A Framework for Efficient and Anonymous Web Usage Mining Based on Client-Side Tracking

Cyrus Shahabi and Farnoush Banaei-Kashani

Department of Computer Science, Integrated Media Systems Center, University of Southern California, Los Angeles, CA 90089-2561, USA
[shahabi,banaeika]@usc.edu

Abstract. *Web Usage Mining (WUM)*, a natural application of data mining techniques to the data collected from user interactions with the web, has greatly concerned both academia and industry in recent years. Through WUM, we are able to gain a better understanding of both the web and web user access patterns; a knowledge that is crucial for realization of full economic potential of the web. In this chapter, we describe a framework for WUM that particularly satisfies the challenging requirements of the web personalization applications. For on-line and anonymous web personalization to be effective, WUM must be accomplished in real-time as accurately as possible. On the other hand, the analysis tier of the WUM system should allow compromise between scalability and accuracy to be applicable to real-life web-sites with numerous visitors. Within our WUM framework, we introduce a distributed user tracking approach for accurate, efficient, and scalable collection of the usage data. We also propose a new model, the Feature Matrices (FM) model, to capture and analyze users access patterns. With FM, various features of the usage data can be captured with flexible precision so that we can trade off accuracy for scalability based on the specific application requirements. Moreover, due to low update complexity of the model, FM can adapt to user behavior changes in real-time. Finally, 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 WUM framework for efficient web personalization.

1 Introduction

In this chapter, 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 [45]. WUM is the process of discovering and interpreting patterns of user access to the web information systems by mining the data collected from user interactions with the system. Knowledge of user access patterns is useful in numerous applications: supporting web-site design decisions such as content and structure justifications [44, 14], optimizing systems by enhancing caching schemes and load-balancing, making web-sites adaptive [34], supporting business intelligence and marketing decisions [8], testing user interfaces, monitoring for security purposes, and more importantly, in web personalization applications such as recommendation systems [36] and target advertising. Commercial products such as *Personify*^{TM} [50], *WebSideStory*^{TM} [51], *BlueMartini*^{TM} [52], and *WebTrends*^{TM} [55], and acquired companies such as *Matchlogic*^{TM}, *Trivida*^{TM}, *Andromedia*^{TM}, and *DataSage*^{TM} are all witnesses of commercial interests in WUM.

Within WUM-based web personalization applications, the user access patterns are utilized to identify needs and preferences of each individual user, and subsequently, customize the content and/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 and to identify the user needs in real-time. The system should treat each user as an anonymous individual and identify user needs per session, otherwise besides violating users privacy by tracking users across sessions, the system will be incapable of distinguishing between various user roles and/or different users using the same client[1].

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, particularly it should be able to identify user needs in a fraction of the session period. In most cases, violation of this time constraint renders the result of the personalization totally useless; for example in a recommendation system, usually there is little use for the recommendations that are generated after 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 visitors every day, generating 48GB clickstream data per hour [54]. Even assuming the analysis model is built off-line, this data cannot be analyzed/classified in real-time unless some features of the data be dropped while interpreting the 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, necessarily we want to consider every single action of a user and to be as accurate as possible, otherwise the customization will be inefficacious. Therefore, for efficient customization, users 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.

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. The WUM framework described in this chapter

[1] In case static profiling is unharmful/desirable, with a hybrid approach anonymous WUM can be employed for more flexible predictions.

comprises an accurate, efficient, and scalable tracking tier, and a flexible and adaptive analysis tier to support anonymous web personalization efficiently. Our tracking approach [39] is 1) *accurate* because it collects accurate data as close as possible to the real data, 2) *efficient* because it imposes minimum amount of overhead to the WUM system to allow real-time analysis, and 3) *scalable* because it scales to be applicable to large-scale applications where volume of the data to be acquired by the WUM system exceeds tens of GBs per day. Our analysis tier takes advantage of a novel model, the *Feature Matrices (FM)* model [40], to capture users navigation. FM is a model with flexible complexity to allow striking a compromise between accuracy and scalability, given the specific requirements of an application. The FM model benefits from two types of flexibility. First, FM is an open model; it can be tuned to capture various features of the web usage data and incorporated new features as required by any particular application domain. Second, by tuning *order* of the FM model, one can determine the accuracy of the data captured about various features of navigation; hence, trade-off between accuracy and scalability is easily accomplished. Besides, the FM model can be updated both off-line and, incrementally, online so that it can adapt to both short-term and long-term changes in user behaviors. Finally, we propose a novel similarity measure, *PPED*, to compare the FM models of partial paths with clusters of paths. With anonymous WUM, accurate classification of partial navigation paths of new users to the cluster representatives is crucial.

