PREDICTING WEB REQUESTS EFFICIENTLY USING A PROBABILITY MODEL

Shanchan Wu, Wenyuan Wang
Department of Automation, Tsinghua University, Beijing 100084, P.R.C.

Keywords: data mining, web mining, probability model, prediction

Abstract: As the world-wide-web grows rapidly and a user's browsing experiences are needed to be personalized, the problem of predicting a user's behavior on a web-site has become important. In this paper, we present a probability model to utilize path profiles of users from web logs to predict the user's future requests. Each of the user's next probable requests is given a conditional probability value, which is calculated according to the function presented by us. Our model can give several predictions ranked by the values of their probability instead of giving one, thus increasing recommending ability. Based on a compact tree structure, our algorithm is efficient. 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. The experiments show that our model has a good performance.

1 INTRODUCTION

Data mining is a process of discovering implicit and useful knowledge from large datasets [Fayyad, et al, 1996]. It is first motivated by the decision support problem faced by many large retail organizations. Now with the development of information technology revolution, we have witnessed an explosive growth in the information available on the World Wide Web (WWW).

Web mining is the application of data mining technology to huge Web data repositories. Basically, there are two domains that pertain to Web mining: the content mining and Web usage mining. The former is the process of extracting knowledge from the content of Web sites, and the latter, also known as Web log mining, is the process of extracting interesting patterns in Web access logs. The purpose of this paper is to explore ways to exploit the information from web logs for predicting users' actions on the web.

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 users next actions based on their previous ones. Almost all of them only give one best guess when given a sequence of previous requests. In this paper, we present a probability model to predict the user's next request. From a sequence of previous requests, instead of giving only one prediction, we can give several predictions ranked by the values of their probability. We present a function to calculate the values of their conditional probability and present an efficient algorithm to implement it. Our algorithm is based on the compact web access pattern tree (WAP-tree) (J.Pei, 2000) structure.

We discuss related work in section 1.1 and discuss preprocessing task in section 2. We describe construction of WAP-tree (J.Pei, 2000) structure in section 3, and introduce our probability model and describe the algorithm of prediction in section 4. In section 5, we evaluate our experiments and provide a summary of this work.

1.1 Related Work

WebWatcher(T. Joachirms, 1997), acts like a web tour guide assistant, it guides the user along an appropriate path through the collection based on the past experiences of the visitor. It accompanies users from page to page, suggests appropriate hyperlinks and learns from experience to improve its advice giving skills.

Syskill & Webert (M. Pazzani, 1996) is designed to help users distinguish interesting web pages on a particular topic from uninteresting ones.

WebTool, an integrated system (F. Masseglia, 1999), is developed for mining either association rules or sequential patterns on web usage mining to

Wu S. and Wang W. (2004).
PREDICTING WEB REQUESTS EFFICIENTLY USING A PROBABILITY MODEL.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 48-53
DOI: 10.5220/0002622000480053

provide an efficient navigation to the visitor. When the navigation matches a rule, the hypertext organization of the document requested is dynamically modified.

WhatNext (Z. Su, 2000) is focused on path-based prediction model inspired by n-gram prediction models commonly used in speech processing communities. The algorithm build is n-gram prediction model based on the occurrence frequency. Each sub-string of length n is an n-gram. 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.

In (R. R. Sarukkai, 2000), the authors proposed to use Markov chains to dynamically model the URL access patterns that are observed in navigation logs based on the previous state. In (Xing Dongshan, 2002), a new a new Markov model is presented.

Another prediction system proposed in (J. Pitkow, 1999) is based on the assumption of mining longest repeating subsequences to predict surfing. In (Xin Chen, 2003), the authors use a popularity-based prediction model for web pre-fetching.

2 PREPROCESSING

There are three main tasks for performing Web Usage Mining or Web usage Analysis: Preprocessing, Pattern Discovery, Pattern Analysis (J.Srivasta, 2000). As Preprocessing is the first step for our task and it is very important, we discuss it in this session.

Preprocessing consist of converting the usage, content, and structure information into the data abstractions necessary for pattern discovery. Typically, only the portion of each user session that is accessing a specific site can be used for analysis, since the access information is not publicly available from the vast majority of Web servers. There are two ways (KhuResearch) to identify server sessions:

(1) IP address associated with a time range:

We assume that from this period of the time, all accesses from a specific machine (unique IP address) belong to a specific user.

