. A point p is **density connected** to a point q if there is a point o such that both p and q are density reachable from o.

The relation is symmetric. It is easy to tell that P_3 and P_4 are density connected with each other.

Definition 5. Let D be a set of points. A **cluster** is a non-empty subset C of D satisfying the following conditions:

1. $\forall p, q$, if $p \in C$ and q is density-reachable from p then $q \in C$.
2. $\forall p, q \in C$, p is density connected to q.

In the example, there is a cluster consisting of P_1, P_3, P_4, P_5. However P_6 and P_2 form clusters of their own. GDBSCAN uses the definition of a cluster as defined above to recursively build a partition of a database.

As mentioned above, a major drawback of GDBSCAN is that the user must provide an appropriate *Eps* value. It turns out from our experiments that choices in EPS-values have very significant influence on the quality of partitions. Thus, the assumption that the user provides the best EPS values is unrealistic. 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 a good partition has 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:

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

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 that 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 numbers 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.

3.2. *Integrating Keywords and Attributes*

As noted in Section 1, the cases in case-based reasoning consist of two types of indexing: attribute/values and keywords. We have so far only discussed how to use attribute knowledge representation to perform clustering. In this section we will discuss how to integrate keywords as well as attributes in case clustering.

For categorical attributes, the distance between two cases can be defined by the number of attributes whose values disagree between the two cases. Let N_i be the number of attributes associated with case i, and A_{ij} the number of attributes that cases i and j have in common. The similarity between cases i and j is defined as:

$$SA(i, j) = \frac{A_{ij}}{\sqrt{N_i * N_j}} \quad (2)$$

Similarly, distances between two sets of keywords associated with cases i and j can be defined as:

$$SK(i, j) = \frac{A'_{ij}}{\sqrt{N'_i * N'_j}} \quad (3)$$

where A'_{ij} is the number of common keywords, and N'_i is the total number of keywords in case i.

Combining these two similarity measures, we obtain the similarity between the case i and case j as

$$S(i, j) = \sqrt{SK(i, j)^2 + SA(i, j)^2} \quad (4)$$

To avoid favoring either attributes or keywords, a weight θ can be introduced. θ is the weight for the keywords and weight $1-\theta$ is the weight for attributes. We assume that the value θ is defined by the domain expert. Initially, it is set to 0.5. Thus, the revised similarity function of 4 is

$$S(i, j) = \frac{\sqrt{(\theta * SK(i, j))^2 + ((1-\theta) * SA(i, j))^2}}{\sqrt{\theta^2 + (1-\theta)^2}} \quad (5)$$

$\sqrt{\theta^2 + (1-\theta)^2}$ is a constant for all the similarities between cases. Ignoring this factor, the similarity function becomes

$$S(i, j) = \sqrt{(\theta * SK(i, j))^2 + ((1-\theta) * SA(i, j))^2} \quad (6)$$

Based on the computation of similarity, we can then define a distance function $d(i, j)$ between cases i and j to be $1/S(i, j)$ when $S(i, j)$ is not zero, and define the distance to be a very large number when $S(i, j)$ is indeed zero.

3.3. *The CBSCAN Algorithm for Clustering Case Bases*

We now introduce the case 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 *MinPts* to 2. To find a value for *Eps* in order to get a good partition, we modify *Eps* by observing changes in the NCC of the resulting partitions. 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* oscillate around this point until an approximate *Eps* value that produces the locally optimal NCC value is found.

Algorithm CBSCAN( ) is described in detail in Fig. 3. In this algorithm, the function GDBSCAN( ) returns the NCC value for the current partition. For

Algorithm CBSCAN(M, D, $EpsInterval$)
Input: A maximal number of iterations M, a Case Base D, a distance function "distance( )" and a small positive value $EpsInterval$ for incrementing the Eps value.
Output: A partition P of the case base D.

```
1. MinPts := 2;
2. Eps := max_{i,j} distance(C_i, C_j), where C_i, C_j are cases in D;
3. BestQuality := 0.001; (or a very small number)
4. Direction := 1; P := ∅;
5. for k := 1 to M do
6.     CurrentQuality := GDBSCAN( D, Eps, MinPts);
7.     if (CurrentQuality ≥ BestQuality) then
8.       BestQuality := CurrentQuality;
9.       BestEps := Eps;
10.    else
11.      Direction := −Direction ;
12.      EpsInterval := EpsInterval/2;
13.    end if
14.    if ( Direction ≥ 0 ) then
15.      Eps := Eps − Epsinterval;
16.    else
17.      Eps := Eps + EpsInterval;
18.    end if
19.end for
20.while D ≠ ∅ do
21.    let CL be a cluster found using the Eps value;
22.    D := D − CL;
23.    P := P ∪ CL;
24.end while
25.return(P);
```

Figure 3. The CBSCAN algorithm.

