A New Markov Model for Web Access Prediction

Accurately predicting Web user access behavior can minimize user-perceived latency, which is crucial in the rapidly growing World Wide Web. Although traditional Markov models have helped predict user access behavior, they have serious limitations. Hybrid-order tree-like Markov models predict Web access precisely while providing high coverage and scalability.

The World Wide Web is a large, distributed hypertext repository of information, which users navigate through links and view with browsers. The heavy Internet traffic resulting from the Web's popularity has significantly increased user-perceived latency. The obvious solution—to increase the bandwidth—is not viable, because we cannot easily change the Web's infrastructure (the Internet) without significant economic cost. However, if we could predict future user requests, we could put those pages into the client-side cache when the browser is free. When a user requests one of the pages, the browser could retrieve it directly from cache.

Much current research has examined modeling and predicting user access behavior on the Web to improve Web prefetching,[1,2] enhance search engines,[3] and understand and influence buying patterns.[4] To predict Web access, we need a method for modeling and analyzing Web access sequences. With this information, we can deduce future user requests.

Some researchers have used traditional Markov models, which are often employed to study stochastic processes and predict user access behavior.[5,6] In general, they use the sequence of Web pages a user has accessed as input, with the goal of building Markov models with which they can predict the page the user will most likely access next. Venkata N. Padmanabhan and Jeffrey C. Mogul used N-hop Markov models to improve prefetching strategies for Web caches;[2] Ramesh R. Sarukkai used Markov models to predict the next page accessed by the user;[7] and Igor V. Cadez and colleagues used Markov models to categorize user sessions.[8] Peter Pirolli and colleagues, however, tested the performance of different-order Markov models for Web access prediction and found traditional Markov models to be inadequate for this purpose.[5] Therefore, we need a new Markov model for Web access prediction.

Our hybrid-order tree-like Markov model can predict Web access precisely, providing high coverage and good scalability. HTMM intelligently merges two methods: a tree-like structure Markov model method that aggregates the access sequences by pattern matching and a hybrid-order method that combines varying-order Markov models. Performance evaluations comparing our HTMM with traditional Markov models confirm its usefulness.

XING DONGSHAN AND SHEN JUNYI
Xi'an Jiaotong University

Limitations of traditional Markov models

Traditional Markov models predict the next Web page a user will most likely access by matching the user's current access sequence with the user's historical Web access sequences.[2,5] Using the models, researchers compare elements of each historical Web access sequence's maximal prefixes with elements from the same-length suffixes of the user's current access sequence and obtain the historical sequences with the highest probability of matching elements.

The 0-order Markov model is the unconditional base-rate probability $p(x_n) = Pr(X_n)$, which is the page visit probability. The 1-order Markov model looks at page-to-page transition probabilities: $p(x_2 \mid x_1) = Pr(X_2 = x_2 \mid X_1 = x_1)$. The K-order Markov model considers the conditional probability that a user transitions to an nth page given his or her previous $k = n - 1$ page visits: $p(x_n \mid x_{n-1},..., x_{n-k}) = Pr(X_n = x_n \mid X_{n-1} = x_{n-1},..., X_{n-k} = x_{n-k})$.

Lower-order Markov models cannot successfully predict future Web page access because they do not look far enough into the past to correctly discriminate users' behavioral modes.[5] Thus, good predictions require higher-order models. Unfortunately, higher-order models result in high state–space complexity and low coverage.[9] Higher-order model states are different combinations of the actions observed in the data set, so the number of states tends to rise exponentially as the model order increases. This dramatic increase can significantly limit the applicability of Markov models for applications in which fast predictions are critical for real-time performance or for applications with tight memory constraints.[2] Furthermore, many examples in the test set might not have corresponding states in higher-order Markov models, thus reducing their coverage.

Tree-like structure Markov model

As mentioned previously, one drawback of higher-order Markov models for prediction is that their size grows rapidly with their order. A tree-like Markov model (TMM) can solve this problem.

