A Web Page Prediction Model Based on Click-Stream Tree Representation of User Behavior

Şule Gündüz
Computer Engineering Department
Istanbul Technical University
Istanbul, Turkey
gunduz@cs.itu.edu.tr

M. Tamer Özsu
School of Computing Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
tozsu@db.uwaterloo.ca

ABSTRACT

Predicting the next request of a user as she visits Web pages has gained importance as Web-based activity increases. Markov models and their variations, or models based on sequence mining have been found well suited for this problem. However, higher order Markov models are extremely complicated due to their large number of states whereas lower order Markov models do not capture the entire behavior of a user in a session. The models that are based on sequential pattern mining only consider the frequent sequences in the data set, making it difficult to predict the next request following a page that is not in the sequential pattern. Furthermore, it is hard to find models for mining two different kinds of information of a user session. We propose a new model that considers both the order information of pages in a session and the time spent on them. We cluster user sessions based on their pair-wise similarity and represent the resulting clusters by a click-stream tree. The new user session is then assigned to a cluster based on a similarity measure. The click-stream tree of that cluster is used to generate the recommendation set. The model can be used as part of a cache prefetching system as well as a recommendation model.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology - classifier design and evaluation

Keywords

Web usage mining, two dimensional sequential model, graph based clustering

1. INTRODUCTION

Web mining is defined as the use of data mining techniques to automatically discover and extract information from Web documents and services [7]. With the rapid growth of the World Wide Web, the study of modelling and predicting a user's access on a Web site has become more important. There are three steps in this process [14]. Since the data source is Web server log data, the first step is to clean the data and prepare for mining the usage patterns. The second step is to extract usage patterns, and the third step is to build a predictive model based on the extracted usage patterns. Fundamental methods of data cleaning and preparation have been well studied [14]. The main techniques traditionally used for modelling usage patterns in a Web site are collaborative filtering (CF), clustering pages or user sessions, association rule generation, sequential pattern generation and Markov models. The prediction step is the real-time processing of the model, which considers the active user session and makes recommendations based on the discovered patterns.

However, the discovery of usage patterns discussed above is not sufficient to accurately describe the user's navigational behavior in a *server session*[1]. An important feature of the user's navigation path is the time that a user spends on different pages [13]. Even the same person may have different desires at different times. The time spent on a page is a good measure of the user's interest in that page, providing an implicit rating for it. If a user is interested in the content of a page, she will likely spend more time there compared to the other pages in her session.

In this paper, we present a new model that uses both the sequences of visiting pages and the time spent on that pages. As far as we know, existing tools for mining two different information types like the order of visited Web pages and the time spent on those pages, are hard to find. Therefore, we concentrate in this study on a model that well reflects the structural information of a user session and handles two-dimensional information.

Our overall approach can be summarized as follows. The user sessions are clustered based on the similarity of the user sessions. When a request is received from an active user, a recommendation set consisting of three different pages that the user has not yet visited, is produced using the best matching user session[2]. For the first two requests of an active user session all clusters are explored to find the one that best matches the active user session. For the remaining requests, the best matching user session is found by explor-

SIGKDD '03, August 24-27, 2003, Washington, DC, USA

[1]The term *server session* is defined as the click stream of page views for a single visit of a user to a Web site [14].
[2]The user session that has the highest similarity to the active user session is defined as the best session.

ing the top-N clusters that have the highest N similarity values computed using the first two requests of the active user session. The rest of the recommendations for the same active user session are made by using the top-N clusters.

The novelty of our approach lies in the method by which we compute the similarity of user sessions and how we cluster them. We propose a method for calculating the similarity between all pairs of user sessions considering both the order of pages, the distance between identical pages, and the time spent on these pages. The distance between identical pages is taken into consideration, because the similarity between two user sessions also reflects the distance between identical pages as measured by the number of user requests between these pages with the same order of occurrence in these sessions. Using these pair-wise similarity values, a graph is constructed whose vertices are user sessions. An edge connecting two vertices in the graph has a weight equal to the similarity between these two user sessions. Using an efficient graph-based clustering algorithm the user sessions are clustered, and each cluster is then represented by a click-stream tree whose nodes are pages of user sessions of that cluster.

