Improving Case Retrieval by Remembering Questions*

Richard Alterman and **Daniel Griffin**
Computer Science Department
Brandeis University
Waltham, MA 02254
{alterman,dang}@cs.brandeis.edu

Abstract

This paper discusses techniques that improve the performance of a case retrieval system, after it is deployed, as a result of the continued usage of the system, by remembering previous episodes of question answering. The user generates a request for information and the system responds with the retrieval of relevant case(s). A history of such transactional behavior over a given set of data is maintained by the system and used as a foundation for adapting its future retrieval behavior. With each transaction, the system acquires information about the usage of the system that is subsequently used to adjust the behavior of the system. This notion of a case retrieval system draws on a distinction between the system in isolation and the system as it is used for a particular set of cases. It also draws on distinctions between the designed system, the deployed system, and the system that emerges as it is used.

Introduction

In this paper we will develop the notion of keeping a history of question/answer transactions as basis for system adaptation as it applies to the construction and usage of a case-based reasoning (CBR) system. Given an existing CBR retriever, the approach we describe is to augment the system with a module that remembers all the questions that were previously asked and the answers/cases that were generated in response, and uses that information to improve the performance of the retriever — we will refer to such a retriever as a *use-adapted case retriever*. Through continued usage, the use-adapted system gradually acquires skill at answering (with cases) the questions that are most frequently asked. Part of the skill is recognizing that a question has been asked and answered before. Another part is knowing how to fill in missing details for a given question in order to make sense of it. A third part is knowing answers that are inappropriate for a given question. We will show that remembering questions enhances the potential of system performance by allowing the system to adapt is behavior. Over time the system gradually begins to reason about future questions not only based on the content of the cases, but also based on the previous questions and their associations to cases or other questions.

*This work was supported in part by DEC (Contract 1717). Additional support for this work came from NSF (ISI-9634102).

Why A Use-Adapted Case Retriever?

In order to build the retrieval system several decisions must be made. Each of these decisions effects the overall performance of the retriever. Data must be collected. The system developer must confront the following issues: 1) deciding what constitutes a case, 2) organizing each 'case' using some uniform syntactic structure, 3) identifying a vocabulary composed of keywords, terms, and expressions, and 4) indexing and clustering the cases. Both automatic and by-hand approaches can be taken to selecting an indexing vocabulary for a given case base. Several vocabularies have been touted for specific kinds of domains (see, e.g., Hammond, 1990; Leake 1991); others have proposed a standard indexing vocabulary that could be used to index cases across a wide selection of domains (Schank et. al, 1990); and still others suggest that features can be ranked or refined dynamically (Rissland & Ashley, 1986, Fox & Leake 1995). After an indexing vocabulary has been established, additional work must be done in identifying synonyms. Next, each of the cases must be indexed; indexing can be done either by hand or automatically using some clustering algorithm. Finally, an interface for retrieval must be developed.

An underlying assumption of this standard process is that there exists a 'right' index for a given case; an alternate possibility, and one implicit to the model we present, is that the index for a given case is dependent on the interaction of the end-user and the retrieval system. Finding a relevant case is dependent on not only the data but also on issues like how does the interface present access to the data, how was the question asked, who asked the question, and for what application. These kinds of issues are contingent on the ongoing interaction between the end-user and the system, and it takes experience and skill to be able to identify

the nature of the question and its application. These are things that can only be learned after the system has begun to be used.

In general, in order to build a useful retrieval system, the system-builder must cycle through the process of building a retriever, deploying it, and then trying again. The problem is that it is not all clear how each change to an indexing vocabulary and resulting classification effects the overall performance of the system. Thus, in many cases, if things are not working, it is very difficult to decide what needs to be fixed. If you change the indexing vocabulary and then reclassify the cases, it is difficult to tell whether you have not fixed one problem by introducing another. (These problems are compounded by the fact that the initial retriever might not be effective at all.) Although this approach allows for some field testing, the vocabulary for indexing and the actual indexing of cases is done mostly independent of any extensive usage of the retriever. In any case, by the time the retriever is deployed, and in use by the end-user, the retriever has pretty much settled into a level of performance.