Get historical Web access sequences from Web logs

When users interact with a Web site, Web server logs store data about their behavior. After data cleaning, we can regard Web logs as a

```
Input  Web logs (after data cleaning)
Output Historical Web access sequence database
Order all records by IP/Agent and Time increasingly
    For each IP/Agent do
        Create a new sequence in  $D_s^{k+1}$
        For i = 1 to the number of records of this IP/Agent do
            If (A_Time(T_{i+1}) − A_Time(T_i)) < τ and i <= (k + 1) Then
                Insert the URL of this record into the sequence
            Else
                Create a new sequence in  $D_s^{k+1}$
            EndIf
        EndFor
    EndFor
```

Figure 1. An algorithm for acquiring the historical Web access sequence database.

user access table V defined by a set of properties $A = \{a_1, a_2,..., a_m\}$, which include the user's access Internet protocol (IP) address, viewing time, required page URL, status, and agent. The historical Web access sequences can be archived from V and then stored in some special structures.[10,11]

Firewalls or agent servers can prevent IP addresses from identifying a user. Therefore, as we describe below in Definition 1, we combine IP addresses with agents to identify the user. The agent indicates the operating system and browser software used by the user. Definition 2 describes our method for getting Web access sequences through a timeout, if the time between page requests exceeds a certain limit. We use the algorithm in Figure 1 to retrieve the user's historical Web access sequences from the Web logs.

- *Definition 1: User identification.* An IP–agent pair determines the user ID. We transfer V to V' by grouping the IP address and the agent. $A_k(V_i)$ is the value of property A_k in the ith V'.
- *Definition 2: Web access sequence.* Web access sequence s is a part of V' with the attributes $A_{IP}(V_{i+1}) = A_{IP}(V_i)$, $A_{Agent}(V_{i+1}) = A_{Agent}(V_i)$, and $A_{Time}(V_{i+1}) - A_{Time}(V_i) < \tau$. V_{i+1} and V_i belong to s, and τ is a given maximum interval (generally 30 minutes[10]).

Constructing a TMM

A K-order TMM is a multiplayer tree formed from a historical Web access sequence database D_s^{k+1}, where each node is a visited page, and each branch is the path to the node, as Figure 2
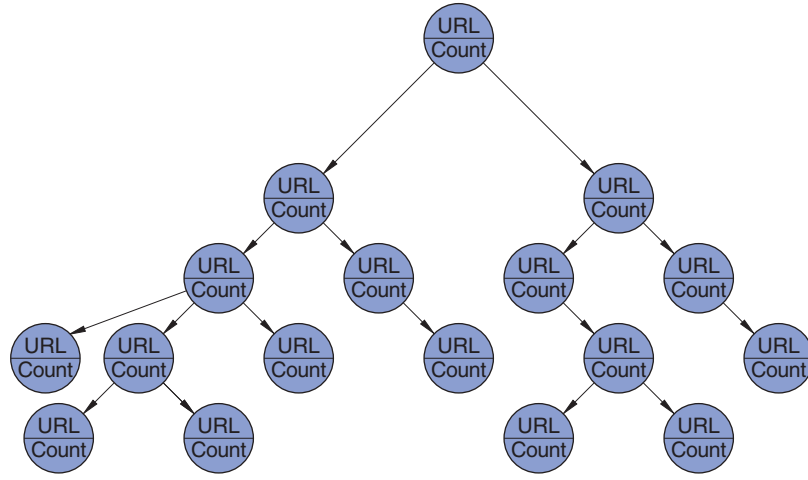
Figure 2. Structure of a tree-like Markov model. In TMM, nodes are visited pages, and branches are paths to nodes.

shows. Each node records the *count*, which is the number of times the user has visited the node along the same route. TMM's major task is to merge Web access sequences into a tier structure, which retains aggregated statistical information. Figure 3 shows our algorithm for constructing a *K*-order TMM.

A *K*-order TMM registers all Web access information. Its height is $(k + 2)$, and its width is no more than the number of sequences in the database D_s^{k+1}. Therefore, a *K*-order TMM cannot generate a large number of nodes.

```
Input  Historical Web access sequence database  D_s^{k+1}
Output   K-order TMM
TMM_Creation (Historical Web access sequence database)
{
For Each sequence i do
       Current = root
       For Each Event j of sequence i do
          If URL of Current node's child nodes = URL of Event j Then
             Count of this child node ++
             Current = this child node
          Else
             Current = Create_New_Node(Current,event j)
          EndIf
       EndFor
EndFor
}
```

Figure 3. Algorithm for constructing a *K*-order tree-like Markov model.

TMMs for predicting user access patterns

Assume the user's current access sequence is $S = u_1 u_2 \ldots u_m$, where $u_i \in$ URLs in a Web site, and for $1 \leq i \leq m$, m is the user access sequence length. When predicting a user's access patterns using a *K*-order TMM, we extract *k* suffixes, $u_{m-k+1} u_{m-k} \ldots u_m$. We call this a *k*-sequence for prediction. Figure 4 shows our algorithm for predicting user access patterns using a *K*-order TMM.