(2) Cookie associated with a time range:

After reconfigure the server, whenever a new visitor comes in, he/she will be set with a cookie. When he/she revisits the server, we could identify him/her by the cookie.

A thirty minute timeout is often used as the default method of breaking a user's click-stream into sessions. Clearing the useless information in the Web logs is needed.

After preprocessing is applied to the original Web log files, pieces of Web logs can be obtained. Each piece of Web log is a sequence of events from one user or session in timestamp ascending order, i.e. the events happened early goes before the events happened late. We use a single letter or a single letter with a number subscript to denote one event, and a sequence of event can be denoted as a sequence of letters or letters with number subscripts. For example, let E be a set of events. A Web log piece or (Web) access sequence $S = e_1 e_2 \cdots e_n$ $(e_i \in E)$ for $(1 \le i \le n)$ is a sequence of events, while n is called the length of the access sequence. Because one can access the same web page more than one time during a single access to the web site, it is not necessary that $e_i \neq e_j$ for $(i \neq j)$ in an access sequence S.

3 CONSTRUCT THE WAP-TREE

As our algorithm is based on the compact web access pattern tree (WAP-tree) (J.Pei, 2000) structure, in this section, we introduce how to construct a WAP-Tree, which is previously presented by J.Pei et al. Due to different purpose, our WAP-Tree has some difference. In order not to lose information, we do not do any truncation to WAP-Tree during construction.

The compactness of the WAP-Tree comes from the fact that if two access sequences share a common prefix P, the prefix P can be shared in the WAP-Tree.

The WAP-Tree can be defined as follows (J.Pei, 2000). The only difference from (J.Pei, 2000) is that we use the complete sequences.

1. Each node in a WAP-Tree registers two pieces of information: label and count, denoted as label: count. The root of the tree is a special virtual node with an empty label and count 0. Every other node is labeled by an event in the event set E, and is associated with a count which registers the number of occurrences of the corresponding prefix ended with that event in the Web access sequence database.

2. The WAP-Tree is constructed as follows: for each access sequence in the database, insert them into WAP-Tree. The insertion of sequences is started from the root of WAP-Tree. Considering the first event, denoted as e, increment the count of child node with label e by 1 if there exist one; otherwise create a child labeled by e and set the count to 1. Then, recursively insert the rest of the sequence to the sub tree rooted at that child labeled e.

3. Auxiliary node linkage structures are constructed to assist node traversal in a WAP-Tree as follows. All the nodes in the tree with the same

label are linked by shared-label linkages into a queue, called event-node queue. The event-node queue with label e_i is also called e_i-queue. There is one header table H for a WAP-Tree, and the head of each event-node queue is registered in H.

The specific process of the algorithm of construction of WAP-Tree can be referred in (J.Pei, 2000). We can also build our WAP-tree from frequent sequences which are mined from original sequences with minimum support. This method will reduce applicability (applicability is defined in session 5), but it will also increase precision and reduce the time consumed in the predicting process. We will discuss this in session 5.

4 PREDICT FUTURE REQUESTS

Using the previous requests to predict the next can be treated as a problem to find the maximum conditional probability. Let E be the set of web pages, and $e_1 e_2 \cdots e_{n-1}$ be the previous requests. Our target is to find e_m which satisfy the following equation:

$$p(e_m \mid e_{n-1} e_{n-2} \cdots e_1) = Max\{p(e_i \mid e_{n-1} e_{n-2} \cdots e_1)\}, \quad e_i \in E$$

Where $p(e_i \mid e_{n-1} e_{n-2} \cdots e_1)$ is the conditional probability of request for page e_i at the next step. Suggest the user will request e_n at the next step. It is not guaranteed that e_n equals e_m, but for all pages in E, from the above equation, e_n is most probably equal to e_m.

Practically, it is not easy to find the ideal maximum probability $P_{\max}(e_i)$ for a certain user, because there are many facts which affect the probability and some of which are difficult to get and describe in math form. For example, different user has different interests and so requests in different mode, and for different request time, the mode will also alter. Here we use the user's previous requests in the same session to predict the next request, without taking personal information into account, which is needed for Syskill & Webert (F. Masseglia, 1999). Comparing to well known WebWatcher (T. Joachirms, 1997), we also do not require the web site link structure. We only use the logs of web site. This greatly simplifies the process of prediction and without losing much accuracy, and even sometimes may increase it.

