A Prediction System for Multimedia Pre-fetching in Internet

Zhong Su*
Department of Computer Science
Tsinghua University
Beijing P.R.China 10084
suzhong_bj@hotmail.com

Qiang Yang *
School of Computing Science
Simon Fraser University
Burnaby BC V5A. 1S6 Canada
qyang@cs.sfu.ca

Hong-Jiang Zhang
Microsoft Research China
5F,Beijing Sigma Center Beijing
P.R.China
hjzhang@microsoft.com

ABSTRACT
The rapid development of Internet has resulted in more and more multimedia in Web content. However, due to the limitation in the bandwidth and huge size of the multimedia data, users always suffer from long time waiting. On the other hand, if we can predict the web object or page that the user most likely will view next while the user is viewing the current page, and pre-fetch the content, then the perceived network latency can be significantly reduced. In this paper, we present an n-gram based model to utilize path profiles of users from very large web log to predict the users' future requests. Our model is based on a simple extension of existing point-based models for such predictions, but our results show that by sacrificing the applicability somewhat one can gain a great deal in prediction precision. Also we present an efficient method to compress the prediction model size so that it can be fitted into the main memory. Our result can potentially be applied to a wide range of applications on the web, including pre-fetching, enhancement of recommendation systems as well as web caching policies. The experiments based on three realistic web logs have proved the effectiveness of the proposed scheme.

Keywords
Web prediction, Markov decision processes, pre-fetching, multimedia.

1. INTRODUCTION
Internet is a global distributed, dynamic information repository that contains vast amount of digitized information, and more and more such information now available in multimedia forms. However, the colossal amount of information also poses great challenge for many

Internet users struggling to access useful information in a timely manner with limited access bandwidth. An effective way to reduce such burden, or at least reduce the perceived latency in browsing or down loading is to pre-fetch or catching the requested multimedia data to the client end while the user is view the current web page or object. To achieve this, we need a scheme to accurately predict a user's future requests. Fortunately, such prediction is increasingly feasible as more and more browsing information becomes available in web logs. This paper present an effective scheme we proposed to predict users' actions in Web browsing based on information from web logs.

Due to the increasing need for pre-fetching web documents, there has been an increasing amount of work on prediction models on the web. One popular way is to predict next URL based only on present URL. This can be regarded as a Markov model in the graph. Given the current node, what is the probability to access another nodes in the list of known URLs? This may require the knowledge of the context of the web page. Examples are the WebWatcher[6] system and Letzia[10] system. Pre-sending systems go a step further --- they focus on making use of the predictions to send documents ahead of time. Accurate prediction can potentially shorten the users' access times and reduce network traffic when pre-sending is handled correctly.
[1]

Prediction models can be either point based or path based. Point-based draw on relatively small amount of information from each session and therefore the prediction can potentially be rather inaccurate. For examples, the best model [1][16] predicts, for a confidence measure of over 50%, future documents with an accuracy of only around 30%. There has been relatively little work on path-based models in the past. These models are built based on the user's previous path data, and can potentially be more accurate. But the general belief is that they may suffer from much lower applicability because sequences with long length are rare. The aim of this paper is to dispel this myth and show that with large enough web access logs one can build an accurate enough prediction models that also come with high applicability.

In web prediction, each URL is a symbol, not numerical values. Therefore, the prediction work based on URL is the task of string symbol prediction, that is to say, predicting the next symbol given past history of symbol appearance in a given string. In this paper, we present a probabilistic path-based prediction model that is inspired by n-gram prediction models commonly used in speech-processing communities [9]. We have found that when using 3-gram and above the prediction accuracy is increased substantially whereas there is only a moderate decrease in applicability. We present a combined approach where multiple high-order n-gram models are organized in a step-wise manner. Experiments show

[*] This work was performed while the author was visiting Microsoft Research China in Beijing.

that this approach achieves a reasonable balance between precision and applicability. We also provide an efficient method to reduce our model size. Our work assumes no knowledge of user profiles as the ones required by Syskill and Webert [12], and no knowledge about linkage structures of web sites as required by WebWatcher. It's only requirement is that user sessions in web access can be logged successfully -- a requirement realistic enough to apply to a wide range of multimedia applications.

This paper is organized as follows. Section 2 relates a list of previous works to our work. Section 3 presents an algorithm for construction of the path-based model, followed by detailed presentation on the proposed prediction algorithms. Section 5 evaluates the performance of the proposed algorithm, and Section 6 presents our method of pre-sending Web document based on our predicate result. Finally the last section provides a summary of this work.

2. RELATED WORK

Prediction has been a topic of interest in traditional computer systems research. Curewitz et al [4] were the first to examine the use of compression modeling techniques to track events and pre-fetch items. They prove that such techniques converge to an optimal online algorithm. They go on to test this work for memory access patterns in an object oriented database and a CAD system. Kroeger et al [7] adapts "Prediction by Partial Match" in a different manner. The problem domain they examine was the file systems access patterns. The hit ratio of 4M caches using PPM is even higher than 90M caches using LRU.

