WhatNext: A Prediction System for Web Requests using N-gram Sequence Models

Zhong Su*
Department of Computing Science and Technology
Tsinghua University
Beijing 100084, China
Suzhong_bj@hotmail.com

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

Hongjiang Zhang
Microsoft Research China
5F, Beijing Sigma Center
#49, Zhichun Road, Haidian District
Beijing 100080, China
hjzhang@microsoft.com

ABSTRACT

As an increasing number of users access information on the web, there is a great opportunity to learn from the server logs to learn about the users' probable actions in the future. In this paper, we present an n-gram based model to utilize path profiles of users from very large data sets to predict the users' future requests. Since this is a prediction system, we cannot measure the recall in a traditional sense. We, therefore, present the notion of applicability to give a measure of the ability to predict the next document. Our model is based on a simple extension of existing point-based models for such predictions, but our results show for n-gram based prediction when n is greater than three, we can increase precision by 20% or more for two realistic web logs. Also we present an efficient method that can compress our model to 30% of its original size so that the model can be loaded in main memory. Our result can potentially be applied to a wide range of applications on the web, including pre-sending, pre-fetching, enhancement of recommendation systems as well as web caching policies. Our tests are based on three realistic web logs. Our algorithm is implemented in a prediction system called WhatNext, which shows a marked improvement in precision and applicability over previous approaches.

Keywords

N-gram model, Web data mining, prediction

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

1. INTRODUCTION

The Internet provides a rich environment for users to retrieve information. At the same time, it also makes it easy for a user to get lost in the sea of information. One way to assist the user with their informational need is to predict a user's future request and use the prediction for pre-fetching, pre-sending, caching and recommendation. Prediction is increasingly feasible to do as more information is tracked through search engines and web servers. The purpose of this paper is to explore ways to exploit the information from web logs for predicting users' actions on the web.

There are generally two types of information source available: query log and server log. A query log tracks users' queries while server log tracks a user's browsing activities on a server. In this paper, we discuss how to exploit server logs of users for the purpose of prediction.

There has been an increasing amount of work on prediction models on the web. In the past, web-log based inference has been focused on prediction models that make best guesses on the user's next actions based on their previous ones. Much work has been done on recommendation systems and pre-sending systems. Recommendation systems rely on a prediction model to make inferences on users' interests based upon which to make recommendations. 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.

Prediction models can be either point based or path based. Point-based prediction models are built on actions that are indexed on time instants and are used to predict the user's next action based on the currently observed action. These models 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 recall 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 recall.

In this paper, we present a simple 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 that this approach achieves a reasonable balance between precision and applicability. Our work assumes very little knowledge about users and target pages while providing high accuracy and maintaining relatively high applicability. The system 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 domains.

Our assumptions, hypothesis and goal may be summarized as follows:

- We assume that we are given a list of sequences of user clicks that correspond to visits to URLs' on the web;
- We assume that the log resides on the server side and that we can observe users' requests made on the server;
- We assume that we have methods of identifying users such that each sequence in the server log corresponds to a unique ID (which may not be a user ID);
- We hypothesize that the users' short request sequences generally correspond to unpredictable requests. We thus only make predictions on users' next actions based on long enough sequences of user requests;

Given the above, our task is to predict the next user request when the users have made long enough sequences of requests.

This paper is organized as follows. Section 2 presents the algorithms for the construction of path-based models. Section 3 presents the prediction algorithms. Section 4 evaluates the performance of the proposed algorithms. Section 5 discusses related work. Finally, section 6 provides a summary of this work.

2. 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 the next click immediately after the sub-string in all sessions. The maximum occurred request is used as the prediction for the sub-string. The algorithm for building a path-based model on sub-strings is described below.

