Personalized Recommendation with Adaptive Mixture of Markov Models

Yang Liu
Computer Science Department, York University, Toronto, M3J 1P3, Canada. E-mail: yliu@cs.yorku.ca

Xiangji Huang
School of Information Technology, York University, Toronto, M3J 1P3, Canada. E-mail: jhuang@cs.yorku.ca

Aijun An
Computer Science Department, York University, Toronto, M3J 1P3, Canada. E-mail: aan@cs.yorku.ca

With more and more information available on the Internet, the task of making personalized recommendations to assist the user's navigation has become increasingly important. Considering there might be millions of users with different backgrounds accessing a Web site everyday, it is infeasible to build a separate recommendation system for each user. To address this problem, clustering techniques can first be employed to discover user groups. Then, user navigation patterns for each group can be discovered, to allow the adaptation of a Web site to the interest of each individual group. In this paper, we propose to model user access sequences as stochastic processes, and a *mixture of Markov models* based approach is taken to cluster users and to capture the sequential relationships inherent in user access histories. Several important issues that arise in constructing the Markov models are also addressed. The first issue lies in the complexity of the mixture of Markov models. To improve the efficiency of building/maintaining the mixture of Markov models, we develop a lightweight adaptive algorithm to update the model parameters without recomputing model parameters from scratch. The second issue concerns the proper selection of training data for building the mixture of Markov models. We investigate two different training data selection strategies and perform extensive experiments to compare their effectiveness on a real dataset that is generated by a Web-based knowledge management system, *Livelink*.

Introduction

To assist Web surfers in browsing the Internet more efficiently, many research efforts have been made to analyze user browsing behavior by learning from Web logs. The navigation patterns extracted from those historical data can be used to predict which pages are likely to be clicked by a user given a sequence of pages that the user already visited. With the prediction, some pages can be recommended to the user by dynamically generating the links to the pages on the current browsing window of the user to help the user find relevant information more efficiently. This process can be interpreted as a *personalized recommendation* as it utilizes a surfer's particular preference obtained through studying his/her previous access data. Many other applications, such as dynamically adjusting the Web site layout for easy navigation or prefetching a document from the database server for reducing the response time for the next request, may also require such prediction results.

Motivation

The work presented in this paper is part of a research project whose general objective is to improve the functionality of *Livelink* using data mining techniques. *Livelink* is a Web-based system that provides automatic data management and information retrieval over an intranet or extranet. Consider a *Livelink* Web user who needs some information for his/her work. After logging into the Web site, he/she starts searching for the documents that are relevant to his/her interest, and this navigation procedure creates a sequence of requests recorded in the Web log.

Figure 1 shows such a sequence over a period of time, where the number in the first line indicates the identification (ID) of the information object that the user requests in a Web page, whose actual meaning in the *Livelink* context is illustrated in the second line. At the end of this sequence, the user has accomplished his/her goal; otherwise, further exploration might have been done to find what the user needs. Nonetheless, at this point, most existing retrieval

Accepted January 4, 2007

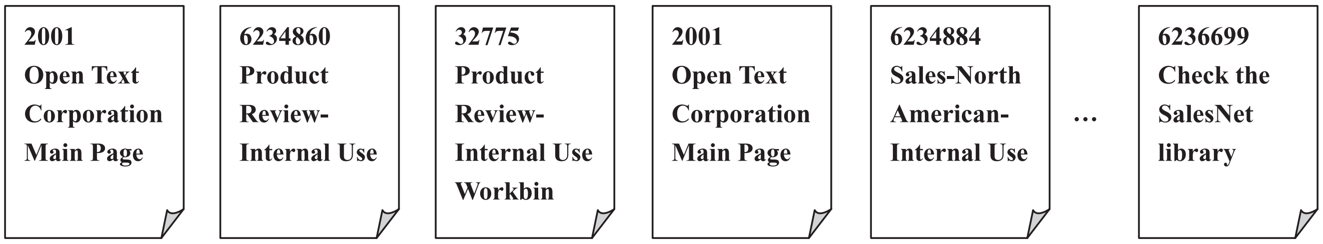


FIG. 1. Example of a clickstream.

systems are inept at providing effective and adaptive results to satisfy different users' individual requirements. For example, when a user has first viewed a series of documents as shown in Figure 1 and then made another query, a traditional retrieval system would return some results based only on this query without considering the fact that his/her previous access history may implicitly determine the intended meaning of the query. Therefore, the objective of our work is to discover distinct navigation patterns from the Web log data so that recommendations can be made to help the user find the target information more effectively.

Main Contributions

Analyzing Web users' navigation behavior with clustering techniques has presented two unique challenges: the immense volume of data and the sequentiality of user access patterns. Traditional distance-based clustering algorithms are ill suited to solve the problem. This is because for complicated data types (e.g., variable length sequences), defining a good distance measure is often data dependent. Another disadvantage of distance-based approaches is that calculating the pairwise distances between objects is computationally inefficient. In our work, we employ a model-based clustering approach, namely, a *mixture of Markov models* based approach, to analyze the user access patterns. Model-based clustering places cluster analysis on principled statistical support. It is based on probability models in which objects are assumed to follow a finite mixture of probability distributions such that each component distribution stands for a unique cluster. In our work, we propose to use Markov-based models to study the user's stochastic navigation processes. In a Markov model, it is assumed that the next requested page/object is only dependent on the last *n* pages/objects, where *n* may be 0 or any positive integer. This feature perfectly matches the typical user's browsing behavior while surfing the Web. With the help of the *mixture of Markov model*, we mainly investigate two research problems in this paper: (1) how to cluster Web users accurately on the basis of their previous navigation patterns and (2) how to predict the next request that is most likely to be accessed by the user. To cluster Web users, we start with analyzing a user's request sequences in the log history, and then find the cluster to which most of his/her sequences would belong. To recommend the user with the most possible request to be selected in the future, we feed the current request session into the mixture model, and let the transition matrix trained with historical data determine the optimal candidate. Through a set of carefully designed empirical studies, we demonstrate that our proposed method can achieve better performance than previous methods, e.g., making recommendation with a single Markov model or with association rule–based methods.

In addition, this paper addresses two important issues that arise in building the Markov models. The first issue is related to the high cost of training a mixture of Markov models in large-scaled applications. For such applications, the number of parameters to be estimated can be huge; therefore, the convergence of the training process can be slow. Also, training the model requires multiple scans of the data, the cost of which can be immense when the dataset is large. This would not be a problem if we were to train the model only once and use it for an extended period, as this cost can be amortized over time. However, in our application, new data arrive continuously; discarding those new data entirely without adjusting the model would adversely affect the performance of our system. On the other hand, using all available data to retrain the model also might be infeasible because of its high cost. Therefore, a more efficient, scalable, and adaptive approach is highly desirable. In this paper, we develop an adaptive algorithm specifically designed to update the parameters in the mixture of Markov models. The computation in the proposed algorithm involves only the new data and the current parameters in the model, does not require any iteration, and is independent of the old data. Therefore, it has very low time and space complexity and is orders of magnitude faster than retraining models from scratch. The second issue is concerned with the proper selection of training data for training the mixture of Markov models. Is it the case that the more recent the training data are, the more relevant they are to users' current interests? Do users' access patterns change over time? In order to answer those questions, we investigate two different training data selection strategies, *pyramidal time frame and uniform selection*, and perform extensive experiments to compare their effectiveness in our application domain.

We also discuss some data preprocessing issues for our real-world application data, which are log data from *Livelink,* a Web-based knowledge management system developed by *Open Text Corporation*. To handle the large volume of the *Livelink* log data, we propose to reduce the state space of Markov models by making use of the hierarchy of

Livelink objects. We also notice that because of the diversity of *Livelink* user navigation patterns, the lengths of request sequences vary greatly. In particular, there exist many short sequences. Most previous Markov-based pattern recognition models treat those short sequences as noise and therefore filter them out. However, they still contain useful information. At the same time, there are many multiple consecutive occurrences of the same object within a session, which may prevent the model from capturing users' actual intentions. To solve these problems, we design a few strategies and justify their effectiveness with experiments.

Our proposed system can also be integrated with traditional retrieval systems in many different ways. For example, (1) we can use the content of the predicted request, which is generated by the mixture of Markov models, to find additional terms to expand the current query; (2) the user clustering information, which is gained through mining surfers' access history, can be used to locate the common interests of a group of surfers and then rerank the query results retrieved with the original query. We show an example to demonstrate some possible usages in the *Livelink* application.

Evaluation

While the Web surfer's access patterns are extracted from Web log data, validation is needed to justify whether these patterns are really meaningful. In our paper, we develop a Web mining system to assist the user's navigation, so the best evaluation method in our case is to test it in the real world. In particular, we utilize the *Livelink* data as the test bed.

Classic distance-based clustering evaluation methods, such as sum of square (SSQ), proposed by Aggarwal, Han, Wang, and Yu (2003), which calculate the distance between an object and the cluster centroid, are not suitable in the model-based clustering domain. Other evaluation metrics, such as the Hubert and Arabie (HA) index proposed *by Milligan and Cooper* (1986), are applicable only when true clusters are known. However, in our case, we do not know the true clusters in our application data set. To solve this problem, we define a metric, called *user clustering accuracy*, which measures the similarity between the clustering result on the training data and the one on the testing data. This evaluation method presents a similar idea to those proposed by Cadez et al. (2000) and Anderson Domingos, and Weld (2002). However, they use the *negative log likelihood score* to measure the results of clustering *sequences*, whereas our strategy is specially designed for user clustering. To evaluate recommendation results, we introduce the concept of *hit ratio*. With this metric, we carefully design a set of experiments to evaluate the adaptive algorithm, which incrementally updates the model parameters, and compare *pyramidal time frame* and *uniform selection strategies* in terms of the recommendation performance. In addition, we compare the experimental results of our *mixture of Markov models* based recommendation method with those of association rule–based recommendation methods.

Organization

The rest of the paper is organized as follows. In the section Background we briefly describe the background. We present the Markov model based clustering algorithm and our adaptive algorithm in the section Mixture of Markov Models for Clustering. We introduce our personalized Web-surfing recommendation method in Personalized Web-Surfing Recommendation. In Application Data Description and Preprocessing we give an overview of the *Livelink* data and report our experience in preprocessing the raw data file and developing different strategies to select training data for the *mixture of Markov models*. Experimental results on the *Livelink* data sets are given in Performance Evaluation on a Real Application Data Set. Next we discuss Related Work then present Conclusion and Future Work.

Background

To gain a better understanding of users' access patterns, various Web clustering techniques that employ different similarity measures have been proposed to study Web surfers' browsing behavior. Current clustering techniques can be classified into two categories in the Web domain: distance-based approaches and model-based approaches (Zhong & Ghosh, 2003). Distance-based clustering computes the distance between pairwise data. Shahabi, Zakesh, Adibi, and Shah (1997) developed a similarity measure based on inner products over a feature space that describes users' navigation paths with viewing time as a primary feature and clustered the session files using the K-means algorithm. However, viewing time may not be a good indicator as users often tend to be distracted from the current navigation when surfing the Web. Fu, Sandhu, and Shih (1999) suggested using uniform resource locators (URLs) to construct a page hierarchy, which is then used to categorize the pages. Unfortunately, this approach works only if the URLs contain useful tokens. Also, the page hierarchy only specifies the hierarchy of the directory structure, not the concept hierarchy of a Web site. Cadez et al. (2000) proposed to use a mixture of Markov models for clustering browsing sessions into categories. However, this approach is only feasible when data dimensionality can be reduced by a prior manual categorization of Web pages. Smyth (1996a) described each time sequence with a hidden Markov model, but such a model is often difficult to build in a Web domain because of lack of sufficient information, and an individual's navigation pattern is likely to be disclosed by this case-based generalization.

Currently, most Web search engines and recommendation systems are designed to serve all users, regardless of the different needs of individuals. To address this problem, there has been work on *Web personalization* and *user modeling* techniques, which utilize Web usage mining algorithms to conduct retrieval for each user incorporating his/her interests. Zukerman and Albrecht (2001) categorized Web page recommendations into two main approaches: a content-based system constructs a model based on the contents of the

Web page, while a collaborative system tries to find similar users and assumes the current user will follow the same patterns already exhibited. A lot of effort has been made to build content-based systems. Yan, Jacobsen, Garcia-Molina, and Dayal (1996) developed a Web personalization system that dynamically provides hypertext links to the user. The whole system consists of two processes: the offline part builds surfer clusters using the feature vectors based on the Web access data, and the online part generates dynamic links that are directed to Web pages. To use the system, a surfer is first assigned to a single cluster according to his/her previous access path, then some dynamic link(s) is retrieved from the pages that have been accessed by other surfers in the same cluster. Schechter, Krishnan, and Smith (1998) attempted to predict the next action of a Web user by constructing a tree that contains the user's access paths. The prediction is conducted by matching the user's previous session as paths in the tree. The maximal prefix of each path (i.e., the first $N - 1$ requests of a length N) of the tree is compared with the same length suffix of user's access sequence, and the path with the highest number of matches is returned. The authors claim that storing longer paths in the tree offers improvements in predictions, but this method might be rigorous and cumbersome since every path has to store all its prefixes. Meanwhile, as a representative of collaborative systems, Shardanand and Maes (1995) employed the k-Nearest-Neighbour algorithm by comparing a given user's record with the historical records of other users in order to identify those users sharing similar interests with the given user. However, this technique suffers from some well-known limitations that are related to the scalability and efficiency of the kNN approach.

Mixture of Markov Models for Clustering

In this section, we first introduce the concept of *mixture of Markov models* and its learning algorithm. Then we present how to use the mixture model to cluster Web users and their request sequences. After that, we present our adaptive clustering algorithm and analyze its time complexity.

Markov Model Based Clustering Algorithm

Given a population of Web users, each user generates a series of requests with various lengths. The problem addressed in our work is to group request sequences and Web surfers themselves into K clusters. A probabilistic model for grouping sequences is that of a finite mixture model with K components

$$p(v|\theta) = \sum_{k=1}^{K} p(c_k|\theta)p(v|c_k,\theta) = \sum_{k=1}^{K} \pi_k p(v|c_k,\theta) \tag{1}$$

where v_i represents a particular request, $v = v_1v_2 \ldots v_L$ denotes a request sequence, $\pi_k = p(c_k|\theta)$ is the weight of the kth component c_k satisfying $\Sigma_k \pi_k = 1$, $p(v|c_k,\theta)$ is the probability of a request sequence v given the kth component, and θ denotes the parameters of the model. Here, we assume that $p(v|c_k,\theta)$ is a *first-order Markov model* with parameters $\theta = \{\theta_1, \ldots, \theta_K\}$, so that the probability distribution over the current state only depends on the previous state. Note that the reason for using a mixture model with multiple components instead of a single Markov model is to distinguish different groups of sequences. Each component represents the characteristics of a group of sequences. A single Markov model would represent the general characteristics of all sequences.

Formally, a first-order Markov model is a triple $<S, \theta^1, \theta^T>$, where

- $S = \{s_1, s_2, \cdots s_M\}$ is a set of states.
- $\theta^1_{k,i} = p(v_1 = s_i|c_k)$ is the probability that the kth component's initial state is s_i, and θ^1 is the collection of $\theta^1_{k,i}$ for all components and states.
- $\theta^T_{k,j\to l} = p(v_t = s_l|v_{t-1} = s_j, c_k)$ is the probability that the kth component transits from state s_j to s_l, and θ^T is the collection of $\theta^T_{k,j\to l}$ for all the components and transitions.

Also, $\theta^1_{k,j}$ and $\theta^T_{k,j\to l}$ satisfy $\sum_{j=1}^{M} \theta^1_{k,j} = 1$, and $\sum_{l=1}^{M} \theta^T_{k,j\to l} = 1$. Given a first-order Markov model, the probability of observing a sequence of states $v = v_1v_2 \ldots v_L$ under the kth component is

$$p(v|c_k,\theta) = p(v_1|\theta^1_k)\prod_{i=2}^{L} p(v_i|v_{i-1},\theta^T_k) \tag{2}$$

where θ^1_k denotes the probability distribution over the initial request among users in cluster k, and θ^T_k denotes the parameters of the probability distributions over transitions from one request to the next by a user in cluster k. Given a mixture of Markov models, we can assign a sequence x to a cluster by computing $p(c_k|x, \theta)$. Details of using this mixture model for sequence and user clustering are discussed in User and Sequence Clustering.

Learning the Model Parameters from Data

In our research, we assume that each individual behaves independently. If request sequences in the training data are denoted by

$$d_{train} = \{d^1_{train}, d^2_{train}, \ldots, d^N_{train}\}$$

we have

$$p(d_{train}|\theta) = \prod_{i=1}^{N} p(d^i_{train}|\theta) \tag{3}$$

For the sake of simplicity, we further assume that given a model and its parameters, each request sequence of an individual is independent of any other request sequence of that individual. To learn the parameters of a mixture model with known number of components, one possible solution is to identify those parameter values for $\theta = <\pi, \theta^1, \theta^T>$ that maximize the likelihood of the training data

$$\theta_{ML} = argmax_{\theta} p(d_{train}|\theta) \tag{4}$$

These parameters are often referred to as a *maximum likelihood (ML)* estimate. Alternatively, we may utilize prior knowledge about the training data and encode this information in the form of a *prior probability distribution* over the parameters (denoted with $p(\theta)$). The parameters of the mixture model can therefore be learned by identifying those parameter values that maximize the posterior probability of θ given the training data

$$\begin{aligned}\theta_{MAP} &= argmax_{\theta} p(\theta|d_{train}) \\ &= argmax_{\theta} \frac{p(d_{train}|\theta)p(\theta)}{p(d_{train})} \\ &= argmax_{\theta} p(d_{train}|\theta)p(\theta)\end{aligned} \tag{5}$$

where the second expression follows from the first one by the Bayes rule. These parameter estimations are referred to as *maximum a posteriori (MAP)* estimates, and we learn the parameters with the EM algorithm. The EM algorithm is an iterative algorithm used to learn the parameters of a mixture Markov model from the training data (Bilmes, 1997). It first initializes the parameters in the mixture model with random values. Then it iteratively performs two steps: the *E* step and the *M* step.

- In the *E* step, we calculate $p(c_k|d^i_{train}, \theta)$ for each training sequence under the model with the current parameter setting, where d^i_{train} is the *i*th sequence in training data set d_{train}

$$p(c_k|d^i_{train}, \theta) = \frac{\pi_k p(d^i_{train}|c_k, \theta)}{\sum_{t=1}^{K} \pi_t p(d^i_{train}|c_t, \theta)} \tag{6}$$

- In the *M* step, for each component (or cluster), parameters θ are updated to maximize the posterior probability of θ as follows

$$\pi_k = \frac{1}{N}\sum_{i=1}^{N} p(c_k|d^i_{train}, \theta) \tag{7}$$

$$\theta^1_{k,j} = \frac{\sum_{i=1}^{N} p(c_k|d^i_{train}, \theta)\sigma(d_{train1}^{\ i} = j)}{\sum_{d=1}^{M}\sum_{i=1}^{N} p(c_k|d^i_{train}, \theta)\sigma(d_{train1}^{\ i} = d)} \tag{8}$$

$$\theta^T_{k,j\to l} = \frac{\sum_{i=1}^{N} p(c_k|d^i_{train}, \theta)\delta_{j,l}(d^i_{train})}{\sum_{d=1}^{M}\sum_{i=1}^{N} p(c_k|d_{train}, \theta)\delta_{j,d}(d^i_{train})} \tag{9}$$

where $\sigma(d^{\ i}_{train1} = j)$ is an indicator function that equals 1 if the first request in d^i_{train} corresponds to state *j* and 0 otherwise, and $\delta_{j,l}(d^i_{train})$ denotes the number of transitions from state *j* to state *l* in d^i_{train}. Each iteration in the *EM* algorithm is guaranteed to increase the θ_{MAP}, and the algorithm finally converges to a local maximum of the likelihood function. These two steps are repeated until all parameters converge. This procedure offers a straightforward method to estimate the parameters $\theta = <\pi, \theta^1, \theta^T>$ of the mixture Markov model and guarantees that the parameters will ultimately converge.

User and Sequence Clustering

In order to describe how users and sequences are clustered, some notations are first defined as follows.

Definition 3.1. ($P^i_{j,k}(\theta)$) Given a mixture of Markov models with *k* components and parameters θ, the likelihood that the *j*th behavior sequence of user *i* (denoted as $Data^i_j$) belongs to cluster *k* is

$$P^i_{j,k}(\theta) = \frac{\pi_k p(Data^i_j|c_k, \theta)}{\sum_{t=1}^{k} \pi_t p(Data^i_j|c_t, \theta)} \tag{10}$$

$P^i_{j,k}(\theta)$ can be regarded as the weight of a sequence belonging to the *k*th cluster.

Definition 3.2. (C^i_j) Given $P^i_{j,k}(\theta)$ for the *j*th behavior sequence of user *i*, $k \in [1, K]$, where *K* is the number of total clusters, and C^i_j is defined as the cluster that has the maximum likelihood over all clusters

$$C^i_j = argmax_k P^i_{j,k}(\theta) \tag{11}$$

With C^i_j, we can assign a sequence into a single cluster. This way of clustering is called *hard clustering*. In this method, a request sequence is uniquely assigned to a cluster with the maximum likelihood $argmax_k P^i_{j,k}(\theta)$. This approach has a crispy boundary where the constituting elements only take two possible options of membership: i.e., they either belong or do not. However, this approach neglects the possibility that the sequence may partially belong to more than one component. For example, there are two components in the mixture of Markov models, and a sequence has the probability distribution of being in the two clusters of $\{P^i_{j,1}(\theta) = 0.51, P^i_{j,2}(\theta) = 0.49\}$. With the hard clustering method, it is classified into component 1, which holds the higher likelihood. However, it is very possible that the sequence also belongs to the second component, since the possibilities of being in the two different components are very close. To solve this problem, we adopt a more realistic soft clustering approach, where each element is given a weight of membership to each component as follows:

Definition 3.3. (CS^i_j) The sequence's probability distribution over all components is defined as

$$CS^i_j = \{P^i_{j,1}(\theta), P^i_{j,2}(\theta), \ldots, P^i_{j,K}(\theta)\} \tag{12}$$

where *K* is the number of total clusters.

In this way, soft clustering represents a partitioning-optimization technique that allows a request sequence to belong to several groups simultaneously. Similarly, we can define hard and soft clustering for users as follows:

Definition 3.4. (C^i) Given C^i_j for any $j \in [1, J]$, where J is the number of sequences user i has, C^i is defined as the cluster that has the maximum likelihood over all sequences of user i

$$C^i = argmax_j C^i_j \tag{13}$$

Definition 3.5. (CU^i) For a specific user i, let $P^i_{j,k}(k = 1, \ldots, K, j \in S_i)$ be the probability that sequence j belongs to cluster k, where S_i is the set of sequences user i has accessed. The probability that the user belongs to the kth cluster is

$$CU^i_k = \frac{\sum_{i \in S_i} P^i_{j,k}}{\sum_{k=1}^{K} \sum_{i \in S_i} P^i_{j,k}} \tag{14}$$

and the user's probability distribution over all the clusters is

$$CU^i = \{CU^i_1, CU^i_2, \ldots, CU^i_K\} \tag{15}$$

With hard clustering, user i is assigned to cluster C^i. With soft clustering, user i's probability distribution CU^i over all the clusters is generated.

Adaptive Clustering Algorithm

After obtaining a trained mixture of Markov models using past access data, we would like to make it adapt to the changes as new access data become available every day. We could conduct an overhaul reconstruction of the model based on both the old and the new data and obtain the new parameters. This approach, however, can be computationally prohibitive because of the large number of parameters usually involved in real applications.

To reduce the computational cost, we propose an adaptive algorithm that can incrementally update the parameters in the mixture of Markov models. As described in the section Learning the Model Parameters From Data, the process of learning parameters in a mixture of Markov models is divided into two steps. E step estimates the objective function $p(c_k|v, \theta)$ for each sequence in the training data, where $v \in d_{train}$, and M step updates unknown parameters $\theta = <\pi, \theta^1, \theta^T>$ for each cluster. From Equations 7, 8, and 9, we notice that if the number of total sequences N is sufficiently large, the addition of a new request sequence is unlikely to have significant influence on the parameters' estimation. Therefore, when a new request sequence v' becomes available, it is possible to update the parameters incrementally (though approximately) without invoking a reconstruction of the model, and the parameter values thus obtained are expected to be close to those obtained by retraining the model from scratch.

Our method works as follows: Once a new sequence v' is recorded, we first calculate $p(c_k|v', \theta)$ for each component $k \in K$ under the existing models. Second, as all three components of the model would change with the objective function $p(c_k|v', \theta)$, the parameters $\theta = <\pi, \theta^1, \theta^T>$ of the mixture of models are incrementally updated as illustrated in Algorithm 1 in Figure 2. In Algorithm 1, lines 3–5 update the objective functions by calculating the probability distribution of v' over all the components with the current parameter θ, and lines 7–17 describe how to reestimate the parameters in detail.

Comparing with the previous parameter estimations, where

$$\pi_k = \frac{\sum_{i=1}^{N} p(c_k|v,\theta)}{N}, \theta^1_{k,j} = \frac{\sum_{i=1}^{N} p(c_k|v,\theta)\sigma(v_1{=}j)}{\sum_{d=1}^{M}\sum_{i=1}^{N} p(c_k|v,\theta)\sigma(v_1{=}d)},$$

and $$\theta^T_{k,j\to l} = \frac{\sum_{i=1}^{N} p(c_k|v,\theta)\delta_{j,1}(v)}{\sum_{d=1}^{M}\sum_{i=1}^{N} p(c_k|v,\theta)\delta_{j,d}(v)},$$

the current ones reflect the changes of weights in each component when a new sequence is accessed.

We use separate "for" loops in Algorithm 1 for obtaining new parameters and for updating original parameters not only to enhance the readability but to maintain the validity of the algorithm. As shown previously, all π' and θ' are first updated with the new objective function, and then all original parameters are rewritten in a batch. This order has to be maintained because the calculation of π' and θ' depends on all the π and θ values obtained in the last iteration. If parameters θ_k and θ'_k were updated in the same "for" loop, for different components, π' and θ' would be generated with different θs, and that is obviously not appropriate. Also note that a "new" sequence in line 1 is defined as "any" sequence that arrives after the model is constructed; it thus can be a totally new sequence that was not presented in the training data or a sequence that is identical to a sequence in the training data.

Complexity Analysis

It is obvious that the incremental algorithm involves only simple arithmetic calculations and can therefore be carried out very efficiently. Retraining the model, on the contrary, requires considerable computational resources.

Generally, constructing an overhaul model requires a runtime of $O(KNL + KM^2)$ per iteration in EM, where M is the number of states in the model, K is the number of components, N is the number of sequences, and L is the average number of requests per sequence. If we further decompose the complexity, $O(KNL)$ involves the cost of computing δ, σ, and $p(c_k|v, \theta)$, whereas $O(KM^2)$ covers π, θ^1, and θ^T.

```
1:  while there exists a new sequence v' do
2:      /*get objective function; calculate the probability distribution of v' over all components*/
3:      for $k = 1$ to $K$ do
4:          $p(c_k|v', \theta) = \frac{\pi_k p(v'|c_k,\theta)}{\sum_{i=1}^{K} \pi_i p(v'|c_i,\theta)}$
5:      end for
6:      /* get new parameters $\theta' =< \pi', {\theta^1_{k,j}}', {\theta^T_{k,j\to l}}' >$ in M-step*/
7:      for $k = 1$ to $K$ do
8:          $\pi'_k = \frac{\sum_{i=1}^{N} p(c_k|v,\theta)+p(c_k|v',\theta)}{N+1}$
9:          ${\theta^1_{k,j}}' = \frac{\sum_{i=1}^{N} p(c_k|v,\theta)\sigma(v_1=j)+p(c_k|v',\theta)\sigma(v'_1=j)}{\sum_{d=1}^{M}\sum_{i=1}^{N} p(c_k|v,\theta)\sigma(v_1=d)+\sum_{d=1}^{M} p(c_k|v')\sigma(v'_1=d)}$
10:         ${\theta^T_{k,j\to l}}' = \frac{\sum_{i=1}^{N} p(c_k|v,\theta)\delta_{j,l}(v)+p(c_k|v',\theta)\delta_{j,l}(v')}{\sum_{d=1}^{M}\sum_{i=1}^{N} p(c_k|v,\theta)\delta_{j,d}(v)+\sum_{d=1}^{M} p(c_k|v',\theta)\delta_{j,d}(v')}$
11:     end for
12:     /* update original parameters $\theta =< \pi, \theta^1, \theta^T >$ with the new ones */
13:     for $k = 1$ to $K$ do
14:         $\pi_k = \pi'_k$
15:         $\theta^1_{k,j} = {\theta^1_{k,j}}'$
16:         $\theta^T_{k,j\to l} = {\theta^T_{k,j\to l}}'$
17:     end for
```

FIG. 2. Adaptive algorithm.

Assuming that it takes X iterations for *EM* to converge, the total cost is $O(X(KNL + KM^2))$. In real applications, where M and N are usually large, it usually takes a long time (i.e., many iterations) for the system to converge. For instance, it takes more than 50 hours to retrain our mixture of Markov models with the *Livelink* data, when $K = 50$, $N = 4000$, and $M = 120$.

In contrast, our adaptive algorithm avoids this obstacle by executing only one iteration, and the total complexity therefore is $O(KL(N + 1) + KM^2)$. Moreover, observing Algorithm 1, we notice that if we keep a record of every δ, σ, and $p(c_k|v, \theta)$ in the previously constructed model, only $\delta_{j,l}(v')$, $\sigma(v'_1 = d)$ and $p(c_k|v', \theta)$ are unavailable in the current parameters' reestimation, where $d, l, j \leq M$. In this way, the runtime cost of $O(KL(N + 1))$ can be further reduced to $O(KL)$, while the cost of obtaining new parameters $\pi'_k, {\theta^1_{k,j}}'$, and ${\theta^T_{k,j\to l}}'$ remains the same as $O(KM^2)$. Finally, the computational complexity of using the incremental algorithm is $O(KL + KM^2)$, and it takes less than a second to update our mixture Markov model with our *Livelink* application data set.

Personalized Web-Surfing Recommendation

In Web mining, different techniques have been used for predicting the action a user will take next given the sequences of actions he/she has already taken. In this section, we present how we make personalized surfing recommendation with a mixture of Markov models. We also describe how to make a Web-surfing recommendation with association rule deductions in previous methods. We will compare the two recommended methods in the experiments section.

Web-Surfing Recommendations with a Mixture of Markov Models

Given a request sequence v, we can make the Web-surfing recommendation (i.e., predicting the most probable next request) by combining the outputs of both components in the mixture. For this purpose, soft clustering is applied in the recommendation process. Suppose the mixture has K components. The contribution (or the weight) of the kth component in the overall recommendation is $p(c_k|v, \theta)$. Meanwhile, for each possible next request, which corresponds to a state in the Markov model, the likelihood of its being the next request according to the kth component can be obtained from the transition matrix θ^T_k. Therefore, given that s_j is the state corresponding to the last request in v, the likelihood that a state s_l (and its corresponding request) is the next state is

$$p(next = s_l) = \frac{\sum_{k=1}^{K} p(c_k|v,\theta) \times \theta^T_{k,j\to l}}{\sum_{l=1}^{M} \sum_{k=1}^{K} p(c_k|v,\theta) \times \theta^T_{k,j\to l}} \quad (16)$$

where the denominator is a normalization factor, and $p(c_k|v, \theta)$ is defined as

$$p(c_k|v,\theta) = \frac{p(c_k|\theta)p(v|c_k,\theta)}{\sum_{j=1}^{K} p(c_j|\theta)p(v|c_j,\theta)} \quad (17)$$

Given M possible states in the whole model, we sort them in decreasing order of their occurrence possibilities, i.e., $\{s_1, s_2, \ldots, s_M\}$, and use the ones with highest probabilities for recommendation[1]. This method of recommendation applies the soft sequence clustering method because it uses the probabilities that a sequence belongs to each component (i.e., $p(c_k|v, \theta)$) when computing the probability of a possible next state. An alternative is to use the hard clustering method to assign v into a cluster c_k with the highest $p(c_k|v, \theta)$ and use c_k's Markov model to make the prediction. However, our experiment (described in the section Comparison with Hard and Soft Clustering Strategy) shows that the averaged prediction (i.e., the one with soft clustering) achieves better prediction performance than the prediction from the most likely cluster. Please also note that we do not need to cluster users when making recommendations with a mixture of Markov models.

In Markov models, the one making the next recommendation by looking at the last action performed by the user is known as the *first-order Markov model*, where each action that can be performed corresponds to a state in the model. A somewhat more complicated model makes the recommendation by looking at the last k actions performed by the user, and this is generalized as the kth order Markov model. In many applications, first-order Markov models cannot successfully recommend the next action to be taken. This is because such models do not look far into the past to discriminate accurately different behavior models belonging to different users. However, making recommendations with higher-order models may suffer from a high state-space complexity, and the problem will become even worse in the Web domain.

Suppose $v = v_1v_2 \ldots v_L$ to be a sequence of length L, where $v_1, v_2, \ldots$, and v_L take values from a set of states S. If we assume the recommendation for the next action is based on a *single* first-order Markov model, by definition,

$$p(v_{L+1}|v) = p(v_{L+1}|v_L) \quad (18)$$

This is different from making recommendations using a first-order *mixture of Markov models*. If we assume that the recommendation for the next action v_{L+1} is based on the mixture of models, by definition,

$$\begin{aligned} p(v_{L+1}|v) &= \sum_{k=1}^{K} p(v_{L+1}, c_k|v) \\ &= \sum_{k=1}^{K} p(v_{L+1}|v, c_k)p(c_k|v) \\ &= \sum_{k=1}^{K} p(v_{L+1}|v_L, c_k)p(c_k|v) \quad (19) \end{aligned}$$

[1] In our experiment, we recommend the one with the highest probability for the purpose of evaluation.

The transition probabilities in the mixture model are a function of the membership weights $p(c_k|v)$ (equivalent to $p(c_k|v, \theta)$ in Equation 16). These weights are in turn a function of the history of the sequence (based on the Bayes rule) and typically depend strongly on the pattern of the behavior before v_L. Thus, the mixture model is more powerful than a single first-order Markov model, and an acceptable complexity can still be held by looking at the last actions from each component model.

Compared to making recommendations with a single Markov model, making recommendations with a mixture of Markov models can be viewed as a personalized recommendation process since it is capable of integrating both group access patterns using the transition matrix $p(v_{L+1}|v_L, c_k)$ and each surfer's individual preference implied by membership weights $p(c_k|v)$.

Previous Methods

Association rule mining is a technique used to capture the relationships among items on the basis of their co-occurrence patterns across transactions (Mobasher, Dai, Luo, & Nakagawa, 2001). Given a set of transactions, where each transaction contains a list of items, an association rule can be denoted as $A \Rightarrow B$, where A and B are disjoined sets of items.[2] Here, A is referred to as the left-hand side (LHS) of the association rule, and B is referred to as the right-hand side (RHS) of the association rule. The intuitive meaning behind such a rule is that transactions containing the items in A are also very likely to contain the items in B. Two common numeric measures used to evaluate the interestingness of each association rule are support and confidence. The *support* of an association rule $A \Rightarrow B$ is defined as the proportion of transactions that contain both A and B, and its *confidence* is defined as the proportion of transactions that contain B among the transactions that contain A. An association rule mining algorithm, such as Apriori (Agrawal et al., 1993), finds association rules that satisfy the user-specified minimal support and confidence thresholds. In general, the recommendation using association rules recommends item b when the user has bought or visited item a, where b is in the RHS of the association rule and a is in the LHS. To evaluate the performance of our mixture Markov model based recommendation method, we compare it with two association rule based recommendation methods. One is called the *global association rule based method*, and the other is called the *cluster-oriented association rule based recommendation method*. We describe the two methods below.

Recommendation with global association rules. In the global association rule based method, recommendations are based on a set of association rules learned from the whole training

[2] We treat each sequence as a transaction when learning association rules.

data. Thus, the rules capture the general patterns of all the users. In implementation of this global recommendation method, we use the following process to make a recommendation based on a set of global rules. When there is a new request sequence A' accessed, we choose the rule $\{L : A \Rightarrow B\}$ whose LHS A has the largest overlap with A'. In case there is more than one association rule that has the largest overlap with A', we take the union of the RHSs of these rules and recommend all the objects in the union to the user.

Recommendation with cluster-oriented association rules. Some recent studies have considered exploiting association rule mining in recommendation systems. However, most of them rely on discovering object correlations on the whole data set for all the users; thus those recommendations are not personalized and cannot exhibit user distinct characteristics. In this study, we also consider user clustering information to boost the recommendation performance as follows:

1. After building the mixture of Markov models, we first apply the soft sequence clustering method on the whole training data set, such that each request sequence from the training data is allocated to one or more clusters according to $P^i_{j,k}\theta$ in definition 3.1. Meanwhile, each cluster in this way accumulates a group of sequences.
2. For each component cluster, we discover a set of association rules under predefined minimal support and confidence thresholds from all the sequences in the cluster. Ideally, these association rules may cover all inherent associations that exist in the access requests of the cluster k. As the number of sequences in different clusters varies, the number of association rules built with those sequences may change dramatically. In our experiments with *Livelink* data sets, some clusters may contain hundreds of rules, but others may have only a few rules.
3. We apply the soft user clustering approach described in definition 3.5 to all training data to assign each user to one or more clusters.
4. When there is a new request sequence A' accessed by a user i, the user's clusters are identified. We select rules that have the largest LHS overlaps with A' among all association rules derived from those clusters. The objects in the union of RHSs of the selected rules are recommended to the user.

In our experiment section, we will compare our mixture Markov model based recommendation method with both global and cluster-based association rule recommendation methods.

Application Data Description and Preprocessing

Livelink is a database driven Web-based knowledge management system developed by *Open Text Corporation*. It provides a Web-based environment (such as an intranet or extranet) to facilitate collaboration between cross-functional employees within an organization. In particular, *Livelink* offers document management, virtual team collaboration, information retrieval, and business process automation. It assists the storage, sharing, and management of critical information and processes for organizations, teams, and individuals.

The Data Set

The data set used in our experiments is the *Livelink* access data over a period of 34 days (Huang, Peng, An, & Schuurmans, 2004). The original data file is about 7 GB, and it describes more than 3 million requests made to the *Livelink* server by 1,157 users from around 5000 different Internet Protocol (IP) addresses. Each request to *Livelink* maps to a data entry in the log data containing an IP address from which the user is making the request, the cookie of the browser, the time a request was issued and responded to, the name of the request's handler, the name of the function used to handle a request, a query string containing the information about the requested object, and other information, such as the server name. A small portion of entries contain user Ids. We extract a complete user request log entry from *Livelink* and illustrate it in Figure 3. For security reasons, some of the lines are removed.

The users in this data set are mostly the employees of Open Text. They belong to different groups, such as human resources and product development. Different groups of users may have different types of authorizations for accessing the information objects managed by *Livelink*. In the section Object Hierarchy, we describe the hierarchy of the information objects maintained by *Livelink*. Some objects in the hierarchy can be accessed by certain groups of users. Thus, a user's access pattern is affected by the group(s) to which the user belong.

Data Preprocessing

Given log data, we are only interested in those relevant to the understanding of users' browsing behavior. To select relevant information, we conduct data preprocessing on the raw data by (1) identifying users from data entries, (2) extracting the time a request is made, (3) extracting the ID of the information object requested in each entry, (4) removing noisy entries (e.g., the requests having no information object), and (5) identifying user sessions. In Web usage mining, a Web log session is defined as a group of requests made by a single user for a single navigation purpose (Huang, Peng, An, & Schuurmans 2004). Users are identified by combining user Ids that appear in some data entries with IP addresses. When an entry does not contain user IDs, we consider it to belong to the user whose ID appeared in a previous data entry that shared the same IP address with the current entry under the condition that the time difference between the two entries is within a specified time interval. This method assumes that a user does not change the IP address within a reasonable period, as is often the case in the *Livelink* environment. After users are identified, the data entries for each user are sorted according to the requesting time, then further grouped into user sessions using the *time-out* session identification method, in which a

```
Wed Apr 10 19:22:52 2002
CONTENT_LENGTH = '0'
func = 'll'
GATEWAY_INTERFACE = 'CGI/1.1'
HTTPS = 'on'
HTTPS_KEYSIZE = '128'
HTTPS_SECRETKEYSIZE = '1024'
HTTPS_SERVER_ISSUER = 'C=US, O="RSA Data Security, Inc.", OU=Secure Server
Certification Authority'
HTTPS_SERVER_SUBJECT = 'C=CA, S=Ontario, L=Waterloo, OU=Terms of use at
www.cibc.com/verisign/rpa (c)99, OU=Authenticated by CIBC'
HTTP_ACCEPT_ENCODING = 'gzip, deflate'
HTTP_ACCEPT_LANGUAGE = 'en-us'
HTTP_CONNECTION = 'Keep-Alive'
HTTP_COOKIE = 'WebEdSessionID=05CAB314874CD61180FE00105A9A1626;
LLInProgress=\%2FI0PiE0OD4iNz4iMzk3Py8vIA;
LLCookie=\%2FI0PiE0OD4iNz4iMzk3MHhvZHd\%2Fb28hbW9uaWVifyEkaWFuYW8gAA;
HTTP_HOST = 'intranet.opentext.com'
HTTP_REFERER =
'https://intranet.opentext.com/intranet/livelink.exe?func=doc.Vie
wDoc&nodeId=12856199'
HTTP_USER_AGENT = 'Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)'
objAction = 'viewheader'
objId = '12856199'
PATH_TRANSLATED = 'C:\Inetpub\wwwroot'
QUERY_STRING = 'func=ll&objId=12856199&objAction=viewheader'
REMOTE_HOST = '24.148.27.239'
REQUEST_METHOD = 'GET'
SCRIPT_NAME = '/intranet/livelink.exe'
SERVER_NAME = 'intranet.opentext.com'
SERVER_PORT = '443'
_REQUEST = 'llweb'
Wed Apr 10 19:22:52 2002 - 638968       Func='ll.12856199.viewheader'
Timing:.140 0A<1,0,'Number_of_Delete_Statements'=0,'Number_of_Insert_Statements'=0,
'Number_of_Other_Statements'=0,'Number_of_Select_Statements'=7,
'Number_of_Update_Statements'=0,'OutputTime'=47,'Total_Execute_Time'=16,
'Total_Fetch_Time'=31,'Total_SQL_Statements'=7,'Total_SQL_Time'=47>
04/10/2002 19:22:52     Done with Request on socket 069DC4B0
04/10/2002 19:22:57     Processing Request on socket 09A87EF8
```

FIG. 3.   A Livelink log entry.

session shift is identified between two consecutive requests if the interval between them is more than a predefined threshold. We then assume that all sequences obtained at this stage are independent and uniform. In this paper, a *request sequenc*e refers to an object sequence in a user session. A piece of the session file after identifying the three sessions is shown as follows:

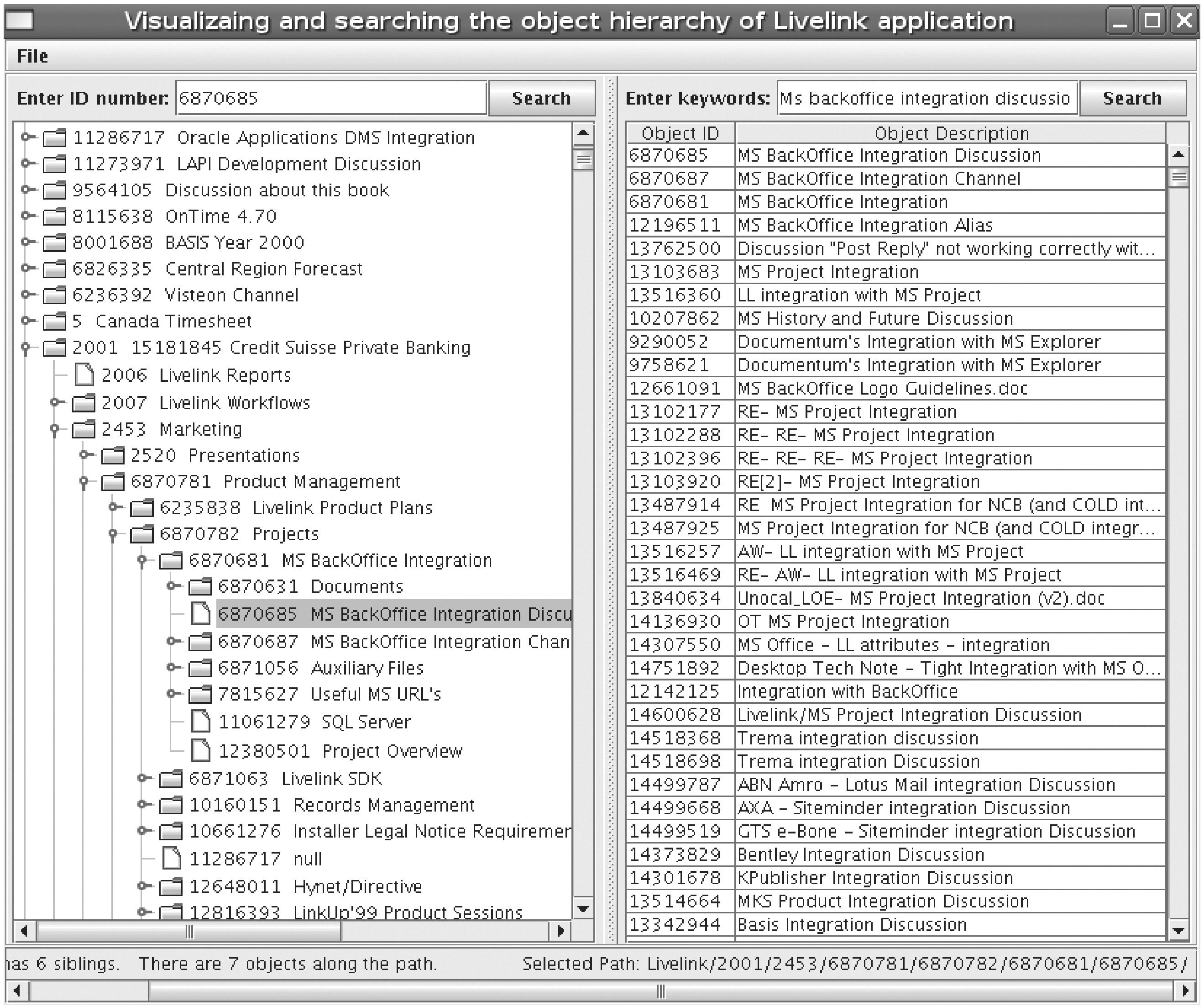
UserId	time	objectId
1	04/12/2002-12:06:25	14655738
1	04/12/2002-12:07:33	15199366
1	04/12/2002-12:07:37	14655738
1	04/12/2002-12:34:53	14655738
1	04/12/2002-12:34:55	14655738
1	04/12/2002-12:35:22	15199366
1	04/12/2002-12:36:38	15199366
1	04/12/2002-12:36:53	15291602
2	04/18/2002-03:01:04	13972781
2	04/18/2002-03:01:54	15291602
2	04/18/2002-03:02:04	15291602
2	04/18/2002-03:02:13	15291602
2	04/18/2002-03:02:22	62349805
2	04/18/2002-03:02:31	13972781

where blank lines separate sessions.

Object Hierarchy

The information objects maintained by *Livelink* are organized into a forest of trees. Each tree in the forest contains objects and folders that relate to a subject. A leaf node in a tree corresponds to a document, such as a Portable Document Format (PDF) file, a Power Point (PPT) document, or a picture. A nonleaf node of a tree indicates a folder that holds links to other folders or objects. This tree structure is constructed for better file storage and is totally transparent to the user. According to our domain experts, there are 2028 different trees maintained by the *Livelink* server, but only about 10% of the trees are "public trees," which are accessible to all or some of the users, and others are "private trees" that can only be accessed by a single user.

Figure 4 shows part of a public tree on its left-hand side. The whole picture shows the user interface for a search function. The right-hand side of the interface offers a keyword search function and shows the search result for the keyword "MS Backoffice Integration Discussion." The search result lists the object ID and the brief description of each retrieved object. With an object ID, we can use the search


Visualizaing and searching the object hierarchy of Livelink application
File
Enter ID number: 6870685
Search
Enter keywords: Ms backoffice integration discussio
Search

```
11286717  Oracle Applications DMS Integration
11273971  LAPI Development Discussion
9564105  Discussion about this book
8115638  OnTime 4.70
8001688  BASIS Year 2000
6826335  Central Region Forecast
6236392  Visteon Channel
5  Canada Timesheet
2001  15181845 Credit Suisse Private Banking
    2006  Livelink Reports
    2007  Livelink Workflows
    2453  Marketing
        2520  Presentations
        6870781  Product Management
            6235838  Livelink Product Plans
            6870782  Projects
                6870681  MS BackOffice Integration
                    6870631  Documents
                    6870685  MS BackOffice Integration Discu
                    6870687  MS BackOffice Integration Chan
                    6871056  Auxiliary Files
                    7815627  Useful MS URL's
                    11061279  SQL Server
                    12380501  Project Overview
                6871063  Livelink SDK
                10160151  Records Management
                10661276  Installer Legal Notice Requiremer
                11286717  null
                12648011  Hynet/Directive
                12816393  LinkUp'99 Product Sessions
```

Object ID	Object Description
6870685	MS BackOffice Integration Discussion
6870687	MS BackOffice Integration Channel
6870681	MS BackOffice Integration
12196511	MS BackOffice Integration Alias
13762500	Discussion "Post Reply" not working correctly wit...
13103683	MS Project Integration
13516360	LL integration with MS Project
10207862	MS History and Future Discussion
9290052	Documentum's Integration with MS Explorer
9758621	Documentum's Integration with MS Explorer
12661091	MS BackOffice Logo Guidelines.doc
13102177	RE- MS Project Integration
13102288	RE- RE- MS Project Integration
13102396	RE- RE- RE- MS Project Integration
13103920	RE[2]- MS Project Integration
13487914	RE MS Project Integration for NCB (and COLD int...
13487925	MS Project Integration for NCB (and COLD integr...
13516257	AW- LL integration with MS Project
13516469	RE- AW- LL integration with MS Project
13840634	Unocal_LOE- MS Project Integration (v2).doc
14136930	OT MS Project Integration
14307550	MS Office - LL attributes - integration
14751892	Desktop Tech Note - Tight Integration with MS O...
12142125	Integration with BackOffice
14600628	Livelink/MS Project Integration Discussion
14518368	Trema integration discussion
14518698	Trema integration Discussion
14499787	ABN Amro - Lotus Mail integration Discussion
14499668	AXA - Siteminder integration Discussion
14499519	GTS e-Bone - Siteminder integration Discussion
14373829	Bentley Integration Discussion
14301678	KPublisher Integration Discussion
13514664	MKS Product Integration Discussion
13342944	Basis Integration Discussion

```
as 6 siblings.   There are 7 objects along the path.        Selected Path: Livelink/2001/2453/6870781/6870782/6870681/6870685/
```

FIG. 4. Hierarchical structure in Livelink data.

function provided on the left-hand side of the interface to locate the object on the tree structure, from which we can trace the ancestors and descendants of the object. In this example, the object that is most relevant to the keyword is "6870685 MS BackOffice Integration Discussion," which is a leaf node in the tree rooted at "2001 15181845 Credit Suisse Private Banking."

State Representation in Markov Models

There are 35,039 information objects in our data set. If we represent the states of the Markov model using the objects, the transition matrix is prohibitively huge and extremely sparse. In order to reduce the number of states, we make use of the *Livelink* Object Hierarchy to generalize the objects by replacing them with their parents or ancestors. We first used the root of each tree to replace the objects in the tree, the process resulted in 655 objects.[3] For instance, as shown in Figure 4, if an object "MS Backoffice Integration Discussion" is requested by a user, its root object "2001 15181845 Credit Suisse Private Banking" could substitute for the original object. To reduce the number states further, we remove the objects that are in private trees, because a private tree is owned by a single user and is not suitable for describing the general property of a group of users. After removing the private objects, 104 objects (which correspond to the roots of 104 trees) are left. However, by observing the resulting session files, we found that over 70% of the requests visit a common tree, which is "2001 15181845 Credit Suisse Private Banking." Thus, a great portion of requests

[3] There are 2028 trees in total. However, not all the trees have objects that appear in our data set.

are identical. Although this fact may potentially increase the clustering or prediction accuracy, it also can be data dependent and does not help distinguish users in real applications. To solve this problem, we move down to the second level of this heavily visited tree and use the children of the root to represent the objects in this tree. In that way, more objects are introduced in our work.

Data Selection

Selecting proper training data for the mixture of Markov models is of critical importance for two reasons. First, there is a possibility that users' access behavior may dynamically change over time. Second, the immensity of users' profiles and the magnitude of servers' computational cost make it unrealistic to use all available information to construct the Markov models. Therefore, only a subset of all available data can be used as training data.

In this section, we study two different training data selection strategies. The first strategy utilizes a *pyramidal time frame*, which gives more recent data greater chances of being included in the training data, while the second strategy, called the *uniform selection strategy*, treats all data equally, regardless of their respective recency, and simply assigns all data the same probability of being selected. The pyramidal time frame strategy, originally used in data stream clustering (Aggarwal, Han, Wang, & Yu, 2003), is studied here to test the hypothesis that more recent data reflect the user's current access pattern more accurately. Without loss of generality, we assume that the starting and ending times of all available data are 0 and T respectively. The proposed method divides the time interval $[0, T]$ into $[\log_2 T]-1$ subintervals, with the length of the ith subinterval $\frac{T}{2^i}$, where $i < [\log_2 T]-1$. That is, the interval $[0,T]$ is divided into $[0, [\frac{T}{2}]], [[\frac{T}{2}] + 1, [\frac{T}{2} + 1 + [\frac{T}{4}]], \ldots, (T - 1,T]$. The training data consist of an equal amount of request sequences drawn from each subinterval. For example, given $T = 55$, the time interval $[0, 55]$ can be divided into $[\log_2 55] - 1 = 4$ subintervals: $\{[0, 28], [29, 45], [46, 53], [54, 55]\}$. For each request sequence in the available data, we assign it into a unique subinterval, so that its starting time is close to the subinterval's lower boundary. For example, if the access time of the first request in a sequence is 30, it should be assigned to the subinterval [29, 45]. In this way, each subinterval covers a list of sequences. Then we can randomly pick an equal amount of sequences in each subinterval, e.g., 500, to form a complete training data set. Clearly, with this data selection method, more recent data have a greater probability of being included in the training data.

For the uniform selection strategy, a random sample of the request sequences is drawn from all the data within the time interval $[0, T]$. This effectively assigns equal weight to all available request sequences and ignores the recency of data.

Performance Evaluation on a Real Application Data Set

In this section, we evaluate our method on a real world application, *Livelink* Web log data. First, we define two evaluation measures, *user clustering accuracy* and *hit ratio*. Then we conduct evaluation on user clustering, recommendation, and our proposed adaptive algorithm in the sections that follow. To avoid the local optimal problem that associates with the EM algorithm, all mixture of Markov model based experimental results shown in this section are an average of five runs. We implemented the Apriori algorithm with C++, and other algorithms with Matlab. All experiments are conducted on our departmental host server with the same central processing unit (CPU) (dual Xeon 3.06 GHz) and memory (2 GB) settings.

Evaluation Measures

In our experiment, we separated our session file into a training data set and a testing data set, where two subsets are of equal size and have no clear difference. The training data are used to train a mixture K-component Markov model. The trained model is used to cluster the users in both training and testing sets. To measure the performance, we introduce the concept of *user clustering accuracy,* which measures the similarity between the clustering result on the training data and the one on the testing data. In addition, the number of components of the mixture model is determined by using cross-validation on the training data.

Definition 6.1 (user clustering accuracy). Let $U = \{u_1, u_2, \ldots, u_N\}$ be the set of users that appear in both training and testing data. Suppose that user u_i ($i \in [1, N]$) is clustered into cluster a_i by the Markov model according to the user's sequences in training data and it is clustered into cluster b_i according to the user's sequences in the test data. Let M be the number of users in U that satisfy $a_i = b_i$ for $i \in [1,N]$. *User clustering accuracy* is defined as the ratio of M to N.

This evaluation approach presents an idea similar to *negative log likelihood* (Cadez et al., 2000; Anderson, Domingos, & Weld 2002). In *negative log likelihood*, a clustering model is first constructed with the training data; then the testing data are used to fit into the model to measure the out-of-sample degree. However, their objective is to evaluate the sequence clustering results, while our method is specially designed for user clustering.

To evaluate the recommendation results, we adopt the concept of *hit ratio*. Hit ratio was originally used as a chief measurement of a cache, which is the percentage of all accesses that are satisfied by the data in the cache. Recently, hit ratio has been used in the Web mining domain to measure the performance of Web page recommendation systems (Gunduz & Ozsu, 2003; Manavoglu, Pavlov, & Giles, 2003). Given a test sequence of requests, a hit is declared if what the system recommends on the basis of a subsequence

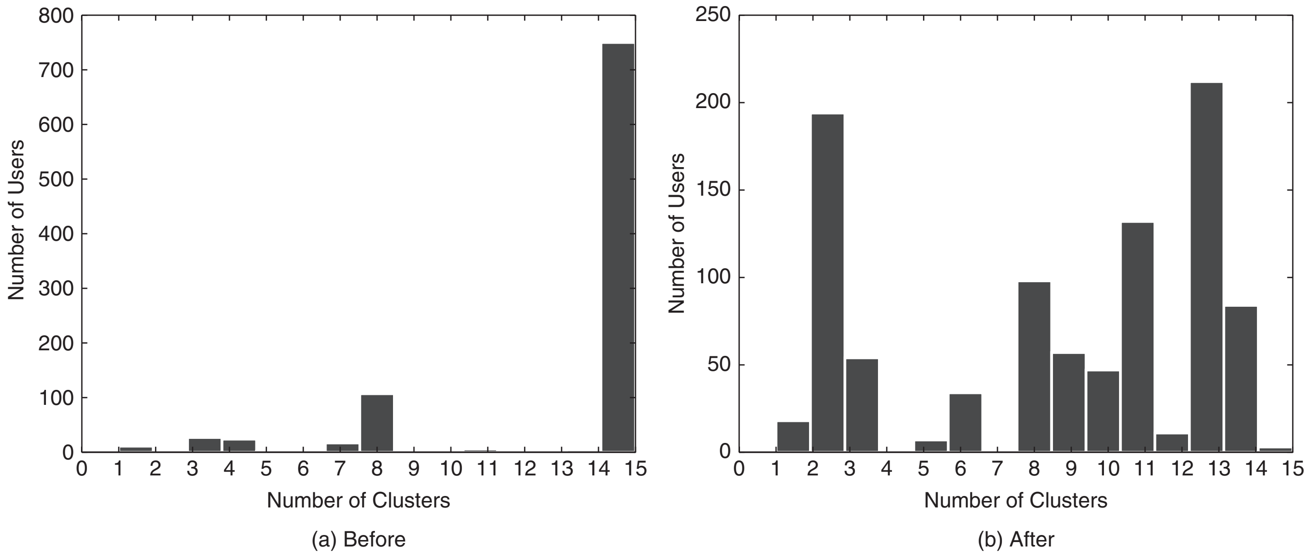


FIG. 5. User clustering before and after adjustment.

covers the next request in the sequence. *Hit ratio* is defined as the number of hits divided by the total number of recommendations made by the system. In this study, we take a request sequence $S = \{a_1, a_2, \ldots, a_n\}$ from the test data, where n is the number of requests, and divide S into two parts: $A = \{a_1, a_2, \ldots, a_{n-1}\}$ and $B = \{a_n\}$. We use A as an input to the trained model and obtain a system recommendation C. Since B is the actual next request, C can be considered a good recommendation if $B \subseteq C$.[4] If $B \subseteq C$, we call C a correct recommendation. Given a set of test sequences, the hit ratio is thus defined as

$$\textit{Hit Ratio} = \frac{\#\textit{ of Correct Recommendations}}{\#\textit{ of Recommendations}} \quad (20)$$

The rationale behind this is that when a user is looking for a document, he/she may make a sequence of requests within a period until the target is found. In Web usage mining, this sequence is referred to as a *user session*, which accomplishes a single task. We can thus assume that after accessing the last object in a session, a user has found the document that is interesting to him/her. Therefore, to evaluate whether the user is satisfied with the recommendation, we simply leave out the last request (i.e., a_n) in a test sequence (which is a session), make a prediction by fitting the previous requests in the sequence into the trained Markov model, and then check whether the recommendation includes a_n.

User Clustering Evaluation

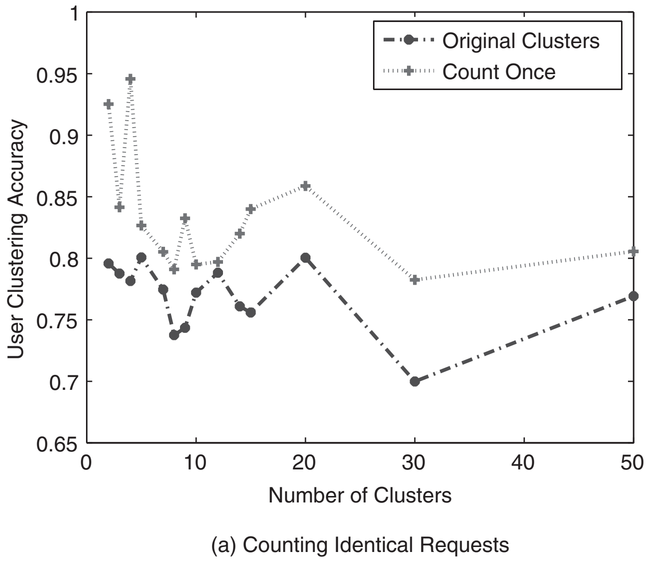
To improve the clustering performance, we design different strategies to preprocess the *Livelink* data sets and show the experimental results in this section.

Remove heavily visited tree in user clustering. As discussed in the section State Representation in Markov Models, more than 70% of the requests visit a common tree in the resulting session files. Therefore, a great portion of requests are identical, making it difficult to distinguish users. To solve this problem, we move down to the second level of this heavily visited tree and use the children of the root to represent the objects in this tree. Figure 5 shows the user distribution among clusters with and without the adjustment after applying the Markov model based user clustering method with 15 components[5] in the mixture model. As shown in Figure 5(a), without the adjustment, the users are not distinguishable since almost all the users are grouped into a single cluster and there are few members in the 14 other clusters. After the adjustment, the users are better distributed as shown in Figure 5(b).

It is also easy to understand that making recommendations with the data set in Figure 5(a) may not satisfy user's various requirements, as the recommendation to be made depends on **each** component's previously trained system parameters. Hence, if all users' accessed sequences are identical, they will be grouped into the same cluster (with the component

[4] Please note that in our experiments we only recommend a single object (with the highest probability) when using the mixture of Markov models based recommendation method. However, for association rule based recommendation method, a set of objects can be recommended. Therefore, the measure of *hit ratio* defined here gives more favor to the association rule based method.

[5] The number of components here is decided by observing the out-of-sample like lihood (Smyth, 1996b) and a 10-fold cross-validation. We understand that it is still an open problem to determine this number in the clustering problem, and there might be other methods such as Bayesian Inference Criterion (BIC; Box & Jenkins, 1994) to use. However, this is not the main concern of the paper.



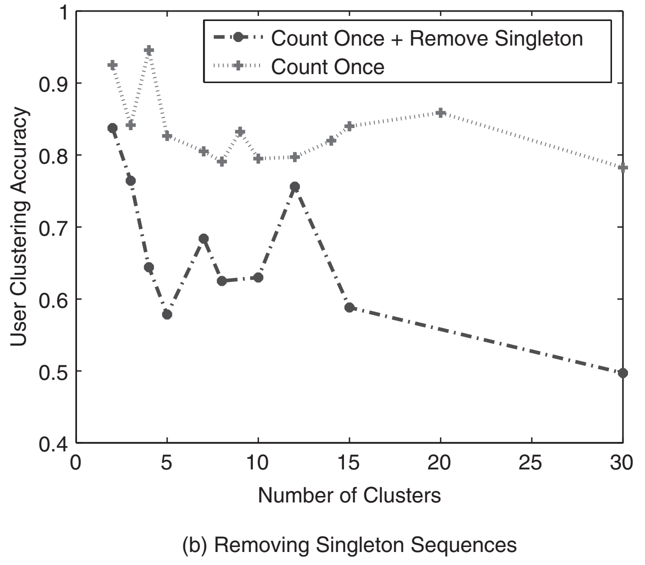


FIG. 6. Counting sequential requests.

parameters), and it is not surprising to see that those users will get the same recommendation in the future. In contrast, after replacing the heavily visited tree with its offspring, as shown in Figure 5(b), the distribution is much balanced; therefore, the underlying predictions might be diversified accordingly.[6]

In addition, in this experiment we apply the same cluster numbers to the two different scenarios for the following concerns. First, the K shown in Figure 5(b) is obtained with the cross-validation method and is a best fit for the data after replacement. Second, as shown in Figure 5(a), most users are grouped into a single cluster, whereas other clusters are almost empty. We thus notice that the selection of cluster numbers from this data set actually does not hurt demonstrating the real cluster distribution, while K is set to be a slightly greater value than it actually is.

Counting sequential requests with different strategies. In our session file, especially after replacing objects with their ancestors, many sessions contain multiple consecutive occurrences of the same objects. That is, the user may request the same object or the objects in the same (sub)tree multiple times within the same session. Learning from this type of data, a Markov model may contain many heavily weighted self-transitions, which may negatively affect the clustering result. This type of data offers us a choice of how to count the objects within a session. We can either combine the multiple consecutive occurrences of the same object into a single object occurrence (denoted as *Count Once*) or leave them as they are (denoted as *Original Clusters*).

Figure 6(a) shows that counting once is apparently superior to the one without removing redundant requests. In addition, we notice that about 20% of the sessions in the *Livelink* data set contain a single request. When combining the consecutive same requests into one, the percentage of singleton sessions is even higher. Since such sessions do not contain state transitions, we consider ignoring such singleton sessions from the training data and comparing the performance with that of the method that keeps the singletons in the training data. As illustrated in Figure 6(b), the performance is not as good as expected, because the size of training data decreases greatly after eliminating the singletons. In contrast to 8225 sequences in our previous training data, there are only 1999 sequences left. On the basis of the experimental results, we find out that *Count Once* achieves the best performance among all the heuristic strategies; thus it is used as a default in other experiments in our work.

Recommendation Evaluation

Comparison of data selection strategies. In this section, we evaluate the recommendation performance of the two data selection strategies described previously, namely, *pyramidal time frame and uniform selection*. We select the same number of training sequences from the data within a 26-day period using the two strategies and compare their average recommendation performance using the same test data set as shown in Table1, where K is set to 15 selected through cross-validation. Table 1 shows that *uniform selection* offers better performance than *pyramidal time frames*, that implies that *Livelink* users' access patterns are quite stable over that time. The result in Table 1 is the average from 10-fold

[6] Note that the user clustering result may be further improved if we consider more trees at the second level. However, this will increase the number of states in the Markov model. Determining how to select an optimal cut-off level for a tree is a challenging issue that needs further investigation, which we do not consider in this paper.

TABLE 1. Performance comparison of two data selection strategies.

Data selection strategy	Hit ratio
Pyramidal time frame	88.41%
Uniform selection	89.23%

cross-validation. We perform a *t-test* to assess whether the means of two groups are statistically different from each other. The p value from the t-test is 0.0138, that means that the recommendation performance with the *uniform selection* data set is significantly superior to that with the *pyramidal time frame* in our study.

Comparison with the single Markov model. A single Markov model can be built from the training set without first clustering the data. It is equivalent to a mixture of Markov models with $K = 1$. Table 2 compares the performance of a mixture of Markov models with $K = 15$ to that of the single Markov model on the *Livelink* data with the uniform data selection strategy. The result shows that clustering sequences before building a Markov model leads to much better recommendation performance than without doing clustering. This indicates that personalized recommendation is much better than recommendation based on the general properties of the whole data set.

Comparison with hard and soft clustering strategy. In the section User and Sequence Clustering, we presented how to use "hard" and "soft" clustering strategies to label access sequences, respectively. In this section, we compare the recommendation performance of using those two methods and report the result in Table 3. With the hard clustering method, we first assign the request sequence to a cluster with the maximum likelihood $argmax_k P^i_{j,k}(\theta)$ according to Definition 3.2, then predict the next request according to the most likely cluster's transition matrix. With soft clustering, the probability that a sequence belongs to each component is used in computing the probability of a possible next state. From Table 3, we observe that recommendation with soft clustering of sequences is better than use of the hard clustering method.

TABLE 2. Performance comparison of the mixture model with the single model.

Markov models	Hit ratio
Mixture of Markov models	89.23%
Single Markov model	31.23%

TABLE 3. Performance comparison of the soft clustering method with the hard clustering method.

Markov models	Hit ratio
Soft clustering	89.23%
Hard clustering	82.37%

Comparison with global association rule based recommendations. In this section, we evaluate the recommendation performance using the global association rules mined from the whole training data without clustering data at first. Assume that all available *Livelink* access sequences are independent. We randomly select 70% of them as training data and take the rest as testing data. With the training data set, we first learn association rules given different minimal support and minimal confidence. Then given a test request sequence $S = \{a_1, a_2, \ldots, a_n\}$, its prefix subsequence $A = \{a_1, a_2, \ldots, a_{n-1}\}$ is used to find rules whose antecedent has the largest overlap with A. All the objects in the consequences of such rules are output as recommendations. If one of those recommendations is identical to the last request a_n, we say that this sequence gets a *hit*. After accumulating results over all the testing data, we calculate hit ratios shown in Figure 7. In addition, as both minimal support and minimal confidence determine the number of generated rules, we are interested to know how the recommendation performance may change with the two thresholds. We show results with a fixed minimal support in Figure 7(a), and those with a fixed minimal confidence in Figure 7(b).

As shown in Figure 7, the best hit ratio, 46.70%, is achieved at minimal support = 2%, and minimal confidence = 50%. In addition, we notice that there is no principled rule we can follow to obtain the best recommendation, as the performance fluctuates dramatically as the values of two threshold parameters change. Moreover, comparing with Table 1 and Table 2, the average performance of using mixture of Markov models to make recommendation is much superior to that of this method. This is because this method focuses on finding the co-occurrence patterns among objects but ignores their sequential relationships.

Comparison with cluster-oriented association rule recommendation. With the same data set used in the preceding section, we evaluate the recommendation performance of the cluster-oriented association rule recommendation method, which uses different sets of rules for different clusters. In contrast to catching co-occurrence relations from the whole data set, this method is expected to identify personalized navigation patterns in each group.

As shown in Table 4, this method yields better performance than the global association rule method, yet it is not comparable with predicting directly with mixture of Markov models. The results indicate that making personalized recommendations with cluster-oriented association rules is better than making recommendations with the general patterns learned from all the users. However, since association rules do not consider the sequential relationships between visited objects, their recommendations are worse than use of the mixture of Markov models, which considers both sequential relationships and personalizations.

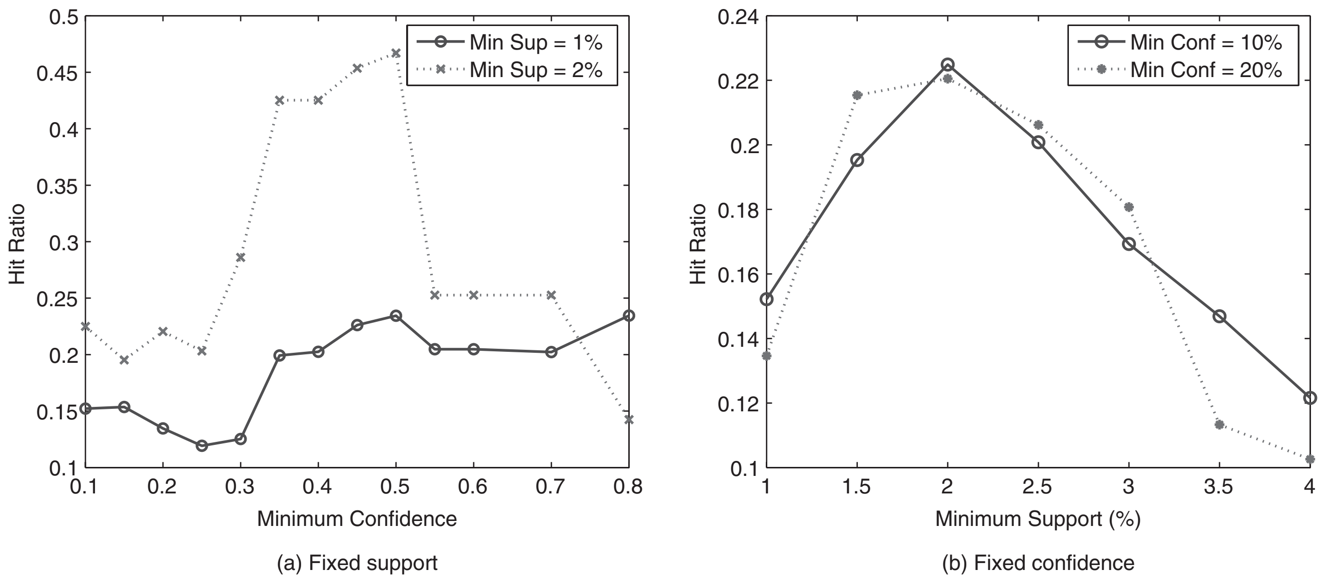


FIG. 7. Recommendation with classic association rule deductions.

TABLE 4. Performance comparison of cluster-oriented association rule recommendations.

(a) Fixed minimal support		(b) Fixed minimal confidence	
Min Sup = 2%	Hit ratio	Min Conf = 20%	Hit ratio
con f = 30%	29.62%	*supp* = 2%	29.54%
con f = 40%	28.52%	*supp* = 2.5%	28.96%
con f = 50%	31.92%	*supp* = 3%	30.36%
con f = 60%	33.28%	*supp* = 3.5%	32.91%
con f = 70%	42.55%	*supp* = 4%	30.58%
con f = 80%	48.90%	*supp* = 4.5%	30.11%
con f = 80%	50.59%	*supp* = 8%	28.82%

Adaptive Algorithm Evaluation

As we have discussed, it is infeasible always to perform an overhaul reconstruction of the mixture of Markov models as new data become available. We proposed an adaptive algorithm to update the model parameters incrementally. In this section, we evaluate the adaptive algorithm's performance by comparing it with that achieved through an overhaul model reconstruction. The previous experiments use 26 days of *Livelink* access data to test the performance of recommendations. In this experiment, we compare the performance of the adaptive algorithm with that of reconstructing the model and that of not reconstructing the model at all using the data from

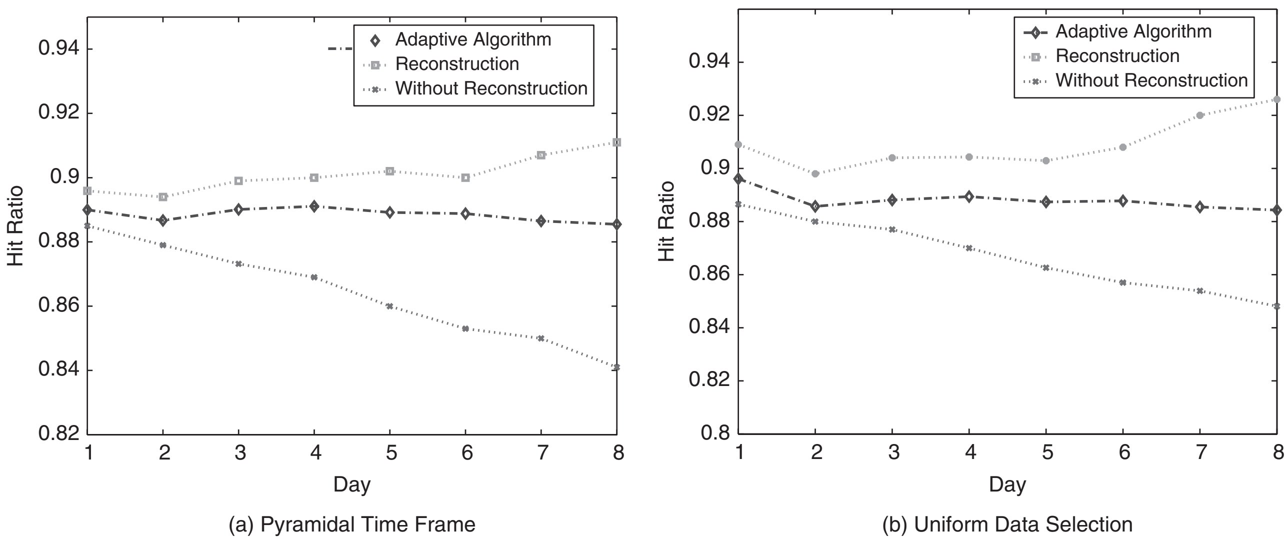


FIG. 8. Adaptive algorithm.

TABLE 5. Adaptive algorithm performance evaluation in terms of time.

Method	Time
Overhaul reconstruction	21.2 hours
Adaptive algorithm	0.3 second

day 27 to day 34, with new data supplied to the training of the model each day. The results are shown in Figure 8.

Compared to the overhaul reconstruction method, the performance of the adaptive algorithm degrades very slowly. The recommendation result is still satisfactory, although it consumes much less time and computational resources. Given $N = 4000$ sequences, where $K = 15$ and $M = 120$, as the training data, when a new request sequence becomes available, the time spent on both methods is shown in Table 5. In addition, we find that the performances of both the adaptive algorithm and the overhaul construction are superior to the performance of the method that does not make any reconstruction at all.

Application of Mixture of Markov Models to Personalized Livelink Search—an Example

In this section, we use an example to demonstrate how our proposed model can be applied to assist personalized retrieval in the *Livelink* application.

As shown in Figure 9, if a user would like to get some marketing information from the local network supported by *Livelink*, he/she may issue a short query with the term *marketing*. However, if such a query is sent to a retrieval system, numerous unwanted returns are possible, as the description is too general and can fit various scenarios. To solve this problem, our system can be utilized to obtain more information about the query's immediate context, then

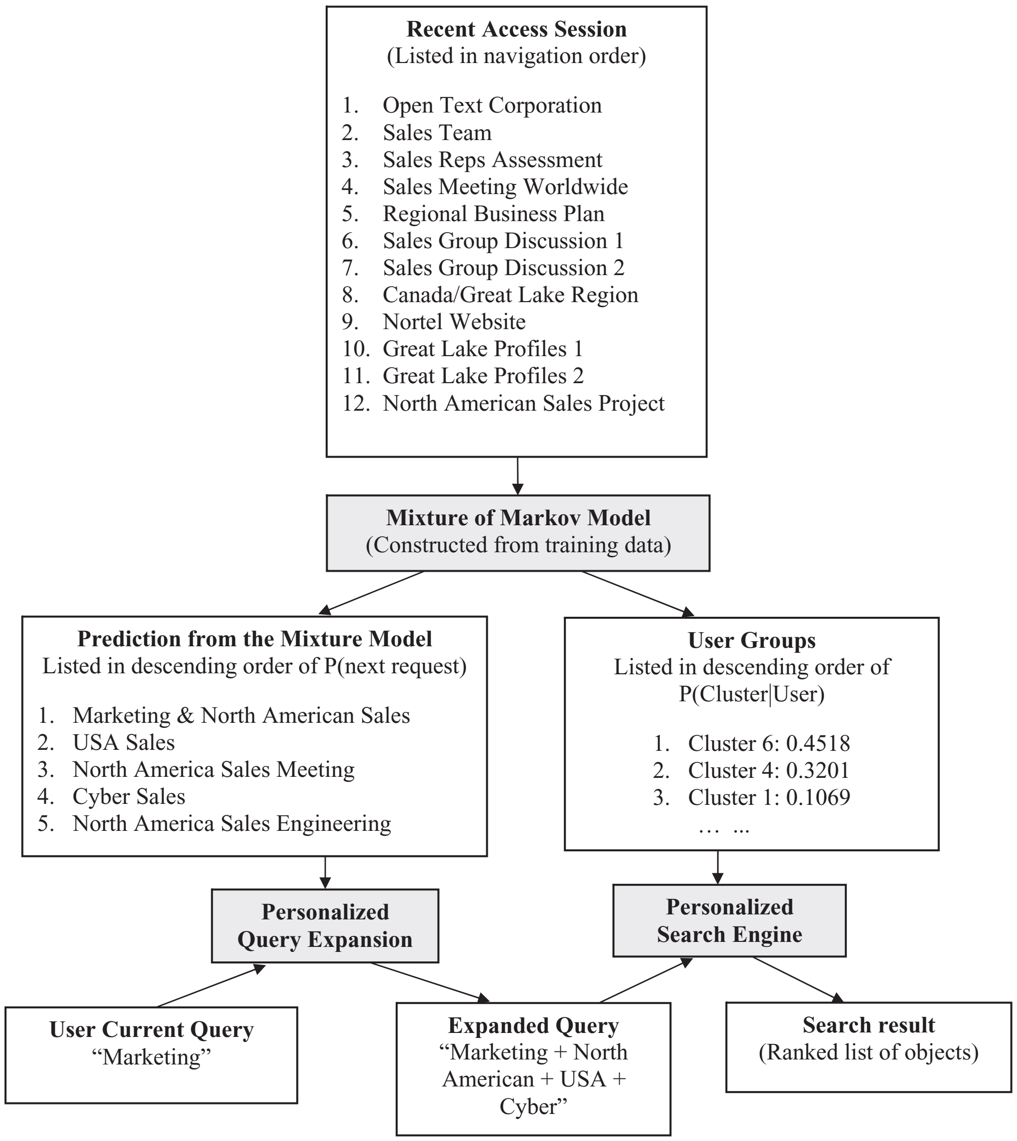

Recent Access Session
(Listed in navigation order)
1. Open Text Corporation
2. Sales Team
3. Sales Reps Assessment
4. Sales Meeting Worldwide
5. Regional Business Plan
6. Sales Group Discussion 1
7. Sales Group Discussion 2
8. Canada/Great Lake Region
9. Nortel Website
10. Great Lake Profiles 1
11. Great Lake Profiles 2
12. North American Sales Project
Mixture of Markov Model
(Constructed from training data)
Prediction from the Mixture Model
Listed in descending order of P(next request)
1. Marketing & North American Sales
2. USA Sales
3. North America Sales Meeting
4. Cyber Sales
5. North America Sales Engineering
User Groups
Listed in descending order of P(Cluster|User)
1. Cluster 6: 0.4518
2. Cluster 4: 0.3201
3. Cluster 1: 0.1069
… ...
Personalized Query Expansion
Personalized Search Engine
User Current Query
"Marketing"
Expanded Query
"Marketing + North American + USA + Cyber"
Search result
(Ranked list of objects)

FIG. 9. An example of integration with the Livelink retrieval system.

infer the user's actual needs. By feeding the user's most recent navigation session into the mixture of Markov models, we can predict which objects are most likely to be accessed next by the user. The contents of those predicted objects can then be analyzed to extract new terms for expansion of the current query. In this example, some candidate terms could be *North American*, *USA*, *Cyber*, etc. In addition, with our user clustering method, a user can be grouped into one or more clusters on the basis of the user's previous navigation history. As shown in Figure 9, it is highly possible that the current user belongs to Cluster 6 and Cluster 4, given the probability threshold of 20%. This user clustering information can be integrated with other techniques, e.g., collaborative learning methods, such that the shared interest of users from the same group can be considered in the retrieval process. For example, the retrieved documents can be reranked by making use of the knowledge about the user's group.

Related Work

Markov-based models have also been widely used in the problem of user modeling. Padmanabhan and Mogul (1996) builts N-gram Markov models and applied the prefetching method to cache the Web pages that are likely to be requested soon. Pirolli and Pitkow (1999) studied the kth order Markov models with k ranging from 1 to 9, and checked which link the surfer followed in the next step given the current page. These approaches all imply a Markov presentation for modeling the surfer's access patterns but focus on a single model to present characteristics. In addition, higher-order Markov models use the strict order of the action sequences, which places a strict restriction on the sequentiality of the data. Sarukkai (2000) constructed first-order Markov models to study the categories of pages requested by a Web surfer. A "personalized" Markov model was trained for each individual and used for predictions in users' future request sessions. In practice, however, it is expensive to construct a unique model for each user, and the problem may worsen when there exist thousands of different users within a big Web site. In this paper, we show that the mixture model is more powerful than a single first-order Markov model, and an acceptable complexity can still be held compared with high-order Markov models.

As shown in Mobasher, Dai, Luo, and Nakagawa (2000), traditional Web personalization strategies could be problematic when a big site evolves or surfers' browsing behavior changes over time. Recently, a wide variety of adaptive algorithms have been proposed in the literature for various learning models to keep track of new available data without incurring overall retraining. Huo, Chan, and Lee (1995) proposed a Bayesian adaptive algorithm for learning the parameters in the *hidden Markov model* for speech recognition Zivkovic (2004) developed an improved method to reestimate recursively the parameters in Gaussian mixture models for background subtraction, which is a common computer vision task. In contrast, our study focuses on finding adaptive algorithms for the mixture of Markov model; however, to the best of our knowledge, there is no exsiting work in this area.

Because of the immensity of available information on the Internet, it becomes crucial to select a part of the useful information from the overall data. In the field of data stream mining, Aggarwal, Han, Wang, and Yu (2003) proposed a data selection method based on the *pyramidal time frame* to test the hypothesis that the more recent data would reflect user access patterns more accurately. In this method, cluster information is maintained as snapshots in time that follow a pyramidal pattern. It provides a trade-off between the storage requirements and the ability to recall summary data from different times. However, this method deals with only data streams, as it requires a large quantity of data at every moment of time. Although the whole investigated *Livelink* data set is huge, it does not have continuous input all the time. In our work, we propose a similar strategy to the *pyramidal time frame*. We first divide the whole duration time into unequal slots, and then randomly pick an equal amount of request sequences in each interval. In this way, it is expected that more recent data could have a higher probability of being selected into the training data.

Conclusion and Future Work

We have presented a *mixture of Markov models* based clustering method for use in a real application. The mixture of Markov models allows us to cluster the user's historical access sequences and Web users themselves into groups and provide personalized recommendations based on the sequential relationships in each component of the models to the Web surfers. This kind of clustering provides advantages over traditional non-model-based clustering approaches. First, it provides a general description of each identified cluster; hence user access patterns are naturally formulated when clustering the user's request sequences. Second, statistical models allow clustering uncertainties in component members, an important feature, especially for those objects that are close to cluster boundaries.

We have made another contribution in providing an adaptive algorithm to keep the system current efficiently. Once the mixture of models has been built with the training data, newly available data can be viewed as a testing data set and applied to the system for clustering within an extended period. Nonetheless, the clustering performance may gradually decrease, if we do not adjust the model as more and more data become available. This is because the system will become out of date and unable to understand the user's current access patterns. Ideally, we could use all data available to date to retrain the model from scratch. However, this is virtually infeasible in view of efficiency concerns. To reduce the computational cost of keeping the model current, we developed a light-weighted adaptive algorithm, which enables the models to make incremental changes as new request sequences are available. Our experiments showed that with the proposed algorithm decent recommendation performance can be achieved without frequent updating of the model.

We discussed two strategies, *pyramidal time frame* and *uniform selection*, for choosing the training data for the mixture of Markov models. Using the *Livelink* data set as the testbed, the *uniform selection* approach exhibited better performance than *pyramidal time frames*, an indication that Web surfers follow a stable access pattern over time in our application domain. We investigated the possibility of combining the model with classic information retrieval systems and demonstrated some integration methods with an example.

We are currently investigating more efficient ways to construct and maintain the mixture of Markov models and further increase the clustering and recommendation accuracies. Moreover, we are interested in the following issues:

1. Data collection: Our proposed method can work well with the *Livelink* data set and the synthetic data set. In the future, we will try to obtain Web data from different sources and test the performance of our algorithms.
2. Non-existent cluster handling: Currently, our system is incapable of making recommendations to request sequences that belong to previously nonexistent clusters. In the future, we will address this problem.
3. Adaptive algorithm: Another interesting problem for future research is related to the adaptive clustering algorithm. According to Figure 8, the performance of the adaptive algorithm gradually degrades as the existing mixture of Markov models becomes out of date. However, it is difficult to determine when the existing models should not be used anymore. In our prototype system, we set a time threshold. The system performs periodic reconstruction of models after the threshold is passed. In the future, it will be interesting to investigate whether there are better approaches to solving this problem.
4. Sequential request counting: In *Livelink* data set, there exist sequences that contain multiple consecutive occurrences of the same objects. If the mixture of Markov model is trained with such data, the transition matrix might be highly skewed and full of heavily weighted self-transitions, i.e., a large number of spots in the matrix that have a very small value. Thus, to determine the probability of observing a sequence v using $p(v|c_k,\theta) = p(v_1|\theta_k^1)\prod_{i=2}^{K} p(v_i|v_{i-1},\theta_k^T)$, we could find $p(v|c_k,\theta) = 0$ while a sequence contains many objects that are not frequently accessed, because $\prod_{i=2}^{K} p(v_i|v_{i-1},\theta_k^T)$ would become close to 0. To alleviate the problem, we added a small value ε on both numerator and denominator of

$$\theta_{k,j\to 1}^T = \frac{\varepsilon + \sum_{i=1}^{N} p(c_k|d_{train}^i,\theta)\delta_{j,l}(d_{train}^i)}{\varepsilon + \sum_{d=1}^{M}\sum_{i=1}^{N} p(c_k|d_{train}\theta)\delta_{j,d}(d_{train}^i)}$$

in Equation 9 to smooth the transition matrix in real implementations. However, the improvement could be limited if the matrix were very sparse. To solve the problem further, we propose several ways to count the requests differently in the section counting sequential requests with different strategies. Through empirical studies, we evaluate their performance in terms of user clustering accuracies. In our study, the result from the Count Once strategy clearly outperforms that from the original counting method. Nonetheless, it still might be suboptimal. Is it the case if an object is accessed consecutively for multiple times by a user, it will be more important to that user? To investigate the possibility, we will develop more strategies in our future work.
5. User evaluation: Judging the usability of a system is an important task in a real application. In the domain of Web usage mining, it is always good to know whether the user is satisfied with the prediction that is provided for him/her. Nonetheless, such an evaluation can only be done by the real users of *Livelink* in our work. In the future, we would like to conduct further user evaluations with our collaborators in Open Text Corporation.

Acknowledgments

This research is supported by a research grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada and the Open Text Corporation. We would like to thank the anonymous reviewers for their excellent and constructive comments. We would also like to thank Gary Promhouse and Mike Dent of Open Text for their valuable comments on this work.

References

Aggarwal, C.C., Han, J., Wang, J., & Yu, P.S. (2003). A framework for clustering evolving data streams. In *VLDB*, pp. 81–92. Berlin, Germany: Morgan Kaufmann.

Agrawal, R., Imielinski, T., & Swami, A.N. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (207–216). Washington, D.C.: ACM Press.

Anderson, C., Domingos, P., & Weld, D. (2002). Relational Markov models and their application to adaptive. Web navigation. In Proceedings of eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 143–152). Madison, Wisconsin: ACM Press.

Bilmes, J. (1997). A gentle tutorial on the em-algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report, University of Berkeley, ICSI-TR-97-021.

Box, G.E.P., & Jenkins, G.M. (1994). Time series analysis: Forecasting and control. Upper Saddle River, NJ: Prentice Hall PTR.

Cadez, L.V., Heckerman, D., Meek, C., Smyth, P., & White, S. (2000). Visualization of navigation patterns on a Web site using model-based clustering. In Knowledge Discovery and Data Mining (pp. 280–284). Boston: ACM Press.

Fu, Y., Sandhu, K., & Shih, M.-Y. (1999). Generalization-based approach to clustering of Web usage sessions. In WEBKDD (pp. 21–38). San Diego: Springer.

Gunduz, S., & Ozsu, M.T. (2003). A Web page prediction model based on click-stream tree representation of user behavior. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 535–540). Philadelphia: ACM Press.

Huang, X, Peng, F., An, A., & Schuurmans, D. (2004). Dynamic Web log session identifcation with statistical language models. Journal of American Society for Information Science and Technology, Special Issue on Webometrics, 55(14), 1290–1303. John Wiley & Sons, Inc.

Huo, Q., Chan, C., & Lee, C.-H. (1995). Bayesian adaptive learning of the parameters of hidden Markov model for speech recognition. IEEE

Transactions on Speech and Audio Processing, 3(5), 334–345. IEEE Signal Processing Society.

Manavoglu, E., Palov, D., & Giles, C.L. (2003). Probabilistic user behavior models. In ICDM'03: Proceedings of the Third IEEE International Conference on Data Mining (p. 203). Melbourne, Fl: IEEE Computer Society.

Milligan, G.W., & Cooper, M.C. (1986). A study of the comparability of external criteria for hierarchical cluster analysis. Multivariate Behavioral Research, 21, 441–458. Society of Multivariate Experimental Psychology.

Mobasher, B., Dai, H., Luo, T., & Nakagawa, M. (2001). Effective personalization based on association rule discovery from Web usage data. In Proceedings of the Third International Workshop on Web Information and Data Management (pp. 9–15). Arlington, VA: ACM Press.

Mobasher, B., Dai, H., Luo, T., Sun, Y., & Zhu, J. (2000). Integrating Web usage and content mining for more effective personalization. In EC-WEB '00: Proceedings of the First International Conference on Electronic Commerce and Web Technologies (pp. 165–176). London: Springer.

Padmanabhan, V.N., & Mogul, J.C. (1996). Using predictive prefetching to improve World Wide Web latency. In Proceedings of the ACM SIGCOMM Comput. Comm. Rev., 26(3): 22–36. New York: ACM Press .

Pirolli, P., & Pitkow, J.E. (1999). Distributions of surfers' paths through the World Wide Web: Empirical characterizations. World Wide Web, 2(1–2), 29–45. Hingham, MA, USA: Kluwer Academic Publishers.

Sarukkai, R.R. (2000). Link prediction and path analysis using Markov chains. In Proceedings of the Ninth International World Wide Web Conference on Computeter Networks: The International Journal of computer and Telecommunications Networking (pp. 377–386). Amsterdam, The Netherlands: North-Holland Publishing Co.

Schechter, S., Krishnan, M., & Smith, M.D. (1998). Using path profiles to predict http requests. In Proceedings of the Seventh International Conference on World Wide Web 7 (pp. 457–467). Brisbane, Australia: Elsevier Science Publishers B.V.

Shahabi, C., Zarkesh, A.M., Adibi, J., & Shah, V. (1997). Knowledge discovery from user's Web-page navigation. In RIDE. Washington, D.C.: IEEE Computer Society.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth." In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems (Vol. 1, pp. 210–217). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Smyth, P. (1996a). Clustering sequences with hidden Markov models. In NIPS (pp. 648–654). Denver, CO: MIT Press.

Smyth, P. (1996b). Clustering using Monte Carlo cross-validation. In Knowledge Discovery and Data Mining (pp. 126–133). Portland, OR: AAAI Press.

Yan, T.W, Jacobsen. M., Garcia-Molina, H., & Dayal, U. (1996). From user access patterns to dynamic hypertext linking. In Proceedings of the Fifth International World Wide Web Conference on Computer Networks and ISBN Systems (pp. 1007–1014). Amsterdam, The Netherlands: Elsevier Science Publishers.

Zhong, S., & Ghosh, J. (2003). A unified framework for model-based clustering. Journal of Machine Learning Research, 4, 1001–1037. Cambridge, MA: MIT Press.

Zukerman, I., & Albrecht, D.W. (2001). Predictive statistical models for user modeling. User Modeling and User-Adapted Interaction, 11(1–2), 5–18. Hingham, MA: Kluwer Academic Publishers.

Zivkovic, Z. (2004). Improved adaptive Gaussian mixture model for background subtraction. In Proceedings of the International Conference on Pattern Recognition. Washington, D.C.: IEEE Computer Society.