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 [6], which makes recommendations on the future hyperlinks that the user might like to visit based on a model obtained through reinforcement learning. Other recommendation systems include the Letizia system that anticipates a user's browsing actions by making forward explorations and the Syskill & Webert system that learns to rate pages on the World Wide Web. Compared to these systems, our path-based prediction model is obtained by building sequences of user requests of long enough length from all user actions in a user log and predicts the actions which may happens in the next m requests (m >= 1) based on statistical analysis of sequence information.

Due to bandwidth limitations, users on the Internet are experiencing increasing delays in obtaining the desired documents. In response, many researchers have designed pro-active systems that make use of predictions from a learned model to pre-fetch or pre-send documents. The work by Zukerman et al. and Albrecht et al. belong to this class. In this work, a Markov model is learned through training on a web server log based on both time interval information and document sequence information. The predicted documents are then sent to a cache of a certain size on the client side ahead of time. Similarly, Lau and Horvitz [5] have classified user queries and built a Bayesian model to predict users' next query goal or intention based on previous queries and time interval. Our work is also related to that of [11] who studied users' complete web search sequences and the work of Silverstein [13] who provided a detailed statistical analysis of log data. Compared to these systems, our algorithm only makes prediction on users' actions when it gathers enough information regarding the users' actions on a long enough sequence of such requests. When the users are observed to make short sequence visits, we do not make any predictions since such users may be making random visits on the web, and thus the next action may not be predictable. Also, two closely related works are that of [14] [15], who presented path-based prediction models. However, they did not compare the power of different n-gram models nor did they offer a hybrid n-gram model such as we do in this paper.

Our work is also related that of the collaborative filtering evaluation work of [3]. However, their work is based on point rather than sequence models. In addition, their correlation model may require too much computational resource to be applied in real time for web prediction. As noted above, our system is also different from many recommendation systems for it does not require the knowledge of web site link structure. Horvitz [5] presented decision-theoretic policies for guiding the pre-fetching decisions web contents in situations of limited or costly bandwidth. He also studied bi-gram models and found that it is useful in prediction.

3. PATH BASED MODEL

Our path-based model is built on a web-server log file L. We consider L to be preprocessed into a collection of user sessions, such that each session is indexed by a unique user id and starting time. Each session is a sequence of requests where each request corresponds to a visit to a web page (an URL). For brevity, we represent each request as an alphabet. The log L then consists of a set of sessions.

Our algorithm builds an n-gram prediction model based on the occurrence frequency. Each sub-string of length n is an n-gram. These sub-strings serve as the indices of a count table T. During its operation, the algorithm scans through all sub-strings exactly once, recording occurrence frequencies of documents' requests of the next m clicks after the sub-string in all sessions. The maximum occurred request (conditional probability > ε) is used as the prediction of next m steps for the sub-string. The algorithm for building a path-based model on sub-strings is described below.

In our experiment, the filtering step on a web log is according to the following steps:

1) We remove data generated by search engines from the server and import them into a database system.
2) Extract sessions from the data. For any two adjacent page visits, if the time interval between the visits is greater than a time threshold T, then these visits are considered to belong to two different sessions. Here one user session is considered as a sequence of URL requests by a single user. We find it reasonable to set the threshold T to be two hours for NASA data that we present later and 24 hours for MSN data.
3) Removes all requests with low visiting frequency according to a certain threshold θ.

All URLs whose access count is below the θ threshold is removed from the log. Based on our empirical tests to be discussed in Section 4, it is reasonable to set θ to 10 times or less in our web server logs. And we can also show that with such

filtering process the precision is raised very much and the applicability is dropped not very much. In our experiments we set θ to 5.

Algorithm ***PathModelConstruction****(**n**: length of n-gram; **m**: predictive steps; **L**: log file)*
Begin
Filter the log file L then extract all the sessions from the L.
Initialize the hash table T that stores the occurrence of the document in m steps after n-grams.
Initialize the hash table H that stores results of this model
For I = 1 to Total Session number
For J =1 to Current Session Length
If (J > n) Then
S = previous n requests from the current position
C = Set of distinct pages that are the next m requests after the current position
For each item C' of C, Do
T [P, C']++;
Update H_{n_m} *(P) so that it gives the C' that has highest T[P,C'] value >* ε *;*
End For
End If
End For
End For
Return H_{n_m}
End

As a very simple example, consider a log file L consisting of the following request paths:

A,B,C,D,E
A,B,C,E,F
A,B,C,E,F
B,C,D,K,A
B,C,D,K,B
B,C,D,F,L

If we were to construct a 3-gram model and set m to two, our algorithm returns the following hash table.

