Decision Method of Training Data for Web Prefetching

Zhijie Ban Feilong Bao
College of Computer Science, Inner Mongolia University, China
e-mail: banzhijie@imu.edu.cn csfeilong@imu.edu.cn

***Abstract*—**Web prefetching is an effective technique to reduce user-perceived latency. Most studies mainly focus on prediction algorithm but they ignore selection strategy of training data which is an important part of web prefetching. This paper presents a decision method based on monitoring prediction precision. It divides user access sequence into different data blocks and the changing features of prediction precision among data blocks indicate whether some training data is outdated. According to the varying trend of prediction precision, some user access requests are inserted into or deleted from training data. We use two real web logs to examine this proposed method and the simulation shows that our method can significantly improve prefetching performance.

Keywords-web prefetching; sliding window; training data

I. INTRODUCTION

Web prefetching technique is one of the primary solutions used to reduce user-perceived latency. The spatial locality shown by user accesses makes it possible to predict future accesses from the previous ones [1][2]. Web prefetching system makes use of these predictions to preprocess user requests before they are actually demanded. Part of the network latency can be hidden if prefetching system perfetchs those pages which are very likely to be demanded in subsequent requests.

To predict the user's next request, a number of prediction approaches were presented, which had achieved an acceptable performance [3]. In the web prefetching technique, part of user access sequence is used as training samples to construct prediction model before user requests are predicted. By training with samples, prediction model includes user access patterns and some important information, which provides a foundation for predicting the user's next request page. Thus training data is very important to correctly predict user requests. However, few studies focus on decision method of training data. Many researchers random select one part of user access sequence as training samples and another part is used as test samples. Nanopoulos et al. used 75 percent of a week Clarknet log available from the site http://ita.ee.lbl.gov/html/traces.html as training data and 25 percent as test data [4]. Sarukkai presented that 40000 samples of the EPA-1995 server log were used as training samples and the remaining as test samples [5]. Shi and Gu used 80 percent of one month's NASA -1995 log to train prediction model and 20 percent as test data [6]. Only the papers slightly talked about training data problem [7][8][9]. In order to verify client-based web prefetching experiments, Lan and Ng [7] obtained a proxy trace whose web pages were requested by different users. Then, the log was partitioned into a number of the single user's access sequences. Finally, they randomly selected continuous 14 days web accesses from every user's log to train prediction model and the fifteenth day's user requests were predicted according to the constructed prediction model. During the experimental period, they found that the web accesses of 14 days were enough for describing user access patterns. So two-week log was selected for every user as training samples. In order to examine the web prefetching performance, Davison shown the prediction model was not trained before predicting the next user request [8]. He considered that this method was better near to the real network environment. But the prediction precision is very low if the prediction model is seldom trained in the real prefetching system. In the low precision's condition, network resources such as network bandwidth are wasted if predicted pages are prefetched. Domènech and Sahuquillo studied how training data to influence prefetching performance with two different prediction models and 4 different logs [9]. They compared prefetching performance using the old and current web log, but they did not study how to decide training data.

This paper presents one decision approach of training data based on our previous work [10]. It partitions the user access sequence into different continous data blocks according to the access time of every request. Based on the changing trend of prediction precision among different data blocks, our method decides whether web accesses are deleted from or added into training data. As a result, prediction model space is decreased and prefetching performance is improved.

The rest of the paper is organized as follows. Section 2 presents the related background. Section 3 describes the decision strategy of training data and its algorithm. Section 4 gives the details of our experiments and testing results. Section 5 is the summary and conclusions.

II. RELATED WORK

There is an important set of research works concentrating on prefetching techniques to reduce the user perceived latency. Various prediction models have been proposed to model and predict a user's browsing behavior on the web. Markatos and Chronaki proposed a Top-10 approach which combined the popular documents of the servers with client

Supported by the program of higher-level talents of Inner Mongolia University (Z20090137) and national innovation experiment program for university student (101012623)