The goal of the use-adapted retriever, seeded by the initial efforts of the system builder, is to continue to improve the performance of the retriever by working on building a useful set of correspondences between external user descriptions and the cases in the case base. Through the usage of a case retriever, the system refines and improves on earlier versions of the retriever. It replaces a process of development that can tend to lack direction, by one that is automatically guided by the successes and failures of the retrieval system. Finding a better way of indexing a given case, learning synonymous expressions, and identifying the questions most frequently asked, are all things that the retrieval system can learn after it has begun to be used.

Outside the research lab

Outside of the research lab, many of the issues involved in building a case-retriever become even more complicated.

- Data may originate from existing data collections; a problem is that the structure and/or meaning of the data may not be clear or easily accessible. The system developers must either work with the data in the form it was collected, or spend time converting the data to some other format which makes it easier to use.
- Even when the data is collected with CBR in mind, cases may have been gathered by different people resulting in some non-uniformity in the vocabulary to describe each case. Although it would be nice to establish a uniform vocabulary in advance, for many reasons, including lack of knowledge about the cases, it is often not possible to do this in advance. Thus a fair amount of time can be spent identifying a useful vocabulary and re-encoding cases for the purposes of establishing some uniformity which is a necessary condition for a successful clustering and retrieval.
- The same data might be used for two different applications, with each application emphasizing different vocabularies and clusterings. Similar problems can exist for situations where you have the same application, but two different end-users of the system.

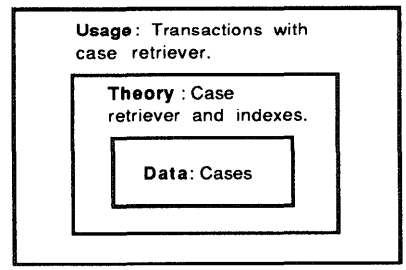

Figure 1: A use-adapted case-based reasoner.

Many of these problems derive from the fact that the data collector, the system builder, and the end-user, may all be different groups of people, with different skills, time constraints, and levels of expertise. By at least in part automating the process of deciding and re-deciding how to get the case retriever to work, a use-adapted case-retriever allows some of the decisions regarding working out the details of the translation to be down-loaded to the period after the system is deployed and in use. At some point one can get out of the cycle of building the retriever and start using it. It is the task of the use-adapted retriever to work out correspondences between queries and cases. Over time, difficulties presented by issues like the non-uniformity of vocabulary, or the biases of different applications, can be gradually improved on by the strategies of the evolving use-adapted system.

Use-Adapted Case-Base Retriever

Our interest is in the pragmatics of how the user accomplishes the task of case retrieval given a particular system and a particular set of cases. Our idea is to look at the particulars of usage of the system as they are represented as episodes of user-system retrieval. The basic idea is depicted in Figure 1. There exists as data a set of cases. A system/theory is built for retrieving cases that are relevant to a given request for information. The usage of the system is characterized as a history of transactions between users of the system and the system over a given set of cases, over an extended period of time.

In this paper, we consider techniques that exploit the history of one sort of transaction between end user and system: question and answer. The user generates a request for information and the system responds with the retrieval of relevant case(s). Given an existing CBR retriever, the approach we describe is to augment the

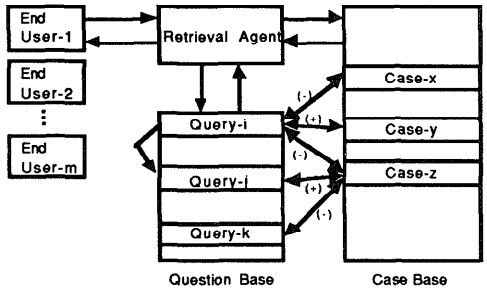

Figure 2: Architecture of use-adapted retriever.

system with a module that remembers all the questions that were previously asked and the answers/cases that were generated in response, and uses that information to improve the performance of the retriever. Each time the end-user asks a question of the system and sifts through the list of cases that are recalled until an appropriate case(s) is found, the system retains a record in the form of a set of transactions between the end-user and the retrieval system, marked for the relevance of the retrieved item to the question, i.e.,

```
(associate <question>
           <ID of case>
           <relevance-of-case-to-question>)
```

Sometimes the association that is retained is not only between a question and a case, but also between two questions.

