NEW ALGORITHMS FOR KNOWLEDGE AUTOMATION OF CBR RETRIEVAL AND ADAPTATION PROCESSES

Abdel-Badeeh M. Salem & A. H. Mohamed
absalem@asunet.shams.enu.eg
amirahmaz@hotmail.com
Computer Science Dept., Ain Shams University,
Faculty of Computer & Information Sciences,
Cairo, Egypt.

ABSTRACT

Recently, Case-Based Reasoning (CBR) has proved its success as reasoning and learning approach. However, there are some knowledge engineering complexity appears in developing the CBR systems. This paper introduces a new CBR system that helps to reduce the knowledge acquisition effort required for building typical CBR systems. The proposed system incorporates the learning techniques into the CBR methodology to automate extracting the features weights of the cases, and to extract the adaptation rules from the case library. This improves the performance of CBR systems by eliminating the need for expert to guide these developing steps, especially for the situations where a little knowledge of the field is known. Also, it increases the accuracy of the achieved solution of the problem to be solved. The proposed system proves its performance when applying for real systems.

KEYWORDS: Case based Reasoning, Retrieval algorithms, Extracting adaptation rules, Introspective learning.

1. INTRODUCTION

Case-Based Reasoning (CBR) is a problem-solving methodology that finds solutions to new problems by analyzing previously solved problems [1]. Given a new problem to be solved, the most similar problems from the case-base are retrieved. Their solutions may be directly applicable to the new problem, although some adaptation of these solutions may be necessary to fit the new problem better. The proposed solution may then be stored in the case-base and in turn be used to solve future problems [2].

CBR has become widely popular because of a number of advantages. Among them, the ability to utilize existing data as cases provides an opportunity to reduce development time. Nevertheless, knowledge engineering effort required for constructing CBR systems for some types of problems may still be significant [3]. Thus, the knowledge contained in a CBR system can be considered as distributed over several “knowledge containers”: cases themselves, case representation, indexing knowledge, similarity knowledge, and adaptation knowledge. Some of this knowledge is easily available, some has to be entered by the expert, and some can be discovered using AI and statistical techniques [2].

Recently work within the CBR community has studied more systematic means of interrelating different types of domain knowledge, tasks, and methods for building and maintaining CBR systems [4]. CBR tools seem to have concentrated on increasing their possible field of application: improving the way they interface with data-base systems, facilitating their deployment in web environments, increasing portability, and improving case representation. However, so far, commercial systems provide few facilities for automating the process of constructing competent case-bases, optimizing retrieval with the goal of retrieving most useful cases, and, in particular, automating the acquisition of adaptation knowledge. These issues are now being addressed in recent research: Smyth and McKenna [5] aim to build compact competent case-bases, Jarmulak et al. [6] introduce ways of automatically optimizing retrieval, and Wilke et al. [7] consider knowledge-light approaches for learning adaptation suitable for incorporation into CBR tools [8].

In this paper, we concern the automation of the knowledge needed in CBR retrieving, and adaptation processes. These types of knowledge have been almostly analyzed by the expert. Therefore, there is a need for methods which contribute to extract the
required knowledge from the available data only without the need of an expert guiding these developing stages. The present research introduces a new methodology that can be used to learn extracting this knowledge from the cases in the library.

Section 2 presents the case based system. While, Section 3 deals with the retrieving and adaptation algorithms of the proposed system. The applicability of the proposed system for the real problem is described in section 4. Section 5 deals with the conclusion.

2. CASE BASED REASONING CYCLE

CBR is a type of problem solving methodology that tries to find a suitable solution to a new problem based on the past experience. The process involved in CBR can be represented by a schematic cycle in the following four steps:

1. RETRIEVE the most similar case(s).
2. REUSE the case(s) to attempt to solve the problem.
3. REVISE the proposed solution, if necessary, and
4. RETAIN the new solution as a part of a new case [9, 10].

3. PROPOSED CASE BASED REASONING SYSTEM

The main task of CBR is developing a solution of a certain problem depending on the stored cases in its library. A key of CBR operation is its retrieving of similar case(s) for the new case from previously solved problems. If needed, the retrieved cases are adapted to suit the requirements of the new case. The proposed system concerns automating the retrieving and the extracting of the adaptation knowledge.