access characteristics [11]. Web servers regularly pushed the most popular documents to web proxies, and then proxies pushed those documents to the active clients. But the approach only made use of page access frequency. In order to solve the problem, the study in [12] presented a prefetching algorithm based on prefetching in the context of file systems [13]. The server built a dependency graph (DG) where an arc from node A to B meant that B was likely to be accessed within a short interval after an access to A. Each arc was labeled the conditional probability. But the DG model was not very accurate in predicting the user browsing behavior because it only considered first-Order dependency [4] and did not look far into the past to correctly discriminate the different observed patterns. Thus, the studies in [14][15] described the use of a kth-Order Markov model for user request patterns. In a kth-Order Markov model, each state represented the sequence of k previous requests, and had conditional probabilities to each of the next possible states. However, it is likely that there will be instances in which the current context is not found in a kth-Order Markov model if the context is shorter than the order of the model. Therefore, the PPM (Prediction by Partial Matching) model [16][17][18] which originates in the data compression community, overcomes the problem. It trained varying order Markov models and used all of them during the prediction phase. Fan et al. studied how user access latency could be reduced for low-bandwidth users by using compression and PPM prediction model between clients and proxies [17]. Bouras et al. studied prefetching's potential in the Wide Area by employing two prediction models [19]. These PPM models do not implement the online update and timely reflect the changing user request patterns. An online PPM with dynamic updating is presented [20]. But most of them arbitrarily take a part of web log as training set and another part as prediction set. Only the studies in [7][8][9] slightly mention the training data problem. Lan and Ng proposed a client-based web prefetching management system, which was based on the caching schema of Netscape Navigator [7]. In the experiments, users submitted their web access requests throw their own machines to the proxy server, and their prefetching system obtained each log file that contained the log of each individual user's web access requests within a 2-week consecutive time period. A 2-week time period was chosen because it was sufficient to show the web access pattern of each individual user based on their observations during the experimental period. Thus, they randomly chosen a 2-week consecutive time period for each user to represent the access history of the user as long as the user accessed the web on the fifteenth day, the day after the 2-week consecutive time period. But Domènech and Sahuquillo considered that the length of training period may impact on prefetching performance, either improving or degrading it [9]. In addition, this length may involve a high amount of information and therefore important computer resources are consumed. Thus, they analyzed that how the training affects the prediction performance using current and old web traces. Their experimental results showed that while in old traces the training, in general, improves performance, when using recent traces this training may degrade performance because users' access pattern had changed. Davison evaluated prediction algorithms without previous training [8]. This procedure was argued to be more realistic than freezing the learning after a training period [9]. But all of them do not study how to dynamically determine training data according to different user access behaviors.

III. DECISION APPROACH

In this section, we specify concept definition, and give decision strategy and decision algorithm.

A. Concept definition

We firstly give some related concepts before decision method is introduced.

Definition 1 *User access sequence* is an orderly sequence composed of a series of two-tuples such as $<T_1, I_1>, <T_2, I_2>, <T_3, I_3>, ...$, where $T_i(i=1,2,3\ldots)$ is the access time, I_i denotes the entity, T_j is larger than T_i if j is larger than i.

The time of two-tuples has strong restriction and denotes the absolute time of user request. The entity of two-tuples represents every request's attributes. Suppose the entity I includes k attributes $\{X_0, X_1, \ldots, X_{k-1}\}$, where the value range of the attribute X_i is $d(X_i)$, the attribute space of the entity I is $\{d(X_0), d(X_1), \ldots, d(X_{k-1})\}$. In the server's log, every $<T,I>$ corresponds to one user request record, where T represents the user absolute request time and I mainly includes IP address, the request page's URL and so on.

Definition 2 *Sliding window* is defined a user access sequence including h user requests, where h is the number of user requests in the sliding window.

Figure 1 gives a sliding window's sketch map with h user requests. In order to describe simple, the two-tuples of user request sequence is denoted as a_j, where j is the relative access time. In the sliding window, a_i is the eldest user request, a_{i+h-1} is the newest one and a_{i+h} is the user request which will slide into the sliding window.

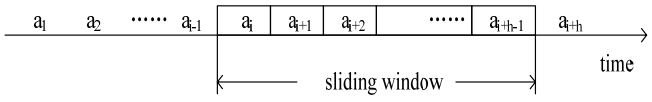


Figure 1. Sliding window with h user requests

Definition 3 *Data block* refers to one user request sequence and all requests are ranked according to the access time from the eldest one to the newest one. Partition of data blocks may take time segment or request number as dimension. We choose the former because there may exist a large number of requests in a short time. When the emergent event happens, data block using fixed request number as dimension can not represent user access behaviors while

data block with the time dimension better reflects user access features.

Definition 4 *Window* includes one user access sequence during a period of time and is partitioned into *n* data blocks according to the same time dimension. The label of data blocks in one window varies from *0* to *n-1*. Figure 2 depicts a window's sketch map with *n* data blocks. In the window, the data block labeled *0* is the eldest and one labeled *n-1* is the newest. The user request number of every data block may be different while the time periods are the same.