What we need to do is to find the maximum conditional probability $Max\{p(e_i \mid e_{n-1} e_{n-2} \cdots e_1)\}$ from the web logs. How to find $Max\{p(e_i \mid e_{n-1} e_{n-2} \cdots e_1)\}$ is the problem we will discuss below.

4.1 Model of Prediction

The previous requests $e_1, e_{n-2}, \cdots, e_{n-1}$ have different influences to the prediction of the future request e_n. We suppose that the most recent request has the strongest influence, and in most the fact is so. This assumption is useful for prediction.

To extend this, we give a coefficient to weigh the influence of every item in the user's previous request. Let $B = b_1 b_2 \cdots b_{n-1} b_n$ denote the n-sequence gotten from web logs, f_B denote the support of sequence B, $e_1, e_{n-2}, \cdots, e_{n-1}$ denote the user's previous request, C_k denote the coefficient called weight here of k's event in the user's previous requests, and Ω_i denote the collection of all n-sequences in web logs which satisfy $b_n = e_i$. In addition, we define $\omega(b_k)$ as following:

$$\omega(b_k) = \begin{cases} 0 & \exists b_j \neq e_j, j \geq k \\ 1 & b_j = e_j, \forall j \geq k \end{cases}$$

We calculate the probability value of e_i to appear at the next step by the following equations:

$$Q(e_i \mid e_{n-1} e_{n-2} \cdots e_1) = \sum_{B \in \Omega_i} \left\{ \frac{f_B}{f_B + M} \cdot \left(\sum_{k=1}^{n-1} \omega(b_k) C_k \right) \right\}$$

M is a constant (1)

$$P(e_i \mid e_{n-1} e_{n-2} \cdots e_1) = \frac{Q(e_i \mid e_{n-1} e_{n-2} \cdots e_1)}{\sum_{k \in E} Q(e_k \mid e_{n-1} e_{n-2} \cdots e_1)} \quad (2)$$

We use the following rule to C_k.

$$C_{k-1} = \alpha \cdot C_k \qquad k = 2,3,\cdots,n-1 \ ,$$

α is a constant. (3)

Function (3) guarantees that no matter what C_{n-1} is, the result will not change if only α is the same. Simply, we set $C_{n-1} = 1$, then $C_k = \alpha^{n-1-k}$.

In order to find the maximum value of $P(e_i \mid e_{n-1} e_{n-2} \cdots e_1)$, We only need to find the maximum value of $Q(e_i \mid e_{n-1} e_{n-2} \cdots e_1)$. We calculate $Q(e_i \mid e_{n-1} e_{n-2} \cdots e_1)$ efficiently basing on the WAP-tree (J.Pei, 2000). For convenient statement, we also mark $Q(e_i \mid e_{n-1} e_{n-2} \cdots e_1)$ as Q_{e_i}. We give an example to illustrate our predicting process below and then in the session 4.2 we will generalize our algorithm.

Example 1. Let $\{a, b, c, d, e, f\}$ be a set of events. From the Web log, we get the Web access sequence as table 1.

From the above Web access sequences, we construct the WAP-tree (J.Pei, 2000) as figure 1.

User ID	Web Access Sequence	User ID	Web Access Sequence
1	abcf	7	fbce
2	abcea	8	bca
3	abcf	9	fbfb
4	abcea	10	fbfb
5	abcea	11	fbfb
6	bca		

Table 1: A database of Web access sequences.