Table 1: Hash table of 3-gram model

N-Gram	Prediction
A,B,C	E(100%)
B,C,D	K(66%)
…	…

4. PREDICTION ALGORITHM

Based on the n-gram prediction model constructed out of the log data, we can then make predictions on a user's clicks in real time. Let H_{n_m} be the prediction model built on n-gram model of predicting which are the most probable requests in next m steps. Our algorithm is as follows:

Algorithm ***m- step n-gram+*** *(**P**: user's current clicking sequence; **n**: minimal path length)*
Begin
If Length of P < m
Then return ("No Prediction");
Else
Begin
For I= max (Length (P), max Length of prediction model) downto n do
If P is an index in hash table H_{I_m} *then*
Prediction = H_{I_m} *[P];*
Return (Prediction);
End If
End For
Return ("No Prediction");
End
End If
End

For comparison purposes, in our experiment we also test individual n-gram algorithms as defined below:

Algorithm ***m- step n-gram*** *(**P**: user's current clicking sequence)*
Begin
If Length (P) >= n and sub_string(P,Length(P)-n+1,n) is an index of H_{n_m} *Then*
Prediction = H_{n_m} *[P];*
Return (Prediction);
End If
Return ("No Prediction");
End

As an example, assume that we have built up 1-step 3-gram and 1-step 2-gram models as H_{3_1} and H_{2_1} in the last section. Suppose that we observe that the current clicking sequence consists of only one click "DBC". In this case, the prediction algorithm checks H_{3_1} first to see if an index "DBC" exists. It finds out that the index does not exist. Therefore, it checks the 1-step 2-gram model H_{2_1} for the index "BC", which exists, thus the predicted next click is "D", according to H_{2_1}.

In the evaluation of the algorithm, we use the following measures. Let $S(m)=\{S_1, S_2,\ldots,S_n\}$ be the set of sessions in a log file that have sequence length greater than m. We build models on a subset of these sequences, known as the training sequences, which are separated from the remaining or the testing sequences. When applying the trained model on the testing sequences, let P^+ be the correct predictions and P^- be the incorrect predictions. Because we remove the infrequent requests, the union of P^+ and P^- is a subset of S(m). Let $|R|$ be the set of all requests. We define the following measures for each learned prediction model H_{n_m}:

$$precision = \frac{P^+}{\left(P^+ + P^-\right)} \quad (1)$$

$$applicability = \frac{P^+ + P^-}{|R|} \quad (2)$$

In the above equations, *precision* has its similar meaning as often used in information retrieval literature, whereas *applicability* is a new measure that is different from *recall*. In particular, the notion of applicability is measures, out of all requests in the original log, the number of requests can be predicted (correctly or wrongly) by our model.

5. DOMAIN ANALYSIS AND EVALUATION

We first analyze the data set under consideration. This is the data set used in Zukerman et al.'s work on predicting user's requests [Zukerman et al. 1999 and Albrecht et al., 1998]. It consists of server log data collected during a 50-day period of time. It includes 525,378 total user requests of 6727 unique URL's (clicks) by 52,455 different IP's, consisting of 268,125 sessions.

One important piece of information about the server data is revealed in Figure 2. In this figure, the horizontal axis shows integer in log scale, designating the number of user visits (to pages). There are two curves in the figure. The upper curve "Page Ratio" depicts, for each value X on the X-axis, the percentage of pages that are visited X times or less by all users. For the same X value, the lower curve "Request Ratio" depicts, shows the percentage of accumulated visits out of all visits in the log on the pages which are visited X times or less.

$$PageRatio(X) = \frac{\sum_{i=1}^{x} |S_i|}{|S|} \quad (3)$$

$$\text{Re}questRatio(X) = \frac{\sum_{i=1}^{x} (|S_i| * i)}{\sum_{i=1}^{\infty} (|S_i| * i)} \quad (4)$$

Thus, for example, X=10 represents a visit count of ten times. The upper curve for X=10 shows that around 60% of pages are each visited 10 times or less, and the total number of such visits represents around 15% of all visits there are.