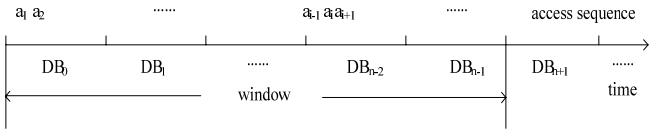


Figure 2. Window with n data blocks

B. Decision strategy

The right training data is important for constructing prediction model and predicting the user next request page. If training data includes too little access requests, the relevant user requests may be forgotten so that some correctly access features may be deleted. If training data includes excessive user accesses, the prediction model do not also represent the browsing characteristics of current users because it may include some outdated user access patterns and browsing infromation.

We use one sliding window *SW* and two windows (W_S and W_L) to dynamically adjust training data to reduce the prediction model's space and improve prefetching performance. The sliding window *SW* includes the total user access sequence in the prediction model. W_S is called the small window which includes some continuous data blocks. W_L is called the large window which includes W_S and other some continuous data blocks. W_S is a part of the large window. The sliding window *SW* includes W_L and the newest user access requests which can not compose one data block. In order to decide training data, the large window size is adjusted according to prediction precision's changing features among data blocks of the small window so that the sizes of the sliding window and W_S change.

Figure 3 gives the relation between the small window and the large window. In Figure 3, the total user access sequence is regarded as a series of user requests. It is denoted a_1, a_2, a_3, …, where a_i stand for usr request and *i* is the relative access time of the *ith* user request. The user access sequence is partitioned into some data blocks according to the same time, where DB_0 is the eldest data block and DB_n is the newest one in the large window. The large window W_L includes *n+1* continuous data blocks and the small window W_S includes *m* continuous data blocks, where *m* is smaller than *n*, and the *m* data blocks are the newest in the large window.

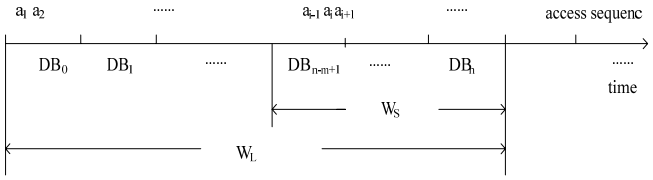


Figure 3. Relation between small window and large window

In order to specify the relation between data blocks of the large window and the user access sequence used to train prediction model, Figure 4 gives the relations among the large window, the small window and the sliding window. By the time dimension, the total user access sequence is partitioned into some continuous data blocks and some subsequent user requests which can not form one data block. The sliding window represents the total user access sequence which is used to construct prediction model. The large window includes all of data blocks labeled from *0* to *n*. The small window is a part of the large window, whose data blocks are labeled from *n-m+1* to *n*.

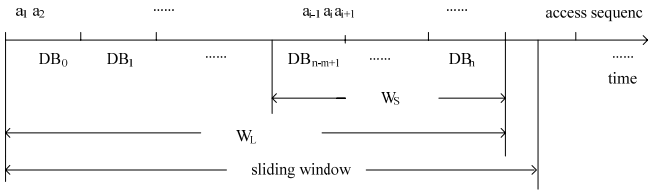


Figure 4. Relations among W_L, W_S and sliding window

For the sake of choosing training data, the total user access requests with access log and current user requests are regarded as one user access sequence which is partitioned into data blocks. W_L ,W_S and sliding window are respectively set the original value. The original prediction model is constructed with the user access sequence in the sliding window according to certain prediction algorithm. Then the sizes of W_L ,W_S and sliding window are adjusted based on prediction precision's changing. The essence of adjusting strategy contains three aspects. First, the sliding window slides ahead and the new user requests are continuous inserted into the sliding window. Second, the prediction model is updated in order to capture the changing user request patterns in time. Third, if the new user access request can compose one new data block, the sizes of W_L ,W_S increase one and these new user requests are inserted into two windows. At the same time, some elder data blocks may be deleted from prediction model according to some rules and windows' sizes will change. The concrete adjusting rules are described as following.

1) If the small window's precision is consistent decrease, the sizes of W_L and W_S are shortened and some elder data blocks are deleted from prediction model. The user access requests in the deleted data blocks are obliterated from the sliding window whose length is reduced accordingly. Consistent decrease indicates that any difference between

the prediction precision of the newest data block in the small window and any other is negative. Consistent decrease shows that the outdated access information reduces prediction precision and the prediction model better represents user access characteristics of the elder data blocks which are not consistent with the new user browsing behaviors. Thus the prediction precision of the newest data block falls.

