Automatic Personalization Based on Web Usage Mining

Bamshad Mobasher
Dept. of Computer Science, DePaul University, Chicago, IL
mobasher@cs.depaul.edu

Robert Cooley, Jaideep Srivastava
Dept. of Computer Science, University of Minnesota, Minneapolis, MN
cooley@cs.umn.edu, srivasta@cs.umn.edu

Introduction

The ease and speed with which business transactions can be carried out over the Web have been a key driving force in the rapid growth of electronic commerce. Business-to-business e-commerce is the focus of much attention today, mainly due to its huge volume. While there are certainly gains to be made in this arena, most of it is the implementation of much more efficient supply management, payments, etc. On the other hand, e-commerce activity that involves the end user is undergoing a significant revolution. The ability to track users' browsing behavior down to individual mouse clicks has brought the vendor and end customer closer than ever before. It is now possible for a vendor to personalize his product message for individual customers at a massive scale, a phenomenon that is being referred to as *mass customization.*

Though the scenario outlined here is from e-commerce, the type of personalization described is applicable to any Web browsing activity. *Web personalization* can be described, as any action that makes the Web experience of a user personalized to the user's taste. The experience can be something as casual as browsing the Web or as (economically) significant as trading stocks or purchasing a car. The actions can range from simply making the presentation more pleasing to an individual to anticipating the needs of the user and providing the right information as well as performing a set of routine book-keeping functions automatically.

Principal elements of Web personalization include modeling of Web objects (pages, etc.) and subjects (users), categorization of objects and subjects, matching between and across objects and/or subjects, and determination of the set of actions to be recommended for personalization. Existing approaches used by many Web-based companies, as well as approaches based on collaborative filtering, e.g. GroupLens [KMM+97, HKBR99] and Firefly [SM95], rely heavily on getting human input, e.g. user profile, for determining the personalization actions. The drawbacks of this are, (a) the input is often a subjective description of the users by the users themselves, and thus prone to biases, and (b) the profile is static, and thus good for personalization for some time after it is collected; but its performance degrades over time as the profile ages.

Recently, a number of approaches have been developed dealing with specific aspects of Web usage mining for the purpose of automatically discovering user profiles. For example, Perkowitz and Etzioni [PE98] proposed the idea of *optimizing* the structure of Web sites based co-occurrence patterns of pages within usage data for the site. Schechter et al [SKS98] have developed techniques for using path profiles of users to predict future HTTP requests, which can be used for network and proxy caching. Spiliopoulou et al [SF99], Cooley et al [CMS99], and Buchner and Mulvenna [BM99] have applied data mining techniques to extract usage patterns from Web logs, for the purpose of deriving marketing intelligence.

Shahabi et al [SZA97], Yan et al [YJGD96], and Nasraoui et al [NFJK99] have proposed clustering of user sessions to predict future user behavior.

In this paper we describe an approach to usage-based Web personalization taking into account the full spectrum of Web mining techniques and activities. Our approach is described by the architecture shown in Figure 1, which heavily uses data mining techniques, thus making the personalization process both automatic and dynamic, and hence up-to-date. Specifically, we have developed techniques for preprocessing of Web usage logs and grouping URL references into sets called *user transactions* [CMS99]. A user transaction is a unit of semantic activity, and performing data mining on them is more meaningful. We describe and compare three different Web usage mining techniques, based on *transaction clustering, usage clustering*, and *association rule discovery*, to extract usage knowledge for the purpose of Web personalization. We also propose techniques for combining this knowledge with the current status of an ongoing Web activity to perform real-time personalization. Finally, we provide an experimental evaluation of the proposed techniques using real Web usage data.

Architecture for Usage-based Web Personalization

The overall process of usage-based Web personalization can be divided into two components. The offline component is comprised of the data preparation tasks resulting in a user transaction file, and the specific usage mining tasks, which in our case involve the discovery of association rules and the derivation of URL clusters based on two types of clustering techniques.

Once the mining tasks are accomplished, the frequent itemsets and the URL clusters are used by the online component of the architecture to provide dynamic recommendations to users based on their current navigational activity. The online component is comprised of a recommendation engine and the HTTP server. The Web server keeps track of the active user session as the user browser makes HTTP requests. This can be accomplished by a variety of methods such as URL rewriting, or by temporarily caching the Web server access logs. The recommendation engine considers the active user session in conjunction with the URL clusters and the discovered association rules to compute a set of recommended URLs. The recommendation set is then added to the last requested page as a set of links before the page is sent to the client browser.

A generalized architecture for the system is depicted in Figure 1. We now discuss the details of each of the architectural components.

Mining Usage Data for Web Personalization

The offline component of usage-based Web personalization can be divided into two separate stages. The first stage is that of preprocessing and data preparation, including, data cleaning, filtering, and transaction identification. The second is the mining stage in which usage patterns are discovered via methods such as association-rule mining and clustering. Each of these components is discussed below.

Preprocessing Tasks

The prerequisite step to all of the techniques for providing users with recommendations is the identification of a set of user sessions from the raw usage data provided by the Web server. Ideally, each user session gives an exact accounting of who accessed the Web site, what pages were requested and in what order, and how long each page was viewed. Two of the biggest impediments to forming accurate user sessions are local caching and proxy servers. In order to improve performance and minimize network