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, $O(m * n * \log n)$.

3.4. Partitioning a Case Base using CBSCAN

After a large case base is partitioned, smaller case bases are formed with a collection of case clusters; we call them the *CC*'s (case clusters). Each *CC* has a case base name and a list of keywords. The case name is a description of the cases in the *CC*. The keywords are a set of the most frequently used keywords by the cases in the case cluster. In addition, there is a set of attributes associated with each case cluster; these are the attributes that are associated with the majority of the cases in the cluster. By this process, we now have a two-level structure of a case base, where at the top level the cases are in fact the case clusters themselves.

3.5. When to Apply the CBSCAN Algorithm?

There are two modes in which one can apply the CBSCAN algorithm to perform case clustering. One is to apply it at every problem solving session. Another is to apply to a very large case base after a relatively longer time interval, so that the clustered results are saved and retrieved during the interval. To clarify, the former approach can be called *short-term maintenance* of the case base, whereas the latter *medium-term*.

For short-term case base maintenance, we have observed that if the number of candidate cases is less than 1000, the clustering time is usually within 10 seconds. Thus, short-term maintenance can be applied every time during problem solving when the number of candidate cases is less than 1000.

In contrast, when the case base size is larger than 1000, the clustering method cannot be efficiently applied every time during real-time problem solving. In

this case, the CBSCAN algorithm can be applied on a daily or weekly basis, producing a collection of case bases. Each case base is one cluster as a result of the partition. These case bases can be stored on different machines on a computer network, creating a distributed case base structure. We are currently evaluating how best to balance the two maintenance strategies.

4. Phase 2: Information-Guided Cluster-Retrieval Algorithm

4.1. Overview

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

- Combine information theory with case cluster retrieval to find the attribute that can distinguish the case clusters the most;
- Deal with the incomplete attribute-value problem; that is, handling the situation where an attribute may not have a value for a subset of case clusters. To cover all case clusters during a retrieval process, a *decision forest* instead of a decision tree is built.

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 clusters that are highly likely to contain the final result based on the answers and allows 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 clusters. For most case base applications, not all cases are associated with all attributes. Therefore, no single attribute may cover all case clusters. Thus, we must select more than one attribute so that the selected set of attributes will cover all the case clusters. This induces a decision forest instead of a decision tree. Second, we still wish the attributes we present to the user to 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. Then we iteratively select a collection of attributes with the highest information gain [13] that form a covering set 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.

There are two special requirements for case-based reasoning in the decision-forest construction process. First, there is a weight associated with the attribute-value pair to indicate the importance of using this attribute-value pair in similarity calculation for cases. We should include this weight in our computation. Second, there are frequently missing values for some attributes. We assume that if an attribute does not have a value for a certain case cluster, the case-cluster weight associated with that specific attribute-value is set to zero. In our computation of information gain values for attributes, we took both considerations into account.

Consider an example, where there are ten cases in a case base (see Table 3). The descriptions of cases are represented by four attributes. In this example, the ten cases are clustered into five case clusters (Table 4). The probability distribution of the partition is $P = (1/5, 1/5, 1/5, 1/5, 1/5)$. The information value of the partition is $Info(P) = 2.5$. The information-gain ratio for the clusters are listed in Table 5.

In our example, the attribute with the largest information-gain ratio for all the case clusters is first

Table 3. Example case base for case cluster retrieval.

Case name	Attr 1	Attr 2	Attr 3	Attr 4	Case cluster 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 4. Case clusters with their attributes and values.

Case cluster no.	Attr 1	Attr 2	Attr 3	Attr 4
CC 1	(a,100)		(a,100)	(a,50) (b,50)
CC 2	(a,100)		(b,100)	(c,50),(d,50)
CC 3		(a,50)	(a,50) (d,50)	(a,50),(b,50)
CC 4		(a,50),(b,50)	(d,50) (c,50)	(a,100)
CC 5		(b,100)		(b,50),(c,50)

Table 5. Sorted attributes information-gain ratio for all case clusters.