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 WUM applications. The remainder of this chapter is organized as follows. Section 2, summarizes the related work. We 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 included in Section 5. Finally, Section 6 concludes the chapter and discusses our future directions.

2 Related Work

Web mining is broadly defined as the discovery and analysis of useful information from the World Wide Web. A detailed taxonomy of Web Mining domains is provided in [26]. Here, first we briefly characterize different domains that pertain to Web Mining. Thereafter we will focus on some of the current researches being performed 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 the web documents [12].

– **Structure data:** The meta data that defines the organization of the web information systems. *Web Structure Mining* is the process of inferring knowledge from the structure of data [21, 18].
– **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 [5].

The idea of exploiting usage data to customize the web for individuals was suggested by researchers as early as 1995 [4, 24, 32]. A comprehensive survey of the existing efforts in WUM is provided by Srivastava et al. [45]. Some of the current approaches with the two tiers of WUM, tracking and analysis, are summarized in the following sections.

2.1 User Interaction Tracking

The usage data are usually obtained from either web server log, at the server side, or web browser, at the client side[2]. Server log is not reliable as a source of usage data for WUM because server log data are not accurate. There are various levels of caching embedded in the web, mainly to expedite users access to the frequently used pages. Those pages requested by hitting the “Back” button, which is heavily used by the web users nowadays [16], are all retrieved from the web browser cache. Also, proxy servers provide an intermediate level of caching in the enterprise level. Unfortunately, cache hits are missing from the server log, rendering it as an incomplete source of information to acquire spatial features of user interactions such as hit-count [13, 35]. Moreover, even for those entries captured by the server log, the temporal aspects of user interactions are recorded inaccurately, whereas temporal features such as view-time of pages are considered highly informative in deducing user preferences [20]. The timestamps recorded for each server log entry includes the network transfer time. It is important to note that due to non-deterministic behavior of the network, the amount of this noise varies rapidly and there is no trivial way to filter it out from the server log data.

Furthermore, data acquisition via server log is inefficient because when server log is used as the data source, preprocessing of the data becomes the prerequisite of the WUM process. Preprocessing imposes many difficulties and results in a large amount of overhead to the actual process of WUM [13], so that practically it renders on-line mining of user behaviors impossible. Specifically, *user session identification* and *data cleansing* are the most difficult and time-consuming tasks performed during preprocessing of the server log. Due to stateless service model of the HTTP protocol, pages requested in a user session are logged independently in the server log. However, for meaningful WUM these requests must be re-identified and re-grouped into user sessions as semantic units of analysis. This

[2] Proxy server log can also be considered as a source of web usage data; however, this source is often used only to characterize browsing behavior of a group of anonymous users sharing a common proxy server. As far as this chapter is concerned, proxy server logs have the same characteristics as web server logs.