Hybrid-order tree-like Markov model

As we described previously, higher-order Markov prediction models are unlikely to match all access sequences (that is, they offer low coverage), which results in low overall accuracy. To overcome this limitation, we train varying-order Markov models and then combine them for prediction. In this scheme, we use all Markov model orders.

The HTMM offers two methods for combining models: our *accuracy voting* method and the *blending models*[12,13] method used in data compression.

Accuracy voting method

The accuracy voting procedure consists of three steps:

1. Set up 1~*N*-order TMMs from Web logs.
2. Extract 1~*N*-suffixes from the user's current access sequence (that is, 1~*N*-sequence) to the 1~*N*-order TMMs for prediction, re-

spectively. Choose the prediction page with the maximum count as the prediction result, then get the prediction set $PS = \{P_1, P_2, \ldots, P_N\}$, where P_k $(1 \leq k \leq N)$ is the prediction result by K-order TMM.

3. Compute the prediction parameter of the pages in PS using Definition 3. We calculate a prediction weight for each prediction result of different-order TMMs using Definition 4. Order accuracy determines prediction weight. Because the TMM prediction accuracy equals that of the same order traditional Markov model, we can get it from Figure 5 directly.

Table 1 shows the results of our predictions.

- *Definition 3: Prediction parameter.*

$$\text{Prediction}(PageX) = \sum_{i=1}^{N} (S_i \times W_i),$$

where

$PageX \in PS,$

$$S_i = \begin{cases} 1 & (P_i = PageX) \\ 0 & (P_i \neq PageX) \end{cases} (0 \leq i \leq N),$$

P_i $(0 \leq i \leq N)$ is the element in PS, and W_i $(0 \leq i \leq N)$ is the prediction weight of P_i.

- *Definition 4: Prediction weight.*

$$W_k = \frac{Accuracy_k}{\sum_i Accuracy_i},$$

where W_k $(1 \leq k \leq N)$ is the prediction weight of the k-order TMM result, and $Accuracy_k$ $(1 \leq k \leq N)$ is the k-order TMM's prediction accuracy.

Finally, we sort the pages by prediction parameters. For example, given a certain user access sequence, PS $\{A, C, B, D, A, B\}$ obtained by 1~6 order TMMs, we get prediction accuracies of 0.13, 0.19, 0.22, 0.35, 0.49, and 0.65, respectively, from Figure 5. We then compute prediction weight, shown in Table 2, using Definition 3. Definition 4 finds that the prediction parameter of A is 0.31, B is 0.43, C is 0.09, and D is 0.17. Thus, B is most likely to be visited, A is next, then D, and last is C.

Blending model method

We can view blending as a bottom-up recur-

```
Input  k-sequence, K-order TMM
Output Prediction page
TMM_prediction(K-order TMM)
{
      Current = root
      For i = 1 to k
           Url = u_i of k-sequence
           If Url = URL of Current node's child nodes then
                Current = this child node
                Isfind = true
           Else
                Current = NULL
                Isfind = false
           EndIf
      EndFor
      If Isfind = false then
           Output NULL
      Else
           Output the child nodes of Current as prediction pages
      EndIf
}
```

Figure 4. Algorithm for predicting a user's next access using a *K*-order TMM.

Table 1. Prediction results.

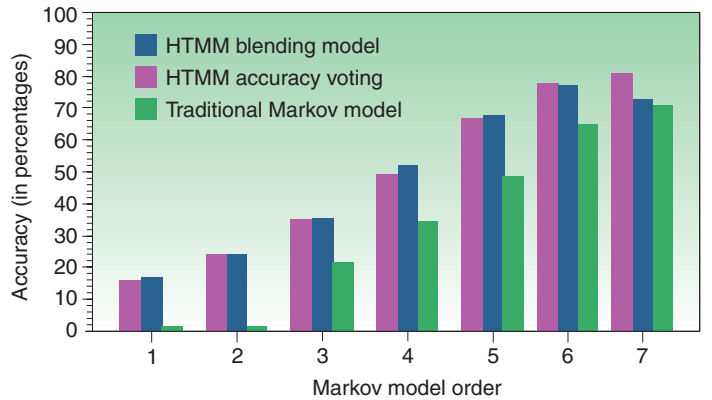| Order	Prediction weight	Prediction page
1	W_1	P_1
2	W_2	P_2
3	W_3	P_3
…	…	…
N	W_N	P_N

Figure 5. Accuracy comparison curves. The two HTMMs are more accurate than traditional Markov models for all model orders.