2) If the small window's precision is consistent increase, the width of W_L and W_S is widened. The large window and small window includes the newest data block and their sizes are increased. Consistent increase indicates that any difference between the prediction precision of the newest data block in the small window and any other is plus. Consistent increase denotes the newest user requests enhance prediction capability of the original prediction model so that training data increases.

3) If the small window's precision is stability, the sizes of W_L and W_S are not changed. The precision stability is defined that any difference is very smaller between the prediction precision of the newest data block in the small window and any other. It shows that the newest user requests are consistent with the original model. Thus the large window and the small window cover the newest data block and the eldest data block are deleted from them. At the same time, the corresponding browsing patterns are deleted from the prediction model and the new user requests are added.

4) The prediction model is in a conversion phase if any instance above mentioned does not happen. In order to avoid forgetting the elder training samples too earlier, the wide of the large window is enlarged and the small window's one is kept.

C. Decision algorithm

Suppose that the length of W_L is n and the length of W_S is m, where n is greater than m . The large window's data blocks from the eldest to the newest are respectively labeled from *0* to *n-1*. When a new user request appears, the sliding window goes forward and the new user request is added into it while the bottom of the sliding window does not change. If the new user requests of the sliding window form one new data block *n*, the changing features of the small window's prediction precision are calculated and the sizes of W_L and W_S are changed according to adjusting rules. Then the length of the sliding window changes and the prediction model's access patterns are updated. In the following section, we specify concrete algorithm and make use of the prediction model which is our previous work [19]. To make this process clear, decision of training data is separated into two steps. First step is the original values of W_L, W_S and the sliding window are respectively set. At the same time, the original prediction model *PM* is constructed with the user access sequence in the sliding window. Second step is to change training data by adjusting the lengths of different windows. The following algorithm *DecisionMethod* gives the adjusting strategy.

Algorithm *DecisionMethod*(W_L, W_S, *SW*, *PM*, *RS*)
Input: W_L is the large window ,W_S is the small window, *SW* is the sliding window, *PM* is prediction model and *RS* is the new user request sequence.
Output: *PM*, W_L, W_S, *SW*

```
BEGIN
  For (every request A of RS)
  BEGIN
   A is inserted into SW and PM
   WHILE (one new data block appears)
     BEGIN
      n=n+1;
      m=m+1;//The sizes of W_L and W_S increases one.
      IF (prediction precision of W_S is consistent decrease)
        BEGIN
          n=n-2;
         //The eldest two data blocks are deleted from W_L
          m=n/2;// To change the size of W_S
         Every request in the deleted data blocks is deleted
         from SW and PM.
        END
      ELSE
      IF (prediction precision of W_S is stable)
        BEGIN
          n=n-1; //The eldest data block is deleted from W_L
          m=m-1;//To keep the size of two windows
         Every request in the deleted data blocks is deleted
         from SW and PM.
        END
      ELSE
      IF (prediction precision of W_S is consistent increase)
          m=n/2;
       ELSE   m=m-1; //To increase the large window's size
      END
END
```

When a new user request appears, we make use of the algorithm in the [20] and its data structure to insert the request into the prediction model so that the changing user behavior patterns are updated in time. When the large windows is shorten, some data blocks are deleted from it and the corresponding user access information is forgotten so that prediction model reduces the outdated browsing patterns and saves space.

IV. EXPERIMENTS

To evaluate our decision method called DM, we adopt Microsoft Visual C++ 6.0 to develop a series of experiments. To compare our method with other, we simulate other system without any training data selection strategy called Non-Selection. DM and Non-Selection both makes use of the prediction model in the [20] during experiments. We compare our approach's performance with Non-Selection from the log day number of training data,

prediction model's space, prediction precision, hit rate, and traffic incremental rate.

A. Logs and parameters set

We do the trace-driven simulation using two real trace files. One file is from Chinese certain medium-sized education institution's proxy server log, called CE log. This trace file is collected by one proxy software from January 1, 2005 to January 26, 2005. Every record includes request object's access information such as IP ,URL and access time. Another file is from American National Lab of Applied Network Research (NLANR) which provides web access logs continuous seven days in one ftp server. We download one proxy server log by authorized username and password, called NLANR_NY, which is composed of continuous user accesses from June 3, 2007 to June 28, 2007.

We remove all dynamically generated files. These files can be in types of ".asp", ".php", ".cgi" and so on. We also filter out embedded image files such as ".gif" and ".jpg" because we believe the image file is an embedded file in the HTML file. Access request sequence of each log file is partitioned into user sessions. One user session is one orderly access sequence from the same user. If a user has been idle for more than two hours, we assume that the next request from the same user starts a new user session. We recognize that the time interval of partitioning sessions may introduce some inaccuracy in the simulator, but it will not affect the evaluation of different models.