*Algorithm **PathModelConstruction** (n: length of n-gram; L: log file of sessions; T: table indexed by all n-grams of all sessions in L)*

Begin

L := Filter(L); // we will explain how to filter log file L later;

T[] := 0; // Initialize Table T to zero for all n-grams

H[] := 0; // result of the model is stored in hash table H(), indexed on n-grams

Max[] := 0 // Max[] records the maximum count for each n-gram

For i:= 1 to |L| Do

S := L[I];

```
For j:=1 to |S| do
  If (|S| - j)>n Then
  // find a sub-string of length n starting at alphabet j
     P := sub-string （S, j, n）;
      C := sub-string(S,j+1,1); // find the next click
      T[P,C] := T[P,C] + 1;
      If T[P,C] > Max[P] Then
        H[P] := C;
      End If
  End If
 End For
End For
Return H[];
End
```

In this algorithm, Filter(L) 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 Filter(L). Based on our empirical tests to be discussed in Section 5, it is reasonable to set θ to 10 times or less in web server logs.

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

A,B,C,D
A,B,C,F
A,B,C,F
B,C,D,G
B,C,D,G
B,C,D,F

If we were to construct a 3-gram model, we have two 3-grams to build our prediction model on. These are

A,B,C; B,C,D

Our application of the algorithm returns the following hash table H3():

N-Gram	Prediction
A,B,C	F
B,C,D	G

However, if we were to build a 2-gram model, then we have the following 2-grams to contend with:

A,B; B,C; C,D

Based on the log data, we can build the following 2-gram prediction model H2():

N-Gram	Prediction
A,B	C
B,C	D
C,D	G

3. 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 Hi() be the prediction model built on i-gram model. Our algorithm is as follows:

```
Algorithm n-gram+ (P: user's current clicking sequence;
n: minimal path length)
Begin
  For i:= |P| downto n do
    If P is an index in hash table Hi Then
        Prediction := Hi[P];
        Return (Prediction);
    End If
    P := the same sequence with the first element removed;
  End For
  Return("No Prediction");
End
```

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

```
Algorithm n-gram (P: user's current clicking sequence)
Begin
  If ((|P| >= n) and (P is an index of Hn[])) Then
        Prediction := Hn[P];
        Return (Prediction);
  End If
  Return("No Prediction");
End
```

As an example, assume that we have built up 3-gram and 2-gram models as H3 and H2 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 H3 first to see if an index "DBC" exists. It finds out that the index does not exist. Therefore, it checks the 2-gram model H2 for the index "BC", which exists, thus the predicted next click is "D", according to H2.

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 Hn[]:

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

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

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.

4. 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.

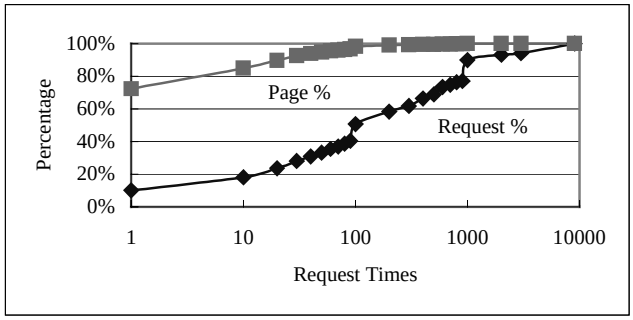

Figure 1. Page vs. request percentage for Monash University data set

One important piece of information about the server data is revealed in Figure 1. 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. 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.

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

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

We have also repeated the page vs. request ratio for two more data sets, NASA and MSN.com data sets, as shown in Figures 2 and 3. The NASA data set contains two months worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida. The log was collected from 00:00:00 August 1, 1995 through 23:59:59 August 31, 1995. In this period there were 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. 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.

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. This 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. These models are stored in memory. To give an indication of their sizes, for the Monash University log our n-gram table sizes are shown in Table 1.