Attaching relevance markers to a given transaction has a relatively low cost for the end-user. During the normal business of retrieving cases, the user must step through the list of retrieved cases and evaluate them. Marking one or another case that is being looked at as relevant (+), not relevant (-), or neutral (0) adds little additional cost for the end-user, yet it is likely to provide improvement in the retrieval system. Even in the case that the end-user is unwilling to evaluate all the cases on the retrieval list for a given question, the system will continue to improve but at a more gradual pace.

Figure 2 shows the basic architecture of a use-adapted case retriever. Interposed between the end-user and the system is a retrieval agent that keeps a history of the interaction between the end-user(s) and the system and uses that information to reason about the retrieval process. Thus the system has two case-bases: one for domain knowledge and a second for the interaction between the user and the system. Relevance links represent the positive and negative associations between pairs of questions and cases.

Basic Case Retrieval System

The first step in our study was to build a base help-desk system that retrieved relevant bug reports. In these experiments, we used a case-base of 293 cases. Each case was a diagnosis of a bug in a release of software consisting of mostly unstructured pieces of text and was entered by one of a number of different people. We identified a list of about 348 keywords used in the description of cases. We used the ID3 algorithm (Quinlan 1986) to build a clustering of the cases over the keywords.

The usage of the base retrieval system can be described as follows. A user specifies a question or information need from which a list of keywords are extracted by the system . These keywords are collected in a query that will be referred to as the *user-query*. The case-base is organized as a binary ID3 tree with nodes separating cases by features (does the case have feature X? Yes or No). The retriever traverses the tree using the user supplied keywords to decide which branch to take at a node. In the situation when an impasse occurs, i.e., when a node in the tree specifies a feature Y not supplied by the user, the system moves into an interactive mode where the user is allowed to say:

1. Take the branch with cases that have feature Y
2. Take the branch with cases that don't have feature Y
3. Take both branches ("I don't know" option)

The user's decisions are recorded in the interactive portion of the retrieval, and along with the initial *user-query* will be referred to as the *user-expanded query*. This process is repeated until cases are reached at the leaf nodes. After retrieval is complete, the user steps through the cases marking each case as relevant (+), not relevant (-), or neutral (0) to the user's question.

Use-Adapted Retrieval Techniques

Recognizing Questions

Our initial idea was that over time the system would become more effective at answering questions, because it would begin to recognize the most frequently asked questions and gradually develop expertise at responding to those questions. Keeping a question-base allows the retrieval system to build up expertise about the cases most regularly asked about. Questions are 'recognized' by retrieving from the question-base using the user-query (see Figure 3). If the question is not 'recognized' as being similar to any previous questions, or if the similar previous questions had no positive associations, then the clustering based on the content of the cases is used to retrieve cases.

Notice that our use of the question-base does not preclude the system from continuing to use the content of the case as a basis for retrieval. This is especially important early on when there are not many previous question and answering episodes available; it is also important on the occasions when the system is evolving due to the development of new applications, interfaces, or kinds of users.

Automatic Question Expansion

Another way that the system can exploit positive associations between previously asked questions and cases

Given a new user query q_{new}, let $QUERY-RS$ be the set of queries retrieved from the query base, and let $CASE-RS$ be the empty set.

1. For each query $q_j \in QUERY-RS$:
   (a) If q_j produced positively evaluated cases, then include them in $CASE-RS$
2. If $CASE-RS = \emptyset$ do normal case base retrieval using q_{new}.
3. Otherwise return $CASE-RS$.

Figure 3: TECHNIQUE 1: Recognize a question.

is a technique by which the system 'fills out' a user question; when an impasse occurs the system attempts to automatically guess the value of the unspecified feature. The basic idea is to retrieve from the question-base a selection of similar questions and poll the positive cases attached to each retrieved question for values to expand the query (see Figure 4).