This approach to recommendation is novel and unique. The experimental results show that using time as a second dimension to the order of user requests improves the accuracy of the prediction of the next request. Equally important, these results are robust across sites with different structures. Clustering of the sessions enables us to reduce the search space and the time for producing the recommendation set.

The rest of the paper is organized as follows. Section 2 presents the proposed model. Section 3 provides detailed experimental results. In Section 4, we examine related work. Finally, in Section 5 we conclude our work.

2. WEB PAGE RECOMMENDATION MODEL

2.1 Data Preparation and Cleaning

In the experimental part of this research (Section 3), we use two sets of server logs. The first is from the NASA Kennedy Space Center server over the months of July and August 1995 [9]. The second log is from ClarkNet Web server which is a full Internet access provider for the Metro Baltimore-Washington DC area [8]. This server log was collected over the months of August and September, 1995. These are well-known data sets that have been used in other studies. For each log data set we apply the same pre-processing steps. Since the cleaning procedure is beyond the scope of this paper, the details of this procedure are not given here.

In this work, *visiting page times*[3], which are extracted during cleaning procedure, are normalized across the visiting times of the pages in the same session, such that the minimum value of normalized time is 1. For evaluating the effect of the normalization values, we try 4 different maximum values: 2, 3, 5 and 10. If a page is not in the user session, then the value of corresponding normalized time is set to 0. This normalization captures the relative importance of a page to a user in a session. The output of this step is a set of user sessions, where each user session of a length m is in the form of: $\langle t_i,\ (p^1_{t_i}, p^2_{t_i}, ..., p^m_{t_i}),\ (T_{p^1_{t_i}}, T_{p^2_{t_i}}, ..., T_{p^m_{t_i}})\rangle$, where t_i is a unique session number, $(p^1_{t_i}, p^2_{t_i}, ..., p^m_{t_i}) \subset P$ is the ordered user requests in session t_i and $(T_{p^1_{t_i}}, T_{p^2_{t_i}}, ..., T_{p^m_{t_i}})$ is the corresponding normalized time values.

[3]It is defined as the time difference between consecutive page requests.

	P_4	P_1	P_2	P_5	P_3	P_6	-
P_1	2	**3**	0	-3	-4	-4	-4
P_2	-1	0	1	-2	-3	-3	-3
P_4	-1	-2	-1	-1	-2	-2	-2
P_5	-3	-2	-1	0	-2	-1	-1
-	-6	-5	-4	-3	-2	-1	0

Table 1: The scoring matrix for two dimensional sequences

2.2 Session Similarity Measure

In this section, we propose a session similarity measure based on FastLSA [3] sequence alignment method. Since user sessions are ordered URL request, we can refer them as sequences of Web pages. The problem of finding the optimal sequence alignment is solved using a dynamic programming. The algorithm uses a matrix where one sequence is placed along the top of the matrix and the other one along the left side of the matrix. There is gap added to the end of each sequence which indicates the starting point of calculation of similarity score. We shall not go into the details of the algorithm since they are given in [3]. For two user session $\langle t_i, (P_1P_2P_4P_5), (1,2,2,2)\rangle$ and $\langle t_j, (P_4P_1P_2P_5P_3P_6), (1,2,2,2,2,2)\rangle$ the score matrix is given in Table 1. We have implement the algorithm with an additional module that takes into account the time spent on matching pages. We use a scoring system which helps to find the score of the optimal alignment between two sessions. In our implementation the identical matching of web pages is given a score $s(m) = 2$. The two dimensional score is calculated for a pair matching pages, $p^l_{t_i}$ and $p^r_{t_j}$ as follows:

$$s(p^l_{t_i}, p^r_{t_j}) = s(m)\frac{min(T_{p^{(l)}_{t_i}}, T_{p^{(r)}_{t_j}})}{max(T_{p^{(l)}_{t_i}}, T_{p^{(r)}_{t_j}})} \quad (1)$$

Each mismatching or gap inserted to the sequences is given a penalty score of -1.