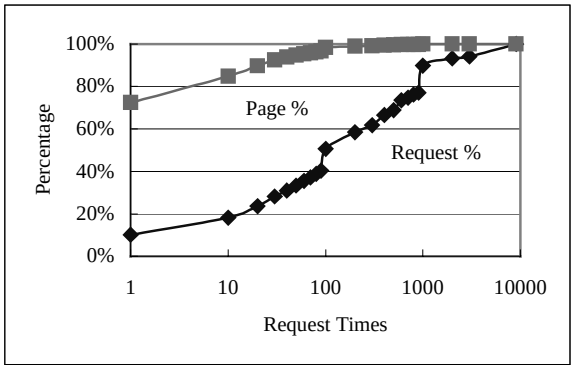

Figure 2. Page vs. request percentage for Monash University data set

We have also repeated the page vs. request ratio for one more data sets, NASA data sets, as shown in Figures 3. The NASA data set contains two months worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida[1]. The log was collected from 00:00:00 August 1, 1995 through 23:59:59 August 31, 1995. In this period there were 1,569,898 requests. Timestamps have 1-second resolution. There are a total of 18,688 unique IP's requesting pages, having a total of 171,529 sessions. A total of 15,429 unique pages are requested. MSN data sets, as shown in Figures 4.The MSN.com log is obtained from the server log of **msn.com**, with all identity of users stripped away. It consists of data collected from Jan 27, 1999 to Mar 26, 1999, with a total of 417,783 user requests. This log contains 722 unique IP's requesting 14,048 unique pages. The MSN.com log is unique in that some requests are from groups of users submitted by Proxies or ISP's. Therefore the lengths of some sessions are long. For example, the long sessions range from 8,384 consecutive requests to 166,073 requests.

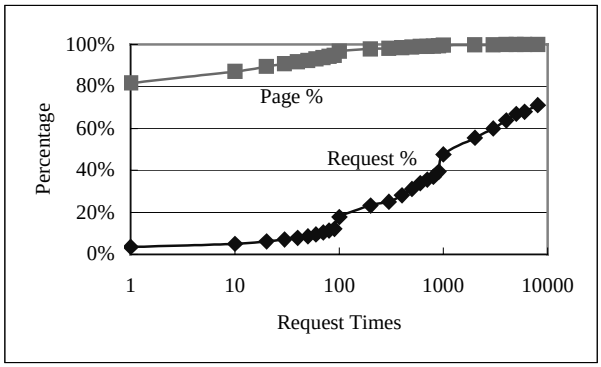

Figure 3. Page vs. request percentage for NASA data set

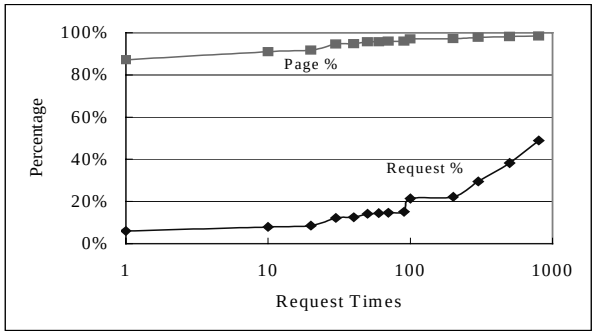

Figure 4. Page vs. request percentage for MSN.com data set

Figures 5 and 6 present the session-length distribution charts for Monash University and NASA data sets. They show, for each session length, the statistical distribution of the sessions having that many consecutive requests on a server. These charts tell us that a significant number of sessions are for one or two requests in a row. However, there are still a sizable number of requests for sessions with lengths greater than three. In fact, our prediction algorithms are aimed at just these sessions. There are several

[1]See http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.

reasons for this choice. We hypothesize that for sessions with lengths less than or equal to three it is difficult to make any predictions with significant accuracy based purely on the statistical information. This hypothesis is supported by our individual n-gram precision experiments in all three domains, as shown in Figures 6, 8 and 10, respectively. It can be seen that for all three domains, the prediction errors are reduced significantly when one predicts based on 3-gram sequences as compared to 1-gram data.

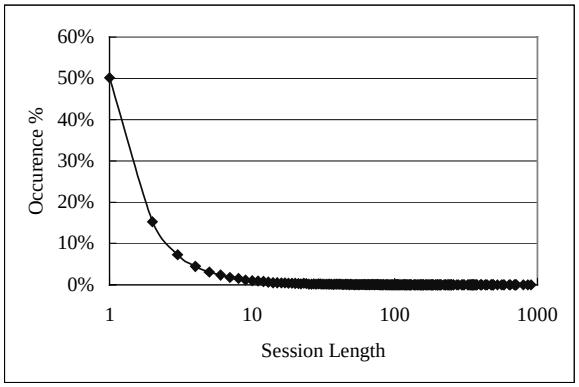

Figure 5. Session length distribution for Monash University data

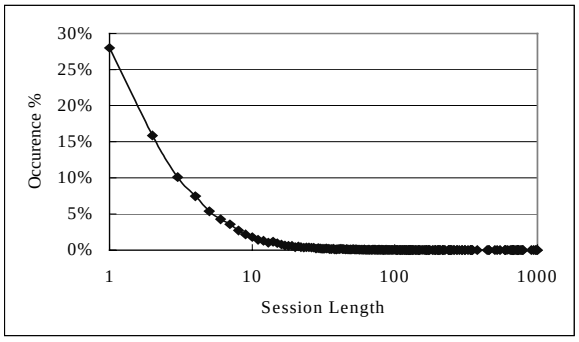

Figure 6. Session length distribution for NASA data

As can be seen from both figures, the data in all three domains follow the same pattern: a large proportion of web pages corresponds to low access ones (less than10 visits per page in the entire log), and together these visits count for a small percentage of total requests as well. Therefore removing them from training set will only decrease precision by a small amount. With the filtering process we can reduce the prediction model to 30% of its original size.