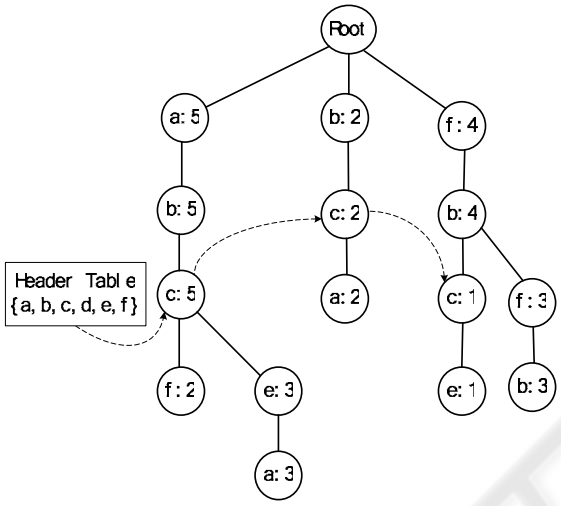

Figure 1: The WAP-tree in Table 1

Suppose a user request the web site in the sequence abc, and we want to predict the next request. Based on function (1), if a sequence in the web tree hasn't the node c, it has no contribution to Q value. This reduces much of the work of calculation. Let α in function (3) equal 2, and M in function (1) equal 3. Then the process of calculating Q in function (1) is as follows.

First, from the header table of the WAP-tree, all of the nodes with c label in the tree is easily reached, which are recorded one after one in the Header Table. The first node labeled c we get is (c: 5). Its parent node (b: 5) is labeled *b* which is the same as the second event of sequence abc. Moreover, the parent node of node (b: 5) is labeled *a* which is the first event of sequence abc. So $\sum_{k=1}^{n-1}\omega(b_k)C_k=\sum_{k=1}^{3}C_k=1+\alpha+\alpha^2=7$. We create a set *L* to hold the optional events. Each node in the set has its event label and the event's *Q* value. The child nodes of (c: 5) are (f: 2) and (e: 3). Now since event f and event e haven't existed in the set *L*, two new nodes are created into the set labeled f and e respectively. According to function (1), set their *Q* values as follows (the Q values would probably be updated during the future steps):

$$Q_f = 0 + \frac{f_f}{f_f + 3}\left(\sum_{k=1}^{3} C_k\right) = \frac{2}{2+3}\cdot 7 = 2.8$$

$$Q_e = 0 + \frac{f_e}{f_e + 3}\left(\sum_{k=1}^{3} C_k\right) = \frac{3}{3+3}\cdot 7 = 3.5$$

The second node labeled c in the Header Table of the WAP-tree is (c: 2). Its parent node (b: 2) is labeled b which is the second event of sequence abc. In addition, the parent node of node (b: 2) is the root of the tree without label a. So in the function (1), $\omega(b_3)=\omega(c)=1, \omega(b_2)=\omega(b)=1, \omega(b_1)=0$, $\sum_{k=1}^{n-1}\omega(b_k)C_k = 1+\alpha = 3$. There is only one child node of node (c: 2). The label of the child node is a, which does not exist in the set *L*. Then a new node labeled *a* is created into the set *L*. Its *Q* value is set as follow (the *Q* value would probably be updated during the future steps):

$$Q_a = 0 + \frac{f_a}{f_a + 3}\left(\sum_{k=1}^{n-1} \omega(b_k) C_k\right) = \frac{2}{2+3}\cdot 3 = 1.2$$

The last nod labeled c in the header table is (c: 1). Its parent node is (b: 4) and the parent node of (b:4) is (f:4) which label is not the same as the first event in the sequence abc. Similar to the above we can get $\sum_{k=1}^{n-1}\omega(b_k)C_k = 1+\alpha = 3$. The node (e: 1) is the only child node of node (c: 1). As the label e has existed in the set *L*, a new node isn't needed to created. What is needed to do is just to update the *Q* value of node e in the set *L*.

$$\Delta Q_e = 0 + \frac{f_e'}{f_e' + 3}\left(\sum_{k=1}^{n-1} \omega(b_k) C_k\right) = \frac{1}{1+3}\cdot 3 = 0.75$$

$$Q_e' = Q_e + \Delta Q_e = 3.5 + 0.75 = 4.25$$

Q_e' is the new value of Q_e. Set $Q_e = 4.25$ to the node e in the set L.

Now in the set *L* there are three nodes {f, Q = 2.8}, {e, Q = 4.25}, {a, Q = 1.2}. The maximum value of *Q* is 4.25. Then we predict the user's next request would be event (page) e.

4.2 Prediction Algorithm

Algorithm 1 (Predicting users' future requests with WAP-tree by calculating the values of conditional probability)

Input: a WAP-tree constructed from web logs, a user's previous request sequence $e_1, e_{n-2}, \cdots, e_{n-1}$, constant M and α, and number n.

Output: the n events of the user's most probable requests at the next step.

Method:

(1). Initialize optional events set $L = \phi$.