Let q_{new} be the user's initial query.
When guessing at decision point for feature X:

1. Retrieve queries from the question base using q_{new}.
2. From these queries, collect the positively associated cases into $CASE-RS$.
3. Use each $c_i \in CASE-RS$ to vote on the value for feature X.
4. If over a certain threshold of the cases vote for a certain value, then guess that value. Otherwise, default to the choice made by the user.

Figure 4: TECHNIQUE 2: Expand a question.

Pruning

Suppose that a case retriever selects a set of cases as relevant to a given question. Figure 5 shows a technique that takes advantage of negative associations between questions and answers. After the initial list of cases are retrieved, they are each evaluated using the list of previous queries associated with that case. If the current query only matches the negatively associated queries attached to a retrieved case, then the case is deleted from the retrieval set. Using previous questions to delete cases can potentially improve the performance of the system by reducing the number of non-relevant cases retrieved; the system learns to recognize incorrect responses to a question.

Evaluation

Seven subjects were enlisted to write queries. All of the subjects wrote queries for seven of the same target cases. Each of the subjects also wrote queries for seven unique cases. Finally the subjects were divided into three groups (two of the groups were size 2 and the

Given a new query q_{new} and a set $CASE-RS$ of retrieved cases:

1. For each case $c_i \in CASE-RS$, let Q_i be the set of queries that have retrieved c_i in the past.
   (a) If Q_i is the empty set do nothing.
   (b) If only negative previous queries match q_{new}, delete c_i from $CASE-RS$.
   (c) Else, do nothing.

Figure 5: TECHNIQUE 3: Pruning cases.

third group was of size 3), and each group wrote queries targeted for 7 cases unique to that group. Thus, in total, each subject wrote queries for 21 cases. After the test subjects finished entering their queries, we had 239 queries. This number is larger than the expected 21 per user because the users were encouraged to try several times if a retrieval was not successful. Of the 239 queries, 132 were used as a training set and the remaining 107 as the test set. The test set was divided into 2 sub-groups: those questions that had no relevant history (numbering 64) and those that did (numbering 43).

We used two standard metrics for evaluating the effectiveness of the techniques: *precision* and *recall*. For a given retrieval, let

a = the number of relevant cases retrieved
b = the number of relevant cases in the case-base
c = the number of cases in the retrieval-set

then $precision = \frac{a}{c}$ and $recall = \frac{a}{b}$.

For the test data we present each query has a single target case. Thus, a high percentage for precision directly translates into a short retrieval list (and less work for the user in evaluating the retrieved cases). Also, since the recall for a given retrieval is either 0 or 1, the average recall number tells you for what percentage of the test queries the target case was retrieved.

We discuss tests on queries with and without history separately. The numbers we present for precision and recall are averages. A precision score of 66.42 should be read as "on average 66.42% of the list of retrieved cases was relevant to the question." A recall score of 72.09 should be read as "on average 72.09% of the test queries retrieved the targeted case." We used paired comparison tests with a 95% confidence interval to verify the significance of our results.

Testing Technique 1: Recognizing Questions
We looked at two versions of the recognition technique. We use the term *short form* to indicate the question as it was initially posed by the user; the term *long form* refers to the query as it was interactively expanded by the user.

Comparing the basic retrieval mechanism (with interactive expansion) to recognition retrieval (short form) gives the following average results for precision

and recall on the questions with a history:

Technique	Precision	Recall
Case-Base (interactive expansion)	15.59	58.14
Recognition (short form)	66.42	72.09

The results show that a dramatic improvement in precision was achieved, as well as a significant increase in recall. Moreover, these numbers are achieved without the user having to do the extra work of interactively expanding the query. When the long form of the query is stored in the question base, the results of comparisons are:

Technique	Precision	Recall
Case-Base (interactive expansion)	15.59	58.14
Recognition (long form)	49.96	69.77

This is still an improvement over case retrieval with interactive expansion, but the results are not as impressive as that for recognition using the short form of the query. A possible explanation for this is that the user is either making "bad guesses" or "guessing both branches" (or both) and these decisions are recorded in the long form of the query. Thus, the variability introduced by user guesswork would make initially similar questions, dissimilar.