All of the models make the following configuration. A global model is constructed for all users in each test. All predictions are based on the model. Because of physical systems limitation (e.g. network bandwidth), each model predicts at most a request according to a user's current request every time. The prefetching cache size is formulated in terms of number of web pages, rather than number of bytes. The approach is more intuitive for interpretation of the results, without altering their significance [16]. The prefetching cache replacement algorithm is LRU. The size of conditional probability threshold affects both hit rate and the amount of traffic increment. A larger threshold allows less data to be prefetched, which is beneficial to traffic, but may decrease hit rate. We take into account a trade-off probability threshold. Thus, conditional probability threshold is set to 0.1.

For our decision method, user access sequence is partitioned into data blocks. Each data block includes one day's user requests so that CE and NLANR_NY both includes 26 data blocks. Each data block is partitioned into some user sessions according to IP address and time threshold. The large window includes 7 data blocks and the small window's length is 3 because people regularly browse web every week.

B. Evaluation parameters

We employ the following four metrics [4][21] in the experiments.

Definition 5 *Precision* is the ratio of the number of correct predictions to the number of total predictions. If users in the subsequent requests access the predicted page that is in the prefetching cache, the prediction is considered to be correct, otherwise it is incorrect. The metric represents the fraction of predicted pages that are actually used.

Definition 6 *Hit rate* refers to the percentage of user access requests that are found in the prefetching cache.

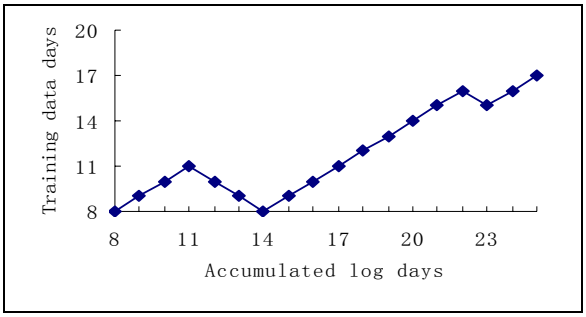
Definition 7 *space* is the required memory allocation measured by the number of nodes for building a prediction model in the web server for prefetching.

Definition 8 *Traffic incremental rate* is the ratio of the traffic from undesired pages to the traffic from the total user requests. Some of the prefetched pages will not be actually requested. Therefore, they increase the network traffic overhead.

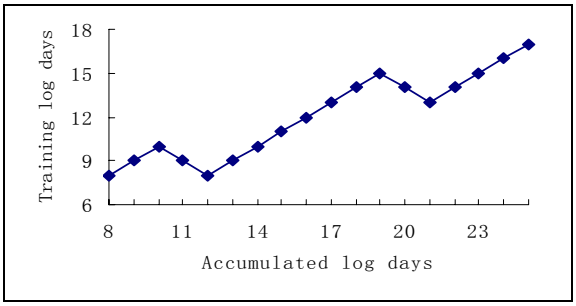
Web prefetching aims at maximizing the first three metrics and minimizing the last one. It is obvious that these metrics are conflicting. The more pages are prefetched, the more probable it is for some of them to be accessed and the hit rate increases. At the same time, precision decreases and network traffic increment is high. Thus, it is a trade-off among these objectives that the model should consider.

C. Decision of training data

In order to decide training data, we respectively choose a part of CE and NLANR_NY to do a series of experiments. Figure 5 depicts the changing process of training data, where abscissa denotes the log's day number and ordinate is the log's day number of training data. For example, abscissa is 20, which denotes that 20 days' log is provided to train prediction model.



(a) Training data versus accumulated log (CE)



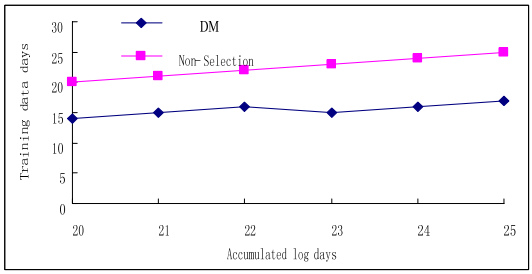
(b) Training data versus accumulated log (NLANR_NY)

