Distrib Parallel Databases (2006) 19: 5–27
DOI 10.1007/s10619-006-6284-1

Incremental click-stream tree model: Learning from new users for web page prediction

Şule Gündüz Öğüdücü · M. Tamer Özsu

Abstract Predicting the next request of a user has gained importance as Web-based activity increases in order to guide Web users during their visits to Web sites. Previously proposed methods for recommendation use data collected over time in order to extract usage patterns. However, these patterns may change over time, because each day new log entries are added to the database and old entries are deleted. Thus, over time it is highly desirable to perform the update of the recommendation model incrementally. In this paper, we propose a new model for modeling and predicting Web user sessions which attempt to reduce the online recommendation time while retaining predictive accuracy. Since it is very easy to modify the model, it is updated during the recommendation process. The incremental algorithm yields a better prediction accuracy as well as a shorter online recommendation time. A performance evaluation of Incremental Click-Stream Tree model over two different Web server access logs indicate that the proposed incremental model yields significant speed-up of recommendation time and improvement of the prediction accuracy.

Keywords Web · Recommendation systems · Web access prediction

1. Introduction

In recent years, the problem of modeling and predicting a Web user's surfing behavior at a Web site has become increasingly important since this can be used to improve Web performance through caching [2, 38] and prefetching [34, 35], to recommend related pages [33], improve search engines [9] and personalize browsing in a Web site [34]. Recommender systems on Internet attempt to determine which Web pages (items) will be accessed (bought) in the near

Ş.G. Öğüdücü
Computer Engineering Department, Istanbul Technical University, Istanbul, Turkey 34390
e-mail: gunduz@cs.itu.edu.tr

M.T. Özsu
School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
e-mail: tozsu@cs.uwaterloo.ca

future, given a user's (who may, for example, be a customer in an e-commerce site) current actions. They help decision making in the complex information space where the volume of information available to users is huge.

There are three steps in this process [40]. The first step is to clean the data collected from several sources 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. Methods of data cleaning and preparation have been well studied (see [40] for a survey). The traditional techniques used for modelling usage patterns in a Web site are collaborative filtering (CF) [5, 36], clustering pages or user sessions [27, 32], association rule generation [31, 39], sequential pattern generation [18], and Markov models [3, 15, 35]. 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.

Most of the works in recommendation systems focus on the use of usage data [3, 4, 15, 18, 27, 28, 32, 35], while some works also consider knowledge associated with the content [6, 7, 7, 12, 29] or the structure [25, 30]. Usage based recommendation systems could be problematic due to content changes or the incomplete and limited usage data [25]. In this paper, we solve these problems by an incremental recommendation system that is updated according to the behavior of new users. It is obvious that the change of the structure or the content of some pages on the web site effects the navigational behavior of users. By adding the navigational path of new users to the system periodically, the system solves both the problem of changing site content or the structure and the incomplete data. Besides this, our systems enriches the usage data by considering the time that a user spends on each page. An important feature of the user's navigation path is the time that a user spends on different pages [37]. Even the same user may have different desires at different times; if the desire of a user was known every time she visits a Web site, one could use this information for recommending pages. Unfortunately, experience shows that users are rarely willing to give explicit feedback. Thus, 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.

We previously proposed a new similarity measure to calculate pair-wise similarities of user sessions derived from Web server access logs and presented a new model that uses both the sequences of visiting pages and the time spent on those pages [20]. In considering a Web server access log, we address an environment with the following characteristics:

1. Derived information is present for analysis;
2. The environment is dynamic, i.e. many updates may occur.

On even a moderately busy Web server, the quantity of information stored in the log files is very large. Consequently, it will be necessary to periodically rotate the log files by moving or deleting the existing logs. In such an environment, all of the extracted usage patterns may change over time. As far as we now, existing tools for recommendation do not use the information derived from updates to Web server log entries. The reason may be that the models are hard to modify. However, the rotation of the log data of the Web site provides useful information for recommendation.

In this paper, we present an *incremental* Web page recommendation model, building on our previous work [20]. Our algorithm is based on Click-Stream Tree (CST) model. Due to the compact structure of CST, efficient algorithms can be given for incremental insertions and deletions to an existing CST. The representation of the behavior of Web users in our model is convenient for incorporating this new information. Our overall approach can be summarized as follows. The user sessions are clustered based on the pair-wise similarities of

the user sessions and each cluster is then represented by a CST whose nodes are pages of user sessions of that cluster. 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.[1] 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 exploring 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 generated by using the top-N clusters. Periodically, new user sessions are added to the most matching cluster and nodes that are not used by former recommendations are deleted from the CSTs. We call this process the *relevance feedback* since the information obtained from new users are added to the model.[2] Thus, the resulting CSTs cover the new user sessions. The model size does not increase as new user sessions are added due to the compact structure of CSTs; in fact, the sizes of the CSTs decrease since unused nodes are deleted. This enables significant speed-up of recommendation time and improvement of the prediction accuracy. We demonstrate the high efficiency of Incremental Click-Stream Tree Model (ICST-Model) on two different Web server access logs and compare our results with three well known recommendation models.

The contributions of this paper are the following:

1. A recommendation model is presented that can be modified according to the updates in Web access logs;
2. A method is proposed to keep the model size under control;
3. The proposed recommendation model has high prediction accuracy and low real time computation overhead.

The rest of the paper is organized as follows. Section 2 briefly reviews the work related to Web usage mining and describes graph-based clustering and sequence alignment methods. Section 3 presents the proposed model. Section 4 provides detailed experimental results. In Section 5, we examine related work. Finally, in Section 6 we conclude and discuss future work.

2. Background

2.1. Web usage mining

In general, Web mining is a common term for three knowledge discovery domains that are concerned with mining different parts of the Web: Web Content Mining, Web Structure Mining, and Web Usage Mining [26]. While Web content and structure mining utilize real or primary data on the Web, Web usage mining works on the secondary data such as Web server access logs, proxy server logs, browser logs, user profiles, registration data, user sessions or transactions, cookies, user queries, and bookmark data. Web usage mining refers to the application of data mining techniques to discover usage patterns from these secondary data, in order to understand and better serve the needs of Web-based applications. The usage data collected at different sources will represent the navigation patterns of different segments of

[1] The user session that has the highest similarity to the active user session is defined as the best session.

[2] Note that this is different from the user feedback discussed earlier where individual users are expected to give explicit feedback about the relevance of individual pages that are visited.

the overall Web traffic, ranging from single-user, single-site browsing behavior to multi-user, multi-site access patterns. The information provided by the data sources can all be used to construct/identify several data abstractions, such as users, server sessions, episodes, click stream, and page views [24].

2.2. One dimensional sequence alignment

In our study, we use sequence alignment techniques to analyze the sequence of user requests that appear in user sessions. For two strings of length m and n, optimal sequence alignment has zero or more gaps inserted into the sequence to maximize the number of positions in the aligned strings that match. For example, consider aligning the sequences "$p_1\ p_2\ p_4\ p_5$" and "$p_4\ p_1\ p_2\ p_5\ p_3\ p_6$". By inserting gaps ($-$) in the appropriate place, the number of positions where two sequences match can be maximized:

-	p_1	p_2	p_4	p_5	-	-
p_4	p_1	p_2	-	p_5	p_3	p_6

Here the aligned sequences match in three positions. Algorithms for efficiently solving this type of problem are well known and are based on dynamic programming [10]. Our method for finding the best alignment for a pair of user sessions uses the basic ideas from these algorithms.

2.3. Graph based clustering

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and dissimilar to objects in other clusters. In this study, we focus on pairwise data clustering. Formally, the pairwise data clustering problem is defined as follows: Given $S = (D, N, W, C)$, where D is a set of data, $N \subseteq D \times D$ is a set of data pairs, W is a symmetric matrix of similarity values of each data pair in N, and C is the clustering criterion function, partition the data set D into clusters such that the criterion function C is optimized.

