Improving Case Retrieval by Remembering Questions *

Daniel Griffin and **Richard Alterman**
Center for Complex Systems
Computer Science Department
Brandeis University
Waltham, MA 02254
dang@cs.brandeis.edu
alterman@cs.brandeis.edu

Abstract

The goal of this research is to develop technology for interactive case retrieval systems that improve their performance, after they have been deployed, by remembering previous questions.

Depending on the origin of the case-base, one can expect a variable vocabulary and structure to the cases, leading to difficulties in determining exactly how the system should use the cases. By remembering questions, structure and uniformity can be built into the case-base, leading to improved system performance.

In this paper, we describe a retrieval system that takes advantage of a question history. Experimental results are presented which confirm that this technique improves precision and recall of the system.

Introduction

The goal of this research is to develop technology for building interactive case retrieval systems that improve their performance, after they have been deployed, by remembering previous questions.

In general we see several advantages to systems that remember questions:

- It allows the system to build expertise for the questions most frequently asked.
- It offers a way to profile and differentiate end-users, classes of end-users, and their associated concerns.
- It allows the system to structure and to smooth out inconsistencies in the data.

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 as well as the answers/cases that were generated in response to the question along with relevance ratings for each case. The module uses that information to improve the performance of the retriever.

In the normal course of events, a CBR system will collect cases and use them to improve its performance. There may come a point, however, when the case-base begins to stabilize and the rate of cases being added to the case-base slows down or even stops. Take for example a CBR system that reasons about legal precedents; there is a fairly stable set of court cases that the system has in its case-base, and these cases are the basis for the reasoning process. It is at this point that the system could continue to learn about its domain by learning about the usage of the cases in its case-base.

The way that cases in the case-base are gathered/structured, may reflect the design time uncertainty as to the eventual usage of the cases. The methods discussed in this paper offer an approach to mitigating the uncertainty in these kinds of domains. The notion is that after the system is built and deployed you want it to continue to develop. Inconsistencies and problems in the data can gradually be overcome through the usage of the system. Thus, for example, a case which initially has a bad index, will eventually acquire a set of questions that can be used as alternative indexes which better reflect the usage of the case.

The notion of remembering questions, as a method for modeling the usage of a given data set, can be applied to a wide range of applications of CBR technology. The key ingredient is to include the interaction between the end-user and the retriever as a part of the evolution of the system. In this paper we will apply this notion to a help desk application, but we are also looking at this as a technique to support multi-agent memory systems and certain online information systems.

Basic Retrieval Mechanism

The base retrieval system can be described as follows. A user specifies a list of keywords which are relevant to his information need. This will be referred to as the *user-query*. Queries are of the form,

$$((keyword_1\ key\text{-}flag_1) \ldots (keyword_n\ key\text{-}flag_n))$$

where $key\text{-}flag_i$ is a boolean value denoting the presence or absence of $keyword_i$ in a desired case. The case-base is organized as a binary ID3 tree (Quinlan, 1986) with nodes separating cases by features (does the case have feature X? Yes or No). The retriever will traverse the tree using the user supplied keywords to

*This work was supported in part by DEC Contract 1717

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 users 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 users question. Although view this evaluation process as minimal work for the end-user, it may not be necessary for the user to evaluate all of the retrieved cases. The system can still use the partial information supplied by the user to improve its performance.

Remembering Previous Episodes

The basic idea is to track the history of questions. Each time the end-user asks a question of the system, he sifts through the list of cases that are recalled marking cases according to their relevance. The system retains a record in the form of a set of transactions between the end-user and the retrieval system. Thus, the basic unit of our analysis is:

```
(associate <question>
           <case and its index>
           <relevance-of-case-to-question>)
```

The system learns associations between questions and cases; those cases that are retrieved and relevant and also cases that are retrieved and that are not relevant are both added to the history. Sometimes the association that is retained is not only between a question and a case, but also between two questions, i.e.,

```
(associate <question> <question>)
```

Such a situation would occur when an initial question does not produce an adequate answer for one reason or another and the end-user tries a different approach.

Building up a history of associations between questions and cases (or between two questions) leads to a different view of the cases then what you get by organizing cases based on their content. By keeping track of questions, the retrieval system can begin to learn the different ways of asking about a case, and also learn about questions which should not retrieve a given case. Remembering questions enhances the potential of system performance by allowing the system 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.

We are exploring two methods for storing previous episodes of question answering (Figure 1). In this paper we will discuss in detail one technique that exploits the first method of storing question-answering histories: recognizing a question.

Techniques that Exploit a Question-Base

Our research at this point has investigated several ways to exploit the existence of a question-base. Here we will discuss one such method.