Retrieving process is mainly depending on the features of the cases, and their weights those affect the solution of the problem. We use cases contained in the case base to achieve the relevance/importance of the case features by determining the weight of each feature, and the best parameters for the retrieval process. This results in self-learning retrieval task. Also, the k-value of the k-nearest neighbor retrieval algorithm that used by the proposed system can be extracted from the stored cases.

While, in adaptation stage, knowledge of the adaptation rules almost acquires and guides by an expert in the field. Learn the system how to extract these rules is considered a very important task in CBR systems. Nevertheless, it has a lot of concern by researchers due to its complexity.

Figure (1) describes a complete diagram for both the retrieval and adaptation processes of the proposed CBR system.

3.1 Retrieval Algorithm

Retrieval process is the core of CBR systems. The success of CBR problem solving methodology depends mainly on the accuracy of its retrieval algorithm. k-nearest neighbor algorithm proves its success as a common CBR retrieval algorithm. Its methodology depends mainly on the feature weights of the cases and the k-voting value. Proposed system can improve its performance by automating the extraction of the feature weights, and k-value. This will lead to avoid the errors due to noisy data, and minimize the knowledge engineering effort required for analyze the weights of the features of the cases.

3.1.1 Feature Weights Learning

Automating the calculation of the feature weights of k-nearest neighbor (k-NN) retrieval algorithm is very important task. Nevertheless, the problem encountered is that it is difficult to determine the important features and adjust their relative importance. The situation is further complicated by the fact that the features are highly context-sensitive; the productiveness of a feature depends heavily on the current context [11].

The proposed system suggests the uses of the introspective learning (IL) to improve the accuracy of the parameter weights. IL refers to an approach of learning problem solving knowledge by monitoring the run-time progress of a particular problem solver. The basic idea behind the introspective learning of parameter weights is to increase or decrease the weights of selected case parameters on the basis of problem solving performance. Parameter weighting methods differ in terms of their learning criteria as well as in terms of their update models.

Two basic learning criteria are used in this topic, they are: failure-driven and success-driven.

1. Failure-driven methods only update parameter weights as a result of a retrieval failure, and confirm to the “if it’s not broken do not fix it” school of thought.

2. Success-driven approaches seek to update parameter weights as a result of a retrieval success [12].

For each approach the weights of matching and unmatching parameters are increased or decreased accordingly. They use so-called pulling and pushing techniques to adjust the feature weights. Given a target case T and two cases A and B, if it is judged that A is a correct solution to T but B is not, the learning method
will push B away from T, and pull A close to T. As to
its weight-updating police, their introspective learning
method uses a decaying learning process as in the
following two formulae.

\[
\text{increase } W_i(t+1) = w_i(t) + \Delta_i F_c/K_c \quad (1)
\]

\[
\text{decrease } W_i(t+1) = w_i(t) - \Delta_i F_c/K_c \quad (2)
\]

where \( K_c \): represents the number of times that a case
has been correctly retrieved.

\( F_c \): represents the number of times that a case
has been incorrectly retrieved.

\( \Delta_i \): determines the initial weight change.

\( F_c/K_c \): the ratio between \( K_c \) and \( F_c \) is used to
reduce the influence of the weight update
as the number of successful retrievals
increases [11].

3.1.2 Learning k-value of k-Nearest Neighbor
Algorithm

k-Nearest Neighbor (k-NN) retrieval algorithm is a
derivative of the common Nearest Neighbor (NN)
algorithm. The only difference is that in the nearest
neighbor the system can retrieve one similar case
depending on measuring of how similar a target case is
to a source one. While, in the k-Nearest Neighbor we
retrieve k most similar neighbors to overcome the
limitation of NN due to the noise or errors in data. Then,
we apply a voting on the k similar cases to classify the
new case to the class with higher existence within the k
nearest neighbors [13].

The number of cases to be retrieved is an issue in
implementing the k-NN algorithm. Typical, k values
depend on the number of cases in the case-base and the
number of identified classes. To apply the CBR system
for different domains of applications, it is difficult to
determine these issues. Therefore, the proposed system
introduces a way to choose the number of cases as described in the following steps:

1. Set \( k = 1 \).
2. Apply k-NN algorithm on case-base using the
current value of \( k \)
3. Testing the correct classification of the cases for
the current \( k \) value. Then, store this \( k \) value, and the
ratio between the correct classification cases
compared to the failed ones.
4. Repeat step (2) for 5 iterations.
5. If these iterations were done without achieving a
better classification ratio, choose this value of \( k \) and
terminate the process. Otherwise, increment the
value of current \( k \) by 2 and repeat steps 2-4.