(2). Following e_{n-1}'s node-queue of the Header Table of the tree., for each node (marked as Θ) in the e_{n-1}'s node queue,

(a) initialize $\lambda = 0$, $\beta = 1$, node $\Theta' = \Theta$, event $e' = e_{n-1}$.mark the parent node of node Θ' as *parent* (Θ'), and mark the event exactly before event e' as *parent* (e').

(b) following node Θ's links in the tree from child to parent.

while (the label of node Θ' is the same as event e')

$$\lambda \leftarrow \lambda + \beta,\ \beta \leftarrow \beta \cdot \alpha$$

$$\Theta' \leftarrow parent(\Theta'),\ e' \leftarrow parent(e')$$

end **while**

(c) for each child node of node Θ,

mark the count value of the child node as f, then

$$\Delta Q = \frac{f}{f + M} \cdot \lambda$$

If the label of the child node hasn't existed in the optional events set L, create a new item with the label that is the same as that of the child, and set it's $Q \leftarrow \Delta Q$.

Otherwise, update Q value of the item with the same label in the set L, $Q \leftarrow Q + \Delta Q$.

(3) For all items in the optional set L, select the n top ones with largest Q values, return their labels, which denote the events.

The algorithm shows that we only need to scan part of the tree once. In addition, the tree isn't needed to be constructed again when making another prediction. Generally, we can put the tree in the memory, and according to the algorithm of constructing WAP-tree, it can be updated easily. We just need to put the user's new sequence to the tree according to the rule of construction.

5 EXPERIMENTAL EVALUATION AND CONCLUSIONS

In the evaluation of the algorithm, we use the following measures. Let $S = \{S_1 S_2, \cdots, S_n\}$ be the set of sequences in a log file. We build WAP-tree Models on a subset of these sequences, known as the training sequences. The remaining is used as testing sequences. If the returned n top events from the algorithm contain the factual next request, we say that it is a correct prediction. Let P^+ be correct predictions, P^- be set of incorrect predictions, and R be the set of all requests. For some requests our algorithm can't give prediction (in the case that in the end the optional set L is empty), so normally $P^+ + P^- \neq |R|$. We use the following measures (Z. Su, 2000):

$$precision = \frac{P^+}{P^+ + P^-},\ applicability = \frac{P^+ + P^-}{|R|}$$

We use Microsoft anonymous web data as experimental data. The data records the use of www.microsoft.com by 38000 anonymous, randomly-selected users visiting in a one-week timeframe in February 1998. It can be downloaded from the website (MSWeb).

As we use the original sequences to build our WAP-tree, almost all the testing requests can be given predictions. In other words, in the end the optional set L isn't empty. So in our experiments, *applicability* = 1. If our WAP-tree is built from frequent sequences, then *applicability* will decrease, but *precision* will increase.

As *applicability* = 1, here we only need to analyze *precision*.

Let $M = 50$, $\alpha = 2$, and we change parameter n which denotes the returning quantity of predicting events from 1 to 10. The value of *precision* percentage is shown as figure 2:

Further, we get different events predicting ability as figure 3. Figure 3 explains that for the n returned events, the greater Q value of one event, the greater probable precision the event can be. This means that Q can approximately represent the actual conditional probability.

As we can see, the event with the greatest Q value still hasn't very great precision percentage. There are some reasons. The important one is that, different users have different request sequences, and even the same user in the different time the request sequence will also change. We can reduce

applicability to increase precision. As we say previously, we can first mine the web logs to collect frequent sequences with certain minimum support

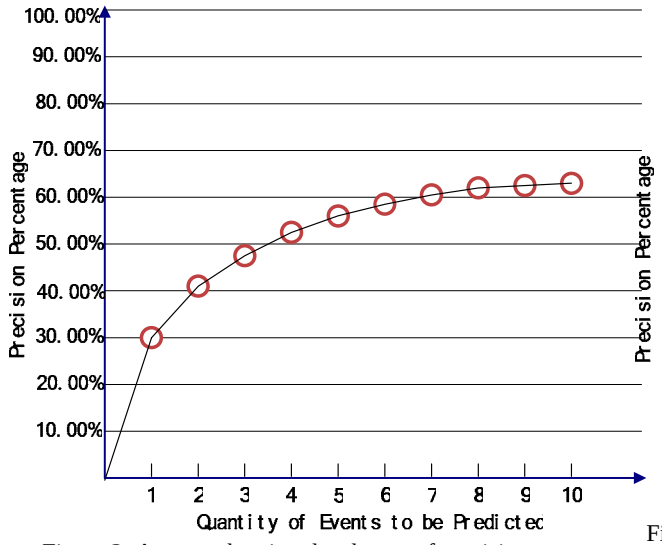

Figure 2: A curve showing the change of precision according to different quantity of returning events.