The similarity between sessions is then calculated such that only the identical matching of sequences has a similarity value of 1. The similarity measure has two components, which we define as *alignment score component* and *local similarity component*. The alignment score component computes how similar the two sessions are in the region of their overlap. If the highest value of the score matrix of two sessions, t_i and t_j, is σ and the number of matching pages is M, then the alignment score component is:

$$s_a(t_i, t_j) = \frac{\sigma}{s(m) * M}$$

The intuition behind this is that the score σ is higher if the sessions have more consecutive matching pages. This value is normalized by the matching score and the number of matching pages. The local similarity component computes how important the overlap region is. If the length of the aligned sequences is L, the local similarity component is :

$$s_l(t_i, t_j) = \frac{M}{L}$$

Then the overall similarity between two sessions is given by

$$sim(t_i, t_j) = s_a(t_i, t_j) * s_l(t_i, t_j) \quad (2)$$

2.3 Pairwise Clustering

A graph is constructed whose vertices are user sessions. There is an edge between two vertices (S_i, S_j) if the similarity value between S_i and S_j computed as described in the previous subsection is greater than 0 and this edge is weighted by this similarity value. The problem of clustering user sessions is formulated as partitioning the graph G into k disjoint subgraphs G_m, $(m \in [1, ..., k])$ by minimizing *Min-MaxCut* function [6]. MinMaxCut function combines both the minimization of similarity between each subgraph and the maximization of similarity within each subgraph and is defined as:

$$minimize \sum_{m=1}^{k} \frac{cut(G_m,\ G - G_m)}{\sum_{v_i, vj \in Gm} sim(v_i, v_j)}$$

where $cut(G_m,\ G - G_m)$is the sum of edges connecting the vertices in G_m to the rest of the vertices in graph $G - G_m$ and $sim(v_i, v_j)$ is the similarity value between vertices v_i and v_j calculated using the similarity metric. In this study an efficient and fast graph partitioning algorithm called Cluto is used for graph partitioning [4].

2.4 Cluster Representation

The clusters created by the graph partitioning algorithm contain user sessions. Each user session in a cluster is a sequence of Web pages visited by a single user and the normalized time spent on those pages with a unique session number. We generate a click-stream-tree for each cluster. Each click-stream tree has a *root* node, which is labelled as "null". Each node except the root node of the click-stream-tree consists of three fields: *data*, *count* and *next_node*. Data field consists of page number and the normalized time information of that page. Count field registers the number of sessions represented by the portion of the path arriving to that node. Next_node links to the next node in the click-stream tree that has the same data field or null if there is any node with the same data field. Each click-stream tree has a *data_table*, which consists of two fields: *data* field and *first_node* that links to the first node in the click-stream tree that has the data field. The tree for each cluster is constructed by applying the algorithm given in Figure 1.

```
1: Create a root node of a click-stream tree, and label it as "null"
2: index ← 0
3: while index ≤ number of Sessions in the cluster do
4:    Active_Session ← t_index
5:    m ← 0
6:    Current_Node ← root node of the click-stream tree
7:    while m ≤ Active_Session length do
8:       Active_Data ← {p^m_{t_index}}-{T^m_{p_{t_index}}}
9:       if there is a Child of Current_Node with the same
         data field then
10:         Child.count++
11:         Current_Node ← Child
12:      else
13:         create a child node of the Current_Node
14:         Child.data = Active_Data
15:         Child.count = 1
16:         Current_Node ← Child
17:      end if
18:      m++
19:   end while
20:   index++
21: end while
```

Figure 1: *Build Click − Stream Tree* **Algorithm**

The children of each node in the click-stream tree is ordered in the count-descending order such that a child node with bigger count is closer to its parent node. The resulting click-stream trees are then used for recommendation.