The strategy for storing the history of questions discussed in this paper is to form a separate case-base of questions. Keeping a question-base allows the retrieval system to build up expertise about the cases most regularly asked about and to recognize different questions and their correct responses. Under this scheme the retriever would first use the question-base to see if it recognized the question. Questions are 'recognized' by first retrieving from the question-base using the user-expanded-query, and then pruning the retrieval list by applying a weighted similarity function to the match between the new question and each previous question. 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 (see Figure 2).

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.

Evaluation

In this experiment, 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 to build a clustering of the cases over the keywords. A simple interface was developed that allowed the user to enter a query in a keyword format.

We then enlisted seven subjects 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 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

1. Create a case-base of previous questions.
   (a) Maintain memory of questions in terms of differences and similarities of questions themselves.
   (b) Alternate view of cases based on questions.
2. Attach questions to cases.
   (a) Organize questions around existing organization of cases.
   (b) Filter a retrieved case using questions attached to the case.

Figure 1: Two approaches to keeping a memory of previous questions.

Given test query q_i, let Q_i be the set of queries retrieved from the query base, and let R be the initially empty set of cases retrieved by this technique.

1. For each query $q_j \in Q_i$:
   (a) If q_j produced positively evaluated cases, then include them in R
   (b) Otherwise include no cases from q_j in R.
2. If $R = \emptyset$ do normal case base retrieval using q_i.
3. Otherwise return R.

Figure 2: TECHNIQUE 1: Recognize a question.

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 our techniques:

Precision is defined as the proportion of relevant cases in the retrieved set of cases.

Recall is defined as the proportion of relevant retrieved cases to the total number of relevant cases in the case-base.

Precision tells you what percentage of the retrieval list is relevant. Recall tells you what percentage of the relevant cases are retrieved. For this particular data set (case-base) there was only one relevant case per test query. This affects the interpretation of of precision and recall in the following ways:

- A high precision directly translates into a short retrieval list (and less work for the user in evaluating the retrieved cases).
- Since for each test query recall is either 0 or 1, when recall is averaged over all of the test queries, the result tells you in what percentage of the test queries the target case was retrieved.

Testing Technique 1: Recognizing Questions

We tested Technique 1 (recognizing questions) as follows. The test set was divided into 2 groups those question that had no relevant history (numbering 64) and those that did (numbering 43). For the purposes of this paper, we will present the results of technique 1 on the queries which had a relevant history only.

This experiment was also run twice, once with unexpanded queries, and the second time with expanded queries. That is, Technique 1 is run twice, once with q_i equal to *user-query* and once with q_i equal to *expanded-query* (see Figure 2). The results of testing technique 1 using user-query retrievals are shown in Table 1.

As you can see, both techniques resulted in high recall. This is due to the fact that user-query is fairly general and the retrievals are therefore very large. Recognition gave no benefit to recall, but it did not decrease it either. Precision, on the other hand, benefits from the recognition technique since when a question is recognized, only the correct answer (case) is returned. This results in a smaller retrieval set, and a higher precision. The precision of the recognition technique is better than that for the case-base retrieval.

The results of testing technique 1 using user-expanded-query retrievals are shown in Table 2. Using the more specific expanded-query, recall is reduced in both the case-base retrieval and the recognition based retrieval, but as you can see, recognition performs better than the case-base retrieval. Precision using recognition has shot up to almost 50% whereas it has remained at 16% for case-base retrieval. Both precision

Technique	Precision	Recall
Case-Base	0.0446	0.9302
Recognition	0.1459	0.9302
Improvement (95% confidence)	$0.0186 \geq \mu_\delta \geq 0.1841$	$0.0000 \geq \mu_\delta \geq 0.0000$

Table 1: Recognize using unexpanded Retrievals

Technique	Precision	Recall
Case-Base (user-expanded)	0.1559	0.5814
Recognition (user-expanded)	0.4996	0.6977
Improvement (95% confidence)	$0.2176 \geq \mu_\delta \geq 0.4698$	$0.0163 \geq \mu_\delta \geq 0.2162$

Table 2: Recognize using user-expanded Retrievals

and recall for recognition are better than that for case-base retrieval within a 95% confidence interval.

Overall this experiment confirms that the technique of recognizing questions does improve the performance of the system for questions that have a relevant history.

Discussion and Related Work

An important feature of our system is that it maintains the difference between questions and the content of the cases. In our view there are several reasons to do this.