Table 2. Prediction weight for a sample sequence using accuracy voting.

Order	Prediction weight	Prediction page
1	.07	A
2	.09	C
3	.11	B
4	.17	D
5	.24	A
6	.32	B

sive procedure for computing a *mixture*—barring some missing terms for each level of the recursion—where a mixture is basically a weighted average of several probability estimates.[14] The procedure uses the escape mechanism to combine different orders of prediction models. When high-order predictions are not yet available, the escape mechanism uses the probability that the model will need to escape to a lower-order model and use a shorter sequence for the remaining predictions. John G. Cleary and Ian H. Witten propose two methods for setting escape probabilities: method A and method B.[12] Method A sets the count of an escape event by allocating one count. In method B, the count equals the number of pages that have been predicted so far, and the count of each prediction page is minus one.

We chose method B to implement our algo-

rithm. We do not explicate the algorithm here because other work has fully described its implementation.[12–14]

Performance evaluation

To test performance, we compared the prediction results of our HTMM with the results of traditional Markov models.[5] Between noon on 19 February 2001 and noon on 18 April 2001, we examined 23,317 records after data cleaning (that is, 23,317 pages visited) from a total of 101,020 records on a 378-page education Web site. We coded all algorithms in Microsoft Visual C++ 6.0 and performed experiments on an X86 733 personal computer with 256 Mbytes RAM.

First, we tested the accuracy of the proposed algorithms. We use 1~7 HTMMs and 1~7-order traditional Markov models to predict Web page accesses. As Figure 5 shows, our HTMM is more accurate than traditional Markov models in all orders.

Of the two HTMM combination methods, accuracy voting is more accurate as the order rises. While the blending model performance also improves as the order rises, it reaches its peak at order six, then deteriorates slightly. Although longer sequences provide more specific predictions, they also are much more likely to fail to give any prediction at all. This causes the blending model to use the escape mechanism more frequently to reduce the sequence length to the point where predictions begin to appear. Each escape operation carries a small penalty in prediction efficiency.

Second, we compared the model-building time and prediction time of the traditional Markov model with the two HTMMs. As Figure 6 shows, the two HTMM combinations have similar model-building time, and, as the order increases, the time increases linearly, while the time for the traditional Markov model increases exponentially. Moreover, the model-building time for the HTMM combination methods is shorter than the time for traditional Markov models regardless of order. As for prediction time, shown in Figure 7, the difference between the traditional Markov model and the two HTMM combination methods is so small we can ignore it.

As a whole, our HTMM is more scalable than traditional Markov models, and it provides efficient modeling and prediction of Web user access behavior.
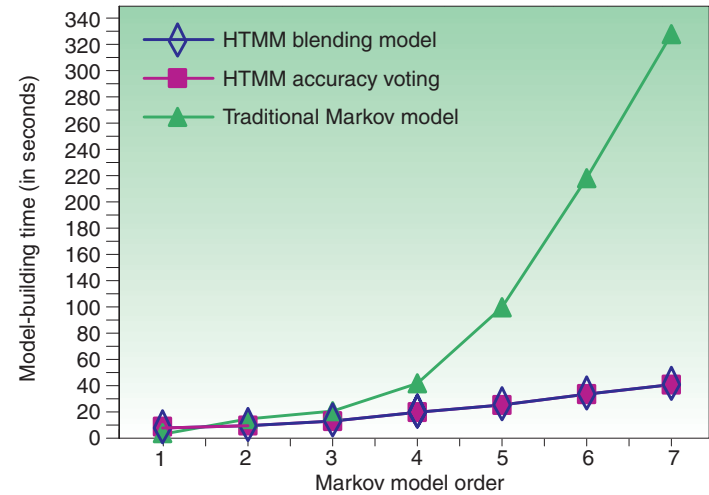

Figure 6. Model-building time comparsion curves. For the two HTMM methods, model-building time increases linearly as the order increases. For the traditional Markov model, model-building time increases exponentially.

W eb access prediction is important for networking, data mining, e-business, and other areas. HTMM is one of several core components of our developing framework for Web usage mining. We plan to perform extensible testing and investigate many possible improvements. Future work will add more factors to our HTMM, such as navigation time and page content, to more accurately reflect user interest. These factors might give us more accurate prediction performance for Web access. Another important research direction for this prediction technology is improvement of the practical application, such as prefetching multimedia content.

Figure 7. Prediction time comparison curves. The difference between prediction times is so small as to be insignificant.