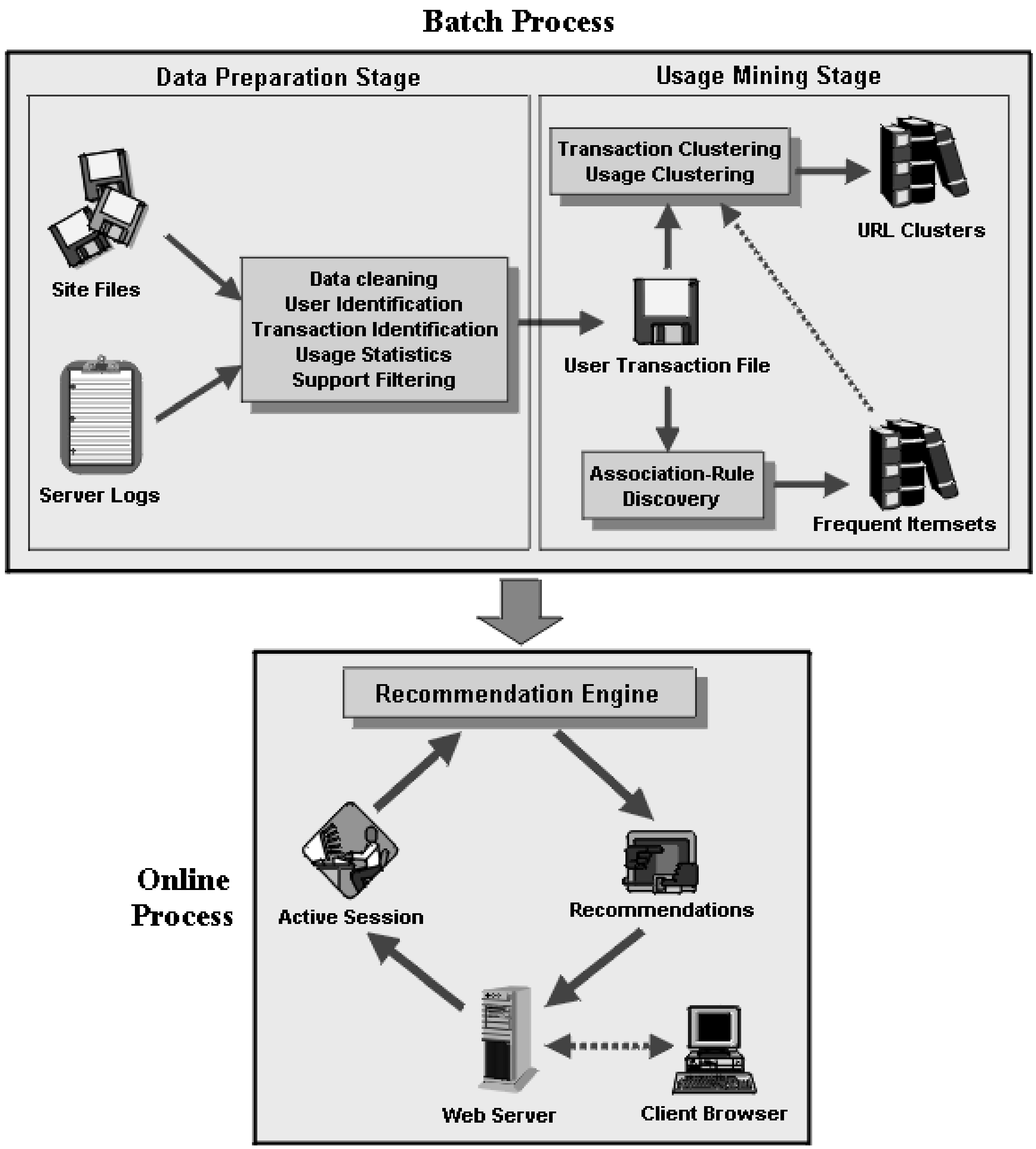


Figure 1. General Architecture for Usage-Based Web Personalization

traffic, most Web browsers cache the pages that have been requested. As a result, when a user hits the "back" button, the cached page is displayed and the Web server is not aware of the repeat page access. Proxy servers provide an intermediate level of caching and create even more problems with identifying site usage. In a Web server log, all requests from a proxy server have the same identifier, even though the requests potentially represent more than one user. Also, due to proxy server level caching, multiple users throughout an extended period of time could actually view a single request from the server. The most reliable methods for resolving a server log into user session are the use of cookies or dynamic URLs with an embedded session ID. However, these techniques are not always available due to privacy concerns of the users, or limitations of the capabilities of the Web server. As described in detail in [CMS99], several simple heuristics using the referrer and agent fields of a Server log can be used to identify user sessions and infer missing references with relative accuracy in the absence of additional information such as cookies.

In addition to identifying user sessions, the raw log must also be cleaned, or transformed into a list of page views. Due to the stateless connection properties of the HTTP protocol, several file requests (HTML, images, sounds, etc.) are often made as the result of a single user action. The group of files that are sent due to a single click are referred to as a *page view*. Cleaning the server log involves removing all of the file accesses that are redundant, leaving only one entry per page view. This includes handling page views that have multiple frames, and dynamic pages that have the same template name for multiple page views. It may also be necessary to filter the log files by mapping the references to the site topology induced by physical links between pages. This is particularly important for usage-based personalization, since the recommendation engine should not provide dynamic links to "out-of-date" or non-existent pages.

Each user session in a user session file can be thought of in two ways; either as a single transaction of many page references, or a set of many transactions each consisting of a single page reference. The goal of transaction identification is to dynamically create meaningful clusters of references for each user. Based on an underlying model of the user's browsing behavior, each page reference can be categorized as a *content* reference, *auxiliary* (or *navigational*) reference, or *hybrid*. In this way different types of transactions can be obtained from the user session file, including content-only transactions involving references to content pages, and navigation-content transactions involving a mix of pages types. The details of methods for transaction identification are discussed in [CMS99]. For the purpose of this paper we assume that each user session is viewed as a single transaction. Finally, the session file may be filtered to remove very small transactions and very low support references (i.e., URL references that are not supported by a specified number of user transactions). This type of support filtering can be important in removing noise from the data, and can provide a form of dimensionality reduction in clustering tasks where URLs appearing in the session file are used as features.

Given the preprocessing steps outline above, for the rest of this paper we assume that there is a set of n unique URLs appearing in the preprocessed log:

$$U = \{url_1, url_2, \ldots, url_n\}$$

and a set of m user transactions:

$$T = \{t_1, t_2, \ldots, t_m\}$$

where each $t_i \in T$ is a non-empty subset of U.

Discovering Frequent Itemsets and Association Rules

The association rule discovery methods such as the Apriori algorithm [AS94], initially find groups of items (which in this case are the URLs appearing in the preprocessed log) occurring frequently together in many transactions. Such groups of items are referred to as *frequent item sets*. Given a set $I = \{I_1, I_2, \ldots, I_k\}$ of frequent itemsets, the *support* of I_i is defined as

$$\sigma(I_i) = \frac{|\{t \in T : I_i \subseteq t\}|}{|T|}$$

Generally, a support threshold is specified before mining and is used by the algorithm for pruning the search space. The itemsets returned by the algorithm satisfy this minimum support threshold. Furthermore, support is downward closed: if an item set does not satisfy the minimum support criteria, then neither do any of its supersets.

Association rules capture the relationships among items based on their patterns of co-occurrence across transactions. In the case of Web transactions, association rules capture relationships among URL references based on the navigational patterns of users. An association rule r is an expression of the form

$$X \Rightarrow Y \quad (\sigma_r, \alpha_r)$$

where σ_r is the support of $X \cup Y$, and α_r is the confidence for the rule r given by $\sigma(X \cup Y) / \sigma(X)$.

Despite some shortcomings (which we point out in the discussion of our experimental results), in many cases frequent itemsets and association rules can be used directly to provide effective recommendations as part of the personalization task. We will describe a simple and efficient technique to do so in the subsequent sections. They are also the foundation of our *usage clustering* technique based on Association-Rule Hypergraph Partitioning [HKKM97, HKKM98], which is used as part of a more general and robust method for computing recommendations.

Clustering Transactions

Traditional collaborative filtering techniques are often based on matching the current user's profile against clusters of similar profiles obtained by the system over time from other users. A similar technique can be used in the context of Web personalization by first clustering user transactions identified in the preprocessing stage. However, in contrast to collaborative filtering, clustering user transactions based on mined information from access logs does not require explicit ratings or interaction with users. In our case, user transactions are mapped into a multi-dimensional space as vectors of URL references. Standard clustering algorithms generally partition this space into groups of items that are close to each other based on a measure of distance. In the case of Web transactions, each cluster represents a group of transactions that are similar based on co-occurrence patterns of URL references.