- There can be large inconsistencies in the data. This can result from different people gathering the data and non-uniformity in the vocabulary used to write-up each case. These kinds of inconsistencies can be filtered out when building the retriever by establishing a uniform vocabulary for retrieval. Thus where the initial cases may be a hodge-podge of vocabulary and description styles, from the vantage point of the questions the cases will appear to have a higher degree of uniformity.
- Different end-users (within the same application) may have different styles of asking questions. Keeping around explicit information about the questions being asked allows the system the potential to learn these kinds of systematic differences and take advantage of them.
- In addition to there being inconsistencies in the data, it may be the case that a CBR system is using already existing data as its case-base, and this data may have little or no structure. By building a collection of questions and answers, we are in effect building structure into the data based upon its usage.

With regards to the literature there are several points to be made:

- 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.
- Improving system performance by adjusting to a particular individual or individuals profile has been investigated (see McTear, 1993). These systems adjust their behavior based upon consideration of the users goals, capabilities, preferences or beliefs. These factors are used along with fairly detailed domain knowledge to adapt system performance to better suit the user. A problem with these techniques is that they use some form of apriori knowledge about how a system is used by users with certain goals, capabilities, preferences or beliefs. This knowledge is not always available or obvious for a given domain, or it may be prohibitively expensive to extract such information.
- In contrast to specific indexing schemes that are claimed to apply to a wide selection of domains (Schank et. al, 1990), the approach to indexing here is the indexing and retrieval is contingent on the ongoing interaction between the end-user and the system. By separating questions from cases, the case retriever can view cases from the perspective of their usage — in principle each question the case is relevant for is another index for the case.
- 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 and/or utility (e.g. Rissland & Ashley, 1986; Veloso & Carbonell, 1993), our approach is to learn the multiple kinds and instances of questions that either positively or negatively retrieve a case.
- Altering case-base indices based upon the utility of cases has also been investigated in the field of CBR (Veloso & Carbonell, 1993). Techniques that adjust the organization of case/index memory based upon the utility of cases work with the assumption that the past usage of a case is how the case should be used. Although we may hope that this is true, it is not true for all situations.
- The idea of attaching relevance/utility ratings to retrieved documents has been investigated in the In-

formation Retrieval (IR) field. Relevance feedback is a technique in which a query is made, the retrieved cases are evaluated, and the evaluations are used to make an improved retrieval (Harman, 1992). Relevance feedback does not take advantage of past relevance ratings. This may result in several iterations of the relevance feedback loop. Using our techniques the amount of work that an end-user has to go through will be reduced by taking advantage of similar past query sessions.

- It is important in general to differentiate CBR from IR. For example, IR technology was developed to work with document file numbering in the tens of thousands, but many CBR application only need work with a few hundred cases — and 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. Another important difference stems from the inclusion of adaptive mechanisms in the CBR paradigm. Thus for 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.

  Despite these differences, there may be some payoff in applying the technology developed here to the task of IR. In principle, one could substitute a information retrieval system for the base case retriever described above, thus potentially improving system performance by allowing the system to gain skill, after it has been deployed, for the questions that are most frequently asked. We have, in fact, run tests that have shown an effective technique for *query expansion*, an important IR research topic.

Ongoing Work and Future Directions

Several other experiments are either in progress or have been planned for the future:

- In addition to technique 1, we have also run tests on using the question base to *expand* the question. In other words, the system, given *user-query*, automatically generates an *expanded-query* based upon the question-base, with reduced work required of the end-user. Space constraints limit discussion of these tests, but the results were favorable.
- Another group of test that we have run have to do with using questions attached to cases (see figure 1 approach two). We are using this memory structure to recognize what answers (cases) are inappropriate for a given question. In other words, we use this structure to prune the retrievals. We have also found favorable results using this technique.
- We have also used the questions attached to cases structure to sort cases by inferred utility (sort by relevance). This has also produced interesting results.
- We plan on expanding the idea of a question beyond the relatively simple keyword format that we have presented in this paper. By doing this, we hope to be able to give even better structure to existing cases and their indexes by remembering questions.
- The same data may be used for two different applications, with each application emphasizing different vocabularies and clusterings for retrieval, even though the cases and their content are stable. Our techniques should be able to recognize the different usage of this data.
- We plan to use the question-base to model users and classes of users by the types of questions that they ask and/or how they ask those questions, and use this information for information filtering.
- We plan to apply all of the techniques to a multi-agent memory system.

References

Carpenter, T., and Alterman, R. 1994. A reading agent. In *AAAI-94*. to appear.

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

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

Harman, D. 1992. Relevance feedback revisited. In *Proceedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval.* New York, New York: ACM Press.

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 Ninth Annual Conference on AI.* AAAI.

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

McTear, M. F. 1993. User modelling for adaptive computer systems: a survey of recent developments. *Artificial Intelligence Review* 7:157–184.

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.*

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

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