Therefore, the proposed system can improve the
performance of the retrieving algorithm in two axes.
Firstly, it uses an introspective learning approach to
extract the weights of the feature from the cases stored
in the case base. This eliminates the needs of the expert
in the field who can determine these weights and
eliminate the knowledge engineering effort required to
analyze the cases to extract these weights. Also, the
accuracy is increased due to eliminating the human
errors. The second axis is automating the determination
of the k-value used in the k-nearest neighbor. These
issues are very important for the little knowledge
domains, and for generalizing the proposed system to be
applied in different domains.

3.2 Adaptation Process

The success of a CBR system often critically
depends on its ability to adapt the solution of a previous
case to suit a new situation that has no exact match with
any stored case in the case base [14]. The simplest and
most widely used form of adaptation knowledge acts to
resolve feature differences between a target problem
and a retrieved case [15]. This type of CBR adaptation
rules is domain-specific and must be hardcoded by the
system developer. Hence, the knowledge-engineering
effort expended in one domain tends not to be re-usable
in other domains [16]. The obvious solution to this
problem is to somehow learn adaptation knowledge.
There are three main ways can be done: to exploit other
domain knowledge, to rely on an expert user or, their
preferred method, and to learn adaptation knowledge
from the case-base.

Adaptation knowledge can be learned using other
domain knowledge if a system is given general
adaptation knowledge rules which when applied, it can
result in specific adaptation cases those can be stored
for future use. The main advantage of this method is
that the domain knowledge used could already be
available in an existing knowledge-based system.
However, its main disadvantage lies in its reliance on a
large amount of such domain knowledge [16].

Adaptation knowledge could also be acquired
directly from the expert, by presenting a target problem
and a retrieved (but unadapted case) together. This
interactive approach is attractive but it would have to be
used in a restrained fashion, to avoid overburdening the
user.

The final possibility, which has received less
attention, is to acquire adaptation rules automatically
from case knowledge. A case-base contains a lot of
implicit domain knowledge and it makes sense to try to
avail of this knowledge in learning adaptation rules
[15].
3.2.1 Learning Adaptation Knowledge

To learn adaptation rules from cases, the system performs pair-wise comparisons of its cases.

1. The feature differences of each pair of cases are noted and become the antecedent part of an adaptation rule, while the consequent part of the adaptation rule constructs from the differences between the solutions in the compared cases.

2. The rule set is then generalized to cover the consequents of rules with matching antecedents. Each rule is assigned a certainty factor based on the number of times it occurred (i.e., the more often an association is found the higher its certainty factor).

This training process is off-line operation. While, at run time the retriever identifies the top $n$ best matches for a given target. Then, selecting pairs for comparison involves comparing each case in the case-base with only its top $n$ best match cases found by the retriever.

3.2.2 Proposed Adaptation Algorithm

The proposed system introduces a new algorithm that can be used in learning adaptation rules from the case knowledge as described in the following steps:

1. The system determines the differences between the new case and the most similar retrieved one(s).
2. Search for a rule that solve all these differences, if it is found use this rule to find the required solution.
3. Having no rule to solve the problem, the system searches the rule set for rules those can substitute one or more differences between the new and retrieved cases.
4. Reduction of the differences continues by dropping one or more differences from the differences list till reach for rule handling one difference only.
5. Repeat step 3-4 till solve all the differences of the list.
6. Rules those solve the problem are combined to generate a new rule that represents the solution of this problem, and store it. Then, display the solution for the user.
7. If all the differences are not covered, system reorder the reminder differences and the corresponding rules used according to the weights of the features of the differences.
8. The partial adapted solutions are displayed for the user to choose the most suitable one.

Best rule can be used in this adaptation process is the simple rules which can be represented in the form (if simple-condition then action). If you have a set of simple rule, it will facilitate to execute this rule one for each difference between parameters. To get this simple rule, the system must go in loop to make substitution by known simple rule in the composite or the rule contains conjunctive condition. This leads to a set of simple rules that contain simple condition with its own action. This set of rules helps in determining a solution for new problems those haven’t achieved before. A main important issue of this algorithm is its suitability to apply for the flat vector domains only.