Given the similarity metric between any pair of data items, the symmetric matrix W forms a weighted adjacency matrix (*weight matrix*) of an undirected graph. Thus, the pairwise data clustering problem becomes a graph partitioning problem. Given a weighted, undirected graph $G = G(V, E)$ with a set of vertices V that represent the data items in D and an edge set E that represent the data pairs in N whose weights are defined by W, we wish to partition it into several subgraphs such that criterion function C is optimized.

3. Incremental click-stream tree model

This section presents the proposed model. As discussed in the Introduction, Web usage mining consists of three steps. In the first step, we use cleaning and filtering methods in order to identify unique users and user sessions. In the second step, we use a similarity measure for calculating the similarities between all pairs of sessions. In the third step, the sessions are clustered based on those similarities using the graph partitioning algorithm and each cluster is represented by a CST. In order to produce the recommendation set, we first select the cluster and then select the path from the CST of that cluster that best matches the active user session. Periodically, the CSTs are updated according to the behavior of new users.

3.1. Data preparation and cleaning

In the experimental part of this research (Section 4), we use two well-known sets of server logs. The first is from the NASA Kennedy Space Center server over the months of July and August 1995 [44]. The second log is from ClarkNet Web server which is a full Internet access provider for the Metro Baltimore-Washington DC area [45]. This server log was collected over the months of August and September, 1995. For each log data set we apply the same pre-processing steps. The log entries are converted into a set of user sessions as follows: the irrelevant log entries are eliminated such that only URL page requests of the form "GET ...html" are maintained. The *visiting page time*, which we define as the time difference between consecutive page requests, is calculated for each page. For the last page of the user session, we set the visiting page time to be the mean of the times for that page taken across all sessions in which it is not the last page request. A new session is created when a new IP-address is encountered or if the visiting page time exceeds 30 minutes for the same IP-address. Thus, a session consists of ordered sequence of page visits. As in other studies [27, 28] short user sessions are removed. For this study, we eliminate sessions whose *session length*[3] is less than or equal to 2 in order to eliminate the effect of random accesses to the Web site. It is important to note that these are only heuristics to identify users and user sessions, and other heuristics may be employed in future studies.

The visiting times are normalized across the visiting times of the pages in the same session, such that the normalized time has a value between 1 and 2. The effect of different values for normalization times are examined in [20, 21] and we show that normalizing time between 1 and 2 improves the prediction accuracy. For that reason, in this study we normalize the visiting page time between 1 and 2. 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.

Eventually, this step produces a set of URL's $P = \{p_1, \ldots, p_n\}$. The output of this step is a set of user sessions $S = \{S_1, S_2, ..., S_{|S|}\}$ where $|S|$ is the number of sessions of S. Each session S_i of the set of sessions S is defined by a tuple $S_i = (PAGES_i,\ NORMS_i)$ where $PAGES_i$ is a subset of P that the user visits in her session S_i and $NORMS_i$ is the normalized visiting times of pages in P:

$$PAGES_i = \{p_{i1}, \ldots, p_{ik}\} \quad i \in [1, \ldots, |S|] \wedge p_{i,k} \subset P$$

$$NORMS_i = \{norm_{p_1}, ..., norm_{p_n}\}$$

3.2. Session similarity measure

In this section, we propose a new session similarity measure. Following the data cleaning and preprocessing steps, we use this measure in the second step for calculating the similarities between all pairs of sessions. Since user sessions are ordered URL requests, we can refer to them as sequences of Web pages. The problem of finding the optimal sequence alignment is solved using dynamic programming approach. Briefly, the algorithm consists of three steps. The first step is initialization where a dynamic programming matrix is created with $K + 1$ columns and $N + 1$ rows where K and N correspond to the sizes of the sequences to be

[3] The length of a session is determined by the number of pages requested by one user within a server session.

aligned. One sequence is placed along the top of the matrix (sequence#1) and the other one along the left-hand-side of the matrix (sequence#2). A gap is added to the end of each sequence which indicates the starting point of calculation of similarity score.

The second step of the algorithm is *FindScore*, in which we modify the dynamic programming algorithm for sequence alignment to calculate the two dimensional similarity of sessions. We implemented the algorithm with an additional module for that step that takes into account the time spent on matching pages. In our implementation, the identical matching of Web pages and time spent on those pages is given a score $s_m = 2$ and each mismatch or gap inserted into the sequences is given a penalty score of -1, i.e. $s_d = s_g = -1$. Then the two dimensional matching score $Score_{l,r} = s(p_{il}, p_{jr})$ of the matrix is calculated for a pair of matching pages, $p_{il} = p_{jr} = p_m$ as follows:

$$Score_{l,r} = s(p_{il}, p_{jr}) = s_m \frac{min(norm_{p_{il}}, norm_{p_{jr}})}{max(norm_{pil}, norm_{p_{jr}})} \tag{1}$$

where $p_m \in P$ and $norm_{p_{il}}$ is the normalized value of time corresponding to the page p_{il} in session S_i, and $norm_{p_{jr}}$ is the normalized value of time corresponding to the page p_{jr} in session S_j. In this step the scoring matrix is filled by starting at the lower right-hand corner in the matrix and finding the maximal score $M_{i,j}$ for each position in the matrix.

The third step is *FindPath* which determines the actual alignment that leads to the maximal score. *FindPath* traverses the matrix beginning from the destination point (upper left corner) to the start point (lower right corner) of the matrix. It takes the current cell and chooses as the next cell one of the neighboring cells that leads to the maximal score.

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, S_i and S_j, is σ and the number of matching pages is M in the aligned sequence, then the alignment score component is:

$$s_a(S_i, S_j) = \frac{\sigma}{s_m * M}$$

The intuition behind this is that score σ is higher if the sessions have more consecutive matching pages. This value is normalized by dividing it 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:

$$s_l(S_i, S_j) = \frac{M}{L}$$

Then the overall similarity between two sessions is given by

$$sim(S_i, S_j) = s_a(S_i, S_j) * s_l(S_i, S_j) \tag{2}$$

Example 1. Let us illustrate the calculation of similarity between two user sessions. Suppose that the set of pages P consists of 10 pages, $P = \{p_1, ..., p_{10}\}$ and the user sessions are as in Table 1. For two user session, S_3 and S_9, the dynamic programming matrix is given in

Table 1 Sample user sessions as running example

S	PAGES	NORMS
S_1	$[p_1, p_5, p_7, p_3, p_4]$	[1, 0, 1, 2, 1, 0, 2, 0, 0, 0]
S_2	$[p_5, p_3, p_7, p_9]$	[0, 0, 1, 0, 1, 0, 2, 0, 2, 0]
S_3	$[p_2, p_8, p_4, p_3]$	[0, 1, 2, 2, 0, 0, 0, 1, 0,0]
S_4	$[p_5, p_3, p_6, p_8]$	[0, 0, 1, 0, 1, 1, 0, 2,0, 0]
S_5	$[p_2, p_8, p_6, p_5]$	[0, 1, 0, 0, 1, 2, 0,2, 0, 0]
S_6	$[p_5, p_3, p_7, p_9]$	[0, 0, 1, 0, 1, 0,2, 0, 2, 0]
S_7	$[p_2, p_8, p_6, p_4]$	[0, 1, 0, 2, 0,1, 0, 2, 0, 0]
S_8	$[p_1, p_7, p_6, p_5]$	[1, 0, 0, 0,2, 2, 1, 0, 0, 0]
S_9	$[p_2, p_8, p_6, p_5, p_3]$	[0,1, 1, 0, 1, 2, 0, 2, 0, 0]

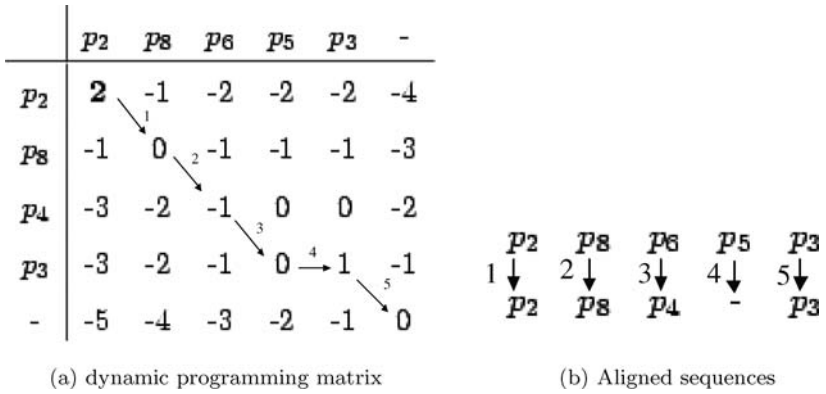
Figure 1(a). The sequence alignment algorithm produces aligned sequences as in Figure 1(b). Since the length of the aligned sequences is 5 and the number of matching pages in these sessions is 3, the overall similarity between these sessions is $sim(1, 2) = (2/(2 * 3)) * (3/5)$.