In Figure 7 we show that with the filtering process the precision is raised in high-order n-grams. In our experiments we set θ to 5 (a path must be repeat more than 4 times before it is indexed). It further justifies our filtering operation in the first step of the **PathModelConstruction** algorithm. After applying the model construction algorithm, we have built different hash tables for storing the learned models. For we have compressed the model by eliminating the infrequently occurred patterns these models can be stored in memory. To give an indication of their sizes, for the Monash University log our **n-gram(1)** table sizes are shown in Table 2. The model size is just about 30% of its original size.

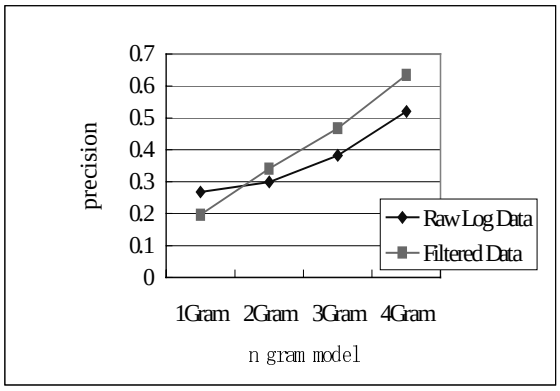

Figure 7. Comparing precision of using raw data and filtered data of different n-gram models for NASA Data

Table 2. Hash table sizes for implementing 1-step n-gram models

Hash Table	1 gram	2 gram	3 gram	4 gram
Table Size	17,434	23.763	22,804	20,958

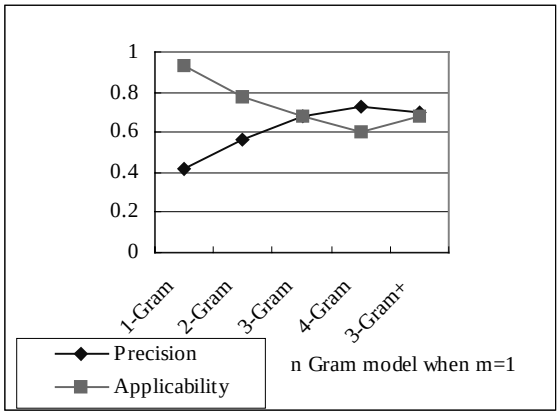

Figure 8. Precision and Applicability as a function of session lengths for Monarsh University data log.1-step n-grams (n=1, 2, 3, 4) represent precision as recorded for sessions having length greater than n.

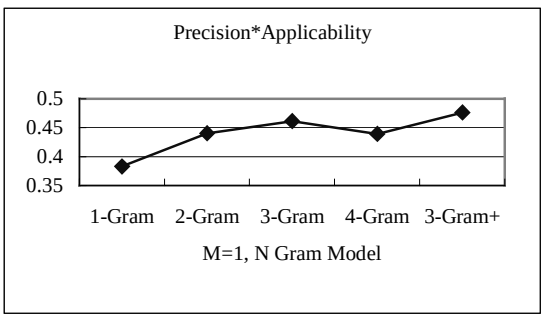

Figure 9. Precision * Applicability for Mosnarsh University data

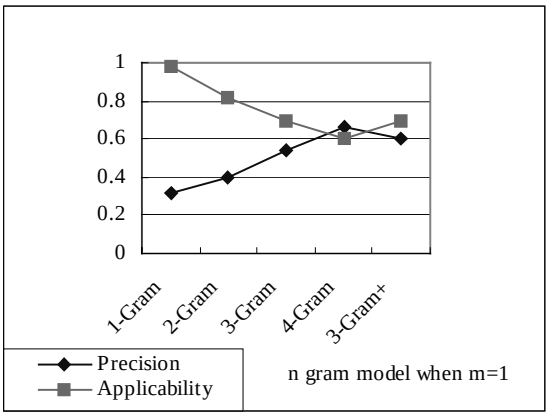

Figure 10. Precision and Applicability as a function of session lengths for NASA data log

