Mining High-Quality Cases for Hypertext Prediction and Prefetching

Qiang Yang, Ian Tian-Yi Li, and Henry Haining Zhang

School of Computing Science
Simon Fraser University
Burnaby, BC, Canada V5A 1S6
(qyang,tlie,hzhangb)@cs.sfu.ca

Abstract. Case-based reasoning aims to use past experience to solve new problems. A strong requirement for its application is that extensive experience base exists that provides statistically significant justification for new applications. Such extensive experience base has been rare, limiting most CBR applications to be confined to small-scale problems involving single or few users, or even toy problems. In this work, we present an application of CBR in the domain of web document prediction and retrieval, whereby a server-side application can decide, with high accuracy and coverage, a user's next request for hypertext documents based on past requests. An application program can then use the prediction knowledge to prefetch or presend web objects to reduce latency and network load. Through this application, we demonstrate the feasibility of CBR application in the web-document retrieval context, exposing the vast possibility of using web-log files that contain document retrieval experiences from millions of users. In this framework, a CBR system is embedded within an overall web-server application. A novelty of the work is that data mining and case-based reasoning are combined in a seamless manner, allowing cases to be mined efficiently. In addition we developed techniques to allow different case bases to be combined in order to yield a overall case base with higher quality than each individual ones. We validate our work through experiments using realistic, large-scale web logs.

1 Introduction

Case-based reasoning (CBR) is a problem-solving framework that focuses on using past experiences to solve new problems [12]. Cognitive evidence points to the approach as a natural explanation for human problem solving. Much work has been done to exploit CBR as a general problem-solving framework, including retrieval [19], conversational CBR [1], case base maintenance [16], and various innovative applications [18].

A prerequisite for successful CBR application is that extensive experience base exists. Such experience base can record users' or systems' behavior in problem solving, in solving traces, consequences, feature selection, and relevance feedback. A alternative to such an experience base is the reliance on individual experts who can articulate their past experiences. However, much empirical work has pointed to the infeasibility of this latter approach, because experts are expensive, subjective and static. In contrast, having an accumulated, extensive experience base enables the design of data mining and knowledge discovery systems that can extract cases from a data set. This conversion, if done successfully and repeatedly, can result in a succinct, highly compact set of problem description and solution pairs that give rise to up-to-date case bases. Therefore, having the data itself is often a point of make-or-break for CBR applications.

Unfortunately, extensive experience base has been rare in practice. Much CBR research still relies on small scale, toy-like problems for empirical tests. This situation is dramatically alleviated, however, with the arrival of the World Wide Web (or the Web in short). Much recent work in Computer Science, and indeed AI itself, has been motivated by this sudden availability of data. On the Web, millions of users visit thousands of servers, leaving rich traces of document retrieval, problem solving and data access.

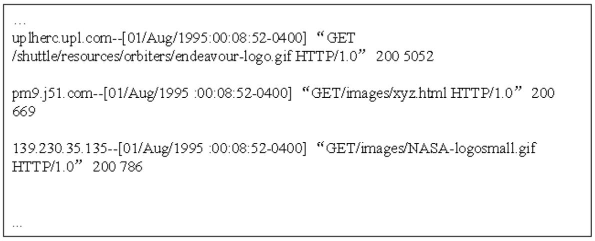
In this paper, we expose the hypertext retrieval on the Web as a potential experience base that is readily available to CBR researchers. Based on vast Web Server logs, we apply CBR in the domain of Web document prediction and retrieval, whereby a server-side application can decide, with high accuracy and coverage, a user's next request for hypertext documents based on past requests. An application program can then use the prediction knowledge to prefetch or presend web objects to reduce latency and network load. Likewise, with highly accurate prediction of users' next possible request, web servers can adapt their user interfaces according to user's interests. Through this application, we demonstrate the feasibility of CBR application in the web-document retrieval context, exposing the vast possibility of using web-log files that contain document retrieval experiences from millions of users. Such web-log files are extensive and up to date (for example, see http://www.web-caching.com for many realistic web logs)

In this framework, a CBR system is embedded within an overall web-server application. A novelty of the work is that data mining and case-based reasoning are combined in a seamless manner, allowing cases to be mined efficiently. In addition, different kinds of data-mined case knowledge from the same data source result in CBR systems with different qualities. Thus, when more than one case base exists, we developed techniques to allow different case bases to be combined in order to yield an integrated case base with higher quality than each individual ones. The integrated case base reasoning system introduces more flexibility and higher quality for CBR application. We validate our work through a series of experiments using realistic, large-scale web logs.