2.5 Recommendation Engine

The recommendation engine is the real time component of the model that selects the best path for predicting the next request of the active user session. There is a trade-off between the prediction accuracy of the next request and the time spent for recommendation. The speed of the recommendation engine is of great importance in on-line recommendation systems. Thus, we propose the clustering of user sessions in order to reduce the search space and represent each cluster by a click-stream tree. Given the time of the last visited page of the active user session, the model recommends three pages. The most recent visited page of the active user session contains the most important information. The click-stream tree enables us to insert the entire session of a user without any information loss. We not only store the frequent patterns in the tree but also the whole path that a user follows during her session. Besides this, the tree has a compact structure. If a path occurs more than once, only the count of its nodes is incremented. Based on the construction of the click-stream tree, a path $(p_1, p_2, ..., p_k), (T_{p_1}, T_{p_2}, ..., T_{p_k})$ occurs in the tree $d_k.count$ times, where d_k is the data field formed by merging the page request $p^k_{t_i}$ and corresponding normalized time value $T_{p^k_{t_i}}$ of the path.

Figure 2 presents the algorithm for finding the path that best matches the active user sessions. For the first two pages of the active user session all clusters are searched to select the best path (line 3). After the second request of the active user top-N clusters that have higher recommendation scores among other clusters are selected (line 29-31) for producing further recommendation sets (line 5). To select the best path we use a backward stepping algorithm. The last visited page and normalized time of that page of the active user session are merged together to build the data field (line 10). We find from the data_table of the click-stream tree of a cluster the first_node that has the same data field (line 11). We start with that node and go back until the root node (or until the active user session has no more pages to compare) to calculate the similarity of that path to the active user session (line 16-19). We calculate the similarity of the optimal alignment. To obtain the recommendation score of a path, the similarity is multiplied by the relative frequency of that path, which we define as the count of the path divided by the total number of paths ($S[cl]$) in the tree (line 20). Starting from the first_node of the data field and following the next_node, the recommendation score is calculated for the paths that contain the data field in the cluster (line 26). The path that has the highest recommendation score is selected as the best path for generating the recommendation set for that cluster (line 21-24). The first three children nodes of the last node of the best path is used for producing the recommendation

set. The pages of these child nodes are recommended to the active user.

```
1: t_a ← Active User Session
2: if t_a.length ≤ 2 then
3:    Clusters = All Clusters
4: else
5:    Clusters = Top − N Clusters
6: end if
7: for i = 0 to NumberOfClusters do
8:    cl = Clusters[i]
9:    Sim[cl] = 0
10:   d_a ← {p^m_{t_a}}-{T_{p^m_{t_a}}}
11:   Node ← data_table[cl](d_a).first_node
12:   path = null
13:   while Node ≠ null do
14:      path = {path} + {Node.data}
15:      Parent_Node ← Node.Parent
16:      while Parent_Node ≠ null do
17:         path = {path} + {Parent_Node.Data}
18:         Parent_Node ← Parent_Node.Parent
19:      end while
20:      Sim(path) = sim(t_a, path)* Node.count/S[cl]
21:      if Sim(path) > Sim[cl] then
22:         Sim[cl] ← Sim(path)
23:         BestPath[cl] ← path
24:      end if
25:      path = null
26:      Node ← Node.next_node
27:   end while
28: end for
29: if t_a.length = 2 then
30:   Top−N Clusters ← N Clusters with highest Sim[cl]
      values
31: end if
```

Figure 2: *Find Best Path* **Algorithm**

3. EXPERIMENTAL RESULTS

In this research we use two different data sets prepared for experiments as mentioned in Section 2. Approximately 30% of these cleaned sessions are randomly selected as the test set, and the remaining part as the training set.

Given the visiting time of a page in the current session, the model recommends three pages. We define the hit-ratio metric and click-soon metric as proposed in [5] to evaluate our method:

Hit-Ratio: A hit is declared if any one of the three recommended pages is the next request of the user. The hit-ratio is the number of hits divided by the total number of recommendations made by the system.

Click-Soon-Ratio: A Click-Soon is declared if any one of the three recommended pages is requested by the user during the active user session. The Click-Soon-Ratio is the number of click-soon divided by the total number of recommendations made by the system.

For each data set we conduct the experiment with a single click-steam tree, without the use of any clusters of user sessions, to compare the performance of the similarity metric and clustering method. The results obtained by using a single tree (see Table 2) gives us the upper bound of the prediction accuracy. In that case we do not have any side effects of the clustering algorithm or the assumptions we made for assigning the active user session to a cluster since the entire tree is searched (with significant run-time overhead, of course).