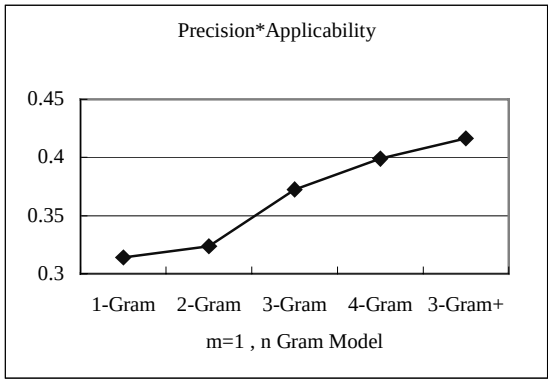

Figure 11. Precision * Applicability for NASA data

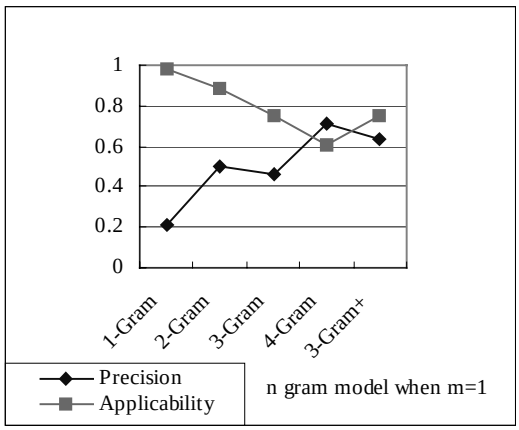

Figure 12. Precision and Applicability as a function of session lengths for MSN data log

For all data, we took 4/5 of the log as training data set and the remaining 1/5 as testing log. For each test, we recorded the precision and applicability information as described by Equations (1) and (2). We have recorded the prediction precision as a function of n where n is the path length of the sequence used for n-gram prediction. Figure 8 and 9 show the precision, the applicability and their product for the Monash University on the same scale. In these and subsequent figures, the x-axes are marked with the length of n-grams (n=1, 2, 3, 4), and the corresponding y-axes represent precision obtained when applying algorithm **n-gram** (1). The remaining mark on x-axes is "3-gram+", which represent experiments on data consisting of sessions having length greater than or equal to three for both training and testing (that is, using **n-gram+(3)** algorithm). As can be seen, our prediction using the combined 3 and 4-gram models achieved a much higher precision than using 1-gram prediction only. Also we can find that "3-gram+" has the highest predictive ability from figure 6. We have also applied the same training and testing to NASA and MSN logs data as shown in Figure 10,11,12,13. The results confirm similar conclusions. For there are too many proxies' visiting logs in MSN data the curve seems a little different. At the place of the three-gram the curve has a sudden drop. However the tendency of the curve is still confirm our conclusions.

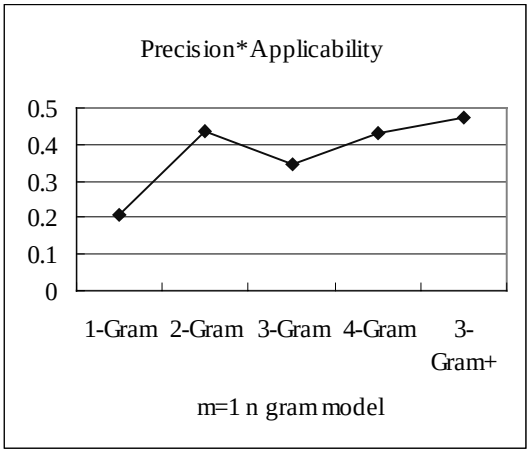

Figure 13. Precision * Applicability for MSN data

In Figures 14 and 15, we compare the precision with the prediction window size m set to 1 and 2. We can observe that the precision increases with larger window sizes.

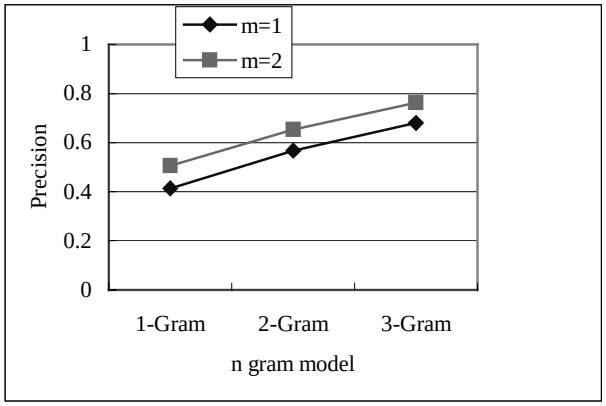

Figure 14. Comparing precision of different window sizes for Monarsh University data

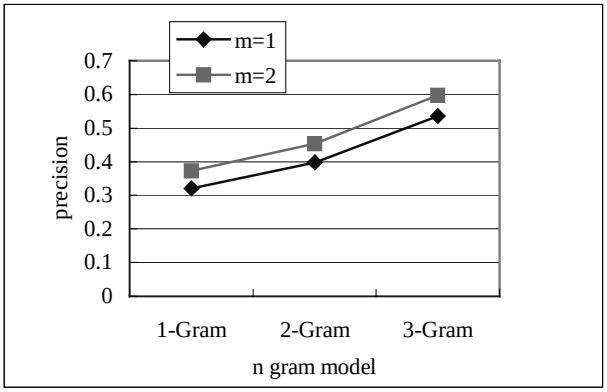

Figure 15. Comparing precision of different window sizes for the NASA data

6. PRE-SENDING DOCUMENTS

Our algorithm predicts, for a given observed path P, the likely documents that are requested by the user within the next window size m. Let the predicted documents that occur within that window be D; D is the set of documents that our n-gram models predict will most likely be accessed. To help the user increase the accessing performance, we can allocate a buffer on the client side, and pre-send these documents to the user via this buffer. Suppose that the buffer has a size of B, and is filled with the top-ranking subset of D. What we are interested in now is how many of the pre-sent documents are actually useful within the next window of size m. This can be measured by the traditional measure of *recall*, defined as the percentage of relevant documents R that are pre-sent and stored in the buffer out of all relevant documents there are.

