Efficient and Anonymous Web-Usage Mining for Web Personalization

Cyrus Shahabi • Farnoush Banaei-Kashani

Department of Computer Science, Integrated Media Systems Center, University of Southern California, Los Angeles, California 90089-2561, USA

shahabi@usc.edu • *banaeika@usc.edu*

The world-wide web (WWW) is the largest distributed information space and has grown to encompass diverse information resources. Although the web is growing exponentially, the individual's capacity to read and digest content is essentially fixed. The full economic potential of the web will not be realized unless enabling technologies are provided to facilitate access to web resources. Currently web personalization is the most promising approach to remedy this problem, and web mining, particularly web-usage mining, is considered a crucial component of any efficacious web-personalization system. In this paper, we describe a complete framework for web-usage mining to satisfy the challenging requirements of web-personalization applications. For on-line and anonymous web personalization to be effective, web usage mining must be accomplished in real time as accurately as possible. On the other hand, web-usage mining should allow a compromise between scalability and accuracy to be applicable to real-life websites with numerous visitors. Within our web-usage-mining framework, we introduce a distributed user-tracking approach for accurate, scalable, and implicit collection of the usage data. We also propose a new model, the feature-matrices (FM) model, to discover and interpret users' access patterns. With FM, various spatial and temporal features of usage data can be captured with flexible precision so that we can trade off accuracy for scalability based on the specific application requirements. Moreover, tunable complexity of the FM model allows real-time and adaptive access pattern discovery from usage data. We define a novel similarity measure based on FM that is specifically designed for accurate classification of partial navigation patterns in real time. Our extensive experiments with both synthetic and real data verify correctness and efficacy of our web-usage-mining framework for anonymous and efficient web personalization.

(*Web Usage Mining; Data Mining; Personalization; Pattern Discovery*)

1. Introduction

The world wide web (WWW) is the largest distributed information space (W3C 2002, Pitkow 1998), which has grown to encompass diverse information resources such as service and product catalogs, digital libraries, personal home pages, Usenet news, etc. More noticeably nowadays, the web is considered as the most appropriate environment for business transactions because it is convenient, fast, and inexpensive to use; hence we see the enormous popularity of electronic commerce and Business-to-Business applications.

Although the web is growing exponentially, the individual's capacity to read and digest content is essentially fixed. The web is a collection of semi-structured and structured information sources often visualized as a huge and complex dynamic mesh. Due to information explosion, constantly changing environment, poor understanding of users' needs and preferences, as well as lack of willingness to modify existing web data models, often web users suffer from information overload. Therefore, the full economic potential of the web has not been realized. The ability to access information in the web efficiently and efficaciously is an enabling technology for realizing its full potential.

Researchers from diverse disciplines such as machine learning, information retrieval, artificial intelligence, data management, etc. are now focusing on this topic in various industrial and academic research centers. Traditionally, search engines have been used to facilitate information access in the web. As the web continues to expand, search engines are becoming redundant because of the large number of pointers they return for a single search. We may think of two other approaches to deal with this problem. On the one hand, we may try to define a reference model intended to streamline the design and implementation of the web information systems, hence, standardizing information access in the web. Faulstich et al. (1997), Scharl (1999), Carchiolo et al. (2000), and Kohonen et al. (2000) are various examples of such an approach. However, since the web is in essence a semi-structured and decentralized environment, globalization of any reference model requires a lot of effort, if it is not simply impossible. On the other hand, with web personalization, or so called mass customization, we intend to customize the web environment for each user. Personalization is achieved by observing needs of the user and providing the preferred information based on user needs. In a personalized web information system, information access is enhanced by optimizing the percentage of relevant information exposed to user. Web personalization is currently known as the key to success in business today and in the future (Allen et al. 2001).

Collaborative filtering (Konstan et al. 1997, Breese et al. 1998), intelligent interfaces / agents and bots (Lieberman 1997, Ackerman et al. 1997, Finin et al. 1997, INT Media Group 2002), and intermediaries (Maglio et al. 2000) are all deployed as the enabling technologies for web personalization. Recently, *web mining*, a natural application of data-mining techniques to the web as a very large and unstructured information source, has made a great impact on web personalization. Through web Mining, we are able to gain a better understanding of both the web and web-user preferences; a knowledge that is crucial for mass customization (Mulvenna et al. 2000, Mobasher et al. 2000a).

In this paper, we describe a complete framework for *web-usage mining (WUM)*, an emergent domain in web mining that has greatly concerned both academia and industry in recent years (Srivastava et al. 2000). WUM is the process of discovering and interpreting patterns of user access to web information systems by mining the data collected from user interactions with the system. A typical WUM system consists of two tiers: 1) *tracking*, in which user interactions are captured and acquired, and 2) *analysis*, in which user access patterns are discovered and interpreted by applying typical data-mining techniques to the acquired data. Knowledge of user access patterns is useful in numerous applications: supporting website design decisions such as content and structure justifications (Spiliopoulou 2000, Drott 1998), optimizing systems by enhancing caching schemes and load-balancing, making websites adaptive (Perkowitz and Etzioni 2000), supporting business intelligence and marketing decisions (Büchner and Mulvenna 1998), testing user interfaces, monitoring for security purposes, and more importantly, in web personalization applications such as recommendation systems (Sarwar et al. 2000) and target advertising. Commercial products such as *Personify*^{TM} (Personify, Inc. 2002), *WebSideStory*^{TM} (WebSideStory Inc. 2002), *BlueMartini*^{TM} (Blue Martini Software, Inc. 2002), and *WebTrends*^{TM} (NetIQ 2002), and acquired companies such as *Matchlogic*^{TM}, *Trivida*^{TM}, *Andromedia*^{TM}, *Rightpoint*^{TM}, and *DataSage*^{TM} are all witnesses of commercial interests in WUM.

Within WUM-based web-personalization applications, the user access patterns discovered through WUM are applied to identify needs and preferences of each individual user, and subsequently, customize the content or structure of the web-information system based on user needs. This process often consists of two components. First, an off-line component learns a comprehensive users-access model by mining typical access patterns from training datasets. Second, once the access model is identified, it is used by an on-line component to interpret navigational behavior of the active users. The on-line component is required for *anonymous* identification of the user needs in real-time. One other typical but naive approach taken by industry to identify user needs is first to build a static profile for each user based on her/his first visit (e.g., Dialpad Communications, Inc. 2002, and MTV Networks 2002). In subsequent visits, user identity is detected using some intrusive technique such as cookies, and the static profile is applied for customization. There are several drawbacks to this approach. Tracking users across sessions violates users privacy. Besides, the static profile cannot distinguish between different roles the user might play during various visits, e.g., buying rock music CDs as a gift versus purchasing classical music for oneself. In addition, if multiple users share the same computer, the profile becomes a mixture of several, possibly conflicting, tastes. Finally, the static profile cannot capture changes in the user preferences. Thus, we believe the system should treat each user as an anonymous individual and identify user needs per session. In cases where the problems with static profiles are tolerable, anonymous web personalization can be applied in parallel to make user profiles adaptive.

Web-personalization applications impose a set of challenging requirements that are partially in conflict with each other. With anonymous web personalization, the on-line component of the system should run in real time, and in particular should be able to identify user needs in a fraction of the session period. Violation of this time constraint usually renders the result of the personalization less useful; for example, in a recommendation system, often there is little use for the recommendations that are generated after the user leaves the site. Thus, the time complexity for the process of applying the access model to interpret active sessions must be sufficiently low. Moreover, on the one hand the volume of navigation data generated per site is usually very large. For example, $Yahoo^{TM}$ has 166 million hits every day, generating 48GB of clickstream data per hour (Yahoo! Inc. 2001). These data cannot be analyzed in real time unless some features of the data are dropped while modeling access patterns and interpreting active sessions. On the other hand, since with anonymous web personalization the available information about a user is limited to the user interactions with the web during an active session period, we necessarily want to consider every single action of a user, or the customization will be inefficacious. Therefore, for efficient customization, user-access model learned in the personalization system should be flexible enough to allow an engineering compromise between scalability and accuracy based on the application specifications.

The WUM framework described in this paper composes an accurate tracking module and a tunable access model to support efficient and anonymous web personalization. Besides, tracking is performed implicitly to avoid reliance on human input (i.e., explicitly requesting users to state their interests). Implicit tracking is considered the ideal solution for the *sparsity* problem in web-personalization applications (Konstan et al. 1997). Also, the access model is adaptive to capture short-term changes in user behaviors. This last feature is particularly useful in delay-sensitive environments such as the stock market. It is important to note that, although we motivate and discuss this framework in the context of web-personalization applications, it is as much applicable for other conventional WUM applications as well.

The remainder of this paper is organized as follows. Section 2 summarizes related work. We briefly explain how to implement the tracking tier of the WUM framework in Section 3. In Section 4, we formally characterize the analysis tier of the WUM framework by defining our access-pattern model, describing our similarity measures, and discussing our dynamic clustering technique. The results of our experiments are in Section 5. Finally, Section 6 concludes the paper and proposes our future directions.

2. Related Work

Web mining is broadly defined as the discovery and analysis of useful information from the WWW. A detailed taxonomy of web-mining domains is provided in Mobasher et al. (1997). Here, we first briefly characterize different domains that pertain to web mining. Thereafter we will focus on some of the current research on WUM.

Target data sets for data mining in the context of the web are classified into the following types:

- **Content data:** The data meant to be conveyed to the web user. Naturally, *web-content mining* is the process of extracting knowledge from the content of web documents (Cohen et al. 2000).

- **Structure data:** The meta data that define the organization of the web information systems. *Web structure mining* is the process of inferring knowledge from the structure of data (Kuo and Wong 2000, Henzinger 2000).

- **Usage data:** The data collected from user interactions with the web. As mentioned before, WUM is the process of discovering and interpreting patterns of user access to the web information system (Baumgarten et al. 2000).

The idea of exploiting usage data to customize the web for individuals was suggested by researchers as early as 1995 (Armstrong et al. 1995, Lieberman 1995, Pazzani et al. 1995). A comprehensive survey of existing efforts in WUM is provided by Srivastava et al. (2000). Some of the current approaches with two tiers of WUM, tracking and analysis, are summarized in the following sections.

2.1 User-Interaction Tracking

Usage data can be collected from various sources: the web browser on the client side, web server logs, or proxy server logs. With anonymous web personalization, accuracy of tracking is of significant concern and should be applied as the main criterion in selecting a data source. There are various levels of caching embedded in the web, mainly to expedite users access to frequently used pages. Pages requested by hitting the "Back" button, which is heavily used by web users (Greenberg and Cockburn 1999), are all retrieved from the web browser cache. Also, proxy servers provide an intermediate level of caching at the enterprise level. Unfortunately, cache hits are missing from proxy and server logs, rendering them incomplete sources of information to acquire spatial features of user interactions. Lin et al. (1999) have tried to remedy this problem by introducing an "access pattern collection server," which is applicable only in environments where user privacy is not a concern. Cooley et al. (1999) explain several heuristics using the referrer and agent fields of a server log to infer missing references with relative accuracy; Spiliopoulou et al. (2002) study performance of various such heuristic techniques. Server and proxy logs are also inaccurate in capturing temporal aspects of user interactions. Timestamps recorded in these logs for page requests incorporate network transfer time. Due to nondeterministic behavior of the network, there is no trivial way to filter out these noise data. On the other hand, if temporal features are captured on the client side, occurrence times of all user interactions can be recorded as exactly as required. Thus, data collected at the client side provide us with the most accurate spatial and temporal information about user interactions with the web.

Moreover, with client-side tracking, preprocessing the data source, which is considered the most difficult task when server/proxy logs are used, can also be performed transparently on the client side. Hence, not only does it help with scalability of the system by distributing the preprocessing load among clients, it eliminates the most critical preprocessing task, i.e. session identification. Due to the stateless connection model of the HTTP protocol, pages requested in a session are logged independently in the server/proxy logs. For meaningful analysis, these requests must be re-identified and re-grouped into sessions as semantic units of analysis (Cooley et al. 1997, Zheng et al. 2002). In Shahabi et al. (1997), we introduce a remote agent that tracks user interactions on the client side. The data captured by each agent are stored as separate semantic units on the server side so that re-identification of the user sessions is not required.

However, client-side tracking has a drawback. It is usually implemented by remote agents developed as Javascripts or Java applets. Employing these agents to collect data at the client side requires user cooperation in enabling Java in their browsers. Considering the popularity of remote agents, on the one hand, and on the other hand benefits of client-side tracking as described above, we believe client-side tracking is the preferred approach, especially for web-personalization applications. In Shahabi et al. (2001), we describe a complete data-acquisition system based on our client-side data-collection idea.