Data Set	NT		UT		Time(ms.)
	H-R	CS-R	H-R	CS-R	
NASA	61.61	99.63	59.9	95	3.5
ClarkNet	55.76	100	51.29	92.25	2

Table 2: Results in % of the recommendation algorithm with one tree. (NT = Normalized Time, UT = Unity Time)

We repeat the experiments with different number of clusters changing from 5 to 30 and N ranging from 1 to 3 after the first 2 requests of the user. As mentioned in Section 2 there is a trade-off between the prediction accuracy and the time spent for recommendation. When we determine the top-N clusters after the first request of the user, the recommendation is faster, but the accuracy is about 4% lower. Thus, we determine top-N clusters after the first 2 requests, since the time spent for recommendation and the decrease of accuracy are in an acceptable range. In order to study the results we repeat the same experiments without considering the normalized time (Unity Time). These experiments show that normalizing time between 1 and 2 improves the prediction accuracy. The same experiments are then performed by normalizing time between 1 and 3, 1 and 5, and 1 and 10. In the case when the time is normalized between 1 and 3, and the number of clusters is 5, the method with time information performs better for NASA data set. But in other cases the method without time information outperforms. Since our method for tree construction merges the page number and time information for creating the data field of nodes, the number of data items corresponding to the same page increases if the values of normalized time changes in a wide range. For example for the NASA data set, if the time is normalized between 1 and 10, the number of data items becomes 920, since the number of pages is 92. Thus, for a page we have 10 different data items. Since we only recommend pages to the user, having different data items corresponding to one page makes the recommendation inefficient. For each experiment we register the average time spent to produce the recommendation set. Due to lack of space, we just present the results of the experiments in which the normalized time has a value between 1 and 2. All Experiments are performed on a Pentium II, 333 MHz computer with a 512 MB main memory running Microsoft Windows 2000. The programs are coded in Java.

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
5	57.41	96.47	59.22	98.80	59.79	99.65
10	54.68	91.10	56.15	93.15	57.18	94.53
15	52.61	88.15	54.65	91.15	55.95	92.43
20	50.79	84.45	52.47	86.37	53.51	87.85
25	49.59	81.47	52.17	85.28	53.11	86.50
30	48.75	80.06	51.29	84.92	52.15	85.77

Table 3: Results in % of the NASA data set. Visiting time is normalized between 1 and 2.

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
5	56.19	91.17	57.23	92.82	57.95	93.89
10	53.92	88.31	55.07	89.9	56.01	91.27
15	52.3	86.36	53.77	88.21	54.68	89.36
20	48.96	80.69	50.58	83.01	52.10	84.20
25	48.67	80.15	50.02	82.45	50.42	82.98
30	48.33	79.50	49.37	81.17	50.58	82.74

Table 4: Results in % of the NASA data set. Time information is ignored.

Table 3 shows the results of the NASA data set where the visiting time of pages are normalized between 1 and 2. The results in Table 4 are obtained without taking time into consideration. Table 5 and Table 6 show the results of ClarkNet data set with normalized time between 1 and 2, and without time information respectively. Figure 3 and 4 present the average time spent to produce one recommendation set using normalized time between 1 and 2. As can be seen from the figures using clustering approach reduces the time for producing the recommendation set whereas the prediction accuracy decreases but is still acceptable. If we do not use the time information, the data field of each node in the click-stream tree consists of only the page number. We perform the experiments for ClarkNet data set with different number of clusters from the experiments of NASA data set. This is done to account for the lower number of sessions and number of pages in the ClarkNet data set. As can be seen from the tables, the method that incorporates time information performs mostly better. Only in the case of large number of clusters does the ClarkNet data set have a lower prediction accuracy with time information. This is likely due to the fact that ClarkNet data set is not cleaned to the same extent as the NASA data set since ClarkNet data set does not exists anymore.