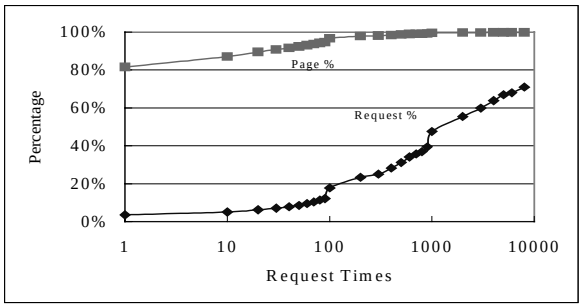

Figure 2. Page vs. request percentage for NASA data set

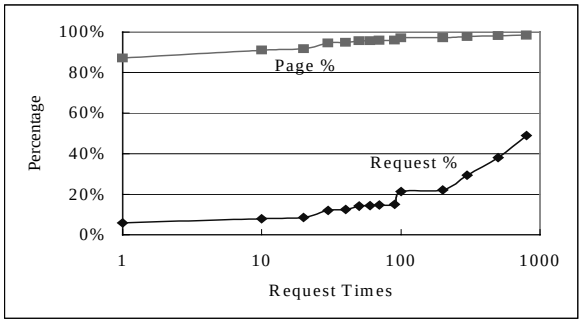

Figure 3. Page vs. request percentage for MSN.com data set

Figures 4 and 5 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 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.

Table1. Hash table sizes for implementing n-gram models

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

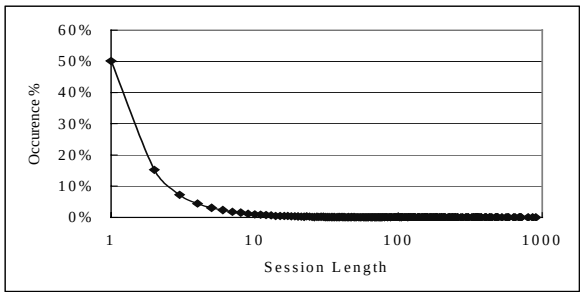

Figure 4. Session length distribution for Monash University data

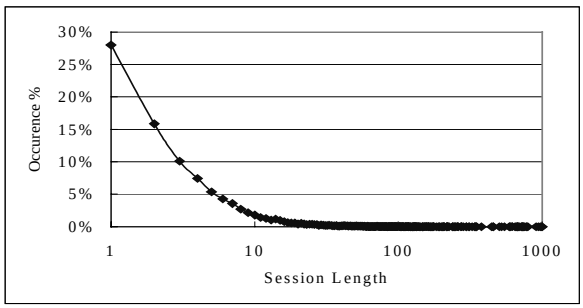

Figure 5. Session length distribution for NASA data

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 6 shows the precision for the Monash University data and Figure 7 shows the applicability 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**(n). 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, setting m to 3 in **n-gram+(m)** 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. We have also applied the same training and testing to NASA log data as shown in Figures 8 and 9. The results confirm similar conclusions.

Finally, we took 10% of the NASA log and run a comparison of n-gram models (**n-gram**(n)) for n between one and seven. The results of precision and applicability are shown in Figure 12. As we can see, as the order of n-gram model increases, the precision is increasing linearly while applicability decreases linearly as well. However, we can see that the decrease in applicability is faster than the increase of precision, indicating an upper limit in which n-gram models should be applied.

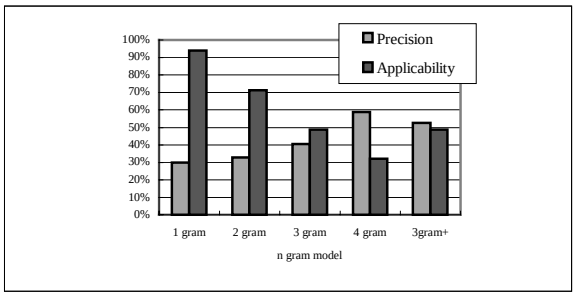

Figure 6. Precision and Applicability as a function of session lengths for Monarsh University data log. n-grams (n=1, 2, 3, 4) represent precision as recorded for sessions having length greater than n