3.3. Pairwise clustering

After calculating pair-wise similarities of all user sessions, 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 the problem of partitioning graph G into k disjoint subgraphs G_m, $(m \in [1, ..., k])$ by minimizing *MinMaxCut* function [16]. 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



(a) dynamic programming matrix (b) Aligned sequences

Fig. 1 Alignment of two user sessions

v_j calculated using the similarity metric. A software package called Cluto is used for graph partitioning.

3.3.1. Cluster representation

The clusters created by the graph partitioning algorithm contain user sessions. Each cluster has the following properties:

1. The order of occurrence of Web pages is similar across the user sessions in the same cluster.
2. The distance between identical Web pages[4] of two user sessions in a cluster is shorter than the distance between identical Web pages of two user sessions in different clusters.
3. The amount of time spent on identical Web pages of two user sessions in a cluster is similar.

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. It is important to represent each user session in a cluster while preserving the properties of a cluster. One way of doing this is to represent each unique user session in a cluster as a branch of a tree, which is what we call the click-stream-tree (CST). Thus, we generate a CST for each cluster. Each CST has a *root* node, which is labelled as "null". Each node except the *root* node consists of four fields: *data*, *count* and *next_node* and *pass*. *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 at that node. *Next_node* links to the next node in the CST that has the same *data* field or null if there is any node with the same *data* field. *Pass* field registers the number of recommendations made by the system using that node. The *pass* field is only used during recommendation and feedback processes. When the CSTs are first created all the *pass* fields of the nodes are set to zero. Each CST has a *data_table*, which consists of two fields: *data* field and *first_node* that links to the first node in the CST that has the *data* field.

The tree for each cluster is constructed by applying the algorithm given in Figure 2. Let each session be in the form of $S_i = (PAGES_i,\ NORMS_i)$, where $PAGES_i = \{p_{i1}, ..., p_{ik}\}$ consists of visited pages in session S_i and $NORMS_i$ is the normalized visiting times of pages in P, as stated in Section 3.1. $S_i.length$ corresponds to the number of pages in $PAGES_i$. We start the algorithm with an empty tree that contains only the *root* node. For each session of a cluster (line 1) and for each request in the session (line 5) the algorithm does the following: Since the algorithm starts to insert a session in the tree from the *root* node, first the *root* node is stored as *Current_Node* (line 4). For each request of the session the requested page and corresponding normalized time value are merged to create the *data* field of a node (line 6) and labelled as *Active_Data*. If the *Current_Node* has a child with the same *data* field as the $Active\_Data$ (line 7), then the *count* field of the child node of the *Current_Node* is incremented by 1 (line 8) and the child node is labelled as *Current_Node* (line 9). If the *Current_Node* does not have a child with the same *data* field as *Active_Data* (line 10) a new node with that *data* field is created as the child node of the *Current_Node* (line 11), its *data* field is set to *Active_Data* (line 12), its count is set to 1 (line 13), its *pass* is set to 0 (line 14) and is labelled as *Current_Node* (line 15). Thus, the next request of a session is inserted always in a node that is a child of the node of the previous request. Such a tree has an inherent orientation,

[4] The distance between identical pages is measured as the number of user requests between these pages with the same order of occurrence in these sessions.

Input : Sessions in a cluster
Output : CST

```
1: for all user sessions in a cluster do
2:   S_i ← next user session in the cluster
3:   m ← 0
4:   Current_Node ← root node of the CST
5:   while m ≤ S_i.length do
6:     Active_Data ← {p_im}-{norm_{p_im}}
7:     if there is a Child of Current_Node with the same data field then
8:       Child.count + +
9:       Current_Node ← Child
10:    else
11:      create a child node of the Current_Node
12:      Child.data = Active_Data
13:      Child.count = 1
14:      Child.pass = 0
15:      Current_Node ← Child
16:    end if
17:    m + +
18:  end while
19: end for
```

Fig. 2 *Build CST* algorithm

since each session is considered from beginning to end. This covers the first property of a cluster. If the distance between identical pages of two user sessions is short, then the number of nodes between these pages in the tree is less compared to the number of nodes of pages that have longer distances. This covers the second property of a cluster. By constructing the CST, the page number and corresponding normalized time values are merged together which forms the *data_field* of a node in the tree. This covers the last property of a cluster.

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

Example 2. Let us illustrate the construction of the CST with an example. Let the sessions in a data set be clustered into 2 clusters and let each cluster consists of the sessions given in Table 2. First the *root* of the tree of the first cluster is created and labelled with null (Figure 3).

Table 2 Sessions of two clusters

Cl.	S	PAGES	NORMS
1	S_1	[p_1, p_5, p_7, p_3, p_4]	[1, 0, 1, 2, 1, 0, 2, 0, 0,0]
1	S_2	[p_5, p_3, p_7, p_9]	[0, 0, 1, 0, 1, 0, 2,0, 2, 0]
1	S_4	[p_5, p_3, p_6, p_8]	[0, 0, 1, 0, 1,1, 0, 2, 0, 0]
1	S_6	[p_5, p_3, p_7, p_9]	[0, 0, 1,0, 1, 0, 2, 0, 2, 0]
1	S_8	[p_1, p_7, p_6, p_5]	[1,0, 0, 0, 2, 2, 1, 0, 0, 0]
2	S_3	[p_2, p_8, p_4, p_3]	[0, 1, 2, 2, 0, 0, 0, 1, 0,0]
2	S_5	[p_2, p_8, p_6, p_5]	[0, 1, 0, 0, 1, 2, 0,2, 0, 0]
2	S_7	[p_2, p_8, p_6, p_4]	[0, 1, 0, 2, 0,1, 0, 2, 0, 0]
2	S_9	[p_2, p_8, p_6, p_5, p_3]	[0,1, 1, 0, 1, 2, 0, 2, 0, 0]