Testing Technique 2: Automatic Expansion
For basic retrieval with interactive expansion, on average we found that the user interactively added 18.7 features per question. This translates into extra work for the end user.

In this part of the analysis we tested the automatic expansion technique as a possible substitute for some interactive expansion. With automatic expansion, we found that the user on average was interactively adding only 4.14 features for the questions that had a history, a clear reduction in work for the user. When we compared automatic query expansion to interactive user expansion, we obtained the following results:

Technique	Precision	Recall
Case-Base (interactive expansion)	15.59	58.14
Automatic Expansion	40.61	69.77

Automatic expansion does significantly better than user expansion on both precision and recall. These results suggest that over time the user can forego some of the effort of interactively expanding the query as that the system will be able to guess at many of the features for those questions that have a history.

Testing Technique 3: Pruning
Where both recognition and automatic expansion take advantage of the positive associations between old questions and cases, pruning is a technique that exploits the negative associations between questions and cases. The results of pruning retrievals produced by each discussed technique are shown below.

Technique	Prec.	Pruned Prec.
Case-Base (interactive expansion)	15.59	37.33
Recognition (short form)	66.42	66.8
Recognition (long form)	49.96	53.23
Automatic Expansion	40.61	64.48

Our results show that pruning improves precision (by removing irrelevant cases from the retrieval list). Since no significant changes were seen in the recall scores, we show only the precision scores.

Tests on Queries with No Relevant History

Another thing we tested for was whether performance on the questions that had no relevant history would somehow degrade as a result of using the history-based techniques. For the 64 test questions that had no relevant history, there was no significant decrease in performance for any of the techniques that used the history of the user's interaction with the system.

Initially we had hoped that either automatic expansion or pruning would somehow improve system performance for questions with no previous history. In neither case was there significant improvement in performance.

Related Work

Relation of Model to CBR Literature

Others have explored the role of 'unanswered' questions in directing learning (Ram & Hunter, 1992) or reading (Ram, 1991; Carpenter & Alterman, 1994). Our interest here is in retaining and using a memory of previously 'answered' questions. The notion of remembering answered questions is different from other approaches to learning that have been applied to case retrieval. Where incremental clusterers, e.g. CYRUS (Kolodner, 1983), UNIMEM (Lebowitz, 1987), and COBWEB (Fisher, 1987), update and re-organize a case-base with the addition of each new case by comparing cases, the model we propose attempts to improve retrieval performance of the system on cases already existing in the case-base by acquiring information about the usage of those cases. There is a difference between updating a system by adding new cases, and updating a system by using it on the cases that already exist in the case-base. The former approach is geared towards updating the clustering of cases by comparing existing cases to new cases as they are acquired. The latter, our approach, improves the quality of retrieval by accumulating information about the usage of a given (existing) case. With our approach the system will continue to improve retrieval performance even after the set of cases in the case-base has stabilized.

In contrast to systems that attempt to learn a best index for a case by refining the index of a case based on retrieval performance (e.g. Rissland and Ashley, 1986; Veloso and Carbonell, 1993; Fox and Leake, 1995), our approach is to learn the multiple kinds and instances of

questions that either positively or negatively retrieve a case.

Yet another view of case retrieval has been that cases have "right indices". The view that cases have "right indices" is inherent to the various vocabularies that have been touted for specific kinds of domains (see e.g. Hammond, 1990; Leake 1991) or specific indexing schemes that are claimed to apply to a wide selection of domains (Schank et. al, 1990). Systems that do indexing based on a clustering of the cases can also end up with a single index for a given case. By separating questions from cases, the case retriever can view cases from the perspective of their usage — in principle each question for which the case is relevant is potentially another index for the case.