Attribute	Information gain ratio
Attribute 2	8.72
Attribute 4	6.99
Attribute 3	4.15
Attribute 1	0

extracted as the root of a decision forest. The retrieval system first sorts all the attributes by their information-gain ratio, as listed in Table 5. As can be seen, attribute Attr2 has the largest information-gain ratio. It is therefore set as the root for the first decision tree in the decision forest. Before Attr2 is returned to the user, it is checked whether this attribute covers the remaining case clusters. From Table 5, we know that Attr2 covers only clusters 3, 4 and 5, but not clusters 1 and 2. However, Attr 3 also covers clusters 1 and 2. Therefore, Attr 2 and Attr 3 are chosen to be presented to the user.

We formalize the above intuitive description of the attribute selection algorithm in Fig. 4. In this algorithm,

Algorithm Atrribute-Selection(CC,A)
Input: A set of case clusters CC, and a set of attributes A.
Output: A subset of A that "covers" all case clusters in CC.

1. $RCC := CC$; (this is the set of remaining clusters)
2. $RA := A$; (this is the set of remaining attributes)
3. $SA := \emptyset$; (this is the selected set of attributes)
4. **while** $RCC \neq \emptyset$ **and** $RA \neq \emptyset$ **do**
5. **for each** $a_i \in RA$ **do**
6. $info_i := InfoGainRatio(a_i, RCC)$;
7. **end for**
8. $a_j := argmax_i info_i$
9. $SA := SA \cup \{a_j\}$;
10. $RCC := RCC - Clusters(a_j)$;
11. **end while**
12. **return** SA;

Figure 4. Attribute-selection algorithm.

it is assumed that subroutine *InfoGainRatio*(a_j, *RCC*) returns the information gain ratio of attribute a_j on case clusters *RCC*, and *Clusters*(a_j) returns a set of case clusters that attribute a_j has an associated value.

5. Experimental Results

To demonstrate the effectiveness of our system, we have run a series of experiments to evaluate the overall system performance. We used an ablation study to determine whether the two individual components, the clustering and information-gain question selection components, are individually useful, and whether their combination itself outperforms the earlier version of CaseAdvisor [34]. In this section, we will clarify the empirical test methodology, and present our test results using several large data libraries. We also discuss our results.

For the ablation study, we are interested in comparing all four system configurations in order to isolate the source of gains in system effectiveness, if any. We run experiments on the following combinations:

1. With clustering and information gain;
2. With clustering only;
3. With info-gain only; and
4. With neither, using the previous version of CaseAdvisor.

In our tests, the effectiveness of the system is defined as the *precision* and *interactive efficiency*. In the context of our system, these concepts can be defined formally.

For each set of data (described below) and experiment we run, we randomly choose a target case *c*. We then randomly choose a subset Q_c of the attributes of *c* and provide them with answers. *c* is the case we wish to find in an interactive CBR retrieval process. During case retrieval for each of the four systems under testing, there is a case list *CL* and a question list *QL* at any moment in time. If no threshold is imposed, the case list contains a listing of all cases in the case base and *QL* all questions that are attributes in the case base. At each step the remaining question in Q_c that is the highest ranked gets a predefined answer. After that cases and questions in *CL* and *QL* respectively may change their ranking accordingly. We will call (*CL*, *QL*) an *interactive-CBR state*, or *I-State*. Thus, each question-answering takes the system from one I-State to another. The experiment loops until all questions in Q_c are answered. For each experiment, there are Q_c transitions of the I-state.

1.$i := 0$;
2.**until** all questions in Q_c are answered, **do**
3. $i := i + 1$;
4. look for the first question q from Q_c that appears in QL;
5. let the question q be ranked at l_ith position in QL;
6. answer q; this produces another I-state;
7.**end until**;
8.let the target case be at jth position in CL ($j = 0$ for the highest ranked case);
9.**stop**

Figure 5. Experimental process.

For fairness in comparison in the computation of the locations of questions and cases, we have arranged so that all cases in the same cluster appear together, and attributes/questions selected by the information-gain based system are placed ahead of other questions in a case-base domain. This arrangement enabled us to measure the precision and interactive efficiency on a common metric, with a common number of total questions and cases.

The experiment for each of the four systems in the ablation study is run as shown in Fig. 5.