Given a user transaction $t \in T$, we can represent the transaction as a (bit) vector

$$\vec{t} = \left\langle u_1^t, u_2^t, \ldots, u_n^t \right\rangle$$

where

$$u_i^t = \begin{cases} 1, & \text{if } url_i \in t \\ 0, & \text{otherwise} \end{cases}$$

Some other proposals [SZAS97, YJGD96] have suggested using, instead of binary weights, feature weights based on the time a user spends on a particular page or the frequency of occurrence of a URL reference within the user transaction. However, neither of these seems intuitively or practically justifiable in the context of Web transactions. For example, studies have suggested [KMM+97] that for a particular user, the amount of time spent on a page may not generally be a good indication of interest. Furthermore, the frequency of a reference is not generally a good measure of importance of page to a user; it only indicates the use of a page as a localized navigational nexus for that particular user. On the other hand, whether the URL reference occurs or whether it does not is clearly important. We have thus chosen to use only binary feature weights for our vector representation.

To cluster transactions we need a measure of distance between two transactions. Given two transactions t and s, we define the similarity $sim(t, s)$ as the normalized cosine of the angle between the two vectors. In case of binary vectors, $sim(t, s)$ can be expressed in terms of the size of the corresponding transactions as:

$$sim(t,s) = \frac{|t \cap s|}{\sqrt{|t| \cdot |s|}}$$

Note that we do not take into account the temporal order of URL references within transactions. While such a constraint can be easily added to the processes of deriving user transactions and clustering of transactions, however, sequential navigational patterns [SF99] seem to play a more important role when the purpose of Web usage mining is to improve the quality of the site design and the flow of traffic rather than providing dynamic recommendations for users.

Based on the above definition, a similarity matrix can be computed and used as the basis for clustering. A variety of clustering techniques can be used for clustering transactions. We use a multivariate k-means algorithm to obtain transaction clusters. However, distance-based clustering methods often break down when handling very high-dimensional data. In the context of clustering user transactions, the number of dimensions are the URLs appearing in the access logs and could number in the thousands. Our experiments suggest that filtering the logs by removing references to low support URLs can provide an effective dimensionality reduction method while improving clustering results.

Transaction clustering will result in a set $C = \{c_1, c_2, \ldots, c_k\}$ of clusters, where each c_i is a subset of T, i.e., a set of user transactions. Essentially, each cluster represents a group of users with "similar" access patterns. However, transaction clusters by themselves are not an effective means of capturing an aggregated view of common user profiles. Each transaction cluster may potentially contain thousands of user transactions involving hundreds of URL references. Our ultimate goal in clustering user transactions is to reduce these clusters into *URL clusters*. The URL clusters associated with each transaction cluster will serve as an aggregated view and a representative of users whose behavior is captured in the transaction clusters.

For each cluster $c \in C$, we first compute the mean vector m_c. The mean value for each URL in the mean vector is computed by finding the ratio of number of occurrence of that URL across transactions in the cluster to the total number of transactions in the cluster. The mean vector can be viewed as a set of URLs each having an associated weight corresponding to the mean value for that URL in the mean vector. We then filter out low-support URLs, i.e. those with mean value below a certain threshold μ. Thus, a URL cluster associated with a transaction cluster c, is the set of all URLs in whose weight is greater than or equal to μ. For example, if the threshold μ is set at 0.5, then each URL cluster will contain only those URL references which appear in at least 50% of transactions within its associated transaction cluster. Each URL cluster, in turn, can be represented as vectors in the original n-dimensional space.

Once the representative URL clusters have been computed, a partial session for the current user (*the active session*) can be assigned to a matching cluster. We must then determine which URLs from this cluster should be provided to the user as recommendations. The details of the matching and recommendation process are discussed in the next section.

Usage Clusters

Another approach we consider is to directly compute clusters of URL references based on how often they occur together across user transactions (rather than clustering transactions, themselves). We call the URL clusters obtained in this way, *usage clusters*. In general, this technique will result in a different type of URL clusters compared to the transaction clustering technique. The URL clusters obtain by reducing transaction clusters group together pages that co-occur commonly across "similar" transactions. On the other hand, usage clusters tend to group together frequently co-occurring items across transactions, even if these transactions are themselves not deemed to be similar. This allows us to obtain clusters that potentially capture overlapping interests of different types of users. The question of which type of clusters are most appropriate for personalization tasks is an open research question, however, the answer to this question, in part, depends on the structure and content of the specific site. We revisit this issue in the discussion of our experimental results.

However, traditional clustering techniques, such as distance-based methods generally cannot handle this type clustering. The reason is that instead of using URLs as features, the transactions must be used as features, whose number is in tens to hundreds of thousands in a typical application. Furthermore, dimensionality in this context may not be appropriate, as removing a significant number of transactions as features may lose too much information.

We have found that the Association Rule Hypergraph Partitioning (ARHP) technique [HKKM97, HKKM98] is well-suited for this task, especially since it provides automatic filtering capabilities, and does not require distance computations. Furthermore, ARHP is particularly useful in efficiently clustering high-dimensional data sets without requiring dimensionality reduction as a preprocessing step. The ARHP has been used successfully in a variety of domains, including the categorization of Web documents [HBG+99].

The set I of frequent itemsets discovered in the mining stage are used as hyperedges to form a hypergraph $H = \langle V, E \rangle$, where $V \subseteq U$ and $E \subseteq I$. A hypergraph is an extension of a graph in the sense that each hyperedge can connect more than two vertices. The weights associated with each hyperedge are computed based on the confidence of the association rules involving the items in the frequent itemset. Note that additional constraints can be placed on the itemsets in I to obtain E. For example, we may be interested in itemsets satisfying criteria such as minimum support, minimum size, or temporal constraints among items within the itemset. The hypergraph H is then partitioned into a set of clusters $C = \{c_1, c_2, \ldots, c_h\}$. The similarity among items is captured implicitly by the frequent item sets. Each cluster c_i represents a group of items (URLs) which are very frequently accessed together across transactions.

Each cluster is examined to filter out vertices that are not highly connected to the rest of the vertices of the partition. The connectivity function of vertex v (a URL appearing in the frequent itemset) with respect to a cluster c is defined as follow:

$$conn(v,c) = \frac{\left|\{e \mid e \subseteq c, v \in e\}\right|}{\left|\{e \mid e \subseteq c\}\right|}$$

Connectivity measures the percentage of edges with which a vertex is associated. A high connectivity value suggests that the vertex has many edges, connecting a good proportion of the vertices in the partition. The vertices with connectivity measure greater than a given threshold value are considered to belong to the partition, and the remaining vertices are dropped from the partition. In the ARHP method, additional filtering of non-relevant items can be achieved using the support criteria in the association rule discovery components of the algorithm. Depending on the support threshold. Items that do not meet support (i.e., URLs appear often in transactions with other URLs) will be pruned.

The connectivity value of an item (URL) defined above is important also because it is used as the weight associated with that item for the cluster. As noted in the case of transaction clustering, the weights associated with URLs in each cluster are used as part of the recommendation process when clusters are matched against an active user session.

The Recommendation Process

The recommendation engine is the online component of the Web personalization system based on usage mining. The task of the recommendation engine is to compute a *recommendation set* for the current session, consisting of links to pages that the user may want to visit based on similar usage patterns. The recommendation set essentially represents a "short-term" view of potentially useful links based on the user's navigational activity through the site. These recommended links are then added to the last page in the session accessed by the user before that page is sent to the user browser.