Other Work

Others have also considered the idea of building retrieval mechanisms that self-customize to a given data set. In Carpenter & Alterman (1994) a reading agent SPRITe is described that automatically builds customized reading plans for returning information from specific sets of instructions. In Armstrong et. al. (1995), WebWatcher is described. Webwatcher is an agent for assisting users in navigating the world wide web, and it can be "attached to any web page for which a specialized search assistant would be useful." Both of these efforts share our intuition that retrieval performance can be tied to the characteristics of particular data sets: for SPRITe the data sets are text and for Webwatcher, a web page.

In IR, there has also been work on query expansion using relevance feedback (Salton & Buckley, 1990) provided by the user. Our work on query expansion offers a history-based approach to expanding a query. A major advantage of our approach is that the work of the end-user is greatly reduced because expansion occurs automatically. Another important issue in IR is deciphering the intended meaning of the query (Sparck Jones, 1992). This research explicitly addresses this issue, as that automatic expansion is an example of a technique that can decipher the meaning of a newly posed question.

Although there is room for technical transfer between IR and CBR, it is also important to differentiate them (see Rissland & Daniels, 1995). One obvious difference is that CBR is not only concerned with retrieval but also with questions of adaptation. Thus in CBR planning, any relevant case may suffice (because it can be adapted), but for IR and document retrieval all relevant cases may need to be retrieved. Another important, and more pertinent, difference concerns the size of the data set. IR technology was developed to work with document files numbering in the tens of thousands, but many CBR applications only need work with a few hundred cases. Technology that is developed towards working with huge numbers of documents will not necessarily *scale down* as the best approach to working with case libraries numbering in the hundreds where, over time, effective retrieval may exhibit the characteristics of a skill acquisition problem. It is not clear that techniques developed in IR that work for constantly changing and growing data sets (e.g., the associated press wire service) are likely to be the ideal method for smaller and more stable data sets.

Summary Remark

In this paper we have shown the efficacy of a use-adapted case retriever. We have presented several techniques and experimentally demonstrated their effectiveness in improving system performance. In each case, these techniques exploited the historic interaction between user and system over a given set of data (not an independent analysis of either the user or the data).

References

Armstrong, R.; Freitag, D.; Joachims, T.; and Mitchell, T. 1995. Webwatcher: A learning apprentice for the world wide web. In *Proceedings of 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments.*

Carpenter, T., and Alterman, R. 1994. A reading agent. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.* Menlo Park, CA: AAAI Press.

Fisher, D. H. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2:139–172.

Fox, S., and Leake, D. B. 1995. Using introspective reasoning to refine indexing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 391–397.

Hammond, K. J. 1990. Case-based planning: A framework for planning from experience. *Cognitive Science* 14:385–443.

Kolodner, J. L. 1983. Reconstructive memory: A computer model. *Cognitive Science* 7:281–328.

Leake, D. B. 1991. An indexing vocabulary for case-based explanation. In Dean, T., and McKeown, K., eds., *Proceedings of the Ninth National Conference on Artificial Intelligence.* AAAI.

Lebowitz, M. 1987. Experiments with incremental concept formation: Unimem. *Machine Learning* 2:103–138.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Ram, A., and Hunter, L. 1992. The use of explicit goals for knowledge to guide inference and learning. *Applied Intelligence* 2:47–73.

Ram, A. 1991. A theory of questions and question asking. *Journal of Learning Sciences* 1:273–318.

Rissland, E., and Ashley, K. 1986. Hypotheticals as heuristic device. In *Proceedings of the Fifth National Conference on Artificial Intelligence.*

Rissland, E. L., and Daniels, J. J. 1995. Using cbr to drive ir. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 400–407.

Salton, G., and Buckley, C. 1990. Improving retrieval performance with relevance feedback. *Journal of the American Society for Information Science* 41(4):288–297.

Schank, R. 1990. Toward a general content theory of indices. In *Proceedings, AAAI Spring Symposium*, 36–40.

Sparck-Jones, K. 1992. Assumptions and issues in text-based retrieval. In Jacobs, P., ed., *Text-Based Intelligent Systems.* Hillsdale, NJ: Lawrence Erlbaum Associates. 157–177.

Veloso, M., and Carbonell, J. 1993. Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. *Machine Learning* 10:249–278.