Figure 5. Decision of training data

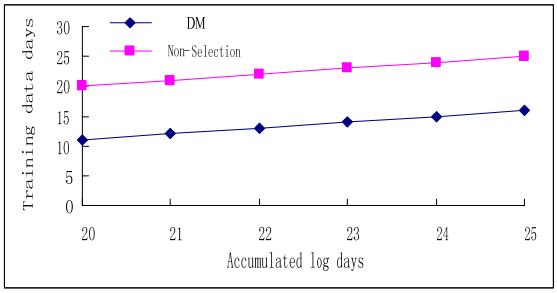
As presented in Figure 5, DM respectively selects the latest 15 days log and 12 days log to construct prediction model for CE and NLANR_NY when the day number of accumulated log is 22. We can conclude that the prediction precision of the small window is consistent increase when the day number of accumulated log varies from 14 to 22 from Figure 5(a) because training data is increasing. At the same time, it also displays that training data decreases when the changing trend of prediction precision is consistent decrease that takes place between days 11 to 14 in Figure 5(a).

D. Comparision of two approaches's training data

In the set of experiments, we display the simulation results of training data comparison among two methods. In the condition of the same day number of accumulated log, Figure 6 gives the log day number of training data with CE and NLANR_NY.



(a) Comparison of training data days using CE log



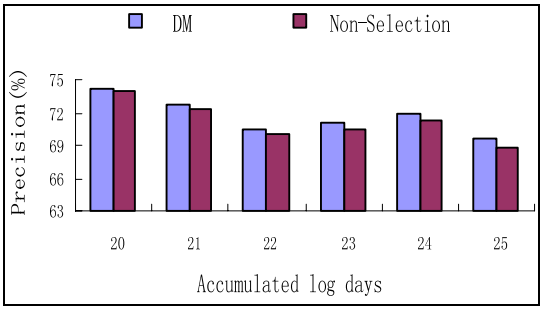
(a) Comparison of training data days using NLANR_NY log

Figure 6. Training data days versus accumulated log days using two logs

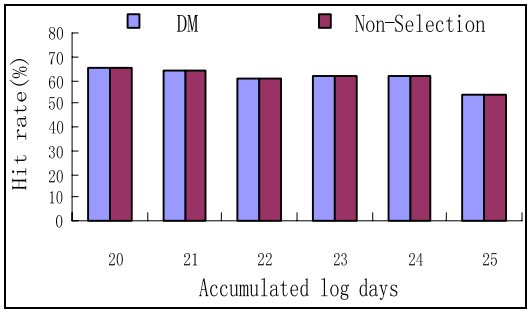
As presented in Figure 6, the day number of training data with DM is less than Non-Selection's when the day number of accumulated log varies from 20 to 25. Because the training data in our method is chosen by the changing trend of the small window's precision during constructing prediction and the outdated user requests are deleted from model and the log day number of training data is reduced.

E. Prefetching performance test

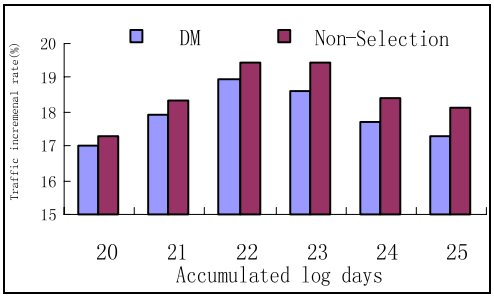
We compare the prefetching performance of DM and Non-Selection with two logs in the condition of the same day number of accumulated log. Figure 7 and Figure 8 respectively show different parameter's comparison of prefetching performance using CE and NLANR_NY. In the two figures, abscissa is the day number of accumulated log and ordinate respectively represents precision, hit rate and traffic incremental rate for (a), (b), (c) of every figure.



(a) Precision comparison



(b)Hit rate comparison

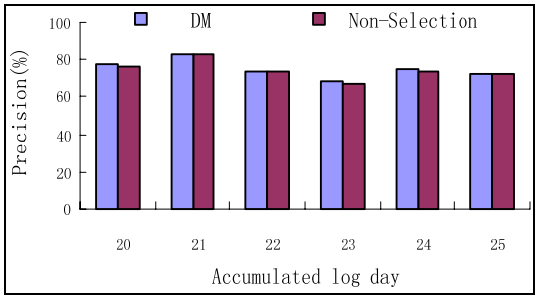


(c) Comparison of traffic incremental rate

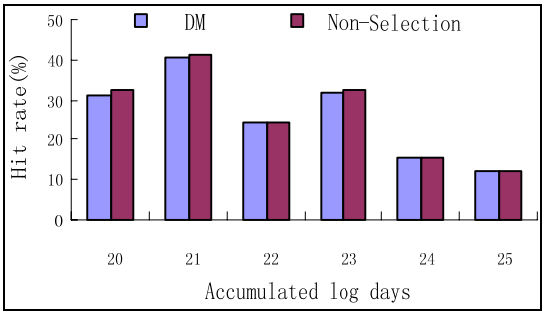
Figure 7. Prefetching performance comparison(CE Log)