In general there are several factors that we would like to consider in determining the recommendation set. These factors include:

1. the matching criteria for each cluster or frequent itemset based on the its similarity to the current active session;
2. whether the candidate URLs for recommendation have been visited by the user in the current active session;
3. a short-term history depth for the current user representing the portion of the user's activity history that we should consider relevant for the purpose of making recommendations; and
4. the length of the physical link path from the current active session window to the candidate URL for recommendation.

Using a fixed-size sliding window over the current active session can capture the current user's history depth. For example, if the current session (with a window size of 3) is <*A*, *B*, *C*>, and the user references the URL *D*, then the new active session becomes <*B*, *C*, *D*>. This makes sense in the context of personalization since most users go back and forth while navigating a site to find the desired information. For example a user might navigate to a particular "content" page using a path of a certain size before deciding to go back and follow a different path to another content page representing an independent piece of information. In many cases these *sub-sessions* have a length of no more than 2 or 3 references. In such a situation, it may not be valuable to use references a user made in a previous sub-session to make recommendations during the current sub-session.

Thus, the sliding window of size n over the active session allows only the last n visited pages to influence the recommendation value of items in the recommendation set. In this context, the notion of a sliding session window is similar to the notion of *N-grammars* discussed in [Cha96]. An important question worth further investigation is the determination of the optimum window size automatically. Our experiments indicate that the average user session length may be a good measure for determining the window size.

The length of the physical link path can be determined by maintaining a directed graph representing the topology of the site. Each node in the graph is a URL representing a page in the site. There is a directed edge from a node x to a node y, if there is a physical link to the page corresponding to y from the page corresponding to x. We would like to weight a candidate URL higher (for the purpose of recommendation), if it is farther away form the current active session. In a sense, we would consider such a candidate URL a better recommendation than one with close proximity to the user's current location.

The *physical link distance* between two URLs u_1 and u_2 is the length of a minimal path from u_1 to u_2 in the site graph described above. Given a site graph G = (V, E), where the set of vertices $V \subseteq URL$, we denote the physical link distance factor between the active session s and a URL $u \notin s$ by $dist(u,s,G)$, which is defined as the smallest physical link distance between u and any of the URLs in s. Then, the *link distance factor* of u with respect to s is defined as:

$$ldf(u,s) = \log(dist(u,s,G)) + 1$$

If the URL u is in the active session, then $ldf(u,s)$ is taken to be 0. We take the log of the link distance so that it does not count too heavily compared to item weights within clusters or the frequent itemsets.

Computing Recommendations Directly from Frequent Itemsets

An efficient method for computing recommendation sets is to directly utilize the frequent itemsets obtained in the offline usage mining stage. After identifying user sessions in the preprocessing steps and then discovering frequent itemsets of URLs, we match the current user session window with itemsets to find candidate URLs for giving recommendations. Given a window size of w, we only consider all itemsets of size $w+1$ satisfying a specified support threshold and containing the current session window. The recommendation value of each candidate URL is based on the confidence of the corresponding association rule whose consequent is the singleton containing the candidate URL. If the rule satisfies a specified confidence threshold requirement, then the candidate URL is added to the recommendation set.

The basic outline of the algorithm is as follows:

```
Input:  an active session window w of fixed maximum size s
        The site graph G (for computing the link distance factor)
        Minimum support threshold σ
        Minimum confidence α

Recommend ← ∅
for each frequent itemset I of size |w|+1 such that w ⊆ I do
    if support(I) ≥ σ then
        let c = confidence(w =>{u})
        if c ≥ α then
            u.rec_score ← c * ldf(u, w)
            Recommend ← Recommend ∪ {u}
        end if
    end if
end for
```

Figure 2. Algorithm for Computing Recommendations Using Frequent Itemsets

Note that the search for itemsets (of size $|w|+1$) containing the current session window can be done efficiently, since during the mining process the discovered itemset can be organized into a lexicographic tree [AAP99]. Each node at depth d in the tree corresponds to an itemset, I, of size d with its children being itemsets of size d+1 that contain I.

It should be noted that, depending on the support threshold used in the above algorithm, it might be difficult to find large enough itemsets that could be used for providing recommendations. This is particularly true for sites with very small average session sizes. An alternative to reducing the support threshold in such cases would be to reduce the session window size. This latter choice may itself lead to some undesired effects since we may not be taking enough of the user's activity history into account. This kind of a situation is further examined in our experimental result section below.

Computing Recommendations Based on URL Clusters

Either of the clustering methods described earlier (usage clustering based on hypergraph partitioning or transaction clustering and computing the mean cluster values) will ultimately result in a set of clusters each containing highly related URLs with respect to user transactions. Depending on the clustering method used, or the mechanism for reducing transaction clusters into a set of URLs, a weight will be associated with each URL u in a cluster c, which we denote by $weight(u, c)$. Recall that in the case of transaction clustering the weight for a URL is its mean value in the cluster mean transaction m_c. In the case of usage clusters obtained using the ARHP method, the weight is the connectivity value of the item within the cluster, $conn(u, c)$.

Each of these URL clusters can be viewed as virtual user profile indicating how various groups of users may access a set of links in the site within their respective user transactions. Once a new user starts a session, our goal is to match, at each step, the partial user session with the appropriate clusters and provide recommendations to the user. The recommendations would be presented as a set of links which will be dynamically added to the top of the page corresponding to the last URL reference made by the user. So, the first task is to obtain a matching score for each cluster based on the cluster similarity to the current active session.

We represent each cluster $c \in C$, as a vector

$$\vec{c} = \left\langle u_1^c, u_2^c, \ldots, u_n^c \right\rangle$$

where

$$u_i^c = \begin{cases} weight(url_i, c), & \text{if } url_i \in c \\ 0, & \text{otherwise} \end{cases}$$

Furthermore, the current active session s is also represented as a vector

$$\vec{s} = \left\langle s_1, s_2, \ldots, s_n \right\rangle$$

where $s_i = 1$, if the user has accessed URL_i in this session, and $s_i = 0$, otherwise.

The cluster matching score is now computed as follows:

$$match(s,c) = \frac{\sum_k u_k^c \cdot s_k}{|s| \cdot \sqrt{\sum_k (u_k^c)^2}}$$

In computing the matching score we normalize for the size of the clusters and the active session. This is important in the context of matching active sessions with clusters in order to make recommendations. During an active session, given two matching clusters with the same non-normalized matching score, the larger cluster should be weighed less. This corresponds to the intuitive notion that we should see more of the user's active session before obtaining a better match with the larger cluster.

We can also impose a minimum threshold τ on the matching score. In this case, we consider a cluster c as a matching cluster for the active session s, only if $match(s, c) \geq \tau$. The value of τ is domain specific and may depend on the statistical properties of the access logs for the site.

Once the matching score is computed for an active session, we have to decide which of the URLs from these matching clusters are to be presented to the user. Thus, our next task is to construct a recommendation set for the user. To compute the recommendation value for a URL within a matching cluster, we take into account both the physical link distance to the current active session and the matching score for the cluster. Given a cluster c and an active session s, we compute a recommendation score for each URL within the cluster:

$$Rec(s,u) = \left(\sqrt{weight(u,c) \cdot match(s,c)}\right) \cdot ldf\,(s,u)$$

Note that if the URL u is in the current active session, then its recommendation value is zero because of the link distance factor. Finally, we can compute the recommendation set $Recommend(s)$ for current active session s by collecting from each cluster all URLs whose recommendation score satisfies a minimum recommendation threshold ρ:

$$Recommend(s) = \{u_i^c \mid c \in C, \text{ and } Rec(s,u_i^c) \geq \rho\}$$

The URLs in the recommendation set are ranked according to their recommendation score, when presented to the user. Furthermore, for each URL that is contributed by several clusters, we use its maximal recommendation score from all of the contributing clusters.