The organization of the paper is as follows. In Section 2 we discuss the web-document retrieval domain, web server logs and case-base representation for the application. In Section 3, we discuss case-knowledge discovery with data mining algorithms. In Section 4, we discuss how different case bases can be combined to give an integrated case base that provides higher quality solutions than individual case bases. In Section 5 we discuss an application of our prediction system in web-document prefetching applications. In Section 6 we conclude the article with a discussion of future work.

2 Web-Document Retrieval and Case Representation

The Web is a globally distributed, dynamic information repository that contains vast amount of digitized information. Every day, more and more information becomes available in multimedia forms. The fundamental framework in which such information is made available to the users is through the well-known client-server models. To retrieve information, a client issues a request that is answered by a server using the HTTP protocol. A by-product of such information exchange is that vast logs are recorded on the server side, indicating the source, destination, file type, time, and size of information transmission. Given a web-server browsing log *L*, it is possible break down a long access trace into sessions, where each session records a single source request in a consecutive sequence of accesses to the same server. These are called user sessions. These user sessions are indexed on the source of requests, and can be discovered by finding out the boundary between short and long requests. An example data log from a NASA web site is shown in Figure 1.



```
…
uplherc.upl.com--[01/Aug/1995:00:08:52-0400] “GET
/shuttle/resources/orbiters/endeavour-logo.gif HTTP/1.0” 200 5052

pm9.j51.com--[01/Aug/1995 :00:08:52-0400] “GET/images/xyz.html HTTP/1.0” 200
669

139.230.35.135--[01/Aug/1995 :00:08:52-0400] “GET/images/NASA-logosmall.gif
HTTP/1.0” 200 786


…
```

Fig. 1. An example web log file

The availability of the web server information allows machine-learning researchers to predict users' future requests and provide better information services according to such prediction. The availability of the web related information has inspired an increasing amount of work in user action prediction. Much work has been done in recommendation systems, which provide suggestions for user's future visits on the web based on machine learning or data mining algorithms. An example is the WebWatcher system [10], which makes recommendations on the future hyperlinks that the user might like to visit, based on a model obtained through reinforcement learning. Albrecht et al. [7] presented a Markov model based approach for prediction using a web-server log based on both time interval information and document sequence information. The predicted documents are then sent to a cache on the client side ahead of time. Based on Web server logs, [11, 15] provided detailed statistical analyses of web log data, pointing out the distribution of access patterns in web accesses and using them for prediction. [17] compared *n*-gram prediction models for different sized *n*, and discuss how the predictions might be used for prefetching for multimedia files, benefiting the network performance. However, none of the above-mentioned work considered integrating the prediction systems with caching and prefetching systems.

The web-document request problem can be stated as the following: given a training web log, construct a predictive model that suggests future web-document accesses based on past accesses. We consider the web-document request prediction problem as a CBR application. In a case base, the most basic representation is that of a case, consisting of a problem description and a solution component. For example, in a Cable-TV help-desk application, a problem description is “VCR not taping correct channels”, and a solution may be “Switch the TV/VCR toggle to VCR, switch to correct channel, and then press Record.” In a structured case, the problem description part of a case is structured into a set of discrete features, with feature-value pairs {<F_i, V_i>, i=1, 2 … n} representing the pattern to be matched against a problem.