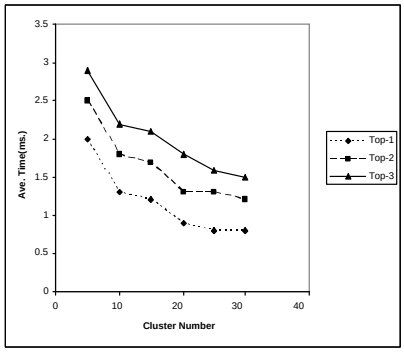

Figure 3: Average time in ms. spent to produce one recommendation set for the NASA data set

For evaluating the performance of our method, we run the experiments with the same training and test examples using 3 other recommendation methods proposed in [10, 11, 12] (these methods are discussed further in the next section). Since the hit-ratio and click-soon-ratio metrics have not performed well for the model in [10], we use the precision metric as proposed in [10] for evaluation. The precision for NASA data set is 4% and for ClarkNet data set 15%. For the method in [11] we use a sliding window with a window

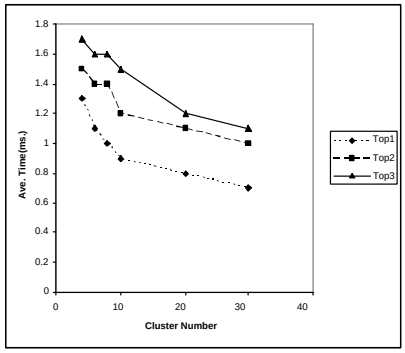

Figure 4: Average time in ms. spent to produce one recommendation set for the ClarkNet data set

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
4	53	100	53.84	100	54.07	100
6	50.53	97.64	50.81	97.86	51.40	98.72
8	49.65	97	50.01	97.20	50.95	97.86
10	48.22	94.07	48.65	94.86	49.01	94.82
20	39.9	76.9	41.51	78.85	42.13	79.50
30	35.65	68.34	37.74	71.75	39.19	73.57

Table 5: Results in % of the ClarkNet data set. Visiting time is normalized between 1 and 2.

size 2. The hit-ratio and click-soon-ratio for NASA data set is 47.84% and 84.41% respectively. The hit-ratio and click-soon-ratio for ClarkNet data set is 49.30% and 92.7% respectively. Since this method does not utilize clustering, we can compare these experiments to our experiments in which we use one tree. Clearly our method is superior. For the last set of experiments we use first order Markov models. The parameters of the Markov model are learned using Expectation-Maximization algorithm. The hit-ratio and click-soon-ratio for NASA data set are 52.6% and 86.3% respectively. The ClarkNet data set has for 4 clusters a hit-ratio of 50.08% and click-soon-ratio of 95.12%. These results prove that our model performs better than the previous proposed models whether we use one click-stream tree or cluster the data set.

Our model has a high click-soon-ratio, in some cases even about 100%. Thus, the model is very useful for a cache prefetching system. Besides this, the clustering approach reduces the search space when working with sites with complex architecture.

4. RELATED WORK

The major classes of recommendation services are based on the discovery of navigational patterns of users. The main techniques for pattern discovery are sequential patterns, association rules, Markov models, and clustering.

There have been attempts to use association rules [11], sequential patterns [1], and Markov models [12] in recommender systems. These techniques work well for Web sites that do not have a complex structure, but experiments on complex, highly interconnected sites show that the storage space and runtime requirements of these techniques increase due to the large number of patterns for sequential pat-

No.Of Clusters	Top-N					
	1		2		3	
	H-R	CS-R	H-R	CS-R	H-R	CS-R
4	49.94	91.38	51	92.95	51.09	92.96
6	48.69	90.13	49.86	92.19	50.4	92.53
8	48.42	90.05	49.63	92.17	50.22	92.79
10	47.18	88.16	48.64	90.78	48.97	91.32
20	40.95	79.22	43.25	81.80	44.45	83.58
30	37.69	73.29	38.48	74.76	41.23	77.12

Table 6: Results in % of the ClarkNet data set. Time information is ignored.

tern and association rules, and the large number of states for Markov models. It may be possible to prune the rule space, enabling faster on-line prediction. Except higher order Markov models, all of these techniques do not capture the entire behavior of a user in a session. Because the number of parameters for higher order Markov models are high, it is not feasible to learn higher order Markov models where the number of Web pages in a site (i.e. the number of states for the Markov model) is big. The compact structure of the click-stream tree of our model makes it possible to keep the entire structure of a user session without any information loss like higher order Markov models (if the length of a user session is n, like $n-th$ order Markov models). Furthermore, the similarity metric we propose in this paper capture both the sequentiality and the time information of user sessions where all of the previous models lack the time information.