Experimental Results

We used the access logs from the University of Minnesota Computer Science Web server to test the three methods discussed earlier. The preprocessed log (for February of 1999) was converted into a session file comprising 14294 user transactions and a total of 4001 unique URLs (before support filtering). We provide a summary of the results below.

Recommendations Based on Usage Clusters

In this experiment using the hypergraph partitioning algorithm as modified by [CC99] in order to take frequent itemsets as the input performed the clustering of URLs. The frequent itemsets were found using the tree projection algorithm described in [AAP99]. Each URL serves as a vertex in the hypergraph, and each edge represents a frequent itemset with the weight of the edge taken as the interest for the set. Since interest increases dramatically with the number of items in a rule, the log of the interest is taken in order to prevent the larger rules from completely dominating the clustering process. For the recommendation process we chose a session window size of 2, since the average session size was 2.4. The recommendation results are given for the sample path `/research => /grad-info => /registration-info`.

Each table below corresponds to one step in the user navigation through the path. In each case the current active session window is given along with the top recommendations. A cut-off value of 0.30 was used for the recommendation score.

Session Window	Recommendation	Score
/research	/newsletter/newfaculty.html	0.73
	/newsletter	0.65
	/faculty	0.55
	/research/cnmrg	0.55
	/research/softeng	0.55
	/research/airvl	0.51
	/research/mmdbms	0.48
	/research/agassiz	0.47
	/personal-pages	0.37
	/registration-info	0.35
	/registration-info/spring99.html	0.32
	/grad-info	0.30
	/registration-info/schedule99-00.html	0.30
	/grad-research	0.30

Session Window	Recommendation	Score
/research /grad-info	/faculty	0.59
	/personal-pages	0.52
	/newsletter/newfac.html	0.52
	/newsletter	0.46
	/grad-info/grad-handbook.html	0.45
	/grad-info/course-guide.html	0.45
	/grad-info/prospective-grads.html	0.40
	/registration-info	0.39
	/research/cnmrg	0.39
	/research/softeng	0.39
	/research/airvl	0.36
	/registration-info/spring99.html	0.35
	/research/mmdbms	0.34
	/research/agassiz	0.33
	/registration-info/schedule99-00.html	0.33

Session Window	Recommendation	Score
/grad-info /registration-info	/faculty	0.51
	/personal-pages	0.45
	/grad-info/grad-handbook.html	0.45
	/grad-info/course-guide.html	0.45
	/grad-info/prospective-grads.html	0.40
	/registration-info/spring99.html	0.36
	/registration-info/schedule99-00.html	0.34

Note that in many cases the obvious recommendations associated with the URLs in the active session window rank lower than URLs for pages that are farther away in the site graph. This variation is mainly due to the link distance factor discussed earlier. In each case the recommendation set is composed of URLs from a number of matching clusters. When `/research` page is requested, the URLs for a number of popular research groups in the department are added to the set. When `/grad-info` is requested some of the frequently visited URLs associated with that page as well as related class registration pages rank higher.

Recommendations Using Transaction Clustering

For this experiment the same session file was clustered into clusters of transactions. We used multivariate k-means clustering for this task. The transaction clusters were converted to URL clusters by computing the mean transaction for each cluster and assigning to each URL in the cluster its associated mean value. Again a cut-off recommendation value of 0.30 was used in the resulting recommendation sets.

Session Window	Recommendation	Score
/research	/faculty	0.62
	/grad-info	0.56
	/grad-research	0.53
	/personal-pages	0.47
	/tech-reports	0.44
	/research/cnmrg	0.40
	/research/mmdbms	0.40
	/research/airvl	0.37
	/research/agassiz	0.31
	/grad-info/grad-handbook.html	0.30

Session Window	Recommendation	Score
/research /grad-info	/grad-info/grad-handbook.html	0.60
	/faculty	0.48
	/grad-research	0.41
	/personal-pages	0.37
	/tech-reports	0.34
	/grad-info/course-guide.html	0.32
	/research/cnmrg	0.31
	/research/mmdbms	0.31
	/registration-info/spring99.html	0.30

Session Window	Recommendation	Score
/grad-info /registration-info	/grad-info/grad-handbook.html	0.61
	/registration-info/spring99.html	0.60
	/grad-info/course-guide.html	0.33
	/registration-info/schedule99-00.html	0.32
	/personal-pages	0.30

In comparing the results with those obtained by usage clustering, we observe that these results (as well as results from other experiments with a variety of usage data), support our intuition that the usage clustering method can capture overlapping interests of different types of users, even if the associated transaction profiles are not considered similar enough. For example, in the first reference to the `/research` page, the usage clustering method, in addition to the core set of recommendations, also provided recommendations for users (mainly graduate students) who may be interested in registering for courses, as well users who may be interested in finding out about research areas of new faculty. Similar observations can be made about the other steps in the sample path. On the other hand, the transaction clustering technique seems to provide a narrower aggregated view of usage activity more directly centered around the a core set of URLs. Which of these methods is more suitable as part of Web personalization may depend on the structure and content of a particular site, as well as the goals of the site designers and operators.

Recommendations Using Frequent Itemsets

For the itemset method the frequent itemsets were discovered using a support threshold of 0.25%. However, only a window size of 1 was used, i.e. the algorithm used itemsets of size 2 to determine the recommendation set. This is because the data set contained very few itemsets of size larger than 2 with sufficient support. A lower support threshold setting would have yielded larger itemsets. A confidence threshold of 0.1 was used in the selection of candidate URL recommendations. The results of the experiments are given below. Note that in general, given the session window $w = \langle s_1, s_2, \ldots, s_n \rangle$, the recommendation score of each URL u_i in the recommendation set is the confidence of the association rule $\{s_1, s_2, \ldots, s_n\} \Rightarrow \{u_i\}$ multiplied by the link distance factor $ldf(w, u)$. The results are summarized below:

Session Window	Recommendation	Score
`/research` `Support = 2.58%`	`/faculty`	0.39
	`/grad-info`	0.34
	`/personal-pages`	0.27
	`/grad-research`	0.22
	`/research/airvl`	0.19
	`/research/cnmrg`	0.19
	`/research/softeng`	0.16
	`/research/agassiz`	0.15
	`/research/mmdbms`	0.15
	`/research/grouplens`	0.13
	`/tech-reports`	0.13

Session Window	Recommendation	Score
`/grad-info` `Support = 7.35%`	`/grad-info/course-guide.html`	0.22
	`/personal-pages`	0.17
	`/grad-info/grad-handbook.html`	0.15
	`/faculty`	0.14
	`/grad-info/prospective-grads.html`	0.14

Session Window	Recommendation	Score
`/registration-info` `Support = 4.20%`	`/registration-info/spring99.html`	0.50
	`/registration-info/schedule99-00.html`	0.20
	`/undergraduate-info`	0.16

While the itemset method is quite efficient and can provide good results in many cases, the above experiment also shows the limitations of this method. For example, if in step 2 a window size of 2 was used (containing `/research` and `/grad-info`), then the resulting recommendation set would have reduced to the set {<`/faculty, 0.50`>, <`/personal-pages, 0.46`>}, thus missing the frequently visited pages descending from `/grad-info`. On the other hand, the window size of one does not take enough of the user session history into account. For example, in the last step, the URL `/undergraduate-info` is given as a recommendation while the hypothetical user is most likely a graduate student. As the results suggest, in cases were the use of the itemset recommendation algorithm is justified, this method can provide even a narrower aggregated view of usage patterns than the transaction clustering method.