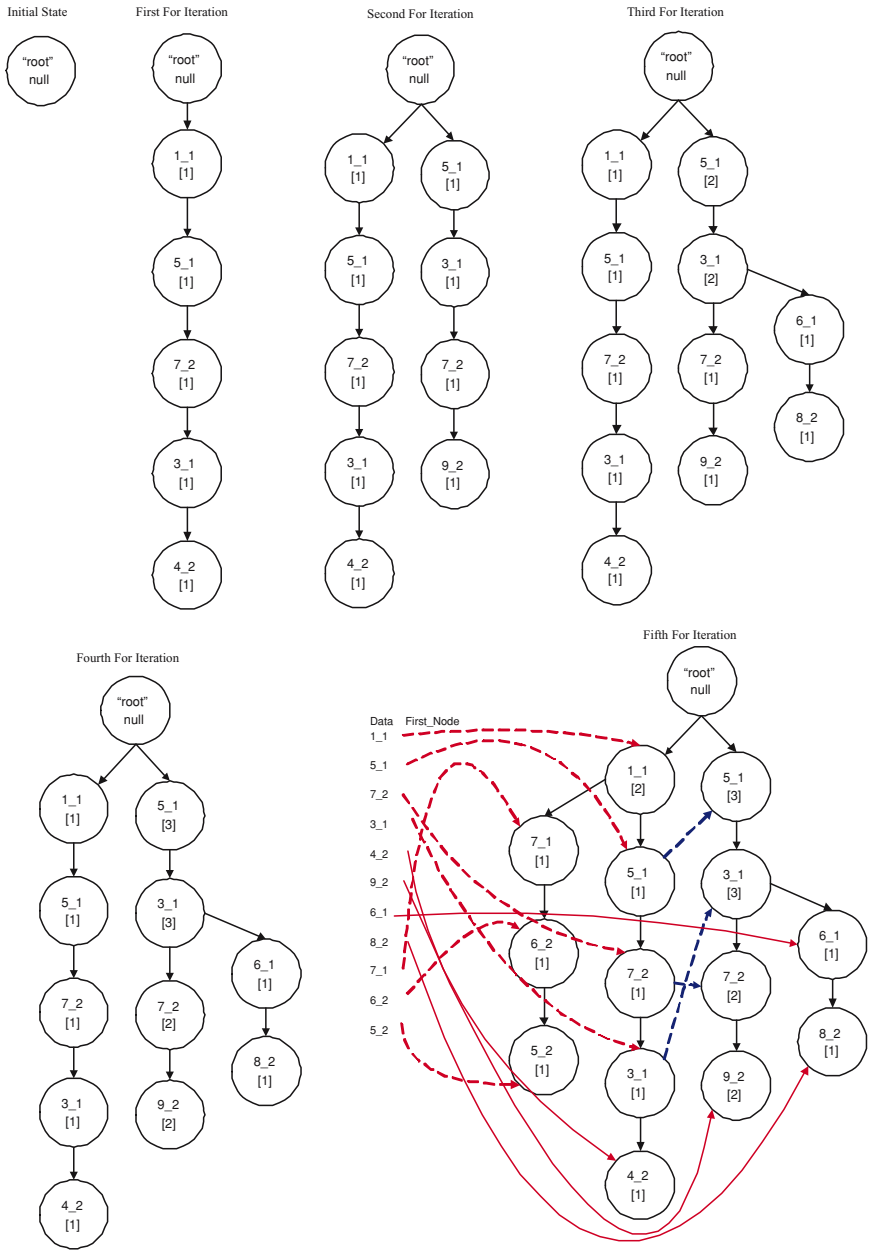


Fig. 3 Construction of the CST of the first cluster

The first page of the session S_1 is inserted to the tree as the child of the *root*, and its *count* field is set to 1. The following pages of this session are inserted as the children of the previous page and their *count* fields are set to one (first For Iteration in Figure 3). For session S_2, a new node is created with *data* field 5_1 as the child node of the *root* node since that node does not exist in the tree. The remaining pages are inserted as the children nodes of the previous page

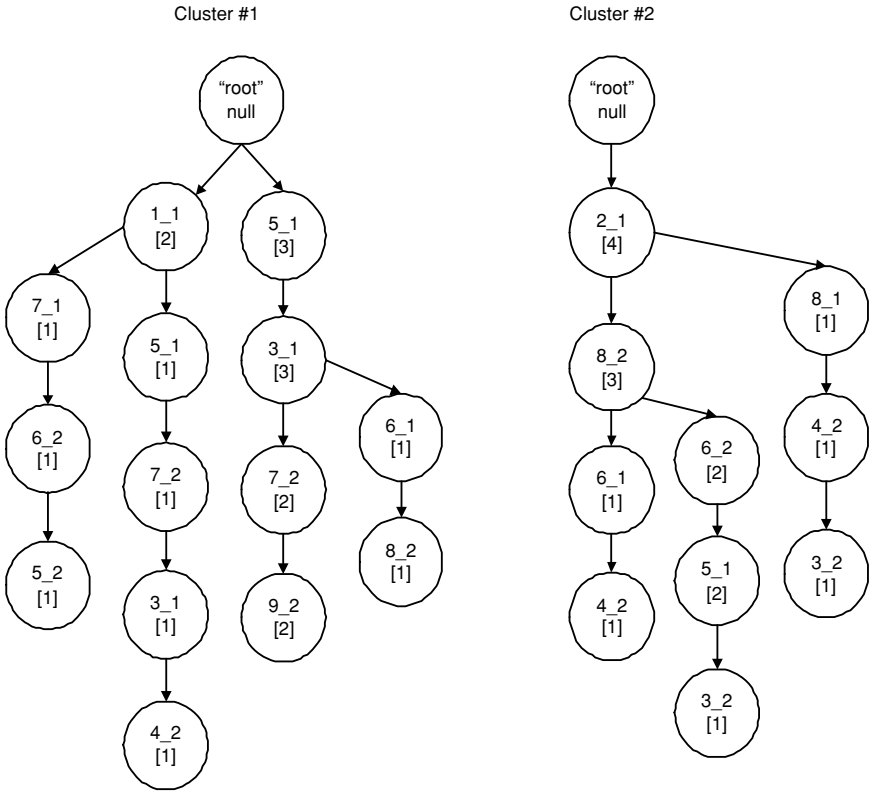


Fig. 4 CSTs of two clusters

in the session (second For Iteration). For session S_8, since the *root* node has a child with the *data* field 1_1, the *count* of that node is incremented by one. A new node is created with *data* field 7_1, and linked as the child of 1_1 (fifth For Iteration). The remaining sessions of that cluster are inserted in the tree in the same manner. The CST of the second cluster is built as the first one. Figure 4 shows the constructed CSTs. Circles represent tree nodes. Each node in a tree has a *data* field (shown in Figure 4 as PageNumber_NormalizedTime in the first line in each node) and a *count* field (shown in Figure 4 as [count] in the second line in each node). For simplicity, PageNumber of the *data* field is represented by only the subscript of pages in *PAGES* field of a user session. Since the *pass* field of all of the nodes are 0, it is not shown in the figure. After inserting all the sessions in a cluster, the CST of one cluster with the associated *next_node* and *data_table* is shown in Figure 3.

3.4. Recommendation engine based on relevance feedback

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 and adds the relevance feedback. The recommendation engine consists of two parts. The first part finds the best matching path for the active user session and generates the recommendation set whereas the second part is responsible for relevance feedback. The complete recommendation algorithm is given in Figure 5. Since it is obvious that modifying the CSTs after each user session is not efficient, it

Input : CSTs
Output : ICSTs

```
1:  count = 0
2:  Buffer ← {∅}
3:  for all new user sessions do
4:    count + +
5:    Find Best Path
6:    if Buffer is full then
7:      Insert CST
8:      Buffer ← {∅}
9:    else
10:     Buffer ← Buffer + {new user session}
11:   end if
12:   if count = y then
13:     for i = 0 to NumberOfClusters do
14:       for all nodes of CST[i] do
15:         if nodes.pass = 0 then
16:           Prune CST
17:         end if
18:       end for
19:     end for
20:     count = 0
21:   end if
22: end for
```

Fig. 5 Recommendation process

would be better to form a buffer for storing new user sessions and corresponding best clusters (line 10). If the buffer is full, user sessions that are stored in the buffer can be inserted into CSTs (line 7). If the number of user sessions is equal to a previously defined number y, then the unused nodes can be pruned from the CSTs (line 12–15). A second alternative for the deletion frequency may be to determine it according to the time needed for generating a recommendation set. If the recommendation time exceeds a specified threshold, the unused nodes can be pruned from ICSTs. In this study we use the first alternative for determining the insertion and deletion frequencies in order to speed up the recommendation time. In this subsection, we first describe the algorithm for finding the best matching path, followed by a description of how we add the relevance feedback to the model.