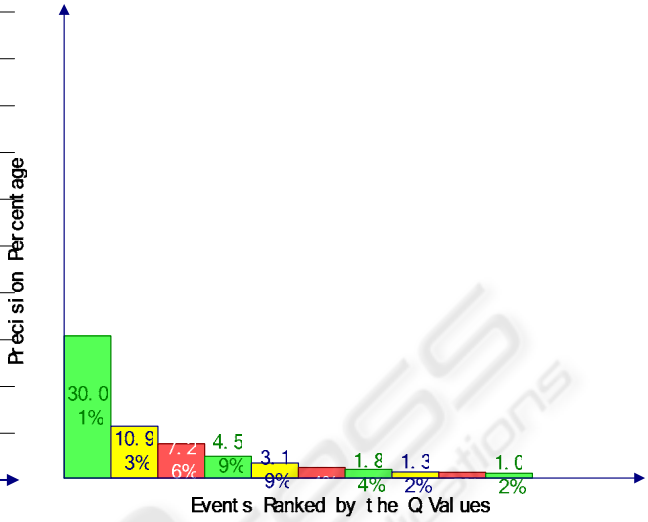

Figure 3: Precision of top events ranked by Q values. Each histogram represents the precision percentage of the event with a certain rank.

and build WAP-tree from the frequent sequence instead of from original sequences. This method not only increase precision, but also decrease the time consumed in prediction. One shortage of this method is that for some previous sequences, it can not give predictions (optional set in our algorithm is finally empty), or we say that applicability will decrease to less than 1. There exists contradiction between applicability and precision. To let applicability equal 1 or approximately equal 1 and still need high precision, we can use n top ones instead of the greatest one. In some cases, we need to compromise between *precision* and *applicability*. This is very useful in recommendation systems.

REFERENCES

J.Pei, J.Han, H.Zhu and B.Mortazavi-asl (2000, April). Mining Access Patters Efficiently from Web Logs. In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD'00), 396-407.

J.Srivasta, R.Cooley, M.Deshpande and P.Tan (2000). Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. In SIGKDD Explorations, 1(2).

KhuResearch, http://www.cs.umbc.edu/~khu1/research/.

T. Joachirms, D. Freitag and T. Mitchell (1997, August). WebWatcher. A Tour Guide for the World Wide Web. In Proceedings of 15th International Joint Conference on Artificial Intelligence, Morgan Kaufrnann, 770-775.

M. Pazzani, J. Muramatsu and D. Billsus (1996). Syskill&Webert: Identifying interesting web sites. In Proceedings of the 13th National Conference on Artificial Intelligence, Portland, OR.

F. Masseglia, P. Poncelet and M. Teisseire (1999, October). Using Data Mining Techniques on Web Access Logs to Dynamically Improve Hypertext Structure. In ACM Sib Web Letters, 8(3), 13-19.

Z. Su, Q. Yang H. Zhang (2000, June). WhatNext: A Prediction for Web Requests using N-Gram Sequence Models. In First International Conference on Web Information Systems and Engineering Conference, HongKong.

R. R. Sarukkai (2000). Link Prediction and Path Analysis Using Markov Chains. In the 9th International WWW Conference.

J. Pitkow and P. Pirolli (1999, April). Mining Longest Repeating Subsequences to predict World Wide Web Surfing. In Proceedings of the 1999 USENIX Technical Conference.

MSWeb,http://kdd.ics.uci.edu/databases/msweb/msweb.html.

Xing Dongshan, Shen Junyi (2002). A new Markov model for Web access prediction. Computing in Science & Engineering [see also IEEE Computational Science and Engineering], 4(6), 34 -39.

Xin Chen; Xiaodong Zhang (2003, March). A popularity-based prediction model for Web prefetching. Computer, 36(3), 63-70.