Other Experiments

A series of other experiments were conducted with data using the site http://acr-news.org which has a dedicated server for the newsletter of the Association for Consumer Research. The site contains a variety of news items, including President's columns, conference announcements, and call-for-papers for a number of conferences and journals. As an example we describe the results of one experiment using transaction clustering technique described in this paper. The session file for the experiments contained

18430 user transactions with a total of 192 unique URLs. Again, based on the average session size, a window of size 2 was chosen for the experiment. The sample path tested was:

```
President's Column => Conference Update => Calls for Papers =>
Special Topics  => Note from the Editor =>ACR News Online Archives =>
ACR 2000 Asia-Pacific Conference
```

A summary of the results is provided in the table below. The recommendations are presented using the title tag of HTML pages associated with the URLs in the recommendation set.

Session Window	**Recommendation**	**Score**
President's Column	News from the Grapevine	0.39
	ACR News Online Archives	0.37
	Note from the Editor	0.35
	Conference Update	0.34
	Special Topics	0.32
President's Column Conference Update	Calls for Papers	0.52
	ACR 1999 European Conference	0.47
	Note from the Editor	0.43
	1999 ACR Annual Conference	0.39
	ACR 2000 Asia-Pacific Conference	0.39
	ACR News Online Archives	0.34
	ACR News Updates	0.33
Conference Update Calls for Papers	ACR 1999 European Conference	0.72
	ACR News Updates	0.64
	1999 ACR Annual Conference	0.57
	ACR 2000 Asia-Pacific Conference	0.56
	Note from the Editor	0.36
	Journal of Psychology and Marketing	0.35
	Seminar in Marketing Communications	0.34
	ACR News Online Archives	0.34
Calls for Papers Special Topics	ACR News Updates	0.63
	ACR 1999 European Conference	0.58
	1999 ACR Annual Conference	0.57
	Journal of Psychology and Marketing	0.50
	ACR News Online Archives	0.43
	Note from the Editor	0.42
	ACR 2000 Asia-Pacific Conference	0.40
	News from the Grapevine	0.39
Special Topics Note from the Editor	ACR News Online Archives	0.46
	News from the Grapevine	0.37
Note from the Editor ACR News Online Archives	Journal of Consumer Psychology	0.41
	President's Column - Sept 1997	0.36
	Journal of Psychology and Marketing	0.36
ACR News Online Archives ACR 2000 Asia-Pacific Conference	1999 ACR Annual Conference	0.76
	Journal of Consumer Psychology	0.41
	Journal of Psychology and Marketing	0.36
	President's Column - Sept 1997	0.34
	Calls for Papers	0.31

System Implementation and Demonstration Site

The ACR News site discussed in the previous section was used to implement a demonstration version of the Web personalization system based on the techniques and the architecture presented in this paper. A local version of the site which uses our recommendation engine is available for demonstration purposes from http://aztec.cs.depaul.edu/scripts/ACR2. For this demonstration, we used a subset of the ACR logs (from June 1998 to June 1999), and used transaction clustering to derive URL clusters. The transaction clustering process yielded 28 URL clusters representing different types of user access patterns. A threshold of 0.5 was used to derive URL clusters from transaction clusters (i.e., URL clusters contained only those URL references appearing in at least 50% of transactions). We used a recommendation threshold of 0.3 as a cut-off point to ensure capturing overlapping user interests. Based on the average session size, the system automatically chose a session window of size of 3 references.

The recommendation engine was implemented as a set of CGI scripts, using cookies to keep track of user's active session. Figures below depict a typical interaction of user with site. The top frame in each window contains the actual page contents from the site, while the bottom frame contains the recommended links. When the user clicks on a link in either frame, the top frame will display the content of the requested page, and the bottom frame is dynamically updated to include the new recommendations. As seen in Figure 3, initially the system does not provide any recommendations until the user has navigated through more pages. Figure 4 shows the recommendations resulting after the user has followed a path to "President's Column" and then to "Online Archives." The recommendations include past President Columns and Editor's Notes (as well as other pages) often visited by users who have shown similar access patterns.

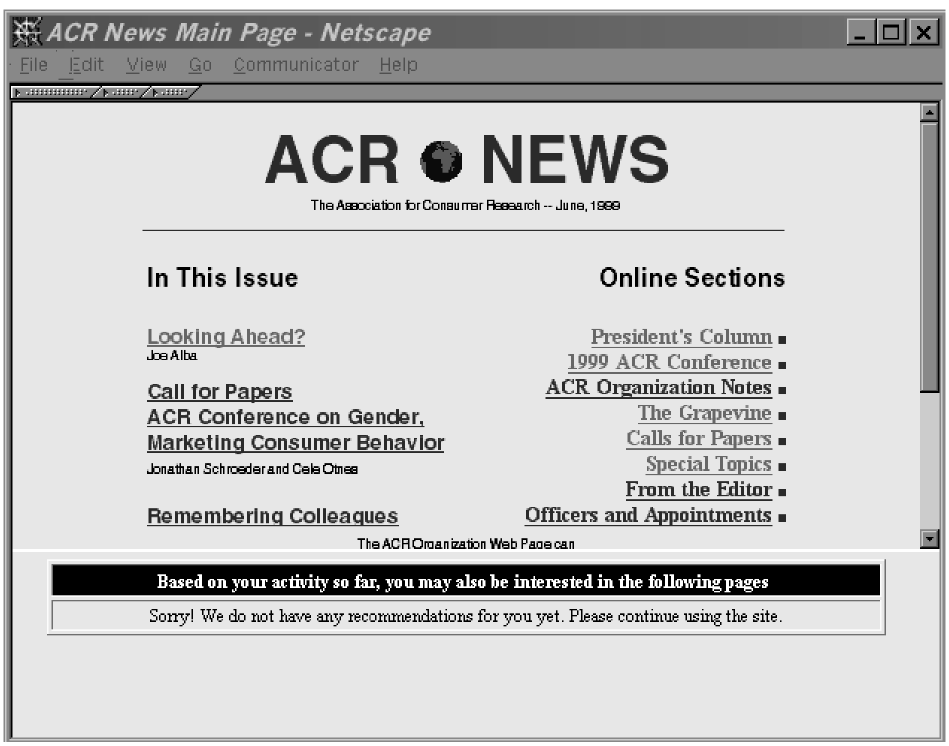
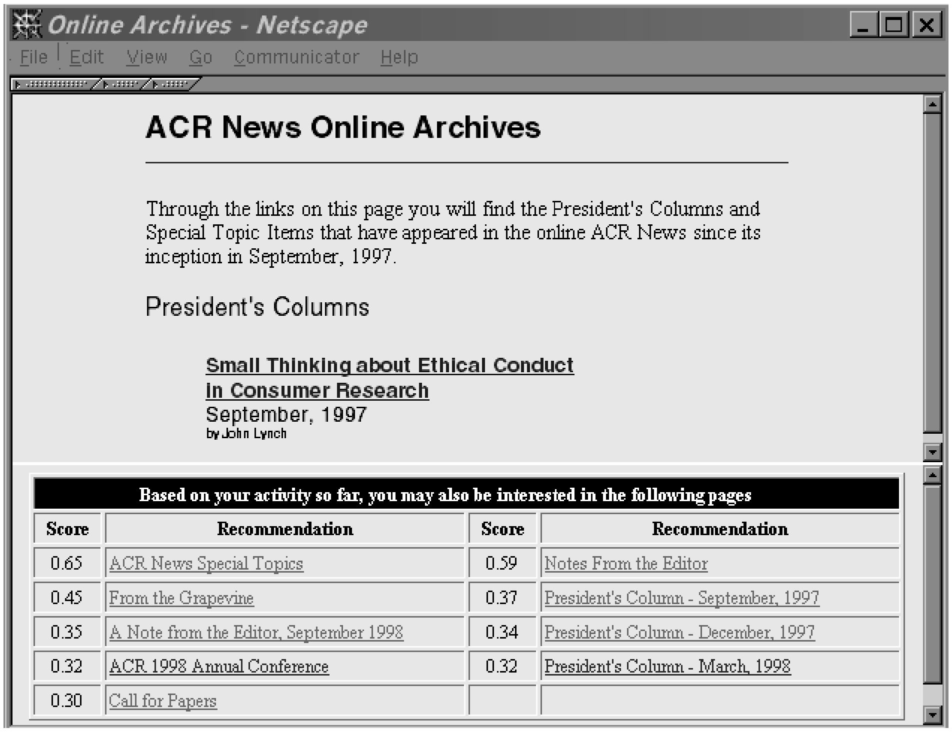