process, the so called user session identification, is usually performed based on the IP address of the client machine recorded in the log entry for each HTTP request. However, since there is a many-to-many relationship between users and IP addresses, this approach cannot provide reliable information. Due to proxy servers and/or IP masquerading, a single IP address can be used by multiple users. On the other hand, some ISPs assign a different IP address to HTTP requests of a user during a single session. Moreover, missing cache hits in the server log makes the user identification process even harder. Researchers have proposed various methods to resolve this problem, but none of these methods are without serious drawbacks. *Cookies*, which allow inter-session tracking of users, violate users privacy [17]; web-sites requiring *user registration* are often neglected by anonymous users; *dynamic URLs* embedding session ID restricts intermediate caching and does not correctly handle the exchange of URLs between people [35]; *cache-busting* defeats the speed up advantage gained by caching; the *hit-metering* scheme proposed by Mogul et al. [29] requires modification to the HTTP protocol; the *heuristics* proposed by Cooley et al. [13] only provide relative accuracy in the absence of additional information; and data acquisition at *application servers* [3] is only possible when actually users interact with the application services of a web-site (many user interactions are directly handled by the front-tier of the web-site; and many web-sites do not have a middle-tier, i.e. application server, at all). Another time-consuming preprocessing task is data cleansing. Often a page request results in recording several extra entries in the server log for the graphics and scripts besides the entry for the actual HTML file. These extra entries should not be included in the input to the analysis tier of the WUM system because they are not indicators of explicit user requests. During data cleansing these entries must be identified and eliminated from the server log.

In [37], we introduced a remote agent that acquires the user interactions from the browser at the client side. In this chapter, we describe a client-side web usage data acquisition system developed based on this remote agent[3]. When a user first enters a web-site enabled with the remote agent, the remote agent is uploaded into the browser at the client side. Thereafter, it captures all required features of user interactions with the web-site such as hits and view-times of web-pages, and transfers the acquired data to a data acquisition server, the *acquisitor*, where data are *directly* dumped into a database to be used by the analysis tier of the WUM system without any further preprocessing. This mechanism satisfies requirements of an accurate, efficient, and scalable data acquisition system. First, since with remote agent the data are collected at the client side, all cache hits are captured and variable network transfer time is excluded from recorded view-times for pages. Second, preprocessing tasks are totally eliminated with this approach. When the agent is uploaded to the browser, it receives a globally unique session ID from the acquisitor and labels all captured data sent to the server with that ID. Thus, the acquisitor can transparently store data captured

[3] This system is currently operational and was demonstrated at VLDB 2000 Conference [38].

by different agents as separate semantic units, i.e., user sessions, in the database without further requirement for user session re-identification. It is important to note that with this approach 1) unlike cookies, session IDs are only valid during a single active user session, so they do not violate users privacy, 2) unlike dynamic URLs and cache-busting, the caching mechanism is not affected/defeated, 3) unlike hit-metering, system works based on the current common protocols and technologies, 4) unlike heuristic algorithms for user session identification, user sessions can be identified with absolute reliability, and 5) unlike application servers, entire user interactions are observed and logged. Thus, this approach provides us with a safe method to superimpose state on the HTTP protocol and perform implicit user session identification with slight overhead on the clients. Moreover, since data to be captured are actively selected by the remote agent, as opposed to the server log that passively records any type of request, the required data cleansing is reduced. Finally, the acquisitor benefits for a scalable architecture to satisfy requirements of large-scale WUM systems. Of course, our approach has some drawbacks that are discussed in Section 3.1.

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 be dropped while modeling access patterns. Page *hit-count*, which indicates frequency of the page visits during a session, has been traditionally considered as an informative indicator of the user preferences [47]. Also, *order* or *sequence* of page accesses is recently identified as an important piece of information [11]. Dependency models such as aggregate tree [43] and hidden Markov model [9] 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 [20]. Although Yan et al. [47] and Levene et al. [23] show that view time has the 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 chapter is defined so that it can capture any number of these and/or any other feature that might seem informative. The features are captured per *segment*, which is the building block of a session (see Section 4.1). Variable number of the features and size of the segment allow 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 the user access patterns. Mobasher et al. [28] have applied the classical association rule Apriori algorithm [2] 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 user based on the observed items in the user session. Mobasher et al. [27] show that clustering techniques provide a better overall performance as compared with association rules when applied in the context of web personalization.

Another set of models, which we denote as 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. [49] and Cadez et al. [9] use Markov model for this purpose. Borges et al. [6] define a probabilistic regular grammar whose higher probability strings corresponds to user's preferred access pattern. Breese et al. [7] perform an empirical analysis of predictive algorithms such as Bayesian classification and Bayesian Network 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, we compare our approach with the Markov model as a typical dependency model.