3.4.1. Recommendation set generation

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 CST. Given the time of the last visited page of the active user session, the model recommends three pages. The most recently visited page of the active user session contains the most important information. Before describing the recommendation algorithm, it is useful to first understand a few special properties of the CST. If a *data* field d_i is found in the tree, all the possible paths that contain d_i can be obtained by following d_i's *next_node*, starting from d_i's *first_node* in the data_table of the tree. The CST

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 user session with the same *PAGES* and *NORMS* fields occurs more than once, only the *count* of its nodes is incremented. Based on the construction of the CST, a user session (p_{i1}, p_{i2}, ..., p_{ik}), ($norm_{p_1}$, $norm_{p_2}$, ..., $norm_{p_n}$) occurs in the tree $d_k.count$ times, where d_k is the *data* field formed by merging the page request p_{ik} and corresponding normalized time value $norm_{p_{ik}}$ of the user session.

Figure 6 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

Input : Active user session, CSTs
Output : Best matching session

```
1: S_a ← Active User Session
2: if S_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:   m ← S_a.length
11:   d_a ← {p_am}-{norm_{p_am}}
12:   Node ← data_table[cl](d_a).first_node
13:   path = null
14:   while Node ≠ null do
15:      path = {path} + {Node.data}
16:      Parent_Node ← Node.Parent
17:      while Parent_Node ≠ null do
18:         path = {path} + {Parent_Node.Data}
19:         Parent_Node ← Parent_Node.Parent
20:      end while
21:      Sim(path) = sim(S_a, path)* Node.count/S[cl]
22:      if Sim(path) > Sim[cl] then
23:         Sim[cl] ← Sim(path)
24:         BestPath[cl] ← path
25:      end if
26:      path = null
27:      Node ← Node.next_node
28:   end while
29: end for
30: for all Nodes of BestPath[cl] with greatest Sim[cl] do
31:   Nodes.pass + +
32: end for
33: if S_a.length = 2 then
34:   Top − N Clusters ← N Clusters with highest Sim[cl] values
35: end if
```

Fig. 6 *Find Best Path* algorithm

path (line 3 in Figure 6). After the second request of the active user, top-N clusters that have higher recommendation scores among other clusters are selected (line 30–32) 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 11). We find from the *data_table* of the CST of a cluster the *first_node* that has the same *data* field (line 12). 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 17–20). 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 21). 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 27). The path that has the highest recommendation score is selected as the best path for generating the recommendation set for that cluster (line 22–25). The *pass* field of the nodes of the best are incremented by one (line 30–32). 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.

Example 3. We illustrate the method for finding the best path with a simple example for top 1 Cluster (Figure 7). Let the active user session be $\langle S_a, [2, 3, 5, 7], [0, 1, 1, 0, 1, 0, 2, 0, 0, 0]\rangle$ and the CSTs be as in Figure 4. The first request of the active user is page 2 with a normalized time value 1. The *data* field is formed by merging 2 and 1. Since only cluster 2 has a node with a *data* field 2_1 only the path $Node(2\_1)$ of cluster 2 is examined and found as the best path with a similarity value of 1. $Node(2\_1)$ in cluster 2 has only children with the page number 8 and page 8 is recommended to the active user. The second request of the active user is page 5 with the normalized time value 1. Thus, the active user path becomes $2\_1 \rightarrow 5\_1$. Starting from the *first_node* of $Node(5\_1)$ on the *data_table* of cluster 1, all the paths that contain $Node(5\_1)$ in cluster 1 are examined by following *next_node* of $Node(5\_1)$. Cluster 2 has only one path with a data field 5_1, $Node(2\_1) \rightarrow Node(8\_2) \rightarrow Node(6\_2) \rightarrow Node(5\_1)$, however the similarity value of this path is $1/8$. Thus the path with the highest similarity value in the first cluster, $Node(5\_1)$, is selected as the best path to produce the recommendation set and page 3 is recommended to the active user. The third request of the active user is page 3 with the normalized time value 1. The active user path becomes $2\_1 \rightarrow 5\_1 \rightarrow 3\_1$. After the first two request of the active user, only top 1 cluster is examined for further recommendation. Since the last best path is from cluster 1, the best path is searched only in cluster 1. Starting from the *first_node* of 3_1 on the *data_table* of cluster 1, two paths are examined by following the next_node of $Node(3\_1)$. The path $Node(5\_1) \rightarrow Node(3\_1)$ is found as the best path with a similarity value of $2/5$. Page 7 and page 6 are recommended to the active user.

3.4.2. Relevance Feedback

As mentioned in Section 1, Web server access log databases are dynamic. Thus, the usage patterns may change over time. It is important to keep the recommendation model up-to-date to improve the prediction accuracy. Every time a new user enters the Web site, the model generates a recommendation set after each request of the user. It is important to observe the behavior of that new user. If she does not request one of the pages in the recommendation set, it means that the recommendation was wrong. This can be due to two reasons: either the recommendation engine is inadequate to generate the correct recommendation set, or the path that this particular user follows does not exist in the CSTs. Let us illustrate this