Figure 3. Main page for the demonstration site. Initially, no recommendations are provided as the active user session does not contain sufficient number of references.


Online Archives - Netscape

File Edit View Go Communicator Help

ACR News Online Archives

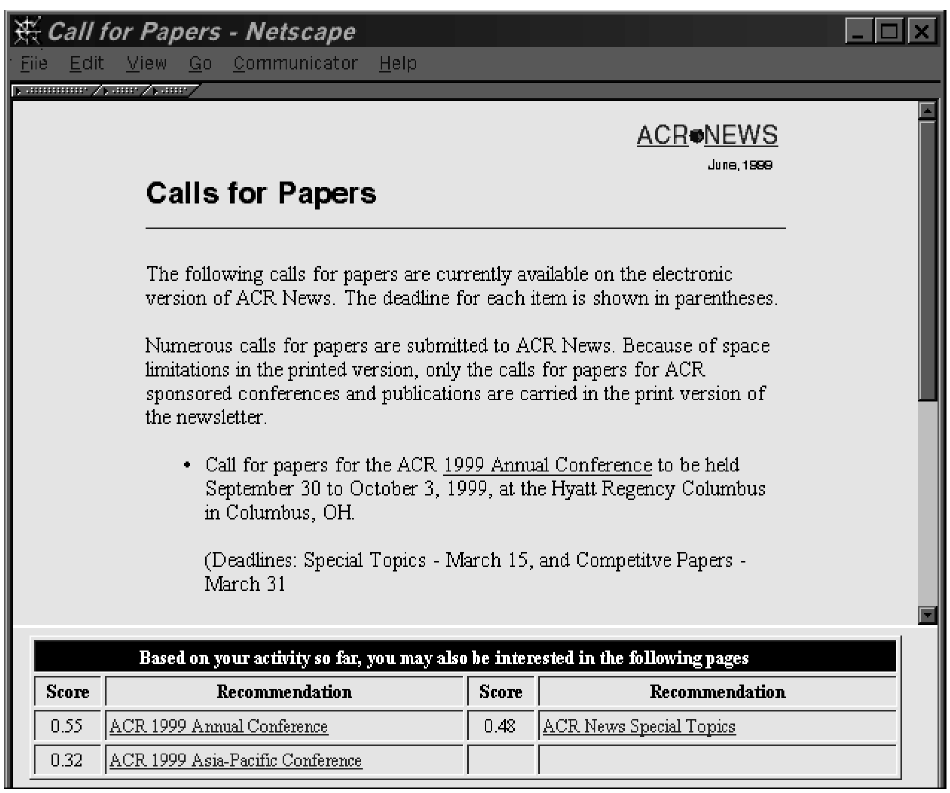
Through the links on this page you will find the President's Columns and Special Topic Items that have appeared in the online ACR News since its inception in September, 1997.

President's Columns

Small Thinking about Ethical Conduct in Consumer Research
September, 1997
by John Lynch

Based on your activity so far, you may also be interested in the following pages			
Score	Recommendation	Score	Recommendation
0.65	ACR News Special Topics	0.59	Notes From the Editor
0.45	From the Grapevine	0.37	President's Column - September, 1997
0.35	A Note from the Editor, September 1998	0.34	President's Column - December, 1997
0.32	ACR 1998 Annual Conference	0.32	President's Column - March, 1998
0.30	Call for Papers		

Figure 4. Dynamic recommendations after the user has navigated through "President's Column" and "Online Archive" pages.


Call for Papers - Netscape

File Edit View Go Communicator Help

ACR●NEWS
June, 1999

Calls for Papers

The following calls for papers are currently available on the electronic version of ACR News. The deadline for each item is shown in parentheses.

Numerous calls for papers are submitted to ACR News. Because of space limitations in the printed version, only the calls for papers for ACR sponsored conferences and publications are carried in the print version of the newsletter.

- Call for papers for the ACR 1999 Annual Conference to be held September 30 to October 3, 1999, at the Hyatt Regency Columbus in Columbus, OH.

  (Deadlines: Special Topics - March 15, and Competitve Papers - March 31

Based on your activity so far, you may also be interested in the following pages			
Score	Recommendation	Score	Recommendation
0.55	ACR 1999 Annual Conference	0.48	ACR News Special Topics
0.32	ACR 1999 Asia-Pacific Conference		

Figure 5. Recommendations based on user navigation through "Conference Update" and "Call for Papers" pages.