In this chapter, we use another classical data mining technique, *clustering*, to mine the usage data. This approach was first introduced by Yan et al. [47]. With this approach, usually user sessions are 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 are investigated to analyze the clustering performance in the context of WUM. Fu et al. [15] employ *BIRCH* [48], an efficient hierarchical clustering algorithm; Nasraoui et al. [30] prefer a fuzzy relational clustering algorithm for WUM because they believe usage data is fuzzy in nature; Perkowitz et al. [33] introduce a new clustering algorithm, *cluster miner*, which is designed to satisfy specific web personalization requirements; Paliouras et al. [31] from 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. [27] 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 chapter is a generalization of the original Vector model introduced by Yan et al. [47], to be flexible in capturing users behavior anonymously and combining various features with tunable order of complexity. VanderMeer et al. [46] study anonymous WUM by considering dynamic profiles of users in combination with static profiles.

We analyze *Dynamic Clustering* as an approach to make the cluster model adaptive to short-term changes in users behavior. We also introduce an accurate similarity measure that avoids overestimation of distance between the partial user sessions and cluster representatives. Distance overestimation has been observed as a classification problem by Yan et al. [47] as well.

3 Tracking Tier

Our tracking system consists of two main components: 1) a remote agent, which is uploaded from the web server to the client machine as soon as client requests the first page of the web-site[4], and 2) a central data acquisition server, termed *acquisitor*, which assigns globally unique IDs to the remote agents migrated to the active clients, and receives and stores the data collected by the remote agents in a database to be analyzed later by the analysis tier of the WUM system. To equip a web-site with this data acquisition system, the code that uploads the remote agent to the client machine should be embedded in the pages of the web-site. We have developed a simple utility that automates this process so that even for large web-sites with huge directories of pages the entire system installation process can be done in a short period.

The data acquisition mechanism in our system is illustrated in Figure 1. As an anonymous user enters an acquisition-enabled web-site by requesting the first page, the system components go through the following steps:

1. The client sends request for the first page to the web server;
2. Web server uploads the first page and the remote agent to the client machine;
3. The remote agent sends a request for an ID to the acquisitor;
4. The acquisitor assigns a globally unique ID to the agent and responds with the ID value;
5. The client's request for the next page causes the agent to send the collected usage data about previous page to the acquisitor before sending the request for the next page to the web server;
6. Request for the next page is sent to the web server;
7. The next page is uploaded to the client machine;

Steps 5 through 7 of this procedure are repeated as long as the user has not left the web-site, i.e. user session is not terminated. Since HTTP is a stateless protocol, web user does not explicitly indicate when she/he actually leaves a web-site. Therefore, termination of a user session can only probabilistically be determined considering a session timeout period. Catledge et al. [10] first measured a period of 25.5 minutes as the optimum timeout for a session; we use the same timeout value in our system. As a remote agent observes an idle period of at least 25.5 minutes during which user stops interaction with the web-site, it reports the session termination to the acquisitor so that it can inform the analysis tier of the WUM system.

It is important to note that browser uploads the remote agent to the client machine only *once*, as the user enters the web-site. Afterwards, the agent stays resident in the client machine for as long as the session is not terminated. Although our tracking mechanism may seem straightforward, its implementation imposes several challenges. In the following sections, we explain the detailed implementation of the remote agent and the acquisitor.

[4] This page can be any page of the web-site, not only the homepage. It is the entry point of the user to the web-site.