In Figure 7, our method and Non-Selection respectively adopt the corresponding result of training data which is displayed in Figure 6 (a). According to the training data, one prediction model is constructed and the succedent one day's user requests are predicted based on the model. For example, DM and Non-Selection respectively use 14 days log and 20 days log to construct the prediction model when abscissa is 20. Then the 21th day's requests are predicted. As presented in Figure 7, the prefetching performance of DM exceeds Non-Selection's. The reason is the Non-Selection method ignores the choosing problem of training data so that the corresponding prediction model does not completely represent the user browsing behaviors. At the same time, DM adopts the technology of adjusting windows to change

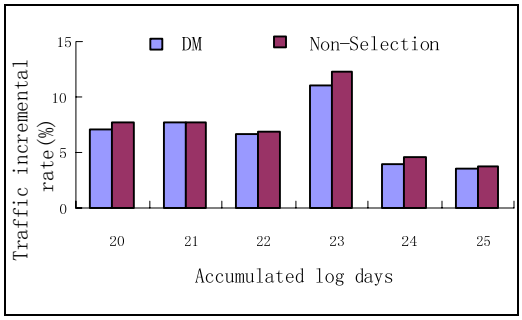
training data so that some outdated access requests are deleted from the prediction model.



(a) Precision comparison



(b) Hit rate comparison



(c) Comparison of traffic incremental rate

Figure 8. Prefetching performance comparison(NLANR_NY Log)

In Figure 8, our method and Non-Selection respectively adopt the corresponding result of training data which is displayed in Figure 6(b). As presented from Figure 8(a) to (c), precision of DM is average higher 0.77% than Non-Selection's and traffic incremental rate is average less 0.49% than Non-Selection's. Although our approach does not have better hit rate or in some cases even worse from the experimental results, it is evident that DM prefetches more web pages correctly than Non-Selection and this is achieved with less cost in the network traffic that has less adverse effect on other network applications. Thus, our algorithm achieves the best performance.

F. Model's space test

We compare prediction model's space of DM with Non-Selection's using two logs in the condition of the same day number of accumulated log. Figure 9 gives the test results using CE and NLANR_NY, where abscissa represents the day number of accumulated log and ordinate denotes the space reduction rate. It is calculated using the following formula.

$$space\ reduction\ rate = \frac{Non-Selection's\ space - DM's\ space}{Non-Selection's\ space}$$

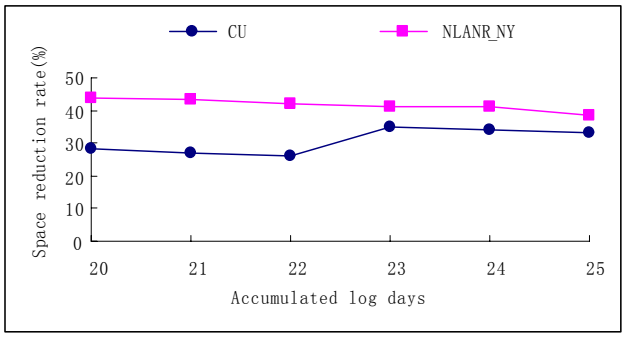


Figure 9. Comparison of prediction model's space using two logs

Figure 9 shows that prediction model's space of DM is always less than Non-Selection's for different logs. The space reduction rate is 25.97% for CE log and it is 41.96% for NLANR_NY log when the day number of accumulated log equals to 22. Figure 9 indicates that our method effectively reduces the prediction model's space.

V. CONCLUSION AND FUTURE WORK

In this paper, we consider the choosing problem of training data and propose a decision method of training data, which is developed according to monitor prediction precision's changing features. It is designed to partition user access sequence into continuous data blocks and makes use of one sliding window, a small window and a large window to capture the precision's characteristics among data blocks so that training data is adjusted. We compare our method with Non-Selection approach from model's space and prefetching performance using two real logs. The experiments show that, for the different day number of accumulated log, our method outperforms Non-Selection's and achieves higher prediction precision with quite low traffic incremental rate and less model's space.

The traces we use are from years ago and some users' behaviors in web surfing could have changed. In the future, we will try to obtain web data from different sources more recent and test the performance of our algorithms.