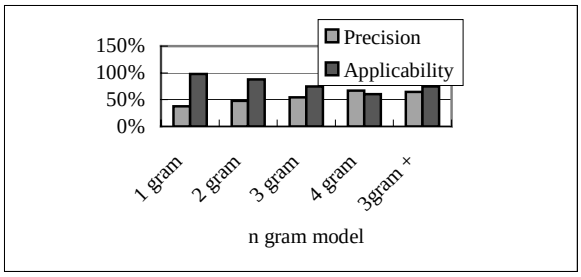

Figure 7. Precision and applicability as a function of session lengths for NASA data log

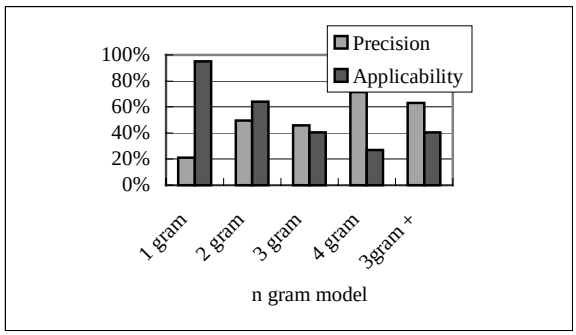

Figure 8. Precision and applicability as a function of session lengths for MSN.com data log

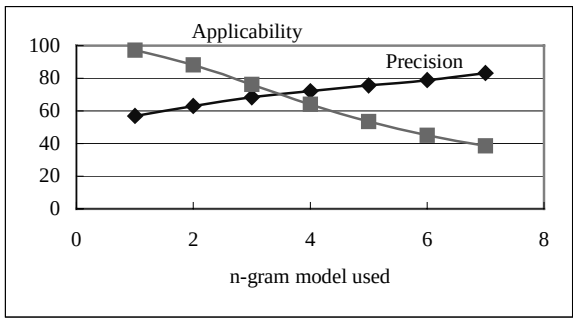

Figure 9. Comparing precision and applicability of different n-gram models for a portion of the NASA data

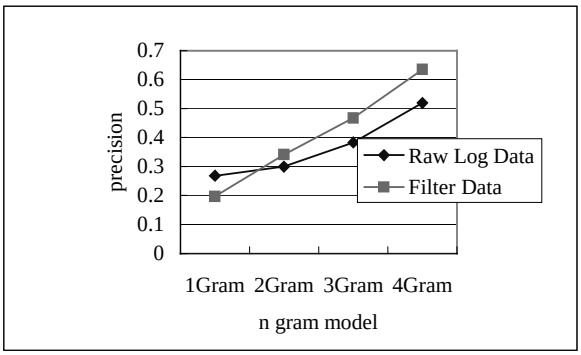

Figure 10. Comparing precision of using raw data and filtered data of different n-gram models for NASA Data

5. RELATED WORK

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. Compared to their work, we focused on the comparison of n-gram models for different n. We also applied a cascading n-gram model (n-gram+) over three realistic web server logs and show that the prediction techniques hold valid for the web domain.

The availability of 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 next action 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 action 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 [8] 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 work focuses on utilizing the server log that contains the users' browsing actions rather than queries submitted to a search service. In addition, 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. The work of [14][15] used similar techniques but did not compare the effectiveness of different n-gram models as we did in this work.

Compared to previous research, our main contributions are:

1. We compared the effectiveness of n-gram prediction for different sequence length n, and found that with an increase in sequence length, there is an increase in precision and decrease in applicability.
2. In response, we formulated a cascading model n-gram+ by including successively lower order models in prediction, and obtained a good balance between predictability and applicability.
3. We preformed experiments on three very different server logs (commercial, university and governmental), and found that our prediction algorithm indeed performed well.

6. Conclusions and Future Work

Our work is aimed at showing that using simple n-gram models for n greater than two will result in significant gain in prediction accuracy while maintaining reasonable applicability. Our results show that for n-gram based prediction when n is greater than three gives a precision gain on the order of 10% or more for the three 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 a 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. In our future work, we wish to apply this algorithm to these domains.

Acknowledgments

We thank Jing Han for her initial involvement in this work, Susan Dumais and Eric Horvitz for their timely feedback on this work. We also 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.

7. 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*