Let N be the number of questions in the case base. Then *interactive efficiency* is defined as

$$1 - \sum_{i=1}^{|Q_c|} (l_i/N)$$

If the target case c appears below the top ten ranking, then we define precision as zero. Otherwise, if c appears within the top ten highest ranked cases, and j is the position of the target case in the final step of the experiment (Fig. 5), then it is

$$Precision = 1 - j/10$$

This definition corresponds to the requirement that the user only wishes to see the top ten cases in the final ranking.

Ideally, we would have wished to use a case base from a help desk domain to perform our empirical tests. However, we have only a Cable TV case base with less than 100 cases C not sufficient to demonstrate the need and scale-up property of our method. Thus, we have chosen some data sets from the UCI Repository of Machine Learning Databases and Domain Theories [35] at the University of California at Irvine. Two data sets are first used for our experiments (the Thyroid Disease Database and the Mushroom (agaricus-lepiota) database). In addition, in order to test the system under the condition that there are missing values for some attributes, we took the union of two different databases "Votes" (house-votes-84.data) and "Soybean" (soybean-large.data) and appended them together into a single one. Each tuple in the appended dataset contains the attributes from each of the two original datasets. This creates a bigger database with many missing values. For each of the three datasets, the number of attributes, cases and the percentage of missing values are shown in Table 6. The percentage of missing values is the proportion of attribute-value pairs that have no values out of all attribute-value pairs. In this table, the number of attributes provides the maximal range of Q_c for the cases.

The test results are shown in Table 7. Each result is the average of 50 random tests. In this table, "CA" stands for running CaseAdvisor alone, "Cluster" stands for running CaseAdvisor with the clustering function turned on. Under this option a cluster is found through an interactive process first, then a case is found using a non-clustering based CBR tool (CaseAdvisor I [34]). Thus, both cluster retrieval and case selection are factored in under this option. "Info-Gain" stands for using CaseAdvisor and Information-Gain Ratio to measure the ranking of attributes, and finally, "Cl + Info" stands for the combined system including both clustering and

Table 6. The three case base domains selected for our experiments.

Case base name	# Attributes	# Cases	Percentage of missing value
Thyroid disease DB	29	2800	4%
Mushroom (agaricus-lepiota)	22	8124	0%
Votes + Soybean	12	740	71%

Table 7. The experimental results for the three selected data sets.

	CA	Cluster	Info-Gain	Cl + Info
Thyroid disease DB				
Precision	0.0%	0.0%	45.0%	44.0%
Interactive efficiency	55.90%	57.50%	97%	95.60%
Time (CPU seconds)	448.18	4.3	62.3	17.5
Vote + Soybean				
Precision	0.0%	84.8%	86.2%	84.8%
Interactive efficiency	58.20%	52.70%	93%	91%
Time (CPU seconds)	2.04	1.74	15	0.96
Mushroom				
Precision	6.00%	83.00%	92.00%	91.80%
Interactive efficiency	58.90%	56.30%	92%	89%
Time (CPU seconds)	5374	29.3	201.7	10.4

information gain calculation. We report on three key properties: Precision, Interactive Efficiency and Total CPU Time.

From the experimental results, one can see the following properties:

- For both precision and interactive efficiency, CaseAdvisor with information-gain and with both clustering and information gain are superior in performance than using CaseAdvisor alone. This is true for all three tests.
- Further, CaseAdvisor with information gain outperforms CaseAdvisor with clustering in both precision and efficiency, but Under-performs the latter in CPU time. This can be explained by the fact that using information gain can help build a finer ranking of the questions after each question is answered, thus raising the ranking of the subsequent questions higher. However, the calculation of decision forest takes much time to complete simply because of the number of attributes and cases under consideration.

  Using clustering, which includes first cluster selection and then case retrieval using a non-clustering based CBR tool, the time for attribute selection is dramatically reduced because only a few attributes and case clusters are under consideration.
- Overall, CaseAdvisor with both clustering and decision forest for attribute selection results in the best performance in combination of precision, interactive efficiency and CPU time.

In summary, these preliminarily tests demonstrate the utility of CaseAdvisor with both clustering and information gain calculation.

6. System Design and Overall Interactive Model

The combined system of CBSCAN and CCRetrieve, known as CASEADVISOR:CASECLUSTER has been implemented in Microsoft Visual C++ (TM). A system architecture diagram for the CASEADVISOR:CASECLUSTER system is shown in Fig. 6. An example clustering result is shown in Fig. 7. As seen in the figure, a partition is shown in the form of a forest. The internal node represents the clusters. A case cluster name is composed of its representative keywords and their frequencies. The keywords of a case cluster are the union of the keywords in the cases. If the user double-clicks a case or case cluster, the detailed information contained in it will be shown.

After a user enters initial queries, cases are retrieved and partitioned to form a collection of case clusters.

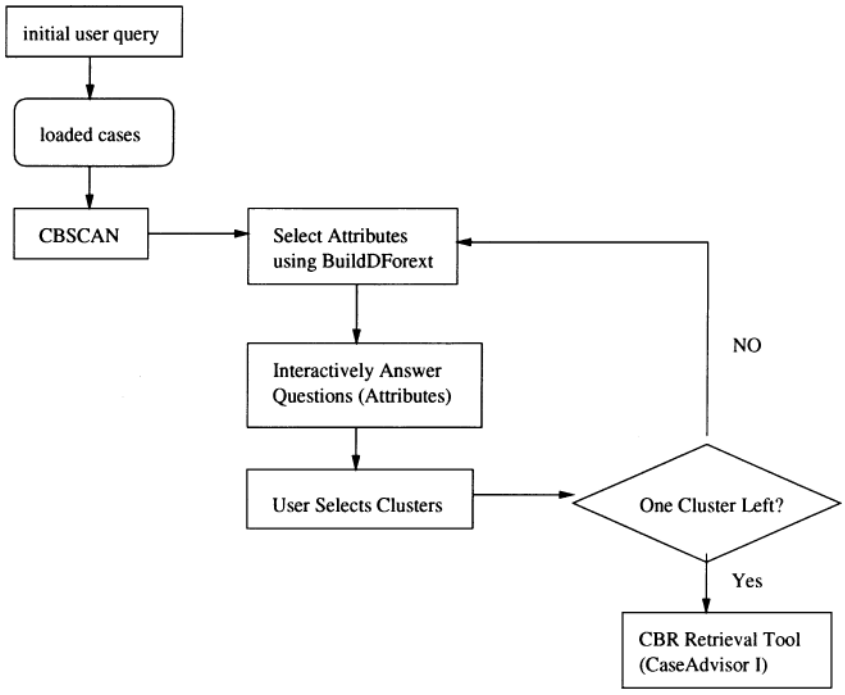

Figure 6. System architecture of the CASEADVISOR:CASECLUSTER system.

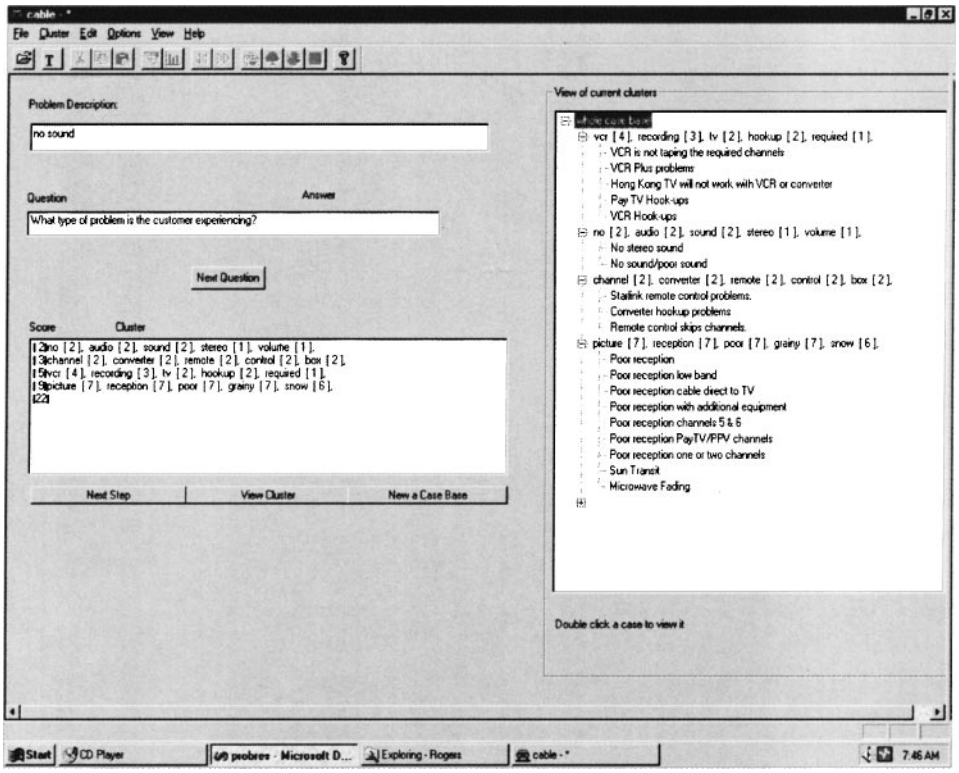

Figure 7. Retrieval process interface.