The methods in [2] cluster user sessions based on a similarity metric between each session. Our method for clustering has a similar basic idea. However, our similarity metric is different from these metric since it considers the distance between matching pages. Furthermore, we extend our work by representing each cluster by a click-stream tree to use these clusters for predicting the user's next request. Page recommendations in [10] are based on clusters of pages found from the server log for a site whereas the recommendations in [11] are based on association rule discovery from usage data. The crucial differences between our model and these previous models are that we consider both the order of pages and the time spent on that pages and our model enables clustering of user sessions which reduces the search space and thus the recommendation time. Furthermore, the click-stream tree in our model represents the behavior of a user from the beginning that a user enters to the Web site to the end of her session in that site. Consequently, as the experiments demonstrate, our model's prediction accuracy is superior.

5. CONCLUSION

We have considered the problem of modelling the behavior of a Web user during a single visit to the Web site. We introduce a similarity metric to find pair-wise similarities between user sessions. This similarity metric compares two user sessions by means of visited pages and visiting times. The third significant element of that metric is that it also reflects the distance between matching pages of two user sessions. We partition user session based on that similarity metric using a graph partitioning algorithm and propose a tree construction for representing the clusters. The experiments show that the model can be used on Web sites with different structures. To confirm our finding, we compare our model to three other recommendation models. Results show that our model improves the efficiency and effectiveness significantly.

6. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the International Conference on Data Engineering (ICDE)*, March 1995. Taipei, Taiwan.

[2] A. Banerjee and J. Ghosh. Clickstream clustering using weighted longest common subsequences. In *Proceedings of the Wokshop on Web Mining, SIAM Conference on Data Mining*, pages 33–40, 2001. Chicago, IL.

[3] K. Cahrter, J. Schaeffer, and D. Szafron. Sequence alignment using fastlsa. In *Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2000)*, pages 239–245, 2000.

[4] Cluto. http://www-users.cs.umn.edu/ karypis/cluto/index.html.

[5] Dan Cosley, Steve Lawrence, and David M. Pennock. REFEREE: An open framework for practical testing of recommender systems using researchindex. In *Proceedings of 28th International Conference on Very Large Databases, VLDB 2002*, Hong Kong, August 20–23 2002.

[6] Chris Ding, Xiaofeng He, Hongyuan Zha, Minh Gu, and Horst Simon. Spectral min-max cut for graph partitioning and data clustering. 2001. Technical Report TR-2001-XX, Lawrence Berkeley National Laboratory, University of CaliforniaBerkeley, CA.

[7] O. Etzioni. The world wide web: Quagmire or gold mine. *Communications of the ACM*, 39(11):65–68, 1996.

[8] ClarkNet WWW Server Log. http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.

[9] NASA Kennedy Space Center Log. http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.

[10] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery of aggregate usage profiles for web personalization. In *Proceedings of the Web Mining for E-Commerce Workshop (WebKDD'2000)*, 2000.

[11] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd ACM Workhop on Web Information and Data Management*, November 2001. Atlanta, USA.

[12] R. R. Sarukkai. Link prediction and path analysis using markov chains. In *Proceedings of the Ninth International World Wide Web Conference*, 2000. Amsterdam.

[13] C. Shahabi, A. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users web-page navigation. In *Proceeding of the IEEE RIDE97 Workshop*, pages 20–29, April 1997. Birmingham, England.

[14] J. Srivastava, R. Cooley, M. Deshpande, and P. N. Tan. Web usage mining: Discovery and application of usage patterns from web data. *ACM SIGKDD Explorations*, 1(2):12–23, 2000.