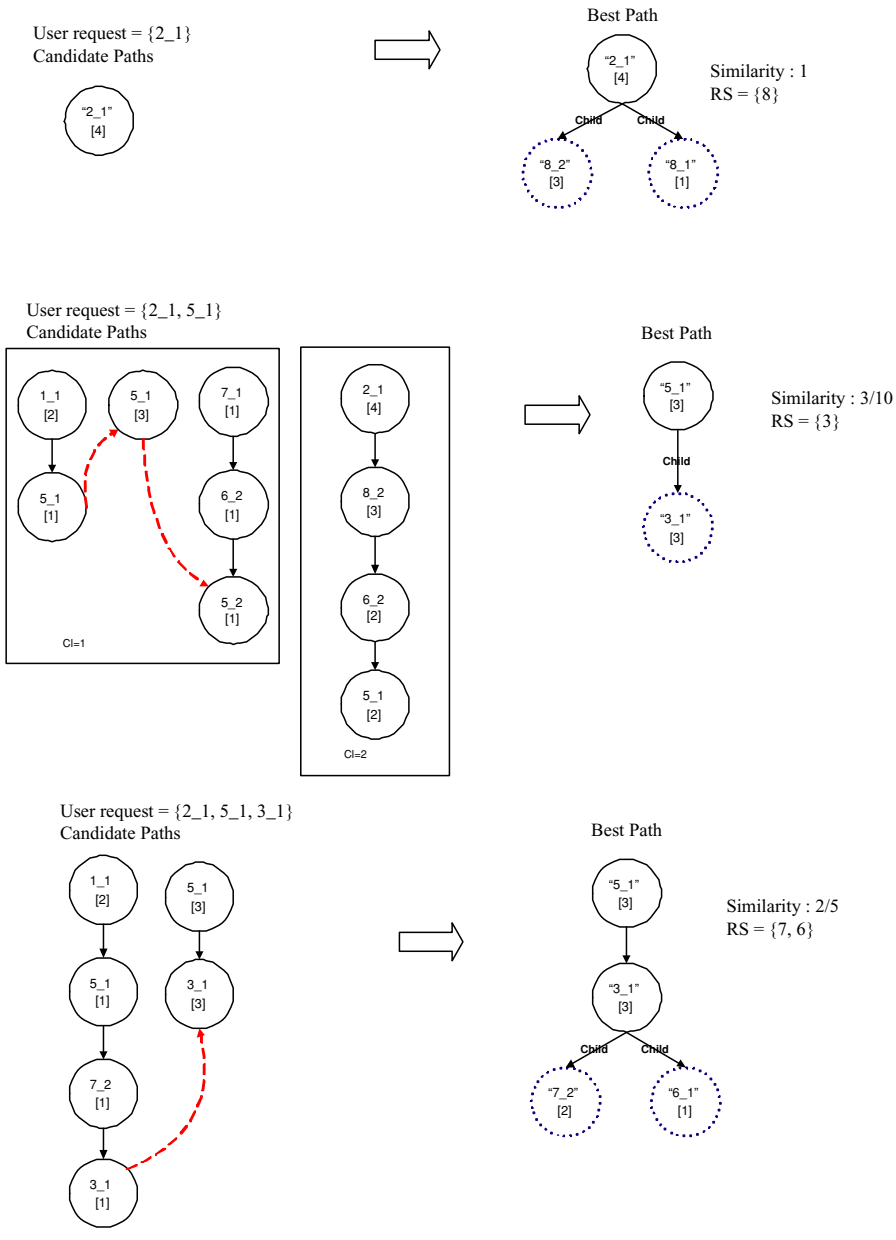


Fig. 7 A sample recommendation set generation

situation with an example. Suppose that a user requests first page p_1 and then page p_2. In order to make a correct recommendation after page p_1, there should be a path in the CSTs that contains the sequence $p_1 p_2$. If not, the recommendation engine would be unable to produce the correct recommendation. An examination of the training set in our experiments shows that the recommendation engine generally selects the path that most fits the active user session. In 93.75% of the cases when the recommendation was incorrect, there does not exist

a path in the CSTs that contains the sequence for producing the correct recommendation. Only in 6.25% of all of cases the recommendation engine is unable to find the most fitting path even though it exists in the CSTs. This investigation leads us to insert new user sessions into CSTs to cover the change of usage patterns over time.

As mentioned before, due to the compact structure of the CST, adding new sessions to the clusters is very simple. However, it is important to prune the nodes that are not used for previous recommendations in order to control the size of the CSTs. If a path in the existing CSTs is not used for generating a recommendation set, it means that the corresponding path is no longer followed by new users. The *pass* field of nodes of a CST provides information about nodes that are used for former recommendations. Thus, deleting the nodes that are not used for recommendation handles the deletion of old entries in the Web server access log database. Furthermore, deleting unused nodes decreases the size of the CSTs which, in turn, leads to a speed-up in recommendation set generation.

4. Experimental results

In this research we use two different Web server access logs (cleaned for experiments as mentioned in Section 3.1). Approximately 30% of these cleaned user sessions are randomly selected as the test set, and the remaining part as the training set. The experiments are repeated with different numbers of clusters and with different values of N for selecting the top-N clusters after the first two pages of the active user session.

Given the visiting time of a page in the current session, the model recommends three pages. We define the hit-ratio metric [11] 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.

In [20], we demonstrated that modeling user sessions with visiting page time information improves the prediction accuracy. Table 3 shows the results of NASA and ClarkNet data sets for different number of clusters without taking visiting page time into consideration. Further experiments conducted in [20] validate the following results:

- The clustering approach reduces the search space when working with sites with complex architecture;
- Normalizing time in a narrow range improves the prediction accuracy;
- CST captures the characteristics of user sessions;
- The proposed similarity measure yields a good prediction accuracy.

Table 3 Hit-Ratio in % of the data sets. Time information is ignored.

	Top-N				Top-N		
No. of Clusters	1	2	3	No. of Clusters	1	2	3
Nasa data set				ClarkNet data set			
5	56.19	57.23	57.95	4	49.94	51	51.09
10	53.92	55.07	56.01	6	48.69	49.86	50.4
15	52.3	53.77	54.68	8	48.42	49.63	50.22
20	48.96	50.58	52.10	10	47.18	48.64	48.97
25	48.67	50.02	50.42	20	40.95	43.25	44.45
30	48.33	49.37	50.58	30	37.69	38.48	41.23

We conduct further experiments to validate our claim that adding new user sessions and deleting unused nodes yields an improvement in the prediction accuracy, as well as speed-up of the recommendation time. The size of the buffer can be determined according to the rotation frequency of Web server logs. We try different buffer sizes and different y values for deleting the nodes from CSTs. The experiments show that determining the buffer size as $1/9$ of the test set and the y value as $1/3$ of the test set decreases the recommendation time significantly with an improvement in the prediction accuracy. The more frequent the insertion to the CSTs, the greater the prediction accuracy. However, deleting the nodes with the same frequency decreases the prediction accuracy. Thus, setting the deletion frequency as three times that of the insertion frequency is reasonable. In real time the insertion frequency can be determined as weekly. All experiments are performed on a Pentium IV, 2.4 GHz computer with a 512 MB main memory running Microsoft Windows XP. The programs are coded in Java without any code optimization.

Table 4 shows the results of the NASA data set where the visiting time of pages are normalized between 1 and 2 and insertion and deletion to the clusters are performed periodically. The results in Table 5 are obtained without modifying the clusters according to the new user sessions. Tables 6 and 7 show the results of ClarkNet data set with and without modifying the CSTs, respectively. The *Time* field in the tables presents the average time spent in milliseconds to produce one recommendation set. As can be seen from the tables,

Table 4 Hit-Ratio in % and time in ms. for the NASA data set. The CSTs are modified

	Top-N					
	1		2		3	
No. of Clusters	H-R	Time	H-R	Time	H-R	Time
5	58.75	0.06	60.02	0.08	60.72	0.10
10	56.07	0.04	57.51	0.05	58.03	0.07
15	53.741	0.03	53.51	0.04	56.37	0.06
20	52	0.03	53.51	0.04	54.85	0.06
25	50	0.03	52.2	0.47	53.60	0.4
30	48	0.02	51.15	0.03	52.3	0.04

Table 5 Hit-Ratio in % and time in ms. for the NASA data set. The CSTs are not modified

	Top-N					
	1		2		3	
No. of Clusters	H-R	Time	H-R	Time	H-R	Time
5	57.41	0.15	59.22	0.19	59.79	0.20
10	54.68	0.08	56.15	0.13	57.18	0.17
15	52.61	0.08	54.65	0.09	55.95	0.13
20	50.79	0.05	52.47	0.07	53.51	0.13
25	49.59	0.05	52.17	0.07	53.11	0.12
30	48.75	0.04	51.29	0.06	52.15	0.09

Table 6 Hit-Ratio in % and time in ms. for the ClarkNet data set. The CSTs are modified

	Top-N					
	1		2		3	
No. of Clusters	H-R	Time	H-R	Time	H-R	Time
4	54.18	0.05	54.83	0.08	55.14	0.08
6	52.34	0.04	52.31	0.07	53.09	0.08
8	52.26	0.04	52.34	0.06	53.02	0.05
10	50.06	0.02	50.74	0.05	51.19	0.07
20	44.14	0.02	45.5	0.03	46.593	0.05
30	35.75	0.01	39.77	0.01	40.76	0.03

Table 7 Hit-Ratio in % and time in ms. for the ClarkNet data set. The CSTs are not modified

	Top-N					
	1		2		3	
No. of Clusters	H-R	Time	H-R	Time	H-R	Time
4	53	0.1	53.84	0.09	54.07	0.14
6	50.53	0.08	50.81	0.11	51.40	0.11
8	49.65	0.08	50.01	0.09	50.95	0.12
10	48.22	0.04	48.65	0.08	49.01	0.10
20	39.9	0.03	41.51	0.03	42.13	0.08
30	35.65	0.02	37.74	0.04	39.19	0.04

adding relevance feedback reduces the time to produce the recommendation set significantly as well as increasing the prediction accuracy. 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 relevance feedback performs mostly better.

4.1. Comparative evaluation

In this subsection, we compare our model with previously proposed models. For comparison it is useful to classify approaches into three groups according to the data structure they use for representing user sessions:

1. Those that represent user sessions using only the time that a user spends on each page during her visit [19, 27];
2. Those that represent user sessions by association rules that capture the relationships among pages based on their patterns of co-occurrence across user sessions [23, 43];
3. Those that represent user sessions by ordering information of Web pages [13].