2.2 Access-Pattern Analysis

As mentioned in Section 1, due to the large volume of usage data, they cannot be analyzed in real time unless some features of the data are dropped while modeling access patterns. Page *hit-count*, which indicates frequency of page visits during a session, has been traditionally considered as an informative indicator of user preferences (Yan et al. 1996). Also, *order* or *sequence* of page accesses has recently been identified as an important piece of information (Chen et al. 1998). Dependency models such as the aggregate tree (Spiliopoulou and Faulstich 1999) and hidden Markov models (Cadez et al. 2000) are used to capture this feature and to predict forward references. In addition to spatial features, temporal features such as page *view time* are also of significant concern, specially in the context of web-personalization applications (Konstan et al. 1997). Although Yan et al. (1996) and Levene and Loizou (2000) show that view time has a Zipfian distribution and might be misleading in cases where long accesses obscure the importance of other accesses, we argue that using view time in combination with other features can alleviate this unwanted effect. The model described in this paper is defined so that it can capture any number of these and any other feature that might seem informative. Features are captured per *segment*, which is the building block of a session (see Section 4.1). Variable number of features and size of the segment allow us to strike a compromise between accuracy and complexity of the model, depending on the application requirements.

Researchers have investigated various models and data-mining techniques to capture these features and to represent user access patterns. Mobasher et al. (2000b) have applied the classical association-rule apriori algorithm (Agrawal and Srikant 1994) to find "frequent item sets" based on their patterns of co-occurrence across user sessions. They deploy association rules to find related item sets to be recommended to the user based on the observed items in the user session. Mobasher et al. (2000a) show that clustering techniques provide better overall performance as compared to association rules when applied in the context of web personalization.

Another set of models, which we call *dependency models*, are applied to predict forward references based on partial knowledge about the history of the session. These models learn and represent significant dependencies among page references. Zukerman et al. (1999) and Cadez et al. (2000) use a Markov model for this purpose. Borges and Levene (1999) define a probabilistic regular grammar whose higher probability strings correspond to users' preferred access patterns. Breese et al. (1998) perform an empirical analysis of predictive algorithms such as Bayesian classification and Bayesian networks in the context of web personalization and demonstrate that performance of these algorithms is dependent on the nature of the application and completeness of the usage data. In Section 4.1.5, we compare our approach with the Markov model as a typical dependency model.

In this paper, we use another classical data-mining technique, *clustering*, to mine usage data. This approach was first introduced by Yan et al. (1996). With this approach, user sessions are usually modeled as vectors. In the original form of the vector model, each element of the vector represents the value of a feature, such as hit-count, for the corresponding web page. A clustering algorithm is applied to discover the user access patterns. Active user sessions are classified using a particular application-dependent similarity measure such as Euclidean distance. Recently, various clustering algorithms were investigated to analyze the clustering performance in the context of WUM. Fu et al. (1999) employ *BIRCH* (Zhang et al. 1996), an efficient hierarchical clustering algorithm; Joshi and Krishnapuram (1999) prefer a fuzzy relational clustering algorithm for WUM because they believe usage data are fuzzy in nature; Strehl and Ghosh (2002) propose relationship-based clustering for high-dimensional data mining in the context of WUM; Perkowitz and Etzioni (1998) introduce a new clustering algorithm, *cluster miner*, which is designed to satisfy specific web-personalization requirements; Paliouras et al. (2000) from the machine-learning community, compare performance of the *cluster miner* with two other clustering methods widely used in machine-learning research, namely *autoclass* and *self-organizing maps*, and show that Autoclass outperforms other methods. Mobasher et al. (2000a) observe that a user may demonstrate characteristics that are captured by different clusters while she/he is to be classified to a single cluster. Thus, they introduce the notion of *usage clustering*, a combination of clustering and association rules, to obtain clusters that potentially capture overlapping interests of different types of users. This goal is equivalently achievable by applying soft classification. The model described in this paper is a generalization of the original vector model introduced by Yan et al. (1996), to be flexible in capturing users' behavior anonymously, and combining various features with a tunable order of complexity. VanderMeer et al. (2000) study anonymous WUM by considering dynamic profiles of users in combination with static profiles. We also analyze *dynamic clustering* as an approach to make the cluster model adaptive to short time changes in users behavior. We also introduce an accurate similarity measure that avoids overestimation of the distance between partial user sessions and cluster representatives. Distance overestimation has been observed as a classification problem by Yan et al. (1996) as well.

3. Tracking Tier

As explained in Section 2.1, if tracking is performed on the client side, acquisition of spatial and temporal features of user interactions can be done accurately. In Shahabi et al. (1997), we proposed a method to run a *remote agent* on the client side without violating the privacy of the user. Here, we provide a brief overview of agent implementation (see Shahabi et al. 2001 for details).

The remote agent is implemented as a Java applet (Sun Microsystems, Inc. 2002). At the server side, there is a peer component to collect data from all remote agents running on the clients. The server-side component is implemented as a multi-threaded Java application with a one-dispatcher/multi-workers architecture. The applet is loaded into the client machine only once when the first page of the website is accessed. Subsequently, every time a new HTML page (say page A) is loaded at the client, the applet will send the system time as $T_{load}(A)$ to the server-side component to be recorded. Similarly, once the page is unloaded, the system time will again be reported to the server as $T_{exit}(A)$. Once we record $T_{load}(A)$ and $T_{exit}(A)$ for each page accessed by the user, we can extract all spatial and temporal features required for session analysis (see Section 4.1.3).

To capture T_{load} and T_{exit} by the Java applet, each HTML page should be modified to incorporate a call to the applet. The following are sample statements that should be added (automatically) to the beginning of every HTML page ("index.html" page in this example):

```
<APPLET CODEBASE="/java"
 CODE="RemoteAgent" WIDTH=1 HEIGHT=1>
 <PARAM NAME="PAGE_NAME"
  VALUE="http://imsc.usc.edu/index.html">
</APPLET>
```

Incorporating the above statements at the beginning of the page results in invocation of the applet "RemoteAgent" immediately when the page is loaded. Subsequently, the applet stops execution when the page is unloaded. The only disadvantage of this method is that the loading time of the applet is considered part of the view time of the first page; thereafter, the applet becomes resident in the client's cache and no more loading time will be encountered. Since the applet loading time affects only the loading time of the first page, we consider it negligible.

We performed experiments to compare the number of hits per page captured by the remote agent with the number of those recorded in a typical server log. The results support correctness and accuracy of our approach (see Section 5.1).

4. Analysis Tier

4.1 The Feature-Matrices Model

Here, we present a novel model to represent both sessions and clusters in the context of WUM. We call this model the *feature matrices* (FM) model. With FM, features are indicators of the information embedded in sessions. To quantify features, we consider a universal set of segments in a concept space as the basis for the session space. Thus, features of a session are modeled and captured in terms of features of its building segments. This conceptualization is analogous to the definition of basis for a vector space, i.e. "a set of linearly independent vectors that construct the vector space." Therefore, the FM model allows analyzing sessions by analyzing features of their corresponding segments.

For the remainder of this section, we explain, analyze, and formalize the FM model. First, we define our terminology. Next, basics of the FM model are explained: the features captured from user interactions, and the main data structure used to present these features. Subsequently, we discuss how to extract the session FM model and the cluster FM model separately. Finally, we analyze complexity and completeness of the model and formalize its uniqueness.

4.1.1 Terminology

Website: A website can be modeled as a set of static or dynamic web pages.

Concept Space (Concept): Each web-site, depending on its application, provides information about one or more concepts. For example, *amazon.com*^{TM} includes concepts such as *Books*, *Music*, *Video*, etc. The web pages within a website can be categorized based on the concept(s) to which they belong. A *concept space*, or simply *concept*, in a website is defined as the set of web pages that contain information about a certain concept. Note that the contents of a web page may address more than one concept, so concept spaces of a website are not necessarily disjoint sets (determination of the concept spaces for a website can be done manually, automatically, or in a hybrid automatic/manual fashion. For example, a possible hybrid approach is to use an automatic content-analysis technique, such as methods commonly employed by search engines, to categorize and classify the pages into different concepts based on their contents. Then, if required, categorization can be fine-tuned based on the application specifications).

Path: A *path* P in a website is a finite or infinite sequence of pages:

$$x_1 \to x_2 \to \cdots \to x_i \to \cdots \to x_s$$

where x_i is a page belonging to the website. Pages visited in a path are not necessarily distinct.

Path Feature (Feature): Any spatial or temporal attribute of a path is termed a *path feature* or simply *feature.* The number of times a page has been accessed, time spent on viewing a page, and spatial position of a page in the path are examples of features.

Session: The path traversed by a user while navigating a concept space is considered a *session.* Whenever a navigation leaves a concept space (by entering a page that is not a member of the current concept), the session is considered terminated. Since each page may belong to more than one concept, several *sessions* from different concepts may be embedded in a single *path.* Also, several sessions from the same concept may happen along a path, while a user leaves and then re-enters the concept. For analysis, we compare sessions from the same concept space with each other. Distinction between the "session" and the "path" notions makes the comparison more efficacious. To identify user behavior, we can analyze all the sessions embedded in his/her navigation path, or prioritize the concepts and perform the analysis on the sessions belonging to the higher priority concept(s). Moreover, among the sessions belonging to the same concept space, we can restrict our analysis to the longer session(s), to decrease complexity of the analysis based on the application specifications. In any case, the result of the analysis on different sessions of the same path can be integrated to provide the final result. For example, in a recommendation system, the recommendation can be generated based on various user preferences detected by analyzing different sessions of the user's navigation path. Thus, we henceforth assume that all sessions belong to the same concept. A similar analysis can be applied to sessions in any concept space.

Session Space: The set of all possible sessions in a concept space is termed *session space.*

Path Segment (Segment): A *path segment*, or simply *segment*, E is an n-tuple of pages: $(x_1, x_2, ..., x_i, ..., x_n)$. We define n to be the *order* of the segment E ($n \geq 1$). Note that there is a one-to-one correspondence between tuples and sequences of pages; i.e. $(x_1, x_2, ..., x_i, ..., x_n) \equiv x_1 \to x_2 \to \cdots \to x_i \to \cdots \to x_n$. We use tuple representation because it simplifies our discussion. Any subsequence of pages in a path can be considered a segment of the path. For example, the path $x_1 \to x_3 \to x_2 \to x_5 \to x_2$ contains several segments such as a first order segment (x_1), a second order segment (x_3, x_2), and a fourth order segment (x_3, x_2, x_5, x_2). We exploit the notion of segment as the building block of sessions to model their features.

Universal Set of Segments $\varepsilon_C^{(n)}$: A universal set of order-n segments is the set of all possible n-tuple segments in the concept space C. Henceforth, since we focus on analysis within a single concept, so we drop the subscript C from the notation.

Cluster: A *cluster* is defined as a set of similar sessions. The similarity is measured quantitatively based on an appropriate similarity measure (see Section 4.2 for discussion of similarity measures).

4.1.2 Basics

Features We characterize sessions through the following features:

- *Hit (H):* A hit is a spatial feature that reflects which pages are visited during a session. The FM model captures H by recording the number of times each *segment* is encountered in a traversal of the session. The reader may consider H as a generalization of the conventional "hit-count" notion. Hit-count counts number of hits per *page*, which is a segment of order one.

- *Sequence (S):* A sequence is an approximation for the relative location of pages traversed in a session. As compared to H, it is a spatial feature that reflects the location of visits instead of the frequency of visits. With the FM model, S is captured by recording the relative location of each segment in the sequence of segments that composes the session. If a segment has been repeatedly visited in a session, S is approximated by aggregating the relative positions of all occurrences. Thus, S does not capture the exact sequence of segments. Exact sequences can be captured through higher orders of H.

- *View Time (T):* View time captures the time spent on each segment while traversing a session. As opposed to H and S, T is a temporal feature.

Features of each session are captured in terms of features of the segments within the session. We may apply various orders of universal sets as a basis to capture different features. In our example, we have used $\varepsilon^{(1)}$ for T, and $\varepsilon^{(2)}$ for H and S, unless otherwise stated. Therefore, we extract the feature T for single-page segments, x_i, and features H and S for ordered page-pair segments (x_i, x_j). In Section 4.1.5, we will explain how using higher-order bases results in a more complete characterization of the session by the FM model at the expense of higher complexity.

The FM model is an open model. It is capable of capturing any other meaningful session features in addition to those mentioned above. The same data structure can be employed to capture new features. This is another option by which completeness of the FM model can be enhanced. However, our experiments demonstrate that the combination of our proposed features is comprehensive enough to detect similarities and dissimilarities among sessions appropriately (see Section 5.2.2).

Data Structure Suppose $\varepsilon^{(n)}$ is the basis to capture a feature F for session U. We deploy an n-dimensional *feature matrix*, $M^F_{r^n}$, to record the F feature values for all order-n segments of U. The n-dimensional matrix M_{r^n} is a generalization of the two-dimensional square matrix M_{r*r}. Each dimension of M_{r^n} has r rows, where r is the cardinality of the concept space. For example, $M_{4\times4\times4}$ is a cube with four rows in each of its three dimensions, and is a feature matrix for a four-page concept space with ε^3 as the basis. The dimensions of the matrix are assumed to be in a predefined order. The value of F for each order-n segment $(x_\alpha, x_\beta, ..., x_\omega)$ is recorded in element $a_{\alpha\beta...\omega}$ of $M^F_{r^n}$. To simplify the understanding of this structure, the reader may assume that rows in all dimensions of the matrix are indexed by a unique order of the concept-space pages; then the feature value for the order-n segment $(x_\alpha, x_\beta, ..., x_\omega)$ is located at the intersection of row x_α on the first dimension, row x_β on the second dimension, ... , and row x_ω on the nth dimension of the feature matrix. Note that M_{r^n} covers all order-n segment members of $\varepsilon^{(n)}$. for instance, in a 100-page concept space with $\varepsilon^{(2)}$ as the basis, M_{100^2} has 10,000 elements. On the other hand, the number of segments existing on a session usually is in the order of tens. Therefore, M_{r^n} is usually a sparse matrix. The elements for which there is no corresponding segment in the session are set to zero.

To map a session to its equivalent FM model, the appropriate feature matrices are extracted for features of the session. The entire set of feature matrices generated for a session constitutes its FM model:

$$U^{fm} = \left\{ M^{F_1}_{r^{n_1}}, M^{F_2}_{r^{n_2}}, ..., M^{F_m}_{r^{n_m}} \right\}.$$

If $n = \max(n_1, n_2, ..., n_m)$ then U^{fm} is an order-n FM model.

In subsequent sections, we explain how values of different features are derived for each segment from the original session, and how they are aggregated to construct the cluster model.

4.1.3 Session Model

In the previous section, we defined the features that characterize a session and the data structure that records them, but we did not explain how values of different features are extracted from a session to form the feature matrices of its FM model. Recall that we record features of a session in terms of features of its segments (if a user session is shorter than a single segment, we can interpret the situation as a lack of information about the user and start analyzing the session when it is at least one segment in length. Alternatively, we can analyze short sessions with low-order FM models). Thus, it suffices if we explain how to extract various features for a sample segment E:

- For a Hit (H), we count the number of times E has occurred in the session ($H \geq 0$). Segments may partially overlap. As long as there is at least one non-overlapping page in two segments, the segments are assumed to be distinct. For example, the session $x_1 \rightarrow x_2 \rightarrow x_2 \rightarrow x_2 \rightarrow x_1$, has a total of four order-two segments, including one occurrence of (x_1, x_2), two occurrences of (x_2, x_2), and one occurrence of (x_2, x_1).

- For sequence (S), we find the relative positions of every occurrence of E and record their arithmetic mean as S for E ($S > 0$). To find the relative positions of segments, we number them sequentially in order of appearance in the session. For example, in the session $x_1 \rightarrow^1 x_2 \rightarrow^2 x_2 \rightarrow^3 x_2 \rightarrow^4 x_1$, the S values for the segments (x_1, x_2), (x_2, x_2), and (x_2, x_1) are 1, 2.5 $(= (2 + 3)/2)$, and 4, respectively.

- For View Time (T), we add up the time spent on each occurrence of E in the session ($T \geq 0$).

Example 1. This example illustrates how to extract the FM model of a sample session. Assume session U is captured in concept space $Z = \{x_1, x_2, x_3, x_4, x_5\}$ as follows:

$$x_1 \rightarrow x_3 \rightarrow x_2 \rightarrow x_2 \rightarrow x_2 \rightarrow x_1 \rightarrow x_3 \rightarrow x_5 \rightarrow x_3.$$

Suppose that the user has spent 20 seconds on each page while navigating the website. To simplify, we are assuming $\varepsilon^{(1)}$ as the basis of T, and $\varepsilon^{(2)}$ as the basis for both H and S:

$$\varepsilon^{(1)} = \{(x_1), (x_2), (x_3), (x_4), (x_5)\},$$

$$\varepsilon^{(2)} = \left\{ \begin{array}{ccccc} (x_1, x_1), & (x_1, x_2), & (x_1, x_3), & (x_1, x_4), & (x_1, x_5), \\ (x_2, x_1), & (x_2, x_2), & (x_2, x_3), & (x_2, x_4), & (x_2, x_5), \\ (x_3, x_1), & (x_3, x_2), & (x_3, x_3), & (x_3, x_4), & (x_3, x_5), \\ (x_4, x_1), & (x_4, x_2), & (x_4, x_3), & (x_4, x_4), & (x_4, x_5), \\ (x_5, x_1), & (x_5, x_2), & (x_5, x_3), & (x_5, x_4), & (x_5, x_5) \end{array} \right\}.$$

To find the relative positions of the segments within the session, first we number them

$$x_1 \rightarrow^1 x_3 \rightarrow^2 x_2 \rightarrow^3 x_2 \rightarrow^4 x_2 \rightarrow^5 x_1 \rightarrow^6 x_3 \rightarrow^7 x_5 \rightarrow^8 x_3.$$

For the sample sequence (x_1, x_3), the H and S values are computed as $H = 1 + 1 = 2$ and $S = (1 + 6)/2 = 3.5$. Also, the value of T for the sample segment (x_1) is calculated as $T = 20 + 20 = 40$.

$U^{fm} = \left\{M^H_{5^2}, M^S_{5^2}, M^T_5\right\}$ is the final FM model extracted for U:

$$M^H_{5^2} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad M^S_{5^2} = \begin{bmatrix} 0 & 0 & 3.5 & 0 & 0 \\ 5 & 3.5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \end{bmatrix}, \quad M^T_5 = \begin{bmatrix} 40 & 60 & 60 & 0 & 20 \end{bmatrix}.$$

4.1.4 Cluster Model

With *clustering*, user sessions are grouped into a set of clusters based on similarity of their features. To cluster sessions, since the FM model is a distance-based model, we need a similarity measure to quantify the similarity between sessions, and a clustering algorithm to construct the clusters. Moreover, we need a scalable model for the cluster. Popular websites are visited by a huge number of users. In such a scale, we may employ any similarity measure and clustering algorithm to group the sessions (or, better to say, session models) into clusters, but merely grouping the sessions is not sufficient. If a cluster is naively modeled as a set of session models, any analysis on a cluster will be dependent on the number of sessions in the cluster, which is not a scalable solution. Particularly, for real time classification of sessions using pre-generated clusters, the cluster model must be a "condensed" model so that the time complexity of the classification is independent of the number of cluster members.

In this section, we describe our cluster model. Subsequently, in Section 4.2, we introduce several applicable similarity measures for the purpose of clustering, and finally, in Section 4.3, we propose a variation on conventional clustering algorithms to make them real time adaptable to varying behaviors.

With our approach to modeling a cluster, we aggregate feature values of all clustered sessions into corresponding feature values of a virtual session, called a *cluster centroid*. The cluster centroid is considered to be a representative of all the sessions in the cluster, or equally, as a model of the cluster. Consequently, the complexity of any analysis on a cluster will become independent of the cluster cardinality.

Suppose we have mapped all the sessions belonging to a cluster into their equivalent session models. In order to aggregate the features of the sessions into the corresponding features of the cluster model, it is sufficient to aggregate features for each basis segment. Assume that we denote the value of a feature F for any segment E in the basis by $F(E)$. We apply a simple aggregation function, namely arithmetic averaging, to the $F(E)$ values in all sessions of a cluster to find the aggregated value of $F(E)$ for the cluster model. Thus, if M^F is the feature matrix for feature F of the cluster model, and M_i^F is the feature matrix for feature F of the i-th session in the cluster, each element of M^F is computed by aggregating corresponding elements of all M_i^F matrices. This procedure is repeated for every feature of the FM model. The final result of the aggregation is a set of aggregated feature matrices that constitute the FM model of the cluster:

$$C^{fm} = \left\{ M^{F_1}, M^{F_2}, ..., M^{F_n} \right\}.$$

Therefore, the FM model can uniquely model both sessions and clusters.

As mentioned before, the aggregation function we use for all features is the simple arithmetic-averaging function. In matrix notation, the aggregated feature matrix for every feature F of the cluster model C^{fm} is computed as follows:

$$M^F = \frac{1}{N} \sum_{i=1}^{N} M_i^F$$

where N is the cardinality of the cluster C. The same aggregation function can be applied incrementally, when the cluster model has already been created and we want to update it as soon as a new session, U_j, joins the cluster:

$$M^F \leftarrow \frac{1}{N+1}\left(N \times M^F + M_j^F\right).$$

This property is termed *dynamic clustering*. In Section 4.3, we leverage on this property to modify the conventional clustering algorithms to become real time and adaptive.

Example 2. This example illustrates how to extract the FM model of a simple cluster. Assume U_1 and U_2 are the only two members of cluster C, and M_1^F and M_2^F are their feature matrices for feature F. The following equations are self-explanatory:

$$M_1^F = \begin{bmatrix} 0 & 2 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad , \quad M_2^F = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$M^F = \frac{1}{2}\left(M_1^F + M_2^F\right) = \begin{bmatrix} 0.5 & 1.5 & 0 \\ 0.5 & 0 & 0.5 \\ 1 & 0 & 0.5 \end{bmatrix}.$$

4.1.5 Analysis of the Model

Cluster-based WUM involves three categories of tasks: constructing clusters of sessions (clustering), comparing sessions with clusters, and integrating sessions into clusters. Regardless of the model employed for analysis and the algorithm used for clustering, the complexity of constructing the clusters is dependent on N, the number of sessions to be clustered. This is true simply because during clustering each session should be analyzed at least once to detect how it relates to other sessions. The FM cluster model is defined so that it reduces the time complexity of the other two tasks. If the complexity of comparing a session with a cluster and integrating it into the cluster is independent of the cluster cardinality, user classification and cluster updating can be fulfilled in real time.

For certain types of data (such as web-usage data), to achieve lower space and time complexity with the data model, one needs to sacrifice *completeness*. If the cluster model is merely the set of member sessions stored in their complete form, although the model is *complete* in representing the member sessions, it does not scale. On the other hand, if we aggregate member sessions to construct the cluster model, the model will lose its capability to represent its members with perfect accuracy. The more extensive aggregation is applied, the less complete the cluster model. The FM model is flexible in balancing this trade-off based on the specific application requirements.

Table 1: Parameters

Parameter	Definition
F_i	i-th feature captured in FM
n_i	Order of the basis used to capture F_i
m	Number of features captured in FM
n	$\max(n_1, n_2, \ldots, n_m)$
r	Cardinality of the concept space
L	Average length of sessions
M	Average cardinality of clusters

FM Complexity versus the Vector and Markov Models: Let $FM^{(n)}$ be an FM model of the order n (see Table 1 for the definitions of terms). In the worst case, $FM^{(n)}$ consists of m n-dimensional matrices M_{r^n}, one for each of the model features. Thus, the *space* cost of $FM^{(n)}$ is $O\left(mr^n\right)$. *Time* complexity for user classification is $O\left(mL\right)$ and for updating a cluster by assigning a new session to the cluster is $O\left(mr^n\right)$. Therefore, the space and time complexity of $FM^{(n)}$ model are both independent of M.

From $O\left(mr^n\right)$, complexity increases exponentially with n, which is the order of the FM model. Property 1 illustrates that as the order n increases, the FM model becomes more *complete* in describing its corresponding session or cluster; hence, it allows better prediction about user/cluster interests. Thus, added complexity is the price for a more accurate model. An appropriate order should be selected based on the accuracy requirements of the specific application.

Property 1. If $p_1 > p_2$ then $FM^{(p_1)}$ is more complete than $FM^{(p_2)}$.

Without loss of generality, we base our discussion on the H feature. The same deduction applies to any other feature. We prove by induction. Suppose $FM^{(1)}$ and $FM^{(2)}$ are used to model the session U into $U^{fm^{(1)}}$ and $U^{fm^{(2)}}$, respectively. Assume we want to reconstruct U based on the data captured in its model:

$$U: \;\; x_1 \to x_2 \to \ldots \to x_i \to x_{i+1} \to \ldots \to x_L.$$

If the first i pages of U are already known, the H matrix of $U^{fm^{(1)}}$ suggests $L-i$ choices for x_{i+1}, while using $U^{fm^{(2)}}$ the number of choices is limited to the sum of elements located in single row of the H matrix, which is indexed by the page x_i. The latter is always less than or equal to $L-i$. This is reasonable because the H matrix in $U^{fm^{(2)}}$ not only records the number of hits on pages, but also implicitly captures information about their *order* in the session, at least to the following page. Since during reconstruction of the session, from the first page to the last, the same deduction holds on the selection of the following pages, the total number of sessions suggested by $U^{fm^{(2)}}$ will always be less than or equal to those suggested by $U^{fm^{(1)}}$. Therefore, by recording information about segments instead of pages, we have achieved a more complete model, though with higher complexity. Higher-order FM models will inductively further restrict the number of choices for each next page, simply because visiting the $(p+1)$ order segment $(x_{i-p+1}, x_{i-p+2}, ..., x_i, x_{i+1})$ is a special case of visiting the p order segment $(x_{i-p+2}, ..., x_i, x_{i+1})$. In the extreme case, when $p = L$, $U^{fm^{(L)}}$uniquely identifies U and the model is absolutely complete.

The other crucial parameter in $O\left(mr^n\right)$ is m, the number of features captured by the FM model. Features are attributes of the sessions, used as the basis for comparison. The relative importance of these attributes in comparing the sessions is application-dependent. The FM model is an open model in the sense that its structure allows incorporating new features as the need arises for different applications. Performing comparisons based on more features results in more accurate clustering, though again we increase the complexity.

Now let us compare the performance of FM with two other conventional models, namely the vector model and the Markov model, mentioned in Section 2.2. The vector model can be considered as one special case of the FM model. As used in Yan et al. (1996), the vector model is equivalent to an $FM^{(1)}$ model with H as the only captured feature. Thus, the vector model scales as $O\left(r\right)$, but as discussed above, since it is an order-1 FM model, it performs poorly in capturing information about sessions. Our experiments illustrate that an $FM^{(2)}$ model with S and H as its features outperforms the vector model in accuracy (see Section 5.2.4). The other model, typically employed in dependency-based approaches, is the "Markov" model. Although whether or not web navigation is a Markovian behavior has been the subject of much controversy (Huberman et al. 1997), the Markov model has demonstrated acceptable performance in the context of WUM (Cadez et al. 2000). The transition matrix of an order-n Markov model is extractable from the H feature matrix of an $FM^{(n+1)}$ model. Thus, the FM model at least captures the same amount of information as does an equivalent Markov model. They also benefit from the same time complexity of $O\left(L\right)$ for dynamic user classification. However, the Markov model cannot be updated in real time because the complexity of updating a cluster is dependent on the cardinality of the cluster. Moreover, the Markov model is not an *open* model, as described for FM, because it is defined to capture order and hit.

Formalization: In this section, we formally prove uniqueness of the FM session and cluster models.

Theorem 1. *Two identical sessions have identical FM models*, i.e.

$$U_1 = U_2 \;\Rightarrow\; U_1^{fm^{(n)}} = U_2^{fm^{(n)}} \;.$$

Based on the session model definition

$$U_1^{fm^{(n)}} = \left\{ M_1^{F_i} \mid i = 1..m \right\} \quad , \quad U_2^{fm^{(n)}} = \left\{ M_2^{F_i} \mid i = 1..m \right\},$$

but if the two sessions U_1 and U_2 are identical, their feature matrices are correspondingly equal. Hence:

$$U_1 = U_2 \;\Rightarrow\; \forall i : 1..m \;\; M_1^{F_i} = M_2^{F_i} \;\Rightarrow\; U_1^{fm^{(n)}} = U_2^{fm^{(n)}}.$$

Theorem 2. *Two identical clusters have identical FM models*, i.e.

$$C_1 = C_2 \;\Rightarrow\; C_1^{fm^{(n)}} = C_2^{fm^{(n)}}.$$

Based on the cluster model definition

$$C_1^{fm^{(n)}} = \left\{ M_1^{F_i} \mid i = 1..m \right\} \quad , \quad C_2^{fm^{(n)}} = \left\{ M_2^{F_i} \mid i = 1..m \right\},$$

but if the two clusters C_1 and C_2 are identical, for each session U_{1j} in C_1 there is an identical session U_{2k} in C_2, and vice-versa:

$$\left. \begin{array}{c} C_1 = \{U_{1j} \mid j = 1..M_{C_1}\} \\ C_2 = \{U_{2k} \mid k = 1..M_{C_2}\} \\ C_1 = C_2 \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} \forall\, U_{1j} \in C_1 \;\Rightarrow\; \exists\, U_{2k} \in C_2 \\ \forall\, U_{2k} \in C_2 \;\Rightarrow\; \exists\, U_{1j} \in C_1 \\ M_{C_1} = M_{C_2} \end{array} \right\}$$

hence:

$$\left. \begin{array}{c} M_1^{F_i} = \sum_{j=1}^{M_{C_1}} M_{1j}^{F_i} \\ M_2^{F_i} = \sum_{k=1}^{M_{C_2}} M_{2k}^{F_i} \end{array} \right\} \Rightarrow\; M_1^{F_i} = M_2^{F_i} \;\Rightarrow\; C_1^{fm^{(n)}} = C_2^{fm^{(n)}}$$

4.2 Similarity Measures

A *similarity measure* is a metric that quantifies the notion of "similarity." To capture behaviors of the website users, user sessions are to be grouped into clusters, such that each cluster is composed of "similar" sessions. Similarity is an application-dependent concept and in a distance-based model such as FM, a domain expert should encode a specific definition of similarity into a pseudo-distance metric that allows the evaluation of the similarity among the modeled objects. With the FM model, these distance metrics, termed *similarity measures*, are used to impose order of similarity upon user sessions. Sorting user sessions based on the similarity is the basis for clustering the users. Some similarity measures are defined to be indicators of dissimilarity instead of similarity. For the purpose of clustering, both approaches are applicable.

In Shahabi et al. (1997), we introduce a similarity measure for session analysis that does not satisfy an important precondition: the basis segments used to measure the similarity among sessions must be orthogonal. Here, we define three other similarity measures based on the FM model. Two of these measures, VA and PED, are frequently used in the context of information retrieval. $PPED$ is a new similarity measure particularly defined to alleviate the overestimation problem attributed to PED in the context of WUM (Yan et al. 1996). All these measures satisfy the mentioned precondition. Before defining the functions of these similarity measures, let us explain how they interpret the FM model.

With all the discussed similarity measures, each feature matrix of FM is considered to be a uni-dimensional matrix. To illustrate, assume all rows of an n-dimensional feature matrix are concatenated in a predetermined order of dimensions and rows. The result will be a uni-dimensional ordered list of feature values. This ordered list is considered to be a vector of feature values in $R^{(r^n)}$, where r is the cardinality of the concept space. Now suppose we want to measure the quantitative dissimilarity between the two sessions U_1^{fm} and U_2^{fm}, assuming that the certain similarity measure used is an indicator of dissimilarity (an analogous procedure applies when the similarity measure expresses similarity instead of dissimilarity). Each session model is composed of a series of feature vectors, one for each feature captured by the FM model. For each feature F_i, the similarity measure is applied to the two F_i feature vectors of U_1^{fm} and U_2^{fm} to compute their dissimilarity, D^{F_i}. Since the dissimilarity between U_1^{fm} and U_2^{fm} must be based on all the FM features, the total dissimilarity is computed as the weighted average of dissimilarities for all features:

$$D^F = \sum_{i=1}^{m} w_i \times D^{F_i} \qquad \left(\sum_{i=1}^{m} w_i = 1\right), \tag{1}$$

where m is the number of features in the FM model. D^F can be applied in both hard and soft assignment of sessions to clusters. The weight factor w_i is application-dependent and is determined based on the relative importance and efficacy of features as similarity indicators. In Section 5.2.2, we report on the results of our experiments in finding the composed set of weight factors for the H and S features.

In the following sections, we introduce our alternative similarity measures. They are classified based on the type of distance they use to estimate similarity between feature vectors. These similarity measures are applicable within any clustering algorithm. Throughout this discussion, assume $\overrightarrow{A}$ and $\overrightarrow{B}$ are feature vectors equivalent to n-dimensional feature matrices M_1^F and M_2^F, and a_i and b_i are their ith elements, respectively. Vectors are assumed to have $N = r^n$ elements, where r is the cardinality of the concept space.

4.2.1 Angular Distance

Vector Angle (VA) VA measures the similarity between feature vectors based on their angular distance (see Figure 1):

$$VA\left(\overrightarrow{A}, \overrightarrow{B}\right) = \cos\varphi = \frac{\overrightarrow{A}.\overrightarrow{B}}{\left|\overrightarrow{A}\right|\left|\overrightarrow{B}\right|} = \frac{\sum_{i=1}^{N} a_i b_i}{\left(\sum_{i=1}^{N} a_i^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{N} b_i^2\right)^{\frac{1}{2}}} \tag{2}$$

where $\varphi = \left\langle \overrightarrow{A}, \overrightarrow{B} \right\rangle$ and $VA \in [0, 1]$ $(a_i, b_i \geq 0)$.

VA expresses the *similarity*; the greater the $VA\left(\overrightarrow{A}, \overrightarrow{B}\right)$, the more *similar* $\overrightarrow{A}$ and $\overrightarrow{B}$. To obtain intuition about VA, recall the structure of a feature vector. A feature vector is a list of values of a certain feature F, each value relevant to a segment in the basis universal set. A combination of these values conveys the status of the corresponding session in terms of the feature F. Since the direction of the feature vector is determined by the feature values of which it is composed, if VA finds two feature vectors close in direction, they must include analogous values. Consequently, the corresponding sessions should be "similar."

However, if we look into VA functionality with more details, we notice that VA cannot

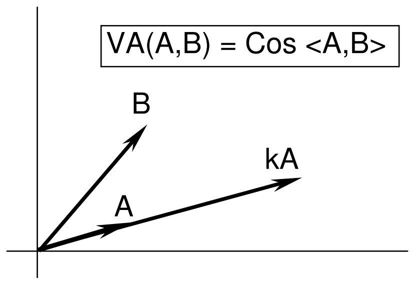


Figure 1: *Vector Angle (VA)* Similarity Measure

differentiate between a vector and its scaled variations:

$$\forall k \in R^{+} \quad VA\left(\overrightarrow{A}, k\overrightarrow{A}\right) = \cos 0 = 1.$$

This characteristic might be undesirable in comparing feature vectors of certain features such as S. For these feature vectors, the mere fact that $\overrightarrow{V_2} = k\overrightarrow{V_1}$ does not necessarily convey similarity between corresponding sessions of $\overrightarrow{V_2}$ and $\overrightarrow{V_1}$, whereas VA finds $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$ identical ($VA\left(\overrightarrow{V_1}, \overrightarrow{V_2}\right) = VA\left(\overrightarrow{V_1}, k\overrightarrow{V_1}\right) = 1$). However, for some other path features such as H, this characteristic may be meaningful. For example, suppose VA is used to compare the H feature vectors of these two sessions:

$$U_1 : \quad x_1 \rightarrow x_2 \rightarrow x_1,$$

$$U_2 : \quad x_1 \rightarrow x_2 \rightarrow x_1 \rightarrow x_2 \rightarrow x_1.$$

Users navigating these paths may have had similar intentions because they have repeatedly traversed a certain set of segments. VA detects this similarity by comparing H vectors of U_1 and U_2:

$$M_1^H = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \Rightarrow \quad V_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix},$$

$$M_2^H = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} \quad \Rightarrow \quad V_2 = \begin{bmatrix} 0 & 2 & 2 & 0 \end{bmatrix},$$

$$(V_2 = 2V_1) \;\Rightarrow\; VA\left(V_1, V_2\right) = 1.$$

$FM^{(n)}$, in the worst case, is composed of m n-dimensional matrices M_{r^n}, one for each of the model features (refer to Table 1 for definitions of terms) . If VA is used to compare two sessions, according to (2) time complexity for user classification is $O\left(mr^n\right)$.

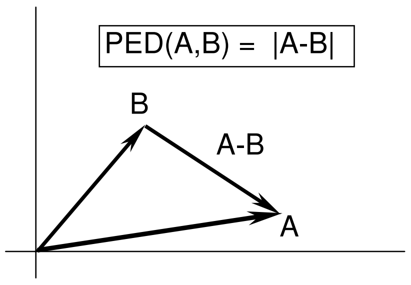


Figure 2: *Pure Euclidean Distance (PED)* Similarity Measure

4.2.2 Euclidean Distance

In this section, we will discuss similarity measures based on Euclidean distance between two feature vectors. These similarity measures particularly quantify the *dissimilarity* between feature vectors; the greater their value, the more *dissimilar* the compared vectors.

Pure Euclidean Distance (PED): PED simply computes the Euclidean distance between two feature vectors (see Figure 2):

$$PED\left(\overrightarrow{A},\overrightarrow{B}\right) = \left|\overrightarrow{A}-\overrightarrow{B}\right| = \left(\sum_{i=1}^{N}(a_i-b_i)^2\right)^{\frac{1}{2}} \tag{3}$$

where $PED \in [0,\infty)$.

As compared to VA, PED can differentiate between a vector and its scaled variations:

$$PED\left(\overrightarrow{A},k\overrightarrow{A}\right) = \left|(k-1)\overrightarrow{A}\right| \neq 0 \qquad \left(if \quad k \neq 1 \ and \ \overrightarrow{A} \neq \overrightarrow{0}\right).$$

However, if PED is used to compare a session with its cluster, the dissimilarity may be overestimated. To illustrate, suppose a user navigates the session U that belongs to cluster C. It is not necessarily the case that the user traverses every segment as captured by C^{fm}. In fact, in most cases the user navigates a path similar to only a subset of the access pattern represented by C^{fm} and not the entire pattern. In evaluating the similarity between U^{fm} and C^{fm}, we should avoid comparing them on that part of the access pattern not covered by U, or else their dissimilarity will be overestimated. Overestimation of dissimilarity occasionally results in failure to classify a session to the most appropriate cluster. Example 3 illustrates this problem.

Example 3. This example demonstrates how the overestimation problem may cause PED to mistarget a session. Suppose clusters C and C' are represented by the following cluster centroids:

$$C: \quad x_2 \rightarrow x_3 \rightarrow x_2 \rightarrow x_3 \rightarrow x_2 \rightarrow x_1$$

$$C': \quad x_3 \rightarrow x_1 \rightarrow x_3$$

and session U is captured as follows:

$$U: \quad x_2 \rightarrow x_3 \rightarrow x_2.$$

The objective is to select the cluster that is more similar to U. Assume that the similarity criterion is based only on the S feature. The S feature vectors of C, C', and U are as follows:

$$M_C^S = \begin{bmatrix} 0 & 0 & 0 \\ 5 & 0 & 2 \\ 0 & 3 & 0 \end{bmatrix} \quad \Rightarrow \quad V_C^S = \begin{bmatrix} 0 & 0 & 0 & 5 & 0 & 2 & 0 & 3 & 0 \end{bmatrix},$$

$$M_{C'}^S = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \Rightarrow \quad V_{C'}^S = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$M_U^S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 2 & 0 \end{bmatrix} \quad \Rightarrow \quad V_U^S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 \end{bmatrix}.$$

As observed from the sequences of pages, U itself is a sub-path of C while it does not have any segment in common with the C' cluster. However, PED wrongly finds U to be more similar to C' rather than to C:

$$PED\left(V_U^S, V_C^S\right) \simeq 5.20 \quad > \quad PED\left(V_U^S, V_{C'}^S\right) \simeq 3.16$$

In next section, we provide a solution for the overestimation problem. Finally, note that according to (3), PED has the same time complexity as does that of VA, namely $O\left(mr^n\right)$ (refer to Table 1 for definitions of terms).

Projected PED (PPED) $PPED$ is a variant of PED that alleviates the overestimation problem. Assume $\overrightarrow{A}$ and $\overrightarrow{B}$ are two feature vectors of the same type belonging to a session and a cluster model, respectively. Each vector is composed of N components. To estimate the dissimilarity between $\overrightarrow{A}$ and $\overrightarrow{B}$, $PPED$ computes the pure Euclidean distance between $\overrightarrow{A}$ and the projection of $\overrightarrow{B}$ on those coordinate planes at which $\overrightarrow{A}$ has non-zero components:

$$PPED\left(\overrightarrow{A},\overrightarrow{B}\right) = \left(\sum_{i=1,a_i\neq 0}^{N} (a_i - b_i)^2\right)^{\frac{1}{2}}, \tag{4}$$

where $PPED \in [0,\infty)$. Note that $PPED$ is not commutative.

Non-zero components of $\overrightarrow{A}$ belong to those segments that exist in the session. Zero values, on the other hand, are related to the remainder of the segments in the basis universal set. By contrasting $\overrightarrow{A}$ with the projected $\overrightarrow{B}$, we compare the session and the cluster based on just the segments that exist in the session and not on the entire basis. Thus, the part of the cluster not covered in the session is excluded from the comparison to avoid overestimation. The impact of overestimation correction of $PPED$ is demonstrated in Example 4.

Example 4. Consider the same scenario described in Example 3. As illustrated below, $PPED$ assigns U to the correct cluster, C:

$$PPED\left(V_U^S, V_C^S\right) = \sqrt{(1-2)^2 + (2-3)^2} \simeq 1.41$$

$$PPED\left(V_U^S, V_{C'}^S\right) = \sqrt{(1-0)^2 + (2-0)^2} \simeq 1.73$$

$$\Rightarrow\ PPED\left(V_U^S, V_C^S\right) \ <\ PPED\left(V_U^S, V_{C'}^S\right).$$

Since $PPED$ can compare sessions with different lengths, it is an attractive measure for real time clustering where only a portion of a session is available at any given time (see Section 4.3). $PPED$ also helps reduce the time complexity of the similarity measurement. According to (4), the time complexity of $PPED$ improves to $O(mL)$ (refer to Table 1 for definitions of terms). In Section 5.2.3, we report on the superiority of $PPED$ performance as compared to that of PED and VA.

4.3 Dynamic Clustering

As discussed in Section 4.1.5, since the FM model of a cluster is independent of the cluster cardinality, any cluster manipulation with FM has reasonably low complexity. Leveraging this property, we can apply the FM model in real time applications.

```
1. Find the distance/similarity between the session and every cluster
   available in the current cluster set using any reasonable similarity
   measure;
   // All similarity measures discussed in this paper are applicable.
   // These similarity measures are defined based on the data structure
   // of the FM model

2. If there is no cluster closer than TDC to the session {
        create a new cluster and use the FM model of the new session as
        the cluster model;
   } else {
        update the closest cluster to the session by joining the session
        to that cluster;
   }
   // TDC is a threshold value specific to Dynamic Clustering. If the
   // distance between the new session and every existing cluster is more
   // than TDC, then it is reasonable to create a new cluster because a
   // new user behavior has been discovered
```

Figure 3: An Algorithm for *Dynamic Clustering*

One benefit of this property is that FM clusters can be updated dynamically and in real time. Note that in most common cluster representations, the complexity of adding a new session to a cluster is dependent on the cardinality of the cluster. Therefore, in large scale systems, they are not practically capable of updating the clusters dynamically. By exploiting *dynamic clustering*, the WUM system can adapt itself to changes in users' behaviors in real time. New clusters can be generated dynamically and existing clusters adapt themselves to the changes in users' tendencies. Delay-sensitive environments such as stock markets are among those applications for which this property is most advantageous. Figure 3 depicts a simple procedure to perform dynamic clustering when a new session is captured.

Periodic re-clustering is the typical approach in updating the clusters. This approach results in high accuracy, but it cannot be performed in real time. According to our experiments to compare the accuracy of dynamic clustering with that of a periodic re-clustering (see Section 5.2.5), dynamic clustering shows lower accuracy in updating the cluster set. In fact, with dynamic clustering, we are trading accuracy for adaptability. Thus, dynamic clustering should not be used instead of classical clustering algorithms, but a hybrid solution is appreciated. That is, the cluster set should be updated in longer periods through periodic re-clustering to avoid divergence of the cluster set from the trends of real user behavior. Meanwhile, dynamic clustering can be applied in real time to adapt the clusters and the cluster set to short-term behavioral changes. Our experiments show that the dynamic clustering algorithm scales as number of the clusters and order of the FM model grows (see Section 5.2.6).

5. Performance Evaluation

5.1 Tracking Tier

To verify correctness and accuracy of our tracking approach, we compared *hit* per page captured by the remote agent with that recorded in a typical server log. For this purpose, we tracked users access to Digimuse (http://digimuse.usc.edu), the public Interactive Art Museum at the University of Southern California, for 20 successive days. This website consists of 70 web pages and it runs Apache web server version 1.3.12. After collecting the data, we computed the average number of hits per page in a period of a day based on the entries recorded in the server log and those reported by remote agents.

Figure 4 illustrates accuracy of our approach, compared to that of the server log, in capturing page requests. The results show that the remote agent captures 47.66% more page requests in average as compared to the server log. The requests missing from the server log are those requests that are retrieved from the caches in the web browsers or the proxy servers.

5.2 Analysis Tier

We conducted several experiments to: 1) compare the efficacy of the path features in characterizing user sessions, 2) study the accuracy of our similarity measures in detecting the similarity among user sessions, 3) compare the performance of the FM model with that of the traditional Vector model, 4) investigate the accuracy of dynamic clustering, 5) study the scalability of the dynamic clustering algorithm, and 6) investigate performance of the FM model in capturing meaningful clusters in real data. Except for the last set of experiments, which verifies capabilities of our system in handling real data, we preferred to use synthetic data with our experiments so that we could have more control over our input characteristics.

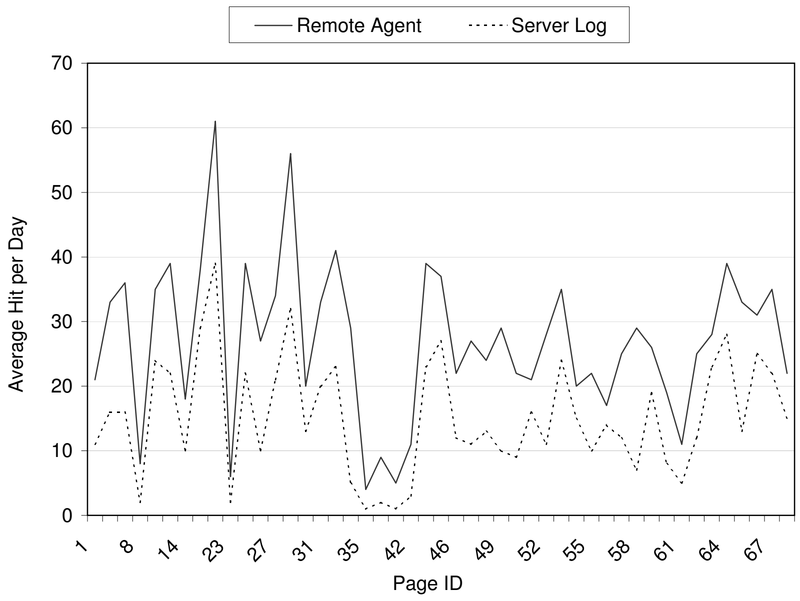


Figure 4: Comparing Accuracy of the Remote Agent with Server Log

5.2.1 Experimental Methodology

To generate N synthetic user sessions, we start by partitioning our user space into k (almost) equal-sized groups. The assumption is that members of the same group are users with similar behavior. Subsequently, we force all users within a group navigate a website almost identically. Here, we use different techniques to introduce noise in these navigations. The objective is to use our FM algorithms to form the clusters as close as possible to the original groups, by just examining the navigations. We use precision and recall metrics to compute the distance between FM clusters and original groups as our measures of success. The website is not real, but simply a randomly generated single-source directed graph (DG).

Our original $k = 18$ groups are formed based on the 18 possible combinations of the following user demographics: sex (male, female), age (young, middle-aged, old), and financial status (poor, middle-class, wealthy). To force similar users to navigate similar paths, a core path was first created for each group as its representative path. Next, each core path was considered the centroid to construct similar sessions around it.

Our website DG consisted of 100 vertices or pages, one of which selected as the source or home page. Each page had exactly 18 edges/links (l_1 to l_{18}) to 18 other distinct pages. The destination pages of the links were selected randomly once during website construction. The core path for a cluster i, C_i, was generated by traversing the link l_i of each page, starting from the home page. The length of a core path was fixed at 15 (15 pages are visited). As a result, we created 18 core paths of length 15, P_1 to P_{18}, all starting from the home-page:

$$P_i : \quad X_{i,1}\,(= X_H) \to X_{i,2} \to ... \to X_{i,j-1} \to X_{i,j} \to ... \to X_{i,15}$$

where $X_{i,j}$ is the destination page of the link l_i in the page $X_{i,j-1}$, and X_H is the home page.

Next, user sessions are constructed around the core paths. There are two knobs v and p to control the similarity of a user session to a core path. v controls the variation in the length of a user session. Specifically, $v = 15 - m$ where m is the minimum length of a session ($m < 15$). Hence, the higher v, the more variations exist in the lengths of the created sessions. p determines how similar a user session is to the core path. Specifically, p is the probability that the user selects an identical link to the core path's link at each and every page. In the extreme case, when $p = 1$, the sessions follow the exact pattern of the core path page-by-page, though they might have different lengths. For example, consider a user session U belonging to cluster C_i with the core path P_i as its representative:

$$U : \quad x_1 \to x_2 \to ... \to x_{j-1} \to x_j \to ... \to x_L \qquad (L \leq 15)$$

where x_j is the page visited at the jth position. Subsequently, x_j will be identical to $X_{i,j}$ of the core path with probability p; otherwise from x_{j-1} we select a wrong link with probability $(1-p)\,/17$ and hence $x_j \neq X_{i,j}$.

In our experiments, we varied v from 2 to 14, and p from 0.8 to 1. Each dataset consists of $200k$ user sessions, where each user is assigned to a cluster randomly. Hence, the size of each cluster is approximately 11,000 users. As mentioned before, in our experiments we used precision and recall to estimate the accuracy (Y-axis in all graphs, in percentage). Whenever these two measures behaved similarly, we combined them via the *harmonic mean (HM)* function, defined as:

$$HM = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \qquad (Precision, Recall, HM \in [0,1])$$

HM assumes a high value only when precision and recall are both high.

Finally, note that for simplicity, T is excluded from the experiments. Thus, analysis is performed based only on spatial features of the sessions. We used order-2 segments to capture both S and H, so, the applied model is $FM^{(2)}$.

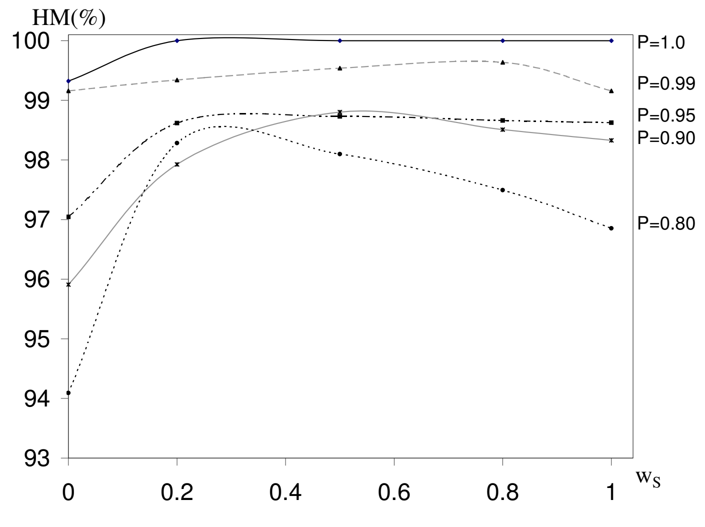


Figure 5: Weight Factors for Path Features

5.2.2 Efficacy of the Path Features

A set of experiments was conducted to study the relative efficacy of the path features H and S in detecting similarities between user sessions. In (1), the weight factor w_i indicates relative importance of the path feature F_i in computing the aggregated similarity measure. The higher weights are assigned to the features that are more effective in capturing similarities. Our experiments were intended to find the set of weight factors w_S (weight factor for S) and w_H that result in the optimum accuracy in capturing the similarities.

With this set of experiments, we used FM to model the sessions, and K-Means with $PPED$ to cluster the sessions. K-Means requires the number of the expected clusters as an input. For real data, the number of clusters should be learned from the dataset by applying an estimation/optimization technique, but for synthetic data it is a priori knowledge. We performed each experiment with a different p value. Figure 5 summarizes the results. In this figure the X-axis is w_S and the Y-axis is HM. Each curve corresponds to a different dataset with a different p value.

As observed in the figure, the accuracy is always above 94%, which indicates that both features (hit and sequence) are equally successful in identifying the spatial similarities, though S demonstrates slight superiority (see the end points). The optimum accuracy is achieved by employing a combination of the similarities detected in hit and sequence. Depending on p,, w_S varies between 0.2 and 0.8. The less distinguishable (smaller p) the dataset is, the

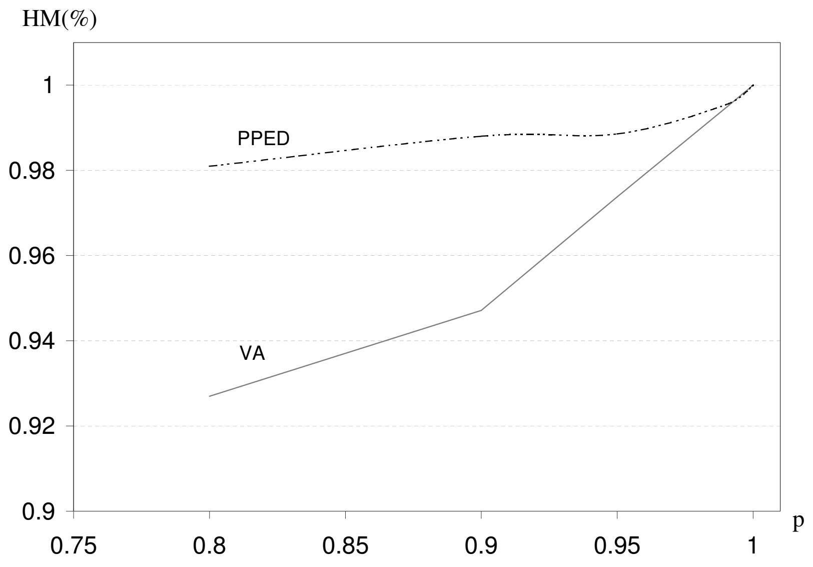


Figure 6: Comparing VA and $PPED$

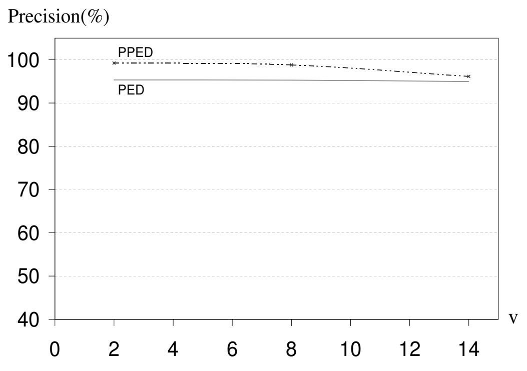
less weight should be assigned to the sequence. That is, when similarity among users of the same group decreases, it is more important to track which pages they visit than where in the session they visit each page. Hence, for real data (that are less distinguishable), setting $w_S \approx 0.2$ may result in the optimum accuracy. In sum, if we assume user behaviors are similar when spatial characteristics of their sessions are similar, using H and S effectively categorizes user behaviors.

5.2.3 Accuracy of the Similarity Measures

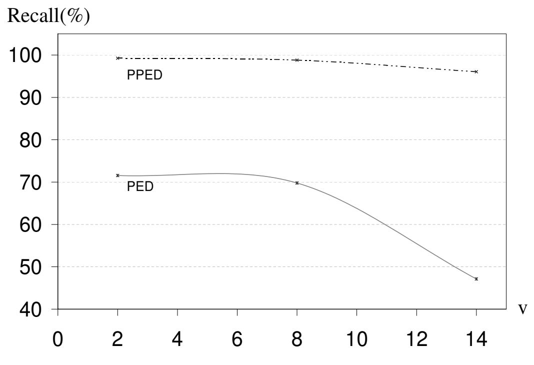
In Section 4.2, we introduced three similarity measures. Here, the accuracy of these similarity measures are compared.

First, we applied VA and $PPED$ to cluster six datasets with different p values: 0.8, 0.9, 0.95, 0.99, 0.9999, and 1. All datasets had $v = 14$. We used FM to model the sessions and K-Means for clustering. Weight factors w_S and w_H were set to 0.5. As observed in Figure 6, $PPED$ outperforms VA. With $p = 1$ both similarity measures performed perfectly, but as datasets become less distinguishable, the margin between their performance increases. Thus, for real data, which assumes low distinguishability, $PPED$ is definitely preferable.

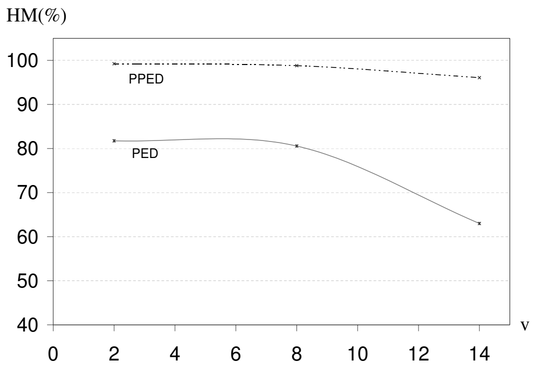
Second, we conducted some experiments to verify the effect of employing $PPED$ to alleviate the overestimation problem with PED. For these experiments, three datasets were



(a)



(b)



(c)

Figure 7: Comparing Performances of PED and $PPED$

used with $p = 0.9$ and various v values: 2, 8, and 14. Accuracy of the clusters generated by applying PED and $PPED$ are contrasted in Figure 7. First, note that $PPED$ outperforms PED in both precision and recall (Figures 7a and 7b). We consider this superiority due to alleviation of the overestimation problem. For $p = 0.9$, accuracy of $PPED$ is at least 96%, though it decreases for higher v values. This is reasonable because higher v implies that more short-length sessions exist in the dataset. The shorter a session is, the less information exists about the user behavior it contains. Thus, shorter sessions are harder to assign to the appropriate clusters. Failure to assign the shorter sessions to their clusters results in lower accuracy. However, PED shows a different behavior. For PED, the decrease in the recall measure at higher v values is more than expected. On the other hand, unlike $PPED$, its precision does not show noticeable variations as v increases. For the remainder of this section, we explain this behavior.

According to the experimental results, when PED is used as the similarity measure

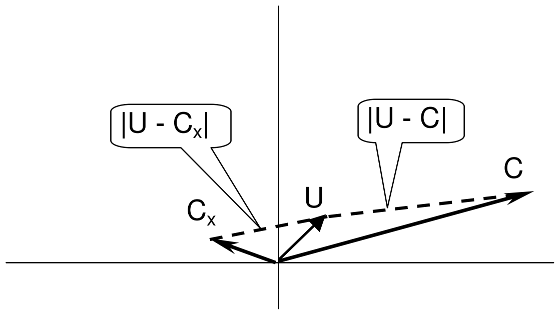


Figure 8: Illustration of the Overestimation Problem

for clustering, sometimes a cluster with a short-length core path attracts several sessions from other clusters, even though they do not have much in common with the cluster. A cluster with a short-length core path represents a group of users about whom we do not have much information. This problem is also explainable when paths are represented by feature vectors. In Figure 8, U and C are the feature vectors for a session and its corresponding cluster, respectively. C_x is the feature vector for a cluster with a short-length core path. As observed from the figure, although C and U are close in direction (so they are "similar"), since C is long and U is short, PED estimates a long distance between them. On the other hand, since C_x is short, it is considered similar to every short-length session regardless of its direction. Thus, PED finds U more similar to C_x than C.

With $PPED$, on the other hand, this problem is avoided by taking the direction of the feature vectors into account. To estimate the similarity, $PPED$ computes Euclidean distance between a session and the projected component of a cluster on the direction of the session. In other words, with $PPED$ similarity between a user and a group is computed based on user characteristics rather than group characteristics. Thus, a user joins the group that has most in common with that user. Therefore, overestimation of the distance between the session and its intended cluster is avoided by disregarding unnecessary group characteristics in distance estimation.

Due to the problem described above, with PED most clusters experience low recall because many of their sessions are accepted by the cluster that is not well characterized. However, since most mis-clustered sessions join a single cluster, the average precision of clustering among 18 clusters is high. Decrease in the recall intensifies with higher v values because short-length sessions are more exposed to mis-clustering. HM combines the precision and recall figures and is a more realistic measure of accuracy for PED and $PPED$ (see Figure

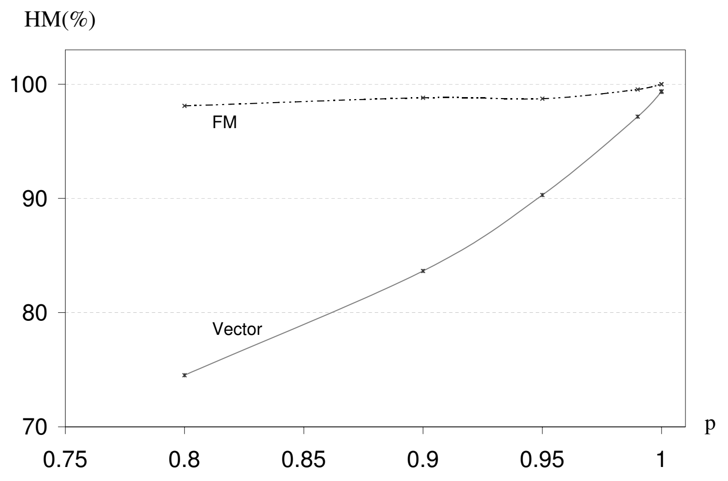


Figure 9: Comparing Performances of the FM Model and the Vector Model

7c).

5.2.4 Performance of the FM Model vs. the Vector Model

We conducted some experiments to compare performances of a sample FM model, namely $FM^{(2)}$ with H and S as its features, with the traditional vector model, which is considered equivalent to $FM^{(1)}$ with H as its only feature.

Results of this study are depicted in Figure 9. As illustrated, performance of the vector model worsens as the user sessions become less distinguishable, while the FM model maintains its accuracy with lower values of p. This superiority is because of: 1) incorporating S into the model, and 2) capturing features based on order-2 segments. Note that even with $p = 1$, the vector model fails to achieve 100% accuracy. That is because of variation of the session lengths. With $p = 1$ all sessions exactly follow the pattern of the core paths, but they might have different lengths. $FM^{(2)}$ is perfect at $p = 1$.

5.2.5 Accuracy of the Dynamic Clustering

In Section 4.3, we introduced *dynamic clustering* as an approach to update cluster models in real time. However, we also mentioned that dynamic clustering trades accuracy for adaptability. We conducted several experiments to study the degradation of the accuracy due to applying the dynamic clustering. For this purpose, we compared dynamic clustering with K-Means.

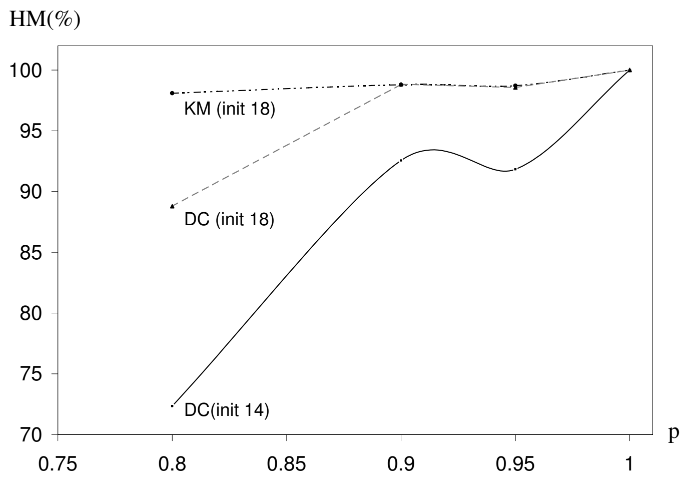


Figure 10: Accuracy of the Dynamic Clustering in Creating (init 14), and Updating (init 18) the Clusters in Real Time

For the experiments, we initiated dynamic clustering in two ways: once we initiated all 18 clusters with the core paths; the other time, 14 clusters were initiated and the remaining 4 clusters were left for dynamic clustering to create. With the former, dynamic clustering is applied to update the existing clusters in real time, and with the latter, besides updating the existing clusters, dynamic clustering also initiates 4 new clusters.

Results of the experiments are illustrated in Figure 10. It is notable that when all clusters are initialized, dynamic clustering performs as accurately as does K-Means at $p = 0.9$ and above, though as expected its accuracy steeply decreases for less distinguishable datasets. Instead, the performance of the dynamic clustering is much better in updating the existing clusters as compared to creating new clusters. Thus, dynamic clustering can be applied to achieve adaptability, but it should be complemented by periodic re-clustering.

5.2.6 Scalability of the Dynamic Clustering

We conducted several experiments to evaluate scalability of our dynamic clustering algorithm as the number of clusters and order of the FM model representing the sessions and clusters grow. We extended the same methodology described in Section 5.2.1 to generate five datasets each consisting of 50,000 sessions ($p = 0.9$ and $v = 11$). The datasets differ in the number of clusters they include: 10, 20, 30, 40, and 50. Then, we used FM with various orders, 1, 2, 3, and 4, to model the sessions in each dataset (in each case, the same order of FM is used to

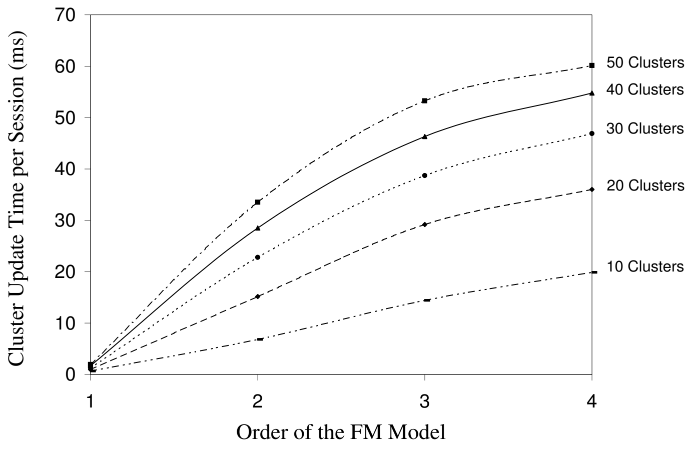


Figure 11: Scalability of the Dynamic Clustering Algorithm

capture all features, and weight factors for the features are selected to be $w_S = w_H = 0.5$). Finally, we applied dynamic clustering initiated with the core paths corresponding to the clusters existing in each dataset to cluster the sessions. We used an IBM™ IntelliStation ZPro-2, Pentium III Xeon with 768 MB RAM, the Windows 2000 Server operating system, and the Java 3.0 compiler to implement all these experiments.

Figure 11 illustrates the results of our experiments. In this figure, the horizontal axis is the order of FM used to model the sessions in a dataset, and the vertical axis is the average amount of time used by the dynamic clustering algorithm to update the cluster set as a new session arrives. As expected, with increase in complexity of the FM model on the one hand, and increase in the number of clusters on the other hand, the update time of the cluster set grows. However, even for the dataset including 50 clusters, modeled with FM order-4, the average cluster update time as a new session arrives is less than 60 ms.

5.2.7 Performance of the FM Model with Real Data

We conducted several experiments to study the performance of the FM model in handling real data. Particularly, we investigated FM capabilities in capturing meaningful clusters in real data. To collect the real data, we tracked the website of a journal, the *Journal of Computer-Mediated Communication* (JCMC) at the University of Southern California (http://www.ascusc.org/jcmc/), for a 15-day period in the Summer of 2001. We used our remote agent described in Section 3 for tracking. The JCMC website provides on-line access

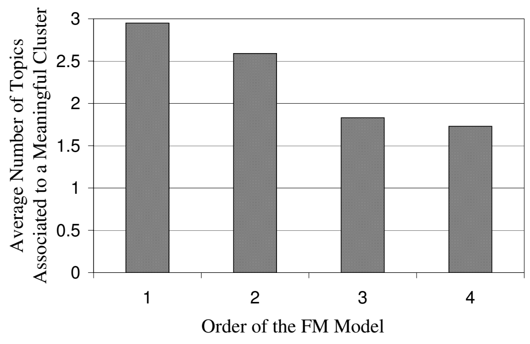
Table 2: Journal Issues and Their Categorization Based on Topic

Category	Topic of the Issue	Associated Web-Page IDs*
(1-1)	Collaborative Universities	63-67 and 465-500
(1-2)	Play and Performance in CMC	69-104
(1-3)	Electronic Commerce	109-127
(1-4)	Symposium on the Net	128-145
(2-1)	Emerging Law on the Electronic Frontier, 1	148-170
(2-2)	Emerging Law on the Electronic Frontier, 2	171-192
(2-3)	Communication in Information Spaces	194-207
(2-4)	Network and Netplay	208-224
(3-1)	Studying the Net	226-237
(3-2)	Virtual Environments, 1	239-248
(3-3)	Virtual Environments, 2	249-260
(3-4)	Virtual Organizations	262-275
(4-1)	Online Journalism	276-290
(4-2)	CMC and Higher Education, 1	291-329
(4-3)	CMC and Higher Education, 2	330-359
(4-4)	Persistent Conversation	360-371
(5-1)	Searching for Cyberspace	372-382
(5-2)	Electronic Commerce and the Web	384-396
(5-3)	Computer-Mediated Markets	398-411
(5-4)	Visual CMC	414-437

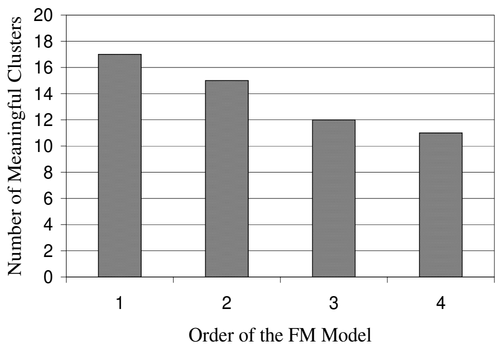
* The page IDs are assigned to the web pages of the website using an off-line utility that crawls the website directory and labels the pages with unique IDs. The same IDs are used by the remote agents to refer to the pages when reporting usage information.

to 20 published issues of the journal. JCMC is a quarterly journal. Publications in each year is dedicated to a particular general topic and each quarterly issue of the journal is devoted to a special topic under the general annual topic (see Table 2 for categorization of the journal issues).

During the tracking period, we collected a total of 502 sessions with a maximum length of 56. The entire website is dedicated to computer-mediated communication issues; hence, we consider all 554 pages of the website to be in the same concept space. Since HTTP is a stateless protocol, the web user does not explicitly indicate when she/he actually leaves the website. Therefore, termination of a user session in a single concept space can be only probabilistically be determined considering a session timeout period. Catledge and Pitkow (1995) first measured a period of 25.5 minutes as the optimum timeout for a session; we use the same timeout value in our tracking system. As a remote agent observes an idle



(a)



(b)

Figure 12: Performance of the FM Model with Real Data

period of at least 25.5 minutes during which the user stops interaction with the website, it reports the session termination. We used all three features, hit, sequence, and view time ($w_H = w_S = w_T = \frac{1}{3}$, after normalization) with the FM of various orders, 1, 2, 3, and 4 (the same order used for all features) to model the captured sessions.

Thereafter, corresponding to the number of topics of the journal issues, we used K-Means with $PPED$ to cluster the sessions into 20 clusters. To discover the relation between the generated clusters and the topics at the website, first we estimated the correspondence between a cluster and each topic by adding up the hit values of those segments (of all sessions included in the cluster) that are completely contained within the range of the pages associated to the topic (see Table 2). Second, using this measure, termed *correspondence*, we determined the topics most related to each cluster by filtering out unrelated (or less-related) topics via a fixed threshold as the minimum acceptable correspondence value (we used $correspondence = 15$ as the threshold value, a threshold selected logically by observing the distribution of the correspondence value. It is important to note that since our analysis is comparative, the actual value of the threshold does not affect the results reported). Finally, to identify the "meaningful" clusters, we manually reviewed the topics associated to each cluster. Assuming that users of the website are often seeking information about a certain topic at each session (a logical assumption for a journal site), we consider a meaningful cluster to be a cluster for which its associated topics are related to each other (according to expert human view).

Figure 12 depicts the results of our experiments with real data. Figure 12a shows that as the order of the FM used to model the sessions increases, the average number of topics associated to each meaningful cluster decreases. On the other hand, Figure 12b illustrates that with increase in the FM order, the total number of meaningful clusters generated decreases. Since higher order FM is more complete, and since with higher FM orders the dimension of the feature vector space grows rapidly, a meaningful cluster of sessions is generated only when a group of sessions are very similar to each other; hence, it is more difficult to generate a meaningful cluster, but, meaningful clusters are more accurately associated with particular topics or user behaviors. One can think of using both a low-order and a high-order FM model to generate two cluster sets for a website. Then, as a new session arrives, it can first be classified using high-order clusters. If it is classified to a meaningful cluster, the user behavior, which is well oriented towards a particular interest, is accurately detected. Otherwise, the session is next classified using low-order clusters to estimate the general characteristics of user behavior as classified to the more general clusters.

6. Conclusions and Future Work

In this paper, we defined a framework for web usage mining (WUM) that satisfies requirements of web-personalization applications. The framework is composed of an accurate tracking technique, and a new model (the FM model) to analyze users' access patterns. The FM model, which is a generalization of the vector model, allows for a flexible, real time, and adaptive WUM. We argued that these characteristics are not only useful for off-line and conventional WUM, but also critical for on-line anonymous web personalization. We demonstrated how flexibility of FM allows conceptualization of new navigation features as well as trading performance for accuracy by varying the *order*. For FM, we proposed a similarity measure, $PPED$, that can accurately classify partial sessions. This property is essential for real time and anonymous WUM. We then utilized $PPED$ within a dynamic clustering algorithm to make FM adaptable to short-term changes in user behaviors. Dynamic clustering is possible since unlike the Markov model, incremental updating of the FM model has a low complexity. Finally, we conducted several experiments that demonstrated the following:

- High accuracy of our tracking technique (47.66% improvement),
- Superiority of FM over the vector model (by at least 25%),

- High precision of session classification when $PPED$ is applied (above 98%),
- Tolerable accuracy of dynamic clustering as compared to K-Means (only 10% worse while being adaptable),
- Scalability of the dynamic clustering algorithm (at most 60ms to update a cluster-set of 50 clusters in order-4 FM), and
- Capabilities of the FM model in capturing meaningful clusters in real data.

We intend to extend this study in several ways. First, we would like to design a recommendation system based on the WUM framework described in this paper. Running a real web personalization application in this framework will allow us to enhance further its capabilities. Second, since the parameter TDC in dynamic clustering algorithm (see Figure 3) is application-dependent, it should be learned through an optimization process. We are looking into various optimization techniques so that we can automatically determine the optimum value for TDC. Third, we plan to investigate other aggregation functions that might be more appropriate for certain features, as opposed to the simple averaging for all the features. Finally, for our cluster and session models, we want to compress the matrix even further, maybe through singular value decomposition (SVD).

Acknowledgments

We are grateful for the assistance given to us by Jabed Faruque and Ming-Chang Lee in conducting the experiments. This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and IIS-0082826, NIH-NLM grant nr. R01-LM07061, DARPA and USAF under agreement nr. F30602-99-1-0524, and unrestricted cash gifts from NCR, Microsoft, and the Okawa Foundation.

References

Ackerman M., D. Billsus, S. Gaffney, S. Hettich, G. Khoo, D. Kim, R. Klefstad, C. Lowe, A. Ludeman, J. Muramatsu, K. Omori, M. Pazzani , D. Semler, B. Starr. 1997. Learning probabilistic user profiles: applications to finding interesting web sites, notifying users of relevant changes to web pages, and locating grant opportunities. *AI Magazine* **18** 47-56.

Agrawal, R., R. Srikant. 1994. Fast algorithms for mining association rules. Proceedings of the 20th VLDB conference, Santiago, Chile, 487-499.

Allen C., D. Kania, B. Yaeckel. 2001. *One-to-One Web Marketing: Build a Relationship Marketing Strategy One Customer at a Time*, 2nd edition. John Wiley and Sons, New York.

Armstrong R., D. Freitag, T. Joachims, T. Mitchell. 1995. WebWatcher: a learning apprentice for the world wide web. AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, Stanford, CA. 6-13.

Baumgarten M., A.G. Bchner, S.S. Anand, M.D. Mulvenna, J.G. Hughes. 2000. Navigation pattern discovery from internet data. M. Spiliopoulou, B. Masand, eds. *Advances in Web Usage Analysis and User Profiling*, *Lecturer Notes in Computer Science* **1836** 70-87.

Blue Martini Software, Inc. 2002. Blue Martini.
http://www.bluemartini.com

Borges J., M. Levene. 1999. Data mining of user navigation patterns. Proceedings of Workshop on Web Usage Analysis and User Profiling (WEBKDD), ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, 31-36.

Breese J.S., D. Heckerman, C. Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. Proceedings of Uncertainty in Artificial Intelligence, Madison, WI. Morgan Kaufmann, San Francisco, CA.

Büchner A.G., M.D. Mulvenna. 1998. Discovering internet marketing intelligence through online analytical web usage mining. ACM SIGMOD Record **27** 54-61.

Cadez I., D. Heckerman, C. Meek, P. Smyth, S. White. 2000. Visualization of navigation patterns on web site using model based clustering. Technical Report MSR-TR-00-18, Microsoft Research, Microsoft Corporation, Redmond, WA.

Carchiolo V., A. Longheu, M. Malgeri. 2000. Extracting logical schema from the web. PRICAI 2000 Workshop on Text and Web Mining, Melbourne, Australia. 64-71.

Catledge L., J. Pitkow. 1995. Characterizing browsing behaviors on the world wide web. *Computer Networks and ISDN Systems* **27** 1065-1073.

Chen M.S., J.S. Park, P.S. Yu. 1998. Efficient data mining for path traversal patterns. *IEEE*

Transactions on Knowledge and Data Engineering **10** 209-221.

Cohen W., A. McCallum, D. Quass. 2000. Learning to understand the web. *IEEE Data Engineering Bulletin* **23** 17-24.

Cooley R., B. Mobasher, J. Srivastava. 1997. Grouping web page references into transactions for mining world wide web browsing patterns. Proceedings of KDEX'97, IEEE, Newport Beach, CA. 2-9.

Cooley R., B. Mobasher, J. Srivastava. 1999. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems* **1** 5-32.

Dialpad Communications, Inc. 2002. Dialpad.
http://www.dialpad.com

Drott M.C. 1998. Using web server logs to improve site design. *Proceedings on the Sixteenth Annual International Conference on Computer Documentation*, Quebec, Canada. 43-50.

Faulstich L.C., M. Spiliopoulou, V. Linnemann. 1997. WIND: a warehouse for internet data. *Proceedings of Fifteenth British National Conference on Databases (BNCOD)*, London, UK. 169-183.

Finin T., C. Nicholas, J. Mayfield. 1997. Agent-based information retrieval. *Special Interest Group on Information Retrieval (SIGIR'97).*
http://agents.umbc.edu/ir/sigir97/

Fu Y., K. Sandhu, M. Shih. 1999. Clustering of web users based on access patterns. International Workshop on Web Usage Analysis and User Profiling (WEBKDD'99), San Diego, CA. 18-25.

Greenberg S., A. Cockburn. 1999. Getting back to back: alternate behaviors for a web browser's back button. *Proceedings of the 5th Annual Human Factors and Web Conference*, Gaithersburg, MD.
http://www.itl.nist.gov/iaui/vvrg/hfweb/proceedings/greenberg/

Henzinger M. 2000. Link analysis in web information retrieval. *Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society* **23** 3-9.

Huberman B., P. Pirolli, J. Pitkow, R. Lukos. 1997. Strong regularities in world wide web surfing. *Science* **280** 95-97.

INT Media Group. 2002. BotSpot.
http://botspot.com/

Joshi A., R. Krishnapuram. 1999. Robust fuzzy clustering methods to support web mining. *Proceedings of SIGMOD Workshop in Data Mining and Knowledge Discovery*, Seattle, WA. 1-8.

Kohonen T., S. Kaski, K. Lagus, J. Salojrvi, V. Paatero, A. Saarela. 2000. Self organization of a massive document collection. *IEEE Transactions on Neural Networks, Special Issue on Neural Networks for Data Mining and Knowledge Discovery* **11** 574-585.

Konstan, J., B. Miller, D. Maltz, J. Herlocker, L. Gordon, J. Riedl. 1997. Applying collaborative filtering to usenet news. *Communications of the ACM* **40** 77-87.

Kuo Y.H., M.H. Wong. 2000. Web document classification based on hyperlinks and document semantics. *PRICAI 2000 Workshop on Text and Web Mining*, Melbourne, Australia. 44-51.

Levene L., G. Loizou. 2000. Zipf's law for web surfers. *Knowledge and Information Systems* **3** 16-24.

Lieberman H. 1995. Letizia: an agent that assists web browsing. *Proceedings of the International Joint Conference on Artificial Intelligence*, Montreal, Canada. 924-929.

Lieberman H. 1997. Autonomous interface agents. *Proceedings of the ACM conference on computers and human interfaces*, Atlanta, GA. 67-74.

Lin I.Y., X.M. Huang, M.S. Chen. 1999. Capturing user access patterns in the web for data mining. *Proceedings of the 11th IEEE International Conference Tools with Artificial Intelligence*, Chicago, IL. 22-29.

Maglio P., R. Barrett. 2000. Intermediaries personalize information streams. *Communications of the ACM*, **43** 96-101.

Mobasher B., R. Cooley, J. Srivastava. 1997. Web mining: information and pattern discovery on the world wide web. *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, Newport Beach, CA. 558-567.

Mobasher B., R. Cooley, J. Srivastava. 2000b. Automatic personalization based on web usage mining. *Communications of ACM* **43** 142-151.

Mobasher B., H. Dai, T. Luo, M. Nakagawa, Y. Sun, J. Wiltshire. 2000a. Discovery of aggregate usage profiles for web personalization. *Proceedings of the Web Mining for E-Commerce Workshop WebKDD'2000*, Boston, MA.
http://maya.cs.depaul.edu/ mobasher/papers/webkdd2000/webkdd2000.html

MTV Networks. 2002. mtv.com.
http://www.mtv.com

Mulvenna M.D., S.S. Anand, A.G. Büchner. 2000. Personalization on the net using web mining: introduction. Communications of ACM **43** 122-125.

NetIQ. 2002. WebTrends.
http://www.webtrends.com

Paliouras G., C. Papatheodorou, V. Karkaletsis, C.D. Spyropoulos. 2000. Clustering the users of large web sites into communities. *Proceedings of the International Conference on Machine Learning*, Stanford, CA. 719-726.

Pazzani M., L. Nguyen, S. Mantik. 1995. Learning from hotlists and coldists: towards a WWW information filtering and seeking agent. *Proceedings of IEEE International Conference on Tools with AI*, Washington, DC. 39-46.

Perkowitz M., O. Etzioni. 1998. Adaptive web sites: automatically synthesizing web pages. *Fifth National Conference in Artificial Intelligence*, Cambridge, MA. 727-732.

Perkowitz M., O. Etzioni. 2000. Toward adaptive web sites: conceptual framework and case study. *Artificial Intelligence* **118** 245-275.

Personify Inc. 2002. Personify.
http://www.personify.com

Pitkow J.E. 1998. Summary of WWW characterizations. *Web Journal* **2** 3-13.

Sarwar, B.M., G. Karypis, J.A. Konstan, J. Riedl. 2000. Analysis of recommender algorithms for e-commerce. *ACM E-Commerce'00 Conference.* Minneapolis, MN. 158-167.

Scharl A. 1999. A conceptual, user-centric approach to modeling web information systems. *Proceedings of the Fifth Australian World Wide Web Conference*, Lismore, Australia.
http://ausweb.scu.edu.au/aw99/papers/scharl/paper.html

Shahabi C., F. Banaei-Kashani, J. Faruque. 2001. A reliable, efficient, and scalable system for web usage data acquisition. *WebKDD'01 Workshop, ACM-SIGKDD 2001*, San Francisco, CA. http://dimlab.usc.edu/Research.html

Shahabi C., A.M. Zarkesh, J. Adibi, V. Shah. 1997. Knowledge discovery from users web page navigation. *Proceedings of the IEEE RIDE97 Workshop*, Birmingham, England. 20-31.

Spiliopoulou M. 2000. Web usage mining for site evaluation: making a site better fit its users. *Communications of ACM* **43** 127-134.

Spiliopoulou M., L.C. Faulstich. 1999. WUM: a tool for web utilization analysis. *Lecture Notes in Computer Science* **1590** 184-203.

Spiliopoulou, M., B. Mobasher, B. Berendt, M. Nakagawa. 2002. Evaluating the quality of data preparation heuristics in web usage analysis. *INFORMS Journal on Computing*, to appear.

Srivastava J., R. Cooley, M. Deshpande, P.N. Tan. 2000. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explorations* **1** 12-23.

Strehl, A., J. Ghosh. 2002. Relationship-based clustering and visualization for high dimensional data mining. *INFORMS Journal on Computing*, to appear.

Sun Microsystems, Inc. 2002. java.sun.com.
http://java.sun.com

VanderMeer D., K. Dutta, A. Datta, K. Ramamritham, S.B. Navanthe . 2000. Enabling scalable online personalization on the web. *Proceedings of the 2nd ACM Conference on Electronic Commerce*, Minneapolis, MN. 185-196.

W3C. 2002. Web Characterization Activity.
http://www.w3.org/WCA/

WebSideStory, Inc. 2002. WebSideStory.
http://www.websidestory.com

Yahoo!, Inc. 2001. Yahoo! reports third quarter 2000 financial results.
http://docs.yahoo.com/docs/pr/release634.html

Yan T.W., M. Jacobsen, H. Garcia-Molina, U. Dayal. 1996. From user access patterns to dynamic hypertext linking. *Fifth International World Wide Web Conference*, Paris, France. 1007-1118.

Zhang T., R. Ramakrishnan, M. Livny. 1996. BIRCH: an efficient data clustering method for very large databases. *SIGMOD '96*, Montreal, Canada. 103-114.

Zheng Z., B. Padmanabhan, S. Kimbrough. 2002. On the existence and significance of data preprocessing biases in web usage mining. *INFORMS Journal on Computing*, to appear.

Zukerman I., D.W. Albrecht, A.E. Nicholson. 1999. *Predicting users' requests on the WWW. Proceedings of the Seventh International Conference on User Modeling (UM-99)*, Banff, Canada. 275-284.

Accepted by Amit Basu; received February 2001; revised June 2001, January 2002; Accepted April 2002.