References

1. S. Schechter, M. Krishnan, and M.D. Smith, "Using Path Profiles to Predict HTTP Requests," *Computer Networks and ISDN Systems*, vol. 30, nos. 1–7, Apr. 1998, pp. 457–467.

2. V.N. Padmanabhan and J.C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *Computer Comm. Rev.*, vol. 26, no. 3, July 1996, pp. 22–36.

3. S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. 7th Int'l World Wide Web Conf.*, Elsevier, New York, 1998, pp. 107–117.

4. E. Chi et al., "Visualizing the Evolution of Web Ecologies," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems* (CHI 98), ACM Press, New York, 1998, pp. 400–407.

5. P. Pirolli and J. Pitkow, "Distribution of Surfers' Paths through the World Wide Web: Empirical Characterization," *World Wide Web*, vol. 2, nos. 1–2, Jan. 1999, pp. 29–45.

6. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw Hill, New York, 1991.

7. R.R. Sarukkai, "Link Prediction and Path Analysis Using Markov Chains," *Computer Networks*, vol. 33, no. 1, June 2000, pp. 377–386.

8. I.V. Cadez et al., *Visualization of Navigational Patterns on Web Site Using Model-Based Clustering*, tech. report MSR-TR-0018, Microsoft Research, Redmond, Wash., 2000.

9. M. Deshpande and G. Karypis, *Selective Markov Models for Predicting Web-Page Access*, tech. report no. 00-056, Dept. of Computer Science, Univ. of Minn., Minneapolis, 2000.

10. R. Cooley, B. Mobasher, and J. Srivastava, "Data Preparation for Mining World Wide Web Browsing Patterns," *Knowledge and Information Systems*, vol. 1, no. 1, Feb. 1999, pp. 5–32.
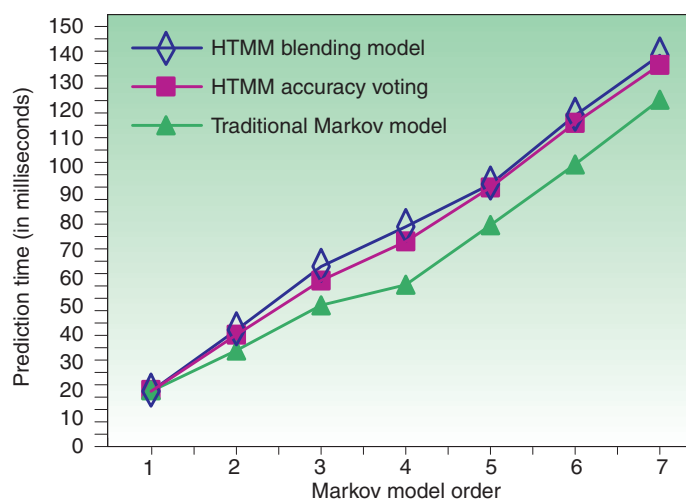
11. J. Pei et al., "Mining Access Patterns Efficiently from Web Logs," *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, Springer-Verlag, New York, 2000, pp. 396–407.

12. J.G. Cleary and I.H. Witten, "Data Compression Using Adaptive Coding and Partial String Matching," *IEEE Trans. Comm.*, vol. 32, no. 4, Apr. 1984, pp. 396–402.

13. J.G. Cleary and W.J. Teshan, "Unbounded Length Contexts for PPM," *Computer J.*, vol. 40, nos. 2–3, Mar. 1997, pp. 67–75.

14. S. Bunton, *A Generalization and Improvement to PPM's Blending*, tech. report UW-CSE-97-01-10, Dept. of Computer Science and Eng., Univ. of Wash., Seattle, Wash., 1997.

Xing Dongshan is currently pursuing a PhD in computer system architecture at Xi'an Jiaotong University. He received BS and MS degrees in automatic control and computer science from Central South University of Technology and Guangxi University, respectively. His research focuses on Web usage mining and Web application systems, especially on user navigation preference, user clustering, and navigation prediction. Contact him at the Inst. of Computer Software, Xi'an Jiaotong Univ., Xi'an 710049, P.R. China; dsxing@hotmail.com or dsxing@163.net.

Shen Junyi is a professor in the Department of Computer Science, Xi'an Jiaotong University. His research focuses on databases, data mining, and software theories. Contact him at the Inst. of Computer Software, Xi'an Jiaotong Univ., Xi'an 710049, P.R. China; jyshen@xjtu.edu.cn.

For more information on this or any other computing topic, please visit our Digital Library at http://computer.org/publications/dlib.