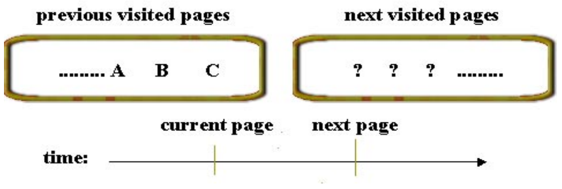


Fig. 2. Moving window algorithm

For the Web-document retrieval problem, our objective is to predict the next document that is going to be retrieved based on users’ previous requests. As shown in Figure 2, our goal is to predict what web pages will most likely be accessed next by a user based on all user’s previous accesses for pages A, B, and C. In the structured case representation, if we want the prediction to be a URL “D”, then we can make “D” to be the solution part of the case, and A, B and C the individual feature-values of the case, as shown in Table 1, Case (a). In this case representation, feature “First” means the first observed request on a page “A”. Likewise, “Second” and “Third” features record the second and third observations before making a prediction. In our representation, it is required that “ABC” is a sub-string ending at the cursor rather than a subsequence, where in the former no other symbols occurs in within the sub string while in the latter, there can be gaps in between. When this case is applied to the problem in Figure 2, the answer for the next visited pages within a given prediction window will be “D”, regardless of where “D” occurs in that window.

This case representation can be generalized so that the number of features can be anywhere from zero to *n*, an integer. If it is required that the observed pages occur next to each other with no “gaps” in between, then the problem-description part is also called an n-gram (3-gram in this example).

This case representation can be generalized such that the *Solution* part of a case includes more than one predicted page. The reason is that when observing a number of URL requests, it is possible to guess at several next pages. Furthermore, these predicted pages do not have to occur in the next instant in time. Instead, they can occur within a given time window from the current time. The time window W_2 can either measure the number of document requests in the near future in which we expect the predicted documents to occur, or a real time in the number of seconds. We call W_2 the *prediction window*, and the window W_1 in which the observation is made that matches that problem description part of the case the *observation window*. Applying this generalization, the case representation is shown in Table 1, under Case (b).

Table 1. A case representation for web-document prediction

Problem Description				Solution
Features	**First**	**Second**	**Third**	**Next Page(s)**
Case (a)	"A"	"B"	"C"	"D"
Case (b)	"A"	"B"	"C"	{"D", "E", "F"}

Given a test sequence of web objects with a certain time cursor representing the current time instant, a case can be used on the sequence for making a prediction if the problem description part of the case matches the observations in the observation window W_1 before the time instance. A prediction can then be made on the next occurrence of web objects within the prediction window W_2. In this generalization, the case *successfully applies* to an instance if, when the problem description matches the observed sequence in W_1, *one* of the web objects "D", "E" or "F" occurs in the prediction window W_2. In this work, we restrict the Solution part of cases to contain only one web document.

We will discuss how to obtain cases from a training web log in the next section. A user session can be considered as a sequence of web object accesses. A testing web log consists of many such sessions. Within each session, we can uncover many cases by moving a "cursor" through the sequence of pages requested; the cursor defines a window pair $<W_1, W_2>$. One consequence of this design is that there will be many different cases, each with different quality. We measure the quality of cases in the same way as that for association rules in data-mining literature [2]: we adopt the concepts of *support* and *confidence* for cases. *Support* for a case is defined as the percentage of strings in the web logs in which the case successfully applies. *Confidence* is defined as the conditional probability that the Solution part of the case falls in window W_2, given that the problem description part of the case fall match the sub-string at the time cursor.

To ensure the quality of a case base, we require that the support for all cases be above a certain user-specified threshold θ, and that the confidence for each case be above a threshold σ. However, finding proper thresholds is difficult in practice, an issue we will address in this work.

We now consider quality measure for a case base. A case base consists of a set of cases. For any given observation sequence within a window size W_1, the *application of the case base onto* W_1 takes all applicable cases in the case base -- cases whose Problem-Description part matches the pages in W_1 ending at the cursor -- and outputs a set of predicted pages based on the solutions of these cases. When there is more than one cases to apply, the decision of how to choose cases among the applicable cases to base predictions on is called the *application policy* of the case base reasoner. Some application policies may opt to output the union of all applicable cases while others may select the most confident case to output. Together, the case base composition and application policy determines the overall quality of a case-base reasoner: the *precision of a case base reasoner is* defined as the conditional probability that the case-base reasoner makes successful predictions for all window pairs $<W_1, W_2>$ in the log. The *coverage* of a case base reasoner, a second quality metric, is the percentage of the testing web log on which the case base reasoner can make predictions.

There are other types of case representations to consider. When the problem-description part of a case base consists of sets of pages rather than the last string of length n in observation window W_1, we have the set-representation of a case. In this representation, <{"A", "B"}, "D"> means that if "A" and "B" are observed in W_1, *regardless* of their relative locations in the window, then "D" is predicted in W_2. In our experiments we have found that this representation has much worse performance than the string-based representation. Likewise, we can include other features such as time interval information and page type information as problem features. For lack of space we do not consider these results and extensions here.

3 Mining Web Logs for Case Bases

Given a web server log file, we first preprocess it by removing all extremely long user sessions that are generated by search engines and crawlers. These do not represent typical user profiles. We also remove objects that are accessed less frequently than the minimum support θ. This is because for any given case in a case base, any individual web object appearing in the case must have support no less than that of the case. This is also the rule used by the well-known Apriori algorithm [2] in association rule mining. In fact, in our experience with very large web server logs, after applying this pre-processing rule with a minimum support of 2%, the total size of the web log is reduced by 50%!

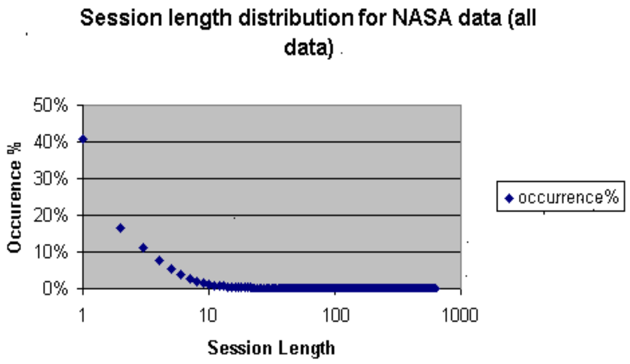


Fig. 3. Session length distribution of a web server log.

We next mine the individual cases, using a *moving-window algorithm*. Briefly, this algorithm scans through an entire user session. For any cursor location in the session, for every string *S* ending at the cursor in the observation window W_1 and a web object *P* in the prediction window W_2, there is a potential case <*S*, *P*>. For this case, a hash table entry is generated and count updated. When the scan is finished, these counts are used to compute support and confidence for the case. Only cases that satisfy the minimum support and confidence requirements are retained in the hash table *T*; *T* is the source knowledge that will be used to generate the final case bases.

Among all potential cases in table T, we also generate a special case *D* with empty problem description part and with maximal support. Such a default case is in fact the most frequent URL in the training web-server log. This case will be used to *catch* the situations when no other cases make a prediction. If we choose to use the default case, then the coverage of the resulting case base will be *100%*.

Through empirical study, we have found that with web-server logs, the number of sessions decreases exponentially with the length of sessions. Figure 3 shows this fact for a NASA web log. From this fact we can be assured that case-based mining using the moving-window algorithm operates in linear time for constant window sizes, in the size of the logs.

To evaluate the predictive power of the case-based reasoning system, we have utilized a realistic web data log, the NASA data (this and many other logs are available at http://www.web-caching.com/). The NASA data set contains one month worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida. The log was collected from 00:00:00 August 1, 1995 through 23:59:59 August 31, 1995. In this period there were totally 1,569,898 requests. After filtering, we reduced the request size down to 479,050 requests. In the web data log, timestamps have 1-second resolution. There are a total of 72,151 unique IP addresses requesting web pages, having a total of 119,838 sessions. A total of 2,926 unique pages are requested. In the following discussions, we will mainly use this log; the observations can be generalized to other logs that we have tested, including an EPA web server log and a university web-server log.

We then apply the moving-window algorithm to get different case bases. In all case bases, we mined the default case *D* to catch the situations where no other cases can be applied for prediction. We partitioned the case base into different size-*n* case bases for *n*=1, 2, 3, 4 and 5, where *n* is the number of pages in the problem description part of a case. We denote a case base consisting only of cases whose problem description parts have *n* consecutive features, with the exception of the default case, a length-*n* case base, or *CB(n)*. Note that our case-mining algorithm is significantly different from the sequential mining algorithm in [5], because our moving window algorithm captures high-frequency strings rather than item-sets in the transaction model.

4 From Case Bases to Case-Based Reasoning

Our task is to predict the future web document accesses based on past cases. Thus, having obtained the case bases is only the first step in constructing a case-based reasoner. A second step is to determine, for a given observation, which case among a set of applicable cases to apply in order to make the prediction. Therefore, the case-based reasoner is a pair, consisting of a case base and a case selection strategy.

Our method of constructing a case-based reasoner is analogous to methods for converting association rules to classifiers in data mining literature [13]. We experimented with two strategies. One strategy selects cases based on confidence, and the other on the length of matching features. To measure the system performance, we use the standard precision measurement, defined as the percentage of correct predictions over all predictions in the testing data set.

Figure 4 shows our precision-comparison on different case bases using the NASA data set. We set a minimum support to be ten occurrences and minimum confidence to be 10%. We set the window size for the prediction window to be one. We took the first 100,000 requests in the NASA log file as training data set and the next 25,000 requests as the testing data set.

In Figure 4, we plot the precision results of all five case bases CB(*i*) with *i* ranges from one to five. As can be seen, the precision of case bases first increases up to *i*=2 and then decreases after *i*=3. We attribute this observation to the fact that when *i*=1 and increases to 2, there are increasingly more high-confidence cases that cover the testing data. However, the situation is not sustained after *i*=3 because when *i* is large, the number of cases increases, causing the number of high-confidence cases for CB(*i*) to decrease rapidly. This has prompted us to study how to integrate the different case bases CB(*i*) in order to obtain a high-quality overall case base reasoning system. We discuss this novel extension in the next section.

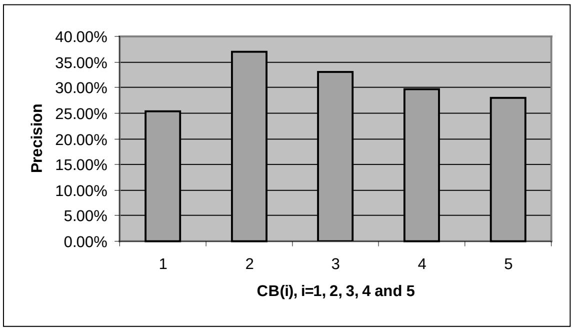


Fig. 4. Precision of case bases with different problem description lengths

5 Integrating Different Case Bases

We wish to combine the power of individual CBR systems in prediction. To do this, we adopt a integrated CBR approach in which we pool the result of each individual CBR for any given observations and then use the integrated case base to select or integrate among the different CBR solutions.

We first study a case-selection method by selecting the most confident case solution for prediction. For any given observation, all CBR systems CB(i), i=1, … 5, operate to make predictions. The case with the highest confidence is selected by the integrated CBR as the overall system prediction. This method is called Most-Confident CBR. A second integrated CBR method will bias toward CBR systems that make longer observations. For this strategy, a case chosen by a highest *i* for whom the solution from CB(*i*) is not the default case is always selected by the CBR. This method prefers longer-length n-grams and is called the longest-match CBR.

Figure 5 shows the result of the CBR in prediction as compared to individual case bases CB(*i*) for a given problem-description length *i*, where LongMatch is the longest-match CBR and MostConf is the most-confident CBR. As can be seen, the selection strategy that chooses the most confident cases for prediction gives the highest precision level compared to the other methods. The longest-match CBR performed a close second in comparison.

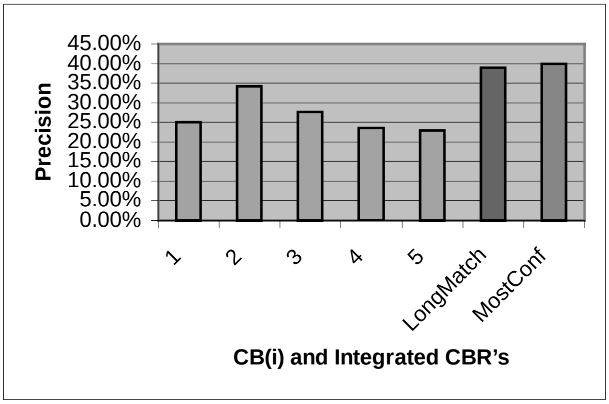


Fig. 5. Comparing CB(i) and Integrated CBR

Both the longest-match and most-confident case-selection strategies for CBR can suggest a single web object as the next URL to be accessed. They are also useful in user-interface agents that help recommend potential hyperlinks for a user. To apply CBR methods in prefetching web objects for enhancing web-access to the problem of caching and prefetching, however, we need an CBR method that recommends more than one web object in the prediction window. In the next section, we highlight this integrated CBR method in the application domain of web-document prefetching and caching.

6 Embedded CBR for Document Prefetching

One way to apply CBR is to embed a CBR application in an integrated system. In this work we are interested in using embedded CBR web-access prediction for Internet caching and prefetching. As the World Wide Web is growing at a very rapid rate, researchers have designed effective caching algorithms to reduce network traffic. The idea behind web caching and prefetching is to maintain a highly efficient but small set of retrieved results in a proxy-server cache, and to prefetch web objects into this cache. An important aspect of proxy-server caching and prefetching is to build an accurate prediction model for each server connected to the proxy server and cache and predict user's requests based on the model.

Lying in the heart of caching algorithms is the so-called ``page replacement policy", which specifies conditions under which a new page will replace an existing one. In proxy-caching, the state of the art techniques are the GD-size policy [8] which considers access costs and varying page sizes, and an enhancement of the GD-size algorithm known as GDSF [3] which incorporates the frequency information. Caching can be further enhanced by prefetching popular documents in order to improve system performance [9]. Our embedded technique will combine the predictions made by different models and give an overall "weight" for each candidate web object, and use the weights to update decisions.

Consider a frequency factor F_i which counts of number of references. With this new factor, the key value can be computed as $K_i = L + F_i * C_i / S_i$ In this formula, K_i is the priority of object i, C_i is the transmission cost of object i, S_i is the size of object i and L is an aging factor such that newer objects receive higher L values. Let $A[i]$ be a sequence of accesses such that $A[1]$ is the most recent access and $A[N]$ the N^{th} past access. Let K be the set of all cases that can be applied to the observations $A[1..N]$ where these cases are suggested by all CB(i) models. The confidence of each case from a case base CB(j) in K with a predicted document D_i is denoted as $P_{i,j}$. The weight W_i for D_i is then sum of all $P_{i,j}$ over all case bases CB(j). We can then update the caching algorithm by a predictive component – by including the predictive weight in the ranking functions for objects in the cache: $K_i = L + (F_i + W_i) * C_i / S_i$

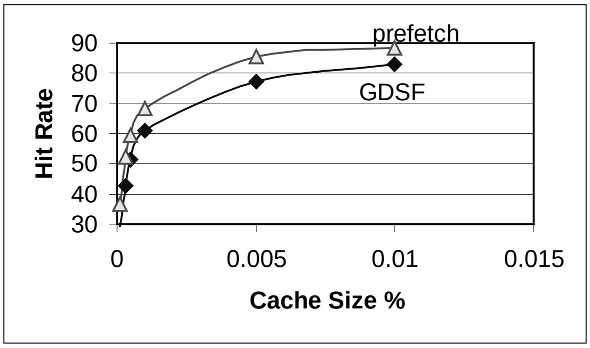


Fig. 6. Comparing prediction-based caching/prefetching, and GDSF caching policy for NASA data.

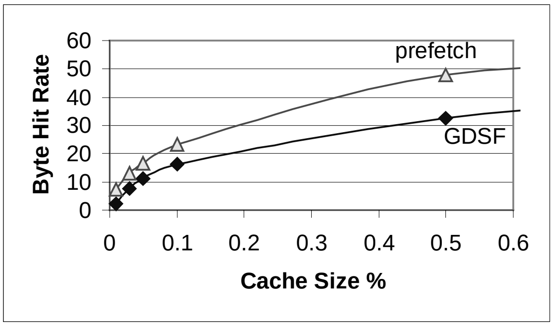


Fig. 7. Comparing prediction-based caching/prefetching, and GDSF caching policy for NASA data on byte hit rate

We follow the same idea with prefetching, by combining the predictions made by all case bases CB(i). The top-N high-probability objects are prefetched and put in a *prefetching buffer*. The buffer serves the same purpose as the cache; when a request arrives, we first check if the object is already in the cache. If not, then we check if the object is in the buffer. If so, then the object returned to the user as a response to the request, and moved into the cache for reuse.

We again used NASA data for experiments; our other experiments including the EPA web logs are not shown here due to space limit. In the experiments, we tested the system performance against two metrics used in network area: hit rate and byte hit rate. The hit rate records the percentage of user requests that can be answered by cache and prefetch buffer, and the byte hit rate measures the percent of bytes that are answered by cache and the prefetch buffer. The results are shown in Figure 6 and 7, where the horizontal axis (Cache Size %) is the size of the cache relative to the size of all objects in testing web logs. As can be seen, using prediction for caching and pre-fetching makes significant improvement to caching performance.

7 Conclusions

In this paper, we have shown how to data-mine web-server logs to get high quality cases. Our approach is to use a simple case representation and to extract only high-confident cases for prediction. Our result shows that using an integrated CBR system with carefully designed selection criteria can provide significant improvements. We also highlighted an application in network caching and prefetching using embedded CBR.

References

[1] D.W.Aha and L.A.Breslow. Refining conversational case libraries. In Proceedings of the Second International Conference on Case-based Reasoning (ICCBR-97), Providence, RI, July 1997.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (ACM SIGMOD '93), Washington, USA, May 1993.

[3] M. Arlitt, R. Friedrich L. Cherkasova, J. Dilley, and T. Jin. Evaluating content management techniques for web proxy caches. In *HP Technical report*, Palo Alto, Apr. 1999.

[4] D. Aha and H. Munoz-Avila. Applied Intelligence Journal, Special Issue on Interactive CBR. Kluwer 2001.

[5] R. Agrawal and R. Srikant. *Mining sequential patterns*. In Proc. of the Int'l Conf. on Data Engineering (ICDE), Taipei, Taiwan, March 1995.

[6] C. Aggarwal, J. L. Wolf, and P. S. Yu. Caching on the World Wide Web. In *IEEE Transactions on Knowledge and Data Engineering*, volume 11, pages 94--107, 1999.

[7] Albrecht, D. W., Zukerman, I., and Nicholson, A. E. 1999. Pre-sending documents on the WWW: A comparative study. *IJCAI99 – Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*.

[8] P. Cao and S. Irani. Cost-aware www proxy caching algorithms. In *USENIX Symposium on Internet Technologies and Systems*, Monterey, CA, Dec. 1997.

[9] E. Markatos and C. Chironaki. A Top Ten Approach for Prefetching the Web. In *Proceedings of the INET'98 Internet Global Summit*. July 1998

[10] Joachims, T., Freitag, D., and Mitchell, T. 1997 WebWatcher: A tour guild for the World Wide Web. *IJCAI 97 – Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 770-775.

[11] T. M. Kroeger and D. D. E. Long. Predicting future file-system actions from prior events. In *USENIX 96*, San Diego, Calif., Jan. 1996.

[12] D. Leake Case-Based Reasoning: Experiences, Lessons, and Future Directions. Menlo Park, CA, AAAI Press. 1996.

[13] B. Liu, W. Hsu, and Y. Ma: "Integrating Classification and Association Rule Mining", Proc. Fourth Int'l Conf. on Knowledge Discovery and Data Mining (KDD), pp. 80-86, AAAI Press, Menlo Park, Calif., 1998.

[14] K. Chinen and S. Yamaguchi. An Interactive Prefetching Proxy Server for Improvement of WWW Latency**.** In *Proceedings of the Seventh Annual Conference of the Internet Society (INEt'97)*, Kuala Lumpur, June 1997.

[15] Pitkow J. and Pirolli P. Mining longest repeating subsequences to predict www surfing. In *Proceedings of the 1999 USENIX Annual Technical Conference*, 1999.

[16] Smyth, B. and Keane, M.T. 1995. Remembering to Forget: A Competence-Preserving Case Deletion Policy for Case-based Reasoning systems. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95, pp. 377-382.

[17] Z. Su, Q. Yang, and H. Zhang. A prediction system for multimedia pre-fetching on the Internet. In *Proceedings of the ACM Multimedia Conference 2000*. ACM, October 2000.

[18] Watson (1997). Applying Case-Based Reasoning: techniques for enterprise systems. Morgan Kaufmann Publishers Inc., San Francisco, USA.

[19] D. Wettscherck, and D.W. Aha 1995. Weighting Features. In Proceedings of the 1st International Conference of Case-Base Reasoning, ICCBR-95, pp. 347-358.