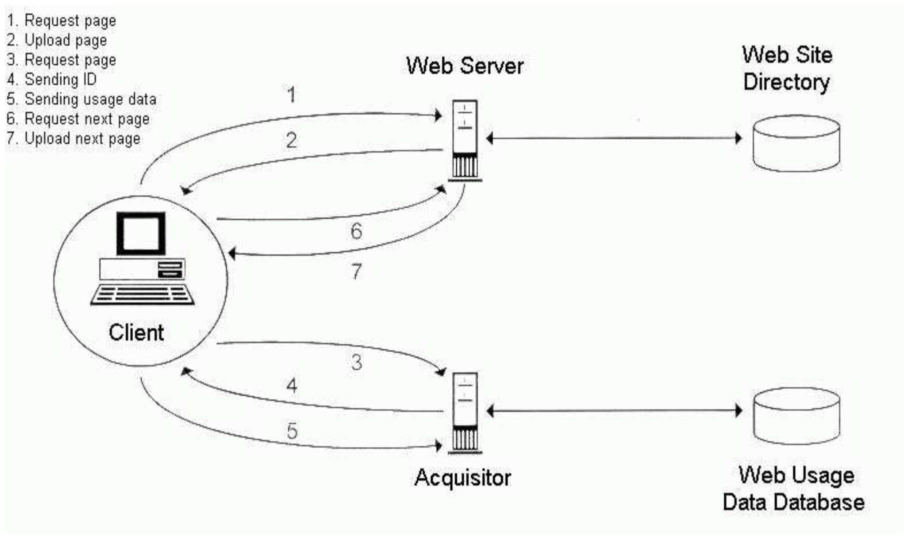


Fig. 1. Tracking Mechanism

3.1 Remote Agent

The existence and activities of the remote acquisition agent should be transparent to users because users usually consider the tracking process as an overhead to their actual web browsing purpose. Specifically, the agent should be implemented as a light-weight piece of code[5] to minimize both the network transmission latency/overhread of uploading the agent to the client machine, and the CPU cycles spent by the client to execute the agent code. Agents implemented as browser plug-in's, such as Syskill and Webert [1], or those developed as separate heavy-weight processes that interact with the browser, such as Letizia [24], are not suitable for this purpose. Also, those agents require users to use special-purpose or modified browsers; therefore, it is difficult to convince users to use the new browser unless enough incentives are offered. Finally, user privacy is an important issue to be considered with any user tracking mechanism. Users are usually reluctant to be monitored; specially they do not want to be tracked from site to site and session to session over long period of time. For instance, the data acquisition system proposed by Lin et al. [25] assumes the environment where user privacy is not a concern; hence, it is not applicable to the web.

We have developed our remote agent as a light-weight Java applet that transparently runs at the client machine. As mentioned before, the agent is uploaded to the client machine only once as the user enters the web-site. Although JavaScript technology provides more facilities to acquire user-web interactions (or browsing events) [55], we prefer the Java applet technology because while it is well supported by all common browsers, it is designed to be secure and hence

[5] Considering both size and execution time of the code.

satisfies users' expectations of privacy [56]. Unlike cookies, which are stored at the client machine to allow keep tracking of user's history of interactions with the web-site from session to session, our agent tracks the user interactions only during a single session and does not store any information at the client machine. Therefore, not only user anonymity is maintained, but also different roles/behaviors of the user in different sessions is identifiable. This characteristic allows *anonymous* WUM.