The CASEADVISOR:CASECLUSTER system then retrieves the most similar case cluster, then it passes the query to the original CASEADVISOR system [34] which performs flat case retrieval in the case cluster. The flat retrieval tool returns the most similar cases to the system, which are returned to the user. Figure 7 is the interface to help a user retrieve a case cluster. The tree on the right side of the window gives the user a general idea of the partition. When a user first gives a high level description about the problem in the query-input text box, the system uses this description to rank the case clusters which might be similar to the current problem description. The scores and the names of case clusters are listed in descending order by score. The score in this module shows the relevancy of each case to the new problem. The higher the score is, the more relevant the case cluster is.

At each iteration, a user may want to specify additional characteristics of the new problem. To do this he can answer questions that are generated by the CCRetrieve module. The system will select the questions automatically by the information-gain ratio as introduced in the last section. This process further reduces the search space. Following the questions, the score of the more relevant case clusters increase. The user can use the backward and forward arrows in the menu to modify his answers. When there is only one case cluster left, the cases in this cluster will be retrieved by a flat retrieval tool; in this case it is CASEADVISOR [34].

7. Conclusions and Future Work

The focus of this work is to provide the user with a powerful interactive tool for managing a very large case base and to facilitate interactive retrieval. We accomplished this task with two novel methods. First, a new case-base clustering algorithm is developed that is both efficient and effective in dealing with large case bases. Second, the clustered case bases are retrieved interactively; the questions which correspond to the attributes are selected using information theory, and are presented to the user to further narrow down. If the size of a cluster is relatively small, any simple CBR retrieval method can be used once a target case base is found in the second step. If, on the other hand, the size of a cluster is large, then other methods must

be applied in finding the target case. This corresponds to a limitation of our system.

This research has enabled us to generate an improved CASEADVISOR system. Currently we are conducting field tests to obtain user feedback in the cable-TV area. In the future, we will extend this work into a distributed case-base management system. In this environment, the input cases are from multiple case bases that are distributed over the Internet. These case bases may in fact overlap with each other. Thus, there is a need to organize a virtual case base space in which the networked case bases are re-indexed through machine learning and database techniques. We consider that this direction represents a major thrust in case-based reasoning in the future.

Acknowledgment

We would like to thank NSERC, Rogers Cablesystems Ltd. Canadian IRIS-III Program and Dr. Madden from Canadian Cable Labs Fund for their support.