More precisely, recall that m is the number of documents that actually occurred after a path P of documents is accessed. Let R^+ be the subset of predicted documents D that actually are requested by the user within the next window. Then recall is defined as:

$$recall = \frac{R^+}{m} \tag{5}$$

Similarly, we can extend our earlier definition of precision by taking into account that, out of *B* documents that we predict will be accessed, R^+ of them are actually accessed. Thus, we have

$$precision = \frac{R^+}{B} \tag{6}$$

As all other IR systems, we expect precision and recall to complement each other when the buffer size increases.

We have run experiments on the NASA data, as shown in Figures 16. In this experiment, we set the window size to 2 (that is, we predict the next document and the one after next), and increase the buffer size from one to 11. As we can see, as the buffer size increases, the precision decreases while the recall value increases dramatically. We can also observe that for buffer size = 2, the 3-gram model makes the best combination of precision and recall. As the n in n-gram model decreases, the precision and recall all correspondingly decrease. This further strengthens the importance of our new concept of *applicability*, which as we learned from the last section has the lowest value for 3-gram models as compared with 1- and 2-gram models. For pre-sending systems, one needs to measure their performance using all three measures of applicability, precision and recall.

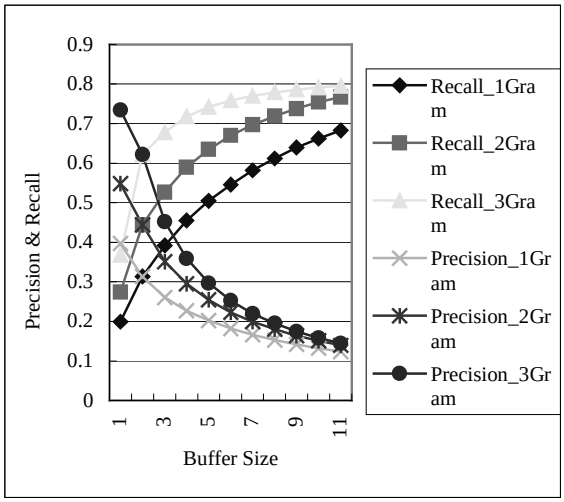

Figure 16. Precision and Recall in the NASA data set for window size two

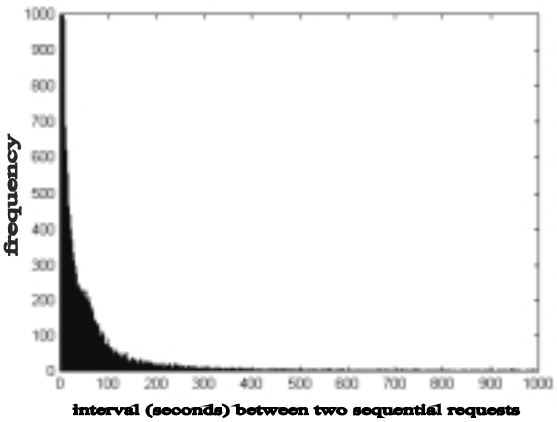

Figure 17. The distribution of users' Requests interval in the NASA's log

Figure 17 shows the distribution of users' requests interval in NASA's log. The horizontal axis is the interval (seconds) between two sequential requests and vertical axis is the frequency of that interval appears in the log. According to NASA's log the average interval between two requests is 95.37865 seconds, which is more than one minute. It will be great helpful if we do pre-fetching during the intervals, especially when user visits page that contains big sized components like image, audio and video data.

We have done a pre-fetching simulation on NASA's data by using 3-Gram+ as the prediction algorithm. We set the buffer size to one in this experiment. Our per-fetching strategy is as follows: once user's requested documents are transferred completely and the next request has not received at the server side we will pre-fetch one document by using our algorithm. The pre-fetching process will be cancelled if the next request from this user is different from our prediction result and the true request has been received at the server side.