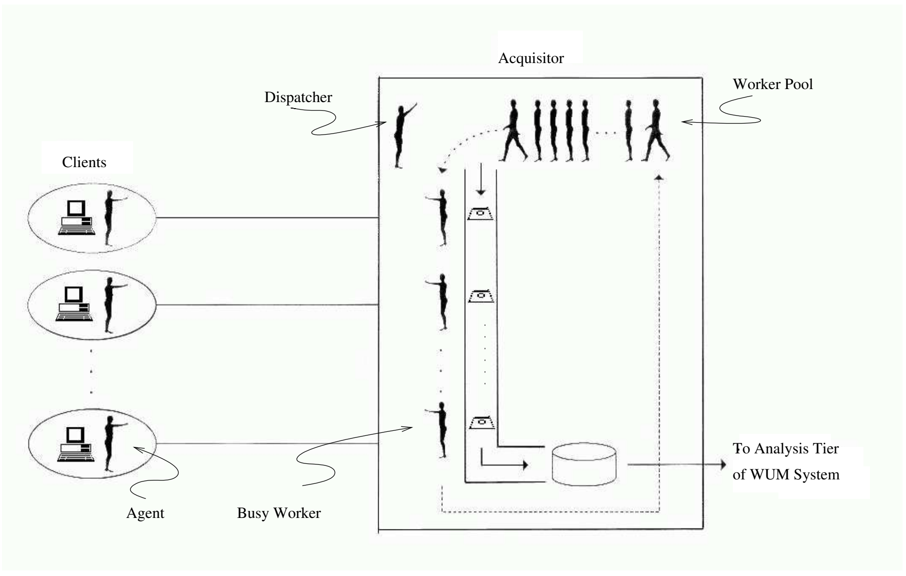


Fig. 2. Architecture of the acquisitor (data acquisition server)

The complete code for the remote agent is included in the appendix of this chapter. Each time the browser loads a web-page, it runs the applet. If it is the first time the applet is executed within the browser's JVM, i.e., when a user enters the web-site, the applet receives a unique ID from the acquisitor[6]. Besides, each time the applet is executed it records the load-time for the current page as it starts execution. Then, when the browser tries to load the next page, before the current page is unloaded, the applet records the unload-time for the page, computes its total view-time, connects to the acquisitor, and transfers a clean set of parameters including the unique agent ID to the acquisitor. Note that since view-time of a page is a time interval relative to the same client's clock (and not an absolute time value), clock synchronization between clients and acquisitor is not an issue. Here, for simplicity we are only capturing the view-time of the page, which provides the WUM system with view-time and hit-count features

[6] Each window of the browser receives a separate ID, which allows distinguishing between various user roles.

of user navigation. However, the same procedure is applicable in acquiring other navigational features.

There are some drawbacks with this approach. First, running the remote agent at the client side requires users cooperation in enabling Java at their browsers. This is the general problem with all client-side acquisition methods. However, considering the popularity of Java applets, Java is enabled by default in all common browsers such as Netscape™ and Internet Explorer™. Second, if a firewall blocks the TCP or UDP port used by the remote agent to communicate with the acquisitor, the captured usage data cannot be collected. This problem can be alleviated using the Remote Scripting technology developed by Microsoft™ [57]. Remote Scripting allows applets to use the HTTP port (port number 80) to communicate with the server. Since usually HTTP port is not blocked by firewalls, this will allow our remote agent to reach the server. Unfortunately, this technology is not supported by all web servers, e.g., Apache™. We are currently studying other technologies that can be used to eliminate this problem. Third, the delay of uploading the agent applet when the user first enters the web-site is undesirable, although with a light-weight agent the delay is typically tolerable. Finally, data traffic between the remote agent and the acquisitor is to be considered as the overhead of data collection for WUM. Since the payload of the packets exchanged between the agent and the acquisitor is light, using light-weight UDP packets for communication can alleviate this problem.

3.2 Acquisitor (Data Acquisition Server)

To be scalable, the data acquisition server should be able to handle requests from large number of active remote agents simultaneously. Our acquisitor is implemented as a multi-threaded daemon process with a standard single-dispatcher/ multiple-worker architecture. This structure is similarly employed in common large-scale servers, e.g. web servers. Figure 2 illustrates the architecture of the acquisitor. All connection requests from the remote agents are received by a single dispatcher thread. The dispatcher simply assigns an idle worker thread from the worker pool to handle the received connection and returns to listen for the next connection request. The worker processes the agent request, i.e., either ID request or request for storing the captured usage data, and returns to the worker pool as soon as the process is finished. The ID is generated using a large global integer variable that is increased each time a new ID is assigned to an agent. The variable is large enough so that by the time it wraps around the old session IDs are analyzed and removed from the database. The usage data are directly dumped into the database as received from the agent.

It is important to note that the acquisitor is independent of the web server. Therefore, it can be executed at a separate machine, if required. Separating the data acquisition server from the web server not only results in a more scalable system, but also allows centralized acquisition of usage data for distributed applications such as distributed web-hosting (e.g., through Akamai [58]). This greatly facilitates usage data collection in such systems.

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 denote this model as the *Feature Matrices (FM)* model. With FM, features are indicators of the information embedded in sessions