4. PROPOSED SYSTEM APPLICABILITY

The suggested system is a general purpose CBR system that can be applied for different domains, solving different problem types. It has been applied for many domains and tasks, and compared its results to See5 system [17, 18]. See5 is a CBR system that analyzes data to produce decision trees and/or rule sets that relate a case’s class to the values of its attributes. See5 depends on inductive retrieval which generates decision trees while our system depends on the k-Nearest Neighbor retrieval algorithm. Running both our system and see5 for the same problems. The results of both systems are compared in the following sections. The main common issue between the two systems, their driving of the rules from cases.

4.1 The Breast-Cancer Domain

The breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. Their cases contain 19 features. While, its case-base contains 699 cases each one is either benign or malignant. the distribution of classes is as follows. Benign: 458 (65.5%) and Malignant: 241 (34.5%). The case-base was divided into 500 cases for training, and 199 cases for testing. The following table demonstrates a comparison between the results of see5 and our system.

<table>
<thead>
<tr>
<th></th>
<th>See5</th>
<th>Proposed CBR System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wrong</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error %</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>497</td>
<td>484</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.032</td>
</tr>
<tr>
<td>Testing</td>
<td>187</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6.0</td>
<td>4.0</td>
</tr>
</tbody>
</table>
4.2 Boston Housing Price Data

In this problem there is two possible classes to be predicted represented by 13 features as: Top 20\% , and Bottom 80\% . The Case base contains 350 for training and 156 case for testing. The following table demonstrates a comparison between the results of see5 and our system.

<table>
<thead>
<tr>
<th></th>
<th>See5</th>
<th>Proposed CBR System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>Wrong</td>
<td>Error %</td>
</tr>
<tr>
<td>Training</td>
<td>340</td>
<td>10</td>
</tr>
<tr>
<td>Testing</td>
<td>139</td>
<td>17</td>
</tr>
</tbody>
</table>

4.3 Preparation of Electronic Laboratory

There are three available classes for this domain. The case base contains 138 for training and 100 case for testing. The following table demonstrates a comparison between the results of see5 and our system.

<table>
<thead>
<tr>
<th></th>
<th>See5</th>
<th>Proposed CBR System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>Wrong</td>
<td>Error %</td>
</tr>
<tr>
<td>Training</td>
<td>130</td>
<td>8</td>
</tr>
<tr>
<td>Testing</td>
<td>75</td>
<td>25</td>
</tr>
</tbody>
</table>

In this domain our system performed worst than see5 since the number of training cases is not enough to cover the features space. So, cases in the test are misclassified due to lack of similar cases in the case-base. Increasing the number of the training cases will increase the performance of the proposed system in this topic.

5. CONCLUSION

Two algorithms were developed in this study. The first is: automating the extraction of the feature weights of the cases, and k value of k-nearest neighbor retrieval algorithm. While, the second is concerned to automate the extraction of the adaptation rules used for adaptation process from the case knowledge. The proposed system is tested for four domains are: The breast-cancer domain, Boston housing price data, and Preparation of electronic laboratory. The results of the suggested system conclude a decrease of the errors of the training and test cases compared to See5 system. Nevertheless, it has less performance for the domains that have low number of stored cases required for the learning task. The advantages of these algorithms are:

1. It has a self-learning extraction of the weights for the retrieval process using the introspective learning technique.
2. It automates extraction of k-values of k-nearest neighbor retrieval algorithm.
3. It can be applied for the little knowledge-domains those are ill in determining the required weights either by the expert or by the statistical technique.
4. It can extract the knowledge for the adaptation rules from the stored cases.
5. Extraction of adaptation rules is off-line process to minimize the run-time of the system.
6. It uses substitution methodology to drive a set of simple rules to be applied for unseen problem in the future.
7. It constructs the rule used to solve the present problem from the used stored rules, and stores it to avoid repeating the same effort again.
8. The updating of the rule set after any new added rule is off-line process to save the run time of the system.
9. It can provide a complete automation of both the main tasks of CBR methodology (e.g. retrieving & adaptation).
10. It can be applied for several domains of applications.

REFERENCES


ICITNS 2003 International Conference on Information Technology and Natural Sciences

Fig. (1): Retrieval and adaptation algorithms