Table 8 Comparison of recommendation models

Data Set	Metric	PACT	Ass. Rules	Markov Chains	ICST. Tree
NASA	Prec.	4	–	–	48.63
	H-R	–	47.84	52.6	60.72
C.Net	Prec.	15	–	–	49.62
	H-R	–	49.3	50.08	55.14

There is no other recommendation model that uses both the ordering information and the time information, which makes a direct comparison difficult. For evaluating the performance of our method, we run the experiments with the same training and test examples using 3 other recommendation methods: Profile Aggregations based on Clustering Transactions (PACT) [27], Personalization Based on Association Rule Discovery from Web Usage Data [28], and Link Prediction and Path Analysis Using Markov Chains [35] (these methods are discussed further in the next section). The first model uses only time information for representing the user sessions. The comparison between the first model and ours will show the effect of using ordering information. The second model uses only association rules and it does not use clustering approach. The comparison between ICST and this model will show the effect of our clustering criteria and similarity measure. The last model, Markov chains, is based only on the ordering information. Since it is unable to use the time information, the comparison will show us the effect of using time information. In addition, this comparison will show the effect of using the whole path of user sessions. Finally, the results will show the effect of relevance feedback, since the ICST model is the only one that can be modified during the recommendation process.

Table 8 shows the comparison of the models. Since the hit-ratio metric has not performed well for the model in [27] (PACT in Table 8), it is not presented here and we use the precision metric as proposed in [27] for evaluation. For this metric, each session is divided into two parts. The first 2 pages of the session are used for generating the recommendation set while the remaining portion of session is used to evaluate the generated recommendations. We obtain the best result with 23 clusters for NASA data set and 4 clusters for ClarkNet data set. The precision is 4% and 15% for NASA and ClarkNet data sets, respectively. The other two models are evaluated using hit-ratio metric since they perform better in this case. For the method in [28] (Ass. Rules in Table 8) we use a sliding window with a window size of 2. The sliding window is the last portion of the active user session to produce the recommendation set. Thus, the model begins to produce the recommendation set after the first two pages of the active user session. We set the support for association rule generation to a low value (such as 1%) in order to have good prediction accuracy. The hit-ratio for NASA and ClarkNet data sets are 47.84% and 49.30%, respectively. ICST has a hit-ratio of 60.72% for NASA data set and 55.14% for ClarkNet data set. Clearly our method is superior. Another difference between our model and this one is that our model begins to produce the recommendation set after the user's first request. For the last set of experiments we use first order Markov models (Markov Chains in Table 8) [35]. The parameters of the Markov model are learned using Expectation-Maximization algorithm [14]. The best results for NASA data set is with 23 clusters giving a hit-ratio is 52.6%, while for the ClarkNet data set it is 4 clusters with a hit-ratio of 50.08%. These results prove that our model performs better than the previous proposed models.

5. Related work

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

Collaborative filtering techniques predict the utility of items of an active user by matching, in real-time, the active user's preferences against similar records (nearest neighbors) obtained by the system over time from other users [5]. A shortcoming of these approaches is that it becomes hard to maintain the prediction accuracy in a reasonable range while handling the large number of items in order to decrease the on-line prediction cost. Some approaches are proposed to handle these problems, but they all have high execution cost [36].

There have been attempts to use association rules [31], sequential patterns [1], and Markov models [15, 35] 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 pattern 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. Other than higher order Markov models, none of these techniques capture the entire behavior of a user in a session. Since the number of parameters for higher order Markov models is 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 large. The compact structure of the CST used in our model makes it possible to keep the entire structure of a user session without any information loss, which is characteristic of higher order Markov models (if the length of a user session is n, like nth order Markov models). Furthermore, the similarity measure 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.

As an alternative to the methods discussed above, the system described in [32] clusters user sessions using a fuzzy clustering algorithm and allows a page or user to be assigned to more than one cluster. However, the work has not been extended to show how these clusters can be used for predicting the user's next request. The methods in [4, 42] cluster user sessions based on a similarity measure between each session. Our method for clustering has a similar basic idea, but our similarity measure is different since it considers the distance between matching pages. Furthermore, we extend our work by representing each cluster by a CST to use these clusters for predicting the user's next request.

Page recommendations in [27] are based on clusters of pages found from the server log for a site whereas the recommendations in [28] are based on association rule discovery from usage data. The former model uses the time information without considering the visiting order of pages. The experimental results of this model are not satisfactory in our case. Although the model based on association rules [28] does not use the time or the order information, it performs better. However, our model outperforms these models since it uses two kinds of information about user sessions: The order of visited pages, and the time spent on them. The model proposed in [35] uses only the order information. But, since it is a first order Markov model it does not use the whole path that a user follows during her visit. It is obvious that the recommendation time would increase if the model uses higher order Markov models. Thus, our ICST outperforms this model due to the fact that ICST uses the whole path of a user as well as the time that a user spends on each page. The crucial differences between our model and these previous models are that we consider both the order of pages and the time spent on those pages and our model enables clustering of user sessions which reduces the search space and thus the recommendation time. Furthermore, the CST 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.

The main difference between all the previous proposed methods and our ICST-model is that ICST is able to easily insert new user sessions and delete old user sessions which keeps the model up-to-date. This property of the ICST-model covers the dynamic characteristic of Web access log data. Consequently, as the experiments demonstrate, our model's prediction accuracy is superior.

6. Conclusion and future work

Web server access logs increase dynamically every time new users enter a Web site which leads to periodic rotation the log files by moving or deleting the existing logs. Thus, usage patterns extracted by the recommendation models may change over time. Then, all usage patterns extracted by some recommendation models have to be updated as well.

In this paper, we presented the first incremental recommendation model, based on CST model, for Web page prediction. The model (abbreviated ICST) represents the behavior of a Web user during a single visit to the Web site and model uses a similarity metric to find pair-wise similarities between user sessions by means of visited pages and visiting times. It also reflects the distance between matching pages of two user sessions. User sessions are partitioned based on that similarity metric using a graph partitioning algorithm. The resulting clusters are represented by CSTs. The compact structure of CSTs enables the insertion of new user sessions or deletion of unused nodes very easily. Thus, the feedback from new users of the site is incorporated to the model.

A performance evaluation of ICST model versus CST model using two different Web server access logs is presented, demonstrating that using the ICST model increases the prediction accuracy while significantly reducing the online recommendation time. We also compare our model to three other recommendation models. Results show that our model improves the efficiency and effectiveness significantly.

We are now extending the model in several ways. We are developing a similarity measure that simultaneously considers the structure and the content of a web site as well as the visiting order of the pages. Besides this, a better clustering algorithm may increase the accuracy of the recommendation model. Since graph clustering problem is NP-hard, heuristic algorithms are required. We are studying evolutionary heuristic algorithms and the preliminary results for this study are promising and suggest further study [22, 41].

References

1. R. Agrawal and R. Srikant, "Effective prediction of web-user accesses: A data mining approach," in : Proc. of the International Conference on Data Engineering (ICDE), Taipei, Taiwan, 1995.
2. C.C. Aggarwal, J.L. Wolf, and P.S. Yu, "Caching on the world wide web," IEEE Transactions on Knowledge and Data Enginnering, Vol. 11, No.1, 1999, pp. 95–107.
3. C.R. Anderson, P. Domingos, and D.S. Weld, "Relational markov models and their application to adaptive web navigation," in Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Edmonton, AB, Canada, 2002, pp. 143–152.
4. A. Banerjee and J. Ghosh, "Clickstream clustering using weighted longest common subsequences," in Proc. of the Wokshop on Web Mining, SIAM Conference on Data Mining, Chicago, IL, USA, 2001, pp. 33–40.
5. J.S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in Proc. of the Fourteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, USA, 1998, pp. 43–52.

6. B. Berendt, "Web usage mining, site semantics, and the support of navigation," in : Proc. of the Web Mining for e-Commenrce—Challengs and Opportinities Workshop (WEBKDD'00), Boston, MA, USA, August 2000.
7. B. Berendt, "Understanding web usage at different levels of abstraction: Coarsening and visualizing sequences," in: Proc. of the Mining Log Data Across All Customer TouchPoints Workshop (WEBKDD'01), San Francisco, CA, USA, August 2001.
8. B. Berendt and M. Spiliopoulou, "Analysis of navigation behaviour in web sites integrating multiple information systems," VLDB Journal, vol. 9, no. 1, 2000 (special issue on "Databases and the Web"), pp. 56–75.
9. S. Brin, and L. Pagepp, "The anatomy of large-scale hypertextual web search engine," in: Proc. Int. World Wide Web Conference, 1998, pp. 107–117.
10. K. Cahrter, J. Schaeffer, and D. Szafron, "Sequence alignment using fastlsa," in: Proc. Int. Conf. on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2000) 2000, pp. 239–245.
11. D. Cosley, S. Lawrence, and D.M. Pennock, "REFEREE: An open framework for practical testing of recommender systems using ResearchIndex," in Proc. of 28th International Conference on Very Large Databases, VLDB 2002, Hong Kong, 2002.
12. H. Dai and B. Mobasher, "Using ontologies to discover domain-level web usage profiles," in: Proc. 2nd Semantic Web Mining Workshop at ECML/PKDD-2002, Helsinki, Finland, August 2002.
13. A. Demiriz, "webSPADE: A parallel sequence mining algorithm to analyze the web log data," in Proc. of The 2002 IEEE International Conference on Data Mining (ICDM '02), Maebashi City, Japan, 2002, pp. 755–759.
14. A. P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," Journal of Royal Statistical Society, vol. 39, no. 1, 1977, pp. 1–38.
15. M. Deshpande, and G. Karypis, "Selective markov models for predicting web-page accesses," in Proc. of the First SIAM International Conference on Data Mining (SDM'2001), Chicago, IL, USA, 2001.
16. C. Ding, X. He, H. Zha, M. Gu, and H. Simon, "Spectral min-max cut for graph partitioning and data clustering," Technical Report TR-2001-XX, Lawrence Berkeley National Laboratory, University of California Berkeley, CA., 2001.
17. O. Etzioni, "The world wide web: Quagmire or gold mine," Communications of the ACM, vol. 39, no. 11, 1996, pp. 65–68.
18. E. Frias-Martinez and V. Karamcheti, "A predicition model for user access sequences," in Proc. International WEBKDD Workshop – Web Mining for Usage Patterns and User Profiles, 2002, Edmonton, Canada.
19. Ş. Gündüz and M.T. Özsu, "A user interest model for web page navigation," in Proc. of International Workshop on Data Mining for Actionable Knowledge (DMAK), Seoul, Korea 2003, pp. 46–57.
20. Ş. Gündüz and M.T. Özsu, "A web page prediction model based on click-stream tree representation of user behavior," in: Proc. Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'03, Washinton DC, USA, 2003, pp. 535–540, .
21. Ş. Gündüz and M.T. Özsu, "A poisson model for user accesses to web pages," in: Proc. Eighteenth International Symposium on Computer and Information Sciences, ISCIS'03, Lecture Notes in Computer Science, vol. 2869, (Springer-Berlin 2003), pp. 332–339.
22. Ş. Gündüz Öğüdücü and A.Ş. Uyar, "A graph based clustering method using a hybrid evolutionary algorithm," WSEAS Transactions on Mathematics, vol. 3, no. 3, 2004, pp. 731–736
23. X. Huang, A. An, N. Cercone, and G. Promhouse, "Discovery of interesting association rules from livelink web log data, in Proc. of The 2002 IEEE International Conference on Data Mining (ICDM'02), Maebashi City, Japan 2002, pp. 763–766.
24. R. Kosala and H. Blockeel, "Web mining research: A survey," ACM SIGKDD Explorations, vol .2, no. 1, 2000, pp. 1–15.
25. J.Li and O. R. Zaïane, "Combining usage, content, and structure data to improve web site recommendation," 5th International Conference on Electronic Commerce and Web Technologies (EC-Web 04), Springer Verlag LNCS 3182, Zaragoza, Spain, August 30- September 3, 2004, pp. 305–315.
26. S.K. Madria, S.S. Bhowmick, W.K. Ng, and E.-P. Lim, "Research issues in web data mining," in: Proc. 1st Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'99), November, 1999, pp. 303–312.
27. B. Mobasher, H. Dai,T. Luo, and M. Nakagawa, "Discovery of aggregate usage profiles for web personalization," in Proc. of the Web Mining for E-Commerce Workshop (WebKDD'2000), Boston, MA, USA, 2000.
28. B. Mobasher, H. Dai,T. Luo, and M. Nakagawa, "Effective personalization based on association rule discovery from web usage data," in Proc. of the 3rd ACM Workhop on Web Information and Data Management, Atlanta, GA, USA, 2001, pp. 9–15.

29. B. Mobasher, H. Dai, T. Luo, Y. Sun, and J. Zhu, "Combining web usage and content mining for more effective personalization," in: Proc. of the Intl. Conf. on ECommerce and Web Technologies (ECWeb), Greenwich, UK, September 2000.
30. M. Nakagawa and B. Mobasher, "A hybrid web personalization model based on site connectivity," in: Proc. In Proceedings of the WebKDD Workshop at the ACM SIGKKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 2003.
31. A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "Effective prediction of web-user accesses: A data mining approach," in Proc. of the WEBKDD Workshop, San Francisco, CA, USA, 2001.
32. O. Nasraoui, R. Krishnapuram, and A. Joshi, "Mining web access logs using a fuzzy relational clustering algorithm based on a robust estimator," in Proc. of Eight International World Wide Web Conference, Toronto, Canada, 1999.
33. P. Pirolli, J. Pitkow, and R. Rao, "Silk from a sow's ear: Extracting usable structures from the web," in: Proc. of CHI'96, 1996, pp.118–125.
34. J. Pitkow, and P.Pirolli, "Mining longest repeating subsequences to predict world wide web surfing," in: Proc. USENIX Symp. on Internet Technologies and Systems (USITS'99), 1999.
35. R.R. Sarukkai, "Link prediction and path analysis using markov chains," in Proc. of the Ninth International World Wide Web Conference, Amsterdam, Holland, 2000.
36. B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system – A case study," in Proc. of the WEBKDD 2000 Workshop at the ACM SIGKDD 2000, Boston, MA, USA, 2000.
37. C. Shahabi, A. Zarkesh, J. Adibi, and V. Shah, "Knowledge discovery from users web-page navigation," in: Proc. 7th Int. Workshop on Research Issues in Data Engineering, 1997, pp. 20–29.
38. J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache algorithms: Design, implementation and performance," IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 4, 1999, pp. 549–562.
39. R. Srikant and R. Agrawal, "Mining generalized association rules," Future Generation Computer Systems, vol. 13, no. (2–3), 1997, pp. 161–180.
40. 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, vol. 1, no. 2, 2000, pp. 12–23.
41. A. Ş. Uyar and Ş. Gündüz Öğüdücü, "A new graph-based evolutionary approach to sequence clustering," to be appear in: Proc. of the Fourth International Conference on Machine Learning and Applications (ICMLA'05), 2005.
42. W. Wang and O. R. Zaïane, "Clustering web sessions by sequence alignment," in Proc. of 13th International Workshop on Database and Expert Systems Applications, DEXA'02, Aix en Provence, France, 2002.
43. Q. Yang, H.H. Zhang, and I.T. Yi Li, "Mining web logs for prediction models in WWW caching and prefetching," in Proc. of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, USA, 2001, pp. 473–478.
44. NASA Kennedy Space Center Log, http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.
45. ClarkNet WWW Server Log, http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.
46. Cluto, http://www-users.cs.umn.edu/ karypis/cluto/index.html.