REFERENCES

[1] J. Domenech, J. A. Gil, et al, "Using Current Web Structure to Improve Prefetching Performance," Science Network, vol. 54, Dec. 2009, pp.1404-1417.

[2] P. Venketesh, D. R. Venkatesan, and L. Arunprakash, "Semantic Web Prefetching Scheme Using Naive Bayes Classifier," International Journal of Computer Science and Applications, vol. 7, 2010, pp. 66-78.

 ISBN: 978-1-61208-124-3

[3] B. Ossa, J. Sahuquillo, et al, “An Empirical Study on Maximum Latency Saving in Web prefetching,” Proc. IEEE Web Intelligence and Intelligent Agent, IEEE Press, Sep. 2009, pp. 556-559.

[4] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, “A Data Mining Algorithm for Generalized Web Prefetching,” IEEE Transactions on Knowledge and Data Engineering, Sep. 2003, Vol. 15, NO. 5, pp. 1155-1169.

[5] R. Sarukkai, “Link Prediction and Path Analysis Using Markov Chains,” Proceedings of the 9th International World Wide Web Conference, Amsterdam, Holand, May 2000, pp. 377-386.

[6] L. Shi, Z. Gu, et al, “Popularity-Based Selective Markov Model,” Proc. IEEE/WIC/ACM International Conference on Web Intelligence, Beijing, China, Sep. 2004, pp. 504-507.

[7] K. Lau and Y. Ng, “A Client-Based Web Prefetching Management System Based on Detection Theory,” Lecture Notes in Computer Science, Springer, 2004, vol. 3293, pp. 129-143.

[8] B. D. Davison, “Learning Web Request Patterns,” Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Springer, 2004, pp. 435-460.

[9] J. Domènech, J. Sahuquillo, et al, “How Current Web Generation Affects Prediction Algorithms Performance,” Proceedings of the 13th International Conference on Software, Telecommunications and Computer Networks, Split, Croatia, Sep. 2005.

[10] Z. Ban, Z. Gu, and Y. Jin, “Selection of Training Period Based on Two-Window,” Proceedings of the 10th International Conference on Advanced Communication Technology, IEEE Computer Society Press, Feb. 2008, pp. 2043-204.

[11] E. Markatos and C. Chronaki, “A Top-10 Approach to Prefetching on the Web,” Proceedings of the Eighth Annual Conference of the Internet Society, Geneva, Switzerland, 1998.

[12] V. N. Padmanabhan and J. C. Mogul, “Using Predictive Prefetching to Improve World Wide Web Latency,” Computer Communication Review, 1996, vol. 26, NO. 3, pp. 22-36.

[13] J. Griffioen and R. Appleton, “Reducing File System Latency Using a Predictive Approach,” Proceedings of Summer USENIX Technical Conference, 1994, pp. 197-207.

[14] J. Borges and M. Levene, “Data Mining of User Navigation Patterns,” Proceedings of WEBKDD, 1999, pp. 92-111.

[15] Z. Su, Q. Yang, et al, “WhatNext: A Prediction System for Web Requests Using N-Gram Sequence Models,” Proceedings of the First International Conference on Web Information Systems Engineering, 2000, pp. 200-207.

[16] T. Palpanas and A. Mendelzon, “Web Prefetching Using Partial Match Prediction” Proceedings of the Fourth Web Caching Workshop, San Diego, California, 1999.

[17] L. Fan, P. Cao, and Q. Jacobson, “Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance,” Proceedings of the ACM SIGMETRICS'99, Atlanta, Georgia, May 1999.

[18] M. Deshpande and G. Karypis, “Selective Markov Models for Predicting Web Page Accesses,” ACM Transactions on Internet Technology, 2004, vol. 4, NO.2, pp. 163-184.

[19] C. Bouras, A. Konidaris, and D. Kostoulas, “Predictive Prefetching on the Web and its Potential Impact in the Wide Area,” World Wide Web: Internet and Web Information Systems, 2004, vol.7, NO. 2, pp. 143–179.

[20] Z. Ban, Z. Gu, and Y.Jin, “An Online PPM Prediction Model for Web prefetching,” Proceedings of the 9th ACM International WoTrkshop on Web Information and Data Management, Nov. 2007, pp. 89-96.

[21] J. Domènech, J. A. Gil, et al. “Web Prefetching Performance Metrics: A Survey,” Performance Evaluation, 2006, vol. 63, NO. 9, pp. 988-1004.

 ISBN: 978-1-61208-124-3