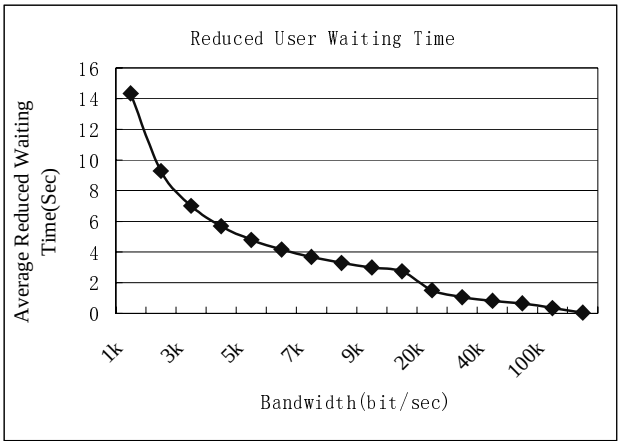

Figure 18. Reduced user requests waiting time under different bandwidth condition by using 3-Gram+ algorithm on NASA's log.

Our experiments test that how much time is saved using pre-fetching this way. Results are shown in Fig 18 as a function of network bandwidth. It can be seen that users' waiting time is reduced by pre-fetching documents using our algorithm, and such reduction increases with bandwidth decreasing.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented an n-gram based model to utilize path profiles of users from very large web log to predict the users' future requests. Our work is aimed at reducing perceived latency when a user is browsing Web content, especially when there are many multimedia objects in a give Web page. Our algorithm is based on the idea that n-gram models for n greater than two will result in significant gain in prediction accuracy while maintaining reasonable applicability. Our experiment results show that for n-gram based prediction when n is greater than 3 gives a precision gain on the order of 10% or more for the two realistic web logs. Our combined algorithm **n-gram+(3)** shows a higher precision than individual 3-gram model and slightly lower than 4-gram model, while at the same time having applicability equal to that of the 3-gram model and higher than that of the 4-gram model. This shows that the **n-gram+(3)** algorithm applies to a significant portion of the web logs for it to be useful. Our results also show that both the training and prediction algorithms can be applied in a real time setting. Our algorithm has immediate applications in web server caching, pre-sending and recommendation systems.

This work is one of several core components of the undergoing framework of adaptive content delivery. To further improve the applicability of the proposed predication algorithm, an important feature work is to develop schemes for integrating the predication algorithm with cost models for catching and pre-fetching of multimedia content.

8. ACKNOWLEDGMENTS

We thank Steven Johnson of MSR Web Support Group and David Abrecht and Ingrid Zukerman from Monash University for sharing their web log data with us.

9. REFERENCES

[1] 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.*

[2] Balabanovic, M. 1998. Exploring versus exploiting when learning user models for text recommendation. *User Modeling and User-adapted Interaction* 8(1-2):71-102.

[3] Breese J., Heckerman D. and Kadie C. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, July, 1998.*

[4] Curewitz K M, Krishnan. P and Vitter. J. S. 1993. Practical Prefetching via Data Compression. *SIGMOD Record,22(2):257-266 . ACM, Jun. 1993*

[5] Horvitz, E. 1998 Continual Computation Policies for Utility-Directed Prefetching. *Proceedings of the Seventh ACM Conference on Information and Knowledge Management, November 1998.*

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

[7] Kroeger T M and Darrell D.E. 1996 Predicting Future file-System Actions From Prior Events. *Proceedings of the USENIX 1996 Annual Technical Conference. Jan 1996*

[8] Lau T., and Horvitz, E., 1999 Patterns of search: analyzing and modeling web query refinement. *User Modeling '99*, pp119-128.

[9] Lee, K. F. and Mahajan, S. 1989. Automatic Speech Recognition: The Development of the SPHINX System. *Kluwer*, Dordrecht, The Netherlands.

[10] Lieberman, H., 1995. Letizia: An agent that assists web browsing. *IJCAI95 – Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 924-929.

[11] Maglio, P. P., and Barrett, R. 1997 How to build modeling agents to support web searchers. *User Modeling: Proceedings of the Sixth International Conference*, UM97, 5-16.

[12] Pazzini, M., Muramatsu, J., Billsus, D., 1996. Syskill and Webert: Identifying Interesting web Sites. *Proceedings of the AAAI 1996.*, Portland, OR., pp54-62.

[13] Silverstein, C., Henzinger, M., Marais, H., and Moricz, M. 1998. Analysis of a very large AltaVista query log. *Technical Report 1998-014*, Digital Systems Research center, Palo Alto, CA.

[14] Pitkow J. and Pirolli P. 1999 Mining Longest Repeating Subsequences to Predict WWW Surfing. *Proceedings of the 1999 USENIX Annual Technical Conference.*

[15] Schechter, S., Krishnan, M., and Smith, M.D. 1998, Using path profiles to predict HTTP requests. *Proceedings of the*

Seventh International World Wide Web Conference Brisbane, Australia.

[16] Zukerman, I., Albrecht. W., and Nicholson, A., 1999. Predicting user's request on the WWW. *UM99 – Proceedings of the Seventh International Conference on User Modeling*.