References

1. J. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann: San Mateo, CA, 1993.
2. D.B. Leake, *Case-Based Reasoning—Experiences, Lessons and Future Directions*, AAAI Press/ The MIT Press, 1996.
3. I. Watson, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann, San Francisco, CA, 1997.
4. M. Lenz, B. Bartsch-Sporl, H.-D. Burkhard, and S. Wess, editors, *Case-Based Reasoning Technology: From Foundations to Applications*, Springer: Berline, 1998.
5. D. Hennessy and D. Hinkle, "Applying case-based reasoning to autoclave loading," *IEEE Expert*, vol. 7, no. 5, pp. 21–27, 1992.
6. J. Kolodner and W. Mark, "Case-based reasoning," *IEEE Expert*, vol. 7, no. 2, pp. 5–6, 1992.
7. E. Simoudis, "Using case-based retrieval for customer technical support," *IEEE Expert*, vol. 7, no. 5. pp. 7–13, 1992.
8. R. Shank, A. Kass, and C. Riesbeck, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates: New Jersey, 1994.
9. B. Smyth and P. Cunningham, "A comparison of incremental case-based reasoning and inductive learning," in *Proceedings of the 2nd European Workshop on Case-Based Reasoning*, 1995, vol. 1, pp. 32–39.
10. A. Ram and J.C. Santamaria, "Continuous case-based reasoning," in *Proceedings of the AAAI-93 Workshop on Case-Based Reasoning*, 1993, pp. 86–93.
11. D. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, pp. 139–172, 1987.
12. P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman, "Autoclass: A Bayesian classification system," in *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI. June 12–14 1988, Morgan Kaufmann: San Francisco, 1988, pp. 54–64.
13. J.R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.
14. M. Malek, "A connectionist indexing approach for CBR systems," in *Case-Based Reasoning Research and Development*, (ICCBR 1995) Sesimbra, Portugal, pp. 520–527, 1995.
15. 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*, Providence RI, USA, 1997, pp. 291–302.
16. 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, (IJCAI95), Aug. 1995, pp. 391-399.
17. D.W. Aha and L. Breslow, "Refining conversational case libraries," in *Second International Conference on Case-Based Reasoning, (ICCBR-97)*, edited by D.B. Leake and E. Plaza, Springer: Providence, RI, USA, vol. 1266, pp. 267–278, July 1997.
18. Z. Zhang and Q. 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*, Springer: Lecture Notes in Computer Science, Sozopol, Bulgaria, vol. 1480, 1998, pp. 489–510.
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*, vol. 1, pp. 377–382, 1995.
20. S. Markovich and P. Scott, "The role of forgetting in learning," in *Proceedings of the Fifth International Conference on Machine Learning*, 1988, vol. 1, pp. 459–465.
21. P. Domingos, "Rule induction and instance-based learning," in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann: San Francisco, CA, USA, 1995, pp. 1226–1232.
22. D.W. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.
23. K. Racine and Q. Yang, "Maintaining unstructured case bases," in *Proceedings of the Second International Conference on Case-Based Reasoning, ICCBR-97*, Providence RI, USA, 1997, pp. 553–564.
24. 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, pp. 196–207, 1998.
25. K. Deng and A.W. Moore, "Multiresolution instance-based learning," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Morgan Kaufmann: Montreal, 1995, pp. 1233–1239.
26. S. Grolimund and J.-L. Ganascia, "Speeding-up nearest neighbor memories: The template tree case memory organization," in *Proceedings of the Thirteenth International Conference on Machine Learning*, Morgan Kaufmann: Bari, Italy, 1996, pp. 225–233.
27. S. Wess, K.-D. Althoff, and G. Derwand, "Using k-d trees to improve the retrieval step in case-based reasoning," in *Topics in Case-Based Reasoning*, edited by S. Wess, K.-D. Althoff, and M.M. Richter, Springer: Heidelberg, Germany, pp. 167–181, 1994.

28. J. Zupan, *Clustering of Large Data Sets*, John Wiley & Sons: Chicester, England, Toronto: Research studies Press, 1982.
29. E. Auriol, S. Wess, M. Manago, K.D. Althoff, and R. Traph, "Inreca: A seamlessly integrated system based on inductive inference and case-based reasoning," in *Case-Based Reasoning Research and Development*, pp. 371–380, 1995.
30. S.K. Bamberger and K. Goos, "Integration of case-based reasoning and inductive learning methods," in *First European Workshop on CBR*, Nov. 1–5, LNCS 837, Springer: Verlag, Berlin, pp. 110–121, 1993.
31. 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*, Seki-Report SR-93-12 University of Kaiserslautern, pp. 313–318, 1993.
32. 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*, vol. 2, no. 2, 1998. Kluwer Academic Publishers.
33. X. Xu, M. Ester, 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*, Orlando FI. (ICDE'98) 1998.
34. Q. Yang, E. Kim, and K. Racine, "Caseadvisor: Supporting interactive problem solving and case base maintenance for help desk applications," in *Proceedings of the IJCAI'97 Workshop on Practical Use of CBR*, Nogoya, Japan, Aug. 1997, pp. 32–44.
35. E. Keogh, C. Blake, and C.J. Merz, "UCI Repository of Machine Learning Databases," [http://www/ics.vci.edu/nmlearn/MLRepository.html]: Icvine CA: University of California, Dept of Intermotion and Computer Science, 1998.

Qiang Yang obtained his PHD from University of Maryland, College Park in 1989. He has since worked at as a professor at University of Waterloo and Simon Fraser University in Canada. His research interests include case-based reasoning, intelligent agents, planning, data mining, web-based information retrieval and machine learning. He is the author of two books published by Springer-Verlag and Kluwer.

Jing Wu obtained her Master's degree from Simon Fraser University in 1998, after she has obtained her B.Sc. degree from Harbin Institute of Technology in Harbin, China in 1986. She has subsequently joined the Amdocs Limited, R&D division working on customer case solutions.