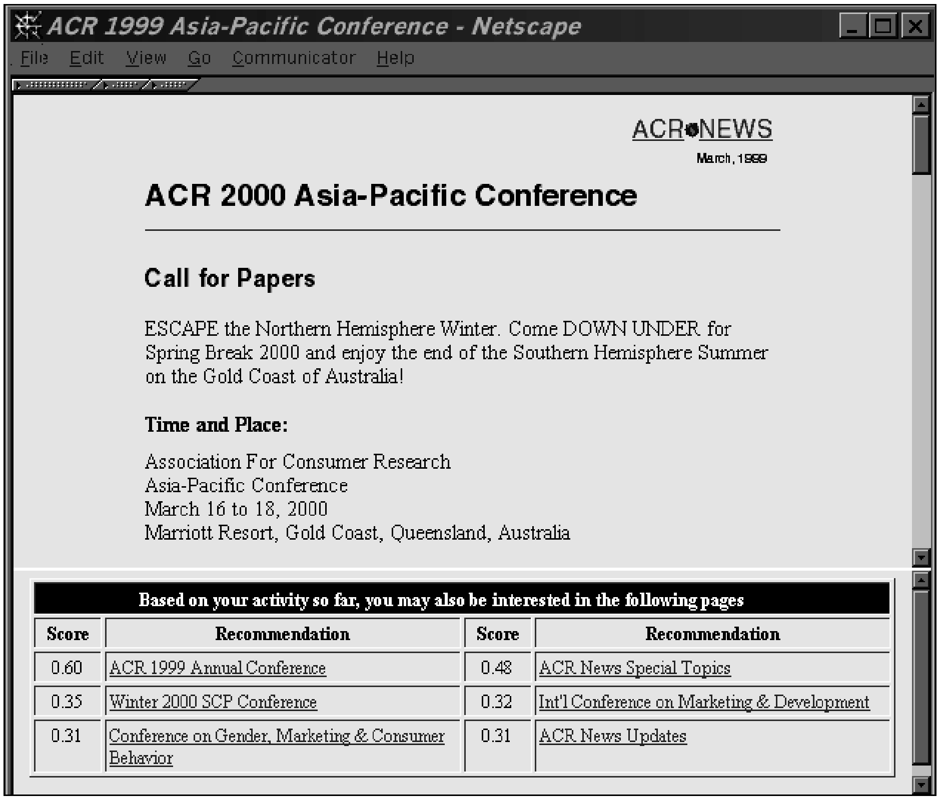


Figure 6. System provides specific recommendations related to conferences, based on user navigation through through "Conference Update," "Call for Papers," and "1999 Asia Pacific Conference" pages.

Figure 5 and 6 show the results of the user navigation through "Conference Update," "Call for Papers," and then "1999 Asia Pacific Conference". As can be seen in these Figures, user's intention of looking for more specific information will result in more specific recommendations. For example, in Figure 5, general recommendations are provided guiding the user to upcoming conferences and news items. When the user accesses a specific conference page (Figure 6), other specific conference information is presented as potentially interesting (e.g., "Winter 2000 SCP Conference" and "Int'l Conference on Marketing and Development").

Conclusions

The Web is providing a direct communication medium between the vendors of products and services, and their clients. Coupled with the ability to collect detailed data at the granularity of individual mouse clicks, this provides a tremendous opportunity for personalizing the Web experience for clients. In e-commerce parlance this is being termed mass customization. Outside of e-commerce also, the idea of Web personalization has many applications. Recently there has been an increasing amount of research activity on various aspects of the personalization problem. Most current approaches to personalization by various Web-based companies rely heavily on human participation to collect profile information about users. This suffers from the problems of the profile data being subjective, as well getting out of date as the user preferences change over time.

We have provided several techniques in which the user preference is automatically learned from Web usage data, by using data mining techniques. This has the potential of eliminating subjectiveness from profile data as well as keeping it up-to-date. We describe a general architecture for automatic Web personalization based on the proposed techniques, and discuss solutions to the problems of usage data preprocessing, usage knowledge extraction, and making recommendations based on the extracted knowledge. Our experimental results indicate that the techniques discussed here are promising, each with its own unique characteristics, and bear further investigation and development.

References

[AAP99] Agarwal, R., Aggarwal, C., and Prasad, V., A tree projection algorithm for generation of frequent itemsets. In *Proceedings of High Performance Data Mining Workshop*, Puerto Rico, 1999.

[AS94] Agrawal, R. and Srikant, R., Fast algorithms for mining association rules. *In Proceedings of the 20th VLDB conference*, pp. 487-499, Santiago, Chile, 1994.

[BM99] Buchner, A. and Mulvenna, M. D., Discovering internet marketing intelligence through online analytical Web usage mining. *SIGMOD Record*, (4) 27, 1999.

[Cha96] Charniak, E., *Statistical language learning*. MIT Press, 1996.

[CC99] Clifton, C. and Cooley, R., TopCat: data mining for topic identification in a text corpus. In *Proceedings of the 3rd European Conference of Principles and Practice of Knowledge Discovery in Databases*, Prague, Czech Republic, 1999.

[CMS99] Cooley, R., Mobasher, B., and Srivastava, J., Data preparation for mining World Wide Web browsing patterns. *Journal of Knowledge and Information Systems*, (1) 1, 1999.

[CPY96] Chen, M. S., Park, J. S., and Yu, P. S., Data mining for path traversal patterns in a Web environment. *In Proceedings of 16th International Conference on Distributed Computing Systems*, 1996.

[HBG+99] Han, E-H, Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., and Mobasher, B., More, J., Document categorization and query generation on the World Wide Web using WebACE. *Journal of Artificial Intelligence Review*, January 1999.

[HKBR99] Herlocker, J., Konstan, J., Borchers, A., Riedl, J.. An algorithmic framework for performing collaborative filtering. To appear in *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, August 1999.

[HKKM97] Han, E-H, Karypis, G., Kumar, V., and Mobasher, B., Clustering based on association rule hypergraphs. In *Proccedings of SIGMOD'97 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'97),* May 1997.

[HKKM98] Han, E-H, Karypis, G., Kumar, V., and Mobasher, B., Hypergraph based clustering in high-dimensional data sets: a summary of results. *IEEE Bulletin of the Technical Committee on Data Engineering*, (21) 1, March 1998.

[KMM+97] Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J., GroupLens: applying collaborative filtering to usenet news. *Communications of the ACM* (40) 3, 1997.

[NFJK99] Nasraoui, O., Frigui, H., Joshi, A., Krishnapuram, R., Mining Web access logs using relational competitive fuzzy clustering. To appear in *the Proceedings of the Eight International Fuzzy Systems Association World Congress*, August 1999.

[PE98] Perkowitz, M. and Etzioni, O., Adaptive Web sites: automaticlly synthesizing Web pages. In *Proceedings of Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998.

[SF99] Spiliopoulou, M. and Faulstich, L. C., WUM: A Web Utilization Miner. *In Proceedings of EDBT Workshop WebDB98*, Valencia, Spain, LNCS 1590, Springer Verlag, 1999.

[SKS98] Schechter, S., Krishnan, M., and Smith, M. D., Using path profiles to predict HTTP requests. In *Proceedings of 7th International World Wide Web Conference*, Brisbane, Australia, 1998.

[SM95] Shardanand, U., Maes, P., Social information filtering: algorithms for automating "word of mouth." In *Proceedings of the ACM CHI Conference*, 1995.

[SZAS97] Shahabi, C., Zarkesh, A. M., Adibi, J., and Shah, V., Knowledge discovery from users Web-page navigation. In *Proceedings of Workshop on Research Issues in Data Engineering*, Birmingham, England, 1997.

[YJGD96] Yan, T., Jacobsen, M., Garcia-Molina, H., Dayal, U., From user access patterns to dynamic hypertext linking. In Proceedings of the 5th International World Wide Web Conference, Paris, France, 1996.

ERROR: undefined
OFFENDING COMMAND: --0FDCB3DF-WebSite-Rules-Byte-Range-Data-0FDCB3DF--

STACK: