Web-Log Mining for Predictive Web Caching

Qiang Yang and Haining Henry Zhang

Abstract—Caching is a well-known strategy for improving the performance of Web-based systems. The heart of a caching system is its page replacement policy, which selects the pages to be replaced in a cache when a request arrives. In this paper, we present a Web-log mining method for caching Web objects and use this algorithm to enhance the performance of Web caching systems. In our approach, we develop an n-gram-based prediction algorithm that can predict future Web requests. The prediction model is then used to extend the well-known GDSF caching policy. We empirically show that the system performance is improved using the predictive-caching approach.

Index Terms—Web log mining, Web caching, prediction models, classification.

1 Introduction

Web caching aims to improve the performance of Web-based systems by keeping and reusing Web objects that are likely to be used often in the near future. It has proven to be an effective technique in reducing network traffic, decreasing the access latency and lowering the server load [9], [10], [6], [16], [13]. Previous research on Web caching has focused on the use of historic information about Web objects to aid the cache replacement policies. These policies take into account not only information about the Web-document access frequency, but also document sizes and access costs. This past information is used to generate estimates on how often and how expensive it is for the objects saved in the cache to be accessed again in the near future.

As pointed out early in caching research [7], the power of caching is in accurately predicting the usage of objects in the near future. In previous work, estimates for future accesses were mostly built on statistical measures such as access frequency and object size and cost. Such measures cannot be used to accurately predict for objects that are likely to be popular but have not yet been popular at any given instant in time. For example, as Web users traverse Web space, there are documents that will become popular soon due to Web document topology, although these documents are not yet accessed often in the current time instant. This gap can be filled with data mining by looking at the association relationship between objects that in Web logs.

In this paper, we present a novel approach to integrating data mining with traditional Web caching to provide more powerful caching policies. In our approach, Web logs are used to train sequential association rules to predict Web users' browsing behavior. These rules are integrated into a cache-replacement policy to obtain a more forward-looking ranking for the cached Web documents and objects. We empirically demonstrate the utility of the method through increased hit rates and byte-hit rates.

To apply data mining, a segment of Web log data is used to train a predictive model. This model is then integrated with an existing caching algorithm for predictive caching. In this work, our technique is primarily aimed at dealing with stationary data, where the Web objects correspond to static Web pages. For nonstationary data such as database-driven Web sites, query-level prediction algorithms can be applied [14].

Our research is novel from both data mining and network systems aspects. Although several approaches have been proposed in the past for Web-log mining [17], [24], few researchers in data mining have integrated prediction models with state-of-the-art caching algorithms. Most previous work on integrating caching and Web-log mining, including our own work [23], has focused on intelligently prefetching Web documents. It remains to show how Web-log mining can be used to extend the best Web caching algorithms.

2 Predictive Caching

2.1 Cache Replacement Policies

A cache stores a finite number of objects that are likely to be reused in the near future. When the cache is full, it is an issue to select an object to remove in order to make space for the new object. The algorithm in selecting the object to be removed is called the cache replacement algorithm. Essentially, the cache replacement algorithm is a way to use past information to predict the future. The more accurate the prediction into the future, the better performance the caching system will have.

We will focus on the GDSF caching policy in this paper [10]. This policy is an improvement on the well-known GD-Size [9] algorithm by incorporating a frequency information about Web objects. The basic idea is to rank objects in the cache based on their likelihood of being reused in the near future by means of a key value. The ranking function for object p is computed as:

$$K(p) = L + F(p) \times C(p)/S(p), \tag{1}$$

where $F(p)$ is the access count of document p, $C(p)$ is the estimated cost of accessing the object p, and $S(p)$ is the document size. The factor L is an aging factor. The frequency count is incremented by one whenever the object is accessed again from the cache. When replacing an object in a cache, the object with the lowest key $K()$ value is removed.

The above Web caching-replacement policies can be considered as making zero-order prediction for future-access frequencies because they simply use the total access count as an estimate for the future. Sequential association-rules allow for further first-order predictions by considering what these past accessed documents may entail. For example, suppose that a document A has been accessed 100 times in the past five minutes. From the zeroth order prediction, we know that A might be accessed often in the next five minutes as well. Suppose that a strong sequential association rule exists: $A \rightarrow B$. Then, we can conclude that, according to this rule, B will also be accessed frequently in the near future, even though B might not have been accessed frequently enough in the past. Thus, by looking into the future through sequential association rules, we may *anticipate* what will come as a result of what has happened. In this section, we will enhance the GDSF caching policy by considering both the zeroth and first order predictions.

We now consider how to anticipate future Web-object accesses from the access history. Let O_j be a Web object under consideration; O_j is not in the cache. Let S_i be a user session for accessing a Web object on a Web server. Let $P_{i,j}$ be the probability predicted by a session S_i for object O_j. If $P_{i,j} = 0$, it indicates that object O_j is not predicted by session S_i. Then, $W_j = \sum_i P_{i,j}$ is the estimated access frequencies from the current sessions executed on the Web server.

With this estimate, we can now extend (1) to incorporate the predicted accesses for object O_p:

$$K(p) = L + (W(p) + F(p)) \times C(p)/S(p). \tag{2}$$

- *Q. Yang is with the Department of Computing Science, Hong Kong University of Science and Technology, Clearwater Bay, Kowloon Hong Kong. E-mail: qyang@cs.ust.hk.*
- *H.H. Zhang is with IBM E-Business Innovation Center, Vancouver, Burnaby BC Canada V5G 4X3. E-mail: haizhang@ca.ibm.com.*

Manuscript received 15 Oct. 2001; revised 3 July 2002; accepted 4 Oct. 2002. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 115198.

In (2), we add $W(p)$ and $F(p)$ together, which implies that the key value of a page p is determined not only by its past occurrence frequency, but also affected by its future frequency. The rationale behind our extension is explained in the beginning of this section: By considering both the absolute count of Web pages and the predicted accesses in the future according to association-rules, we can enhance the priority of those cached objects that may not have been accessed frequently enough, but will be in the near future according to the association rules. The addition of $W(p)$ and $F(p)$ in (2) reflects an expansion of the access frequencies by both the zeroth and the first-order estimates. By adding these two measures together, we integrate two information sources for frequency estimates by adding an optional $W(p)$ weight to a base count for each cached object. This will promote objects that are potentially popular objects in the near future even though they are not yet popular in the past.

Our extension can be seen as an attempt to obtain more accurate estimates on future accesses [7]. Our work is related to many past efforts in building predictive models for Web prefetching [15], [17]. In the Web caching area, the work of Bonchi et al. [8] extended the LRU policy of Web and proxy servers using frequent patterns and decision trees obtained from Web log data. This idea is similar to ours presented here, a difference being that the Web caching model in [8] was designed to extend the LRU policy rather than the more powerful GDSF policy.

2.2 Estimating Future Frequencies through Association-Rule Mining

How do we obtain the conditional probability information P_{ij} for an object O_j? We can obtain this by discovering a set of sequential association rules from a previously obtained Web log. In this section, we discuss how to obtain $W(p)$ for a page p.

In many data mining works in the past, Web access logs are mined to infer the association relationship between Web document and object requests in order to build a prediction model [21], [24]. Related to this effort is the general area of Web log mining and Web caching, which has seen increased activity in recent years [1], [12], [19], [18], [20], [22], [21]. In a Web log, we define a user session to be a sequence of Web requests from a user. In a Web log, requests emanating from the same IP address are grouped into a session. If the time interval between two consecutive requests exceeds a threshold T, we consider them to belong to two different sessions. For example, for the NASA Web log, we set T to be at two hours. By experimenting with different values of T, we did not notice much difference in the prediction system's performance.

Our data mining algorithm is a special case of an association-rule mining algorithm. It constructs an n-gram model by counting the occurrence frequency in order to build the conditional probability estimates P_{ij}. Within each user session, every substring of length n is regarded as an n-gram, which serves as a left-hand-side (LHS) of an association rule. The right-hand-side (RHS) of a rule predicts what documents or objects are most likely to occur after observing the LHS of the rule. We call these rules n-gram rules. In Web logs, we have found that the n-gram rules are more accurate than general sequential association rules. This is because the most recently accessed Web objects are more important indicators for what is to come.

Our n-gram rule-mining algorithm scans through all substrings of a length up to n in every user session, accumulating the occurrence counts of distinct substrings. The substrings with counts lower than a lower bound support value are pruned. The resulting count table is then used to build our prediction model. We call these association rules the n-gram rules since the LHS of the rules are strings instead of arbitrary item sets.

The general association-rule mining literature studied efficient ways to obtain general association rules through variants of a priori and other methods [3], [4], [2], [5], [11]. Our n-grams are a special class of association rules in which the LHSs are strings rather than subsets or subsequences; here, a string is an *adjacent* sequence of items accessed just before the prediction. This restriction that the LHS be the latest substrings enforces the domain knowledge in Web access prediction that the most important predictors come from the most recent events in accessing history. An important side effect of considering only n-grams as LHS is that the time complexity of the n-gram-mining algorithm becomes linear in the size of the Web logs. This reduction in complexity is important for predictive Web caching because Web logs are typically very large in size.

For every n-gram rule $LHS_l \rightarrow O_j$, we can obtain a *confidence* measure for the rule as

$$conf_{lj} = \frac{|LHS_l + \{O_j\}|}{|LHS_l|}.$$

Then, for each user session S_i, the weighted sum of all $conf_{lj}$, where LHS_l is contained in S_i, gives the conditional probability estimate P_{ij}.

3 Experiments

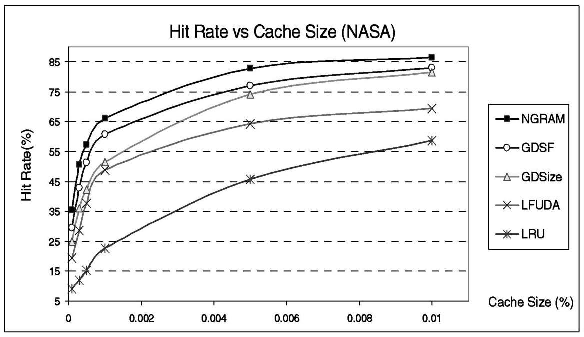
3.1 Data and Evaluation Criteria

Three Web logs are used in our study. One is from a US Environmental Protection Agency (EPA)s Web server located at Research Triangle Park, North Carolina. This Web log contains all the HTTP requests for a full day, from 23:53:25 on Tuesday, 29 August 1995, to 23:53:07 on Wednesday, 30 August 1995. In total, there are 43,202 requests. The second one is from a NASA Kennedy Space Centers Web server in Florida. This Web log contains all the HTTP requests collected from 00:00:00, 1 July 1995, to 23:59:59, 17 July 1995. In this period, there were 1,140,284 requests. The third log is from the School of Computing Science at Simon Fraser University. It was collected during the period from 00:00:00, 1 February 2001, to 23:59:59, 9 February 2001. It contained a total of 468,732 requests. The timestamps of these logs have a 1-second precision. In our experiments, we use half of each Web log to train our model and use the other half to do testing.

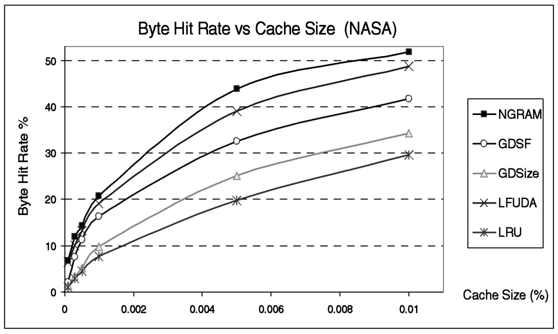
In this work, we assume that documents requested by users must either be retrieved entirely from the original Web servers or from a Web cache before in order to be sent to the users. In a caching system, the requests issued by users can be viewed as a sequence of references to Web documents. Web documents often have different sizes, while a cache only has a finite size. If a requested document is in the cache, the request can be satisfied immediately, which is called a *hit*; otherwise, the document has to be fetched from the original server, which is termed a *miss*. The hit rate is the ratio between the number of requests that hit in the proxy cache and the total number of requests. The byte-hit rate is an even more realistic measure of performance for Web caching; it is the ratio between the number of bytes that hit in the proxy cache and the total number of bytes requested.

Experiments are conducted with different cache sizes. The size of the cache is expressed in terms of the percentage of the total number of bytes of all objects in a Web log. In the NASA data, the cache size ranges from 0.0001 to 0.01 percent, between 24 KB to 2.4 MB. In the EPA data, the cache size ranges from 0.01 to 1.0 percent, between 31 KB to 3.1 MB. In SFU's log, the cache size varies from 0.01 to 1.0 percent, which is approximately 580 KB to 58 MB.

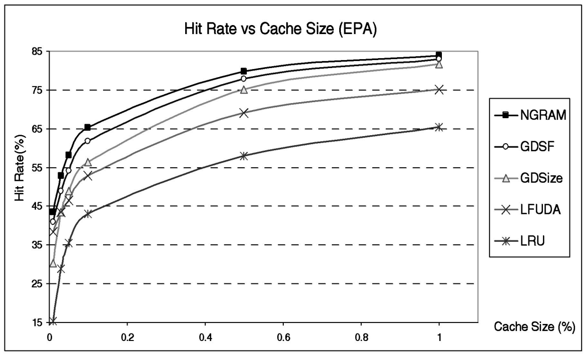
For comparison purposes, we also run simulations using some other major cache replacement algorithms. These algorithms include GDSize [9], GDSF [10], LFUDA [6], and LRU. The first three are Greedy-Dual-based algorithms and have been proven



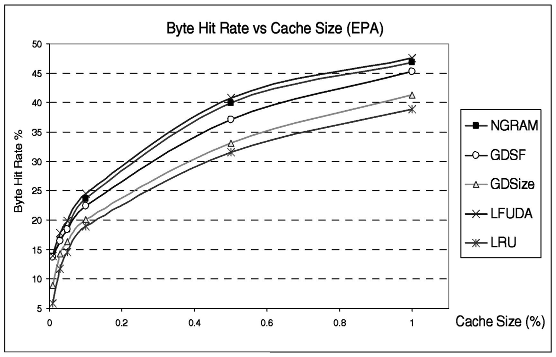
(a)



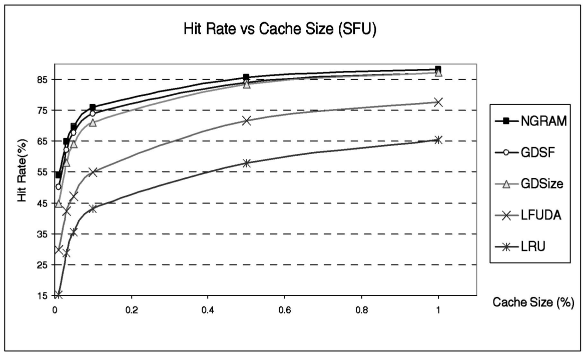
(b)



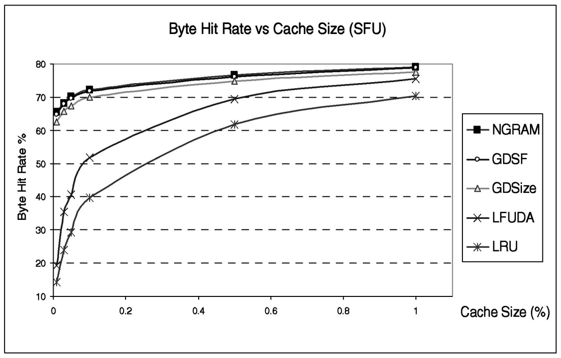
(c)



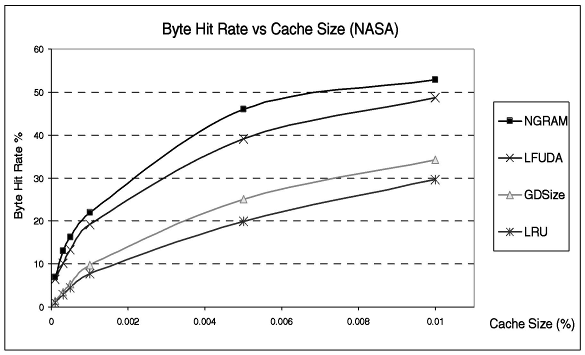
(d)



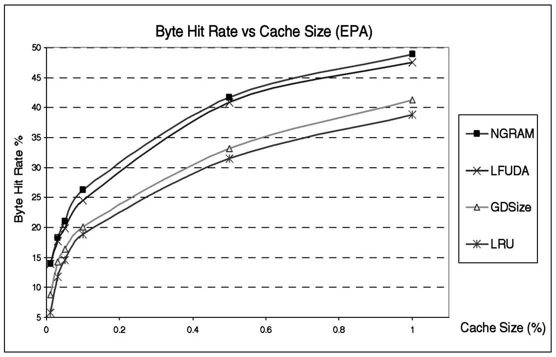
(e)



(f)



(g)



(h)

Fig. 1. Hit rate and byte hit rate comparison across data sets.

among the best page replacement strategies. LRU is the most basic algorithm in the caching domain and it is involved as a baseline of performance measurement. Two series of experiments have been conducted to evaluate the hit rates and byte hit rates.

To achieve higher hit rates, we set $C(p) = 1$ in N-gram, GDSF, and GDSize. The caching algorithm is hereafter referred to as N-gram(1). When $C(p) = 1$, the costs to retrieve all pages are the same, i.e., the connection time dominates the network latency.

Figs. 1a, 1b, 1c, 1d, 1e, and 1f show the hit rate and byte-hit rate on three data sets. Overall, in terms of hit rate, the N-gram(1) algorithm outperforms the other algorithms using all of the selected cache sizes. GDSF(1) is the second best and GDSize(1) is the third. LRU has the lowest hit rates, which suggests this widely used algorithm is not optimal in terms of hit rates.

In terms of byte hit rate, N-gram(1) does not outperform LFUDA in all data sets. Fig. 1d shows that N-gram(1) obtains a lower byte-hit rate than LFUDA. This is natural because N-gram(1) considers the benefit of a cache hit as one, regardless of the size of a page. It achieves higher byte-hit rate at the cost of lower hit rates.

To achieve better byte-hit rate, let $C(p)$, the cost value, be the size of the page. This is the situation of a network where data transmission time contributes most to the network latency. We denote a caching algorithm in this situation as N-gram(Size). The LFUDA actually becomes the GDSF(Size) algorithm. Figs. 1g and 1h plot the byte-hit rates on the two data sets. They show that, if we set $C(p)$ to the size of the document, N-gram(Size) achieves a higher byte hit rate than LFUDA.

To sum up, we empirically showed that the system performance is improved by our predictive caching algorithm. N-gram(1) achieves a better hit rate, while N-gram(Size) achieves a higher byte-hit rate. Our results also suggest that training on larger Web logs results in significant performance improvement for Web caching.

4 Discussions and Conclusions

In this paper, we integrated a Web caching algorithm with a Web log mining algorithm. A novelty in our work is to combine a prediction model learned from Web log data with the state-of-the-art GDSF cache-replacement algorithm. Our experiments demonstrated that the association-rule based predictive algorithm improves both the hit rate and the byte-hit rate in several experiments.

In the future, we plan to study how to deal with nonstationary data, such as caching, for dynamically changing database-driven Web sites. For such Web sites, query-level prediction algorithms need to be studied.

Acknowledgments

The authors thank Tianyi Li and Zhen Zhang for their helpful discussions in this work.

References

[1] C. Aggarwal, J.L. Wolf, and P.S. Yu, "Caching on the World Wide Web," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, pp. 94-107, 1999.

[2] C.C. Aggarwal and P.S. Yu, "A New Approach to Online Generation of Association Rules," *IEEE Trans. Knowledge and Data Eng.*, vol. 13, pp. 527-540, 2001.

[3] R. Agrawal, T. Imielinski, and R. Srikant, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD Conf. Management of Data*, pp. 207-216, May 1993.

[4] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. Very Large Data Base Conf.*, pp. 487-499, Sept. 1994.

[5] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. 11th Int'l Conf. Data Eng.*, P.S. Yu and A.S.P. Chen, eds., pp. 3-14, 1995.

[6] M. Arlitt, R. Friedrich, L. Cherkasova, J. Dilley, and T. Jin, "Evaluating Content Management Techniques for Web Proxy Caches," *HP Technical Report*, Apr. 1999.

[7] L.A. Belady, "A Study of Replacement Algorithms for Virtual Storage Computers," *IBM Systems J.*, vol. 5, no. 2, pp. 78-101, 1966.

[8] F. Bonchi, F. Giannotti, C. Gozzi, G. Manco, M. Nanni, D. Pedreschi, C. Renso, and S. Ruggieri, "Web Log Data Warehousing and Mining for Intelligent Web Caching," *Data & Knowledge Eng.*, vol. 39, no. 2, pp. 165-189, 2001.

[9] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," *Proc. USENIX Symp. Internet Technology and Systems*, pp. 193-206, Dec. 1997.

[10] L. Cherkasova, "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy," *HP Technical Report*, Nov. 1998.

[11] J. Han, J. Pei, and Y. Yin, " Mining Frequent Patterns without Candidate Generation," *Proc. 2000 ACM SIGMOD Int'l Conf. Management of Data*, W. Chen, J. Naughton, and P.A. Bernstein, eds., pp. 1-12, 2000.

[12] T. Joachims, T Freitag, and D. Mitchell, "Webwatcher: A Tour Guide for the World Wide Web," *Proc. 15th Int'l Conf. Artificial Intelligence (IJCAI '97)*, pp. 770-777, 1997.

[13] P. Krishnan and J.S. Vitter, "Optimal Prediction for Prefetching in the Worst Case," *Proc. SODA: ACM-SIAM Symp. Discrete Algorithms (A Conf. Theoretical and Experimental Analysis of Discrete Algorithms)*, 1994.

[14] Q. Luo and J.F. Naughton, "Form-Based Proxy Caching for Database-Backed Web Sites," *The VLDB J.*, pp. 191-200, 2001.

[15] E.P. Markatos and C.E. Chronaki, "A Top 10 Approach for Prefetching the Web," *Proc. INET '98: Internet Global Summit*, July 1998.

[16] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson, *Adaptive Web Caching: Towards a New Caching Architecture*, Nov. 1998.

[17] V. Padmanabhan and J. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *Proc. ACM SIGComm*, pp. 22-36, 1996.

[18] M.J. Pazzani, J. Muramatsu, and D. Billsus, "Syskill & Webert: Identifying Interesting Web Sites," *Proc. Am. Assoc. Artificial Intelligence*, pp. 54-61, 1996.

[19] M. Perkowitz and O. Etzioni, "Adaptive Web Sites: Concept and Case Study," *Artificial Intelligence*, vol. 118, nos. 1-2, pp. 245-275, 2001.

[20] J. Pitkow and P. Pirolli, "Mining Longest Repeating Subsequences to Predict World Wide Web Surfing," *Proc. Second USENIX Symp. Internet Technologies and Systems*, pp. 139-150, 1999.

[21] S. Schechter, M. Krishnan, and M.D. Smith, "Using Path Profiles to Predict http Requests," *Proc. Seventh Int'l World Wide Web Conf.*, pp. 457-467, Apr. 1998.

[22] J. Srivastava, R. Cooley, M. Deshpande, and P. Tan, "Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data," *ACM SIGKDD Explorations*, vol. 1, no. 2, pp. 12-13, 2000.

[23] Q. Yang, H.H. Zhang, and I.T.Y. Li, "Mining Web Logs for Prediction Models in WWW Caching and Prefetching," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 473-478, Aug. 2001.

[24] I. Zukerman, D.W. Albrecht, and A.E. Nicholson, "Predicting Users' Request on the WWW," *Proc. (UM '99) Seventh Int'l Conf. User Modeling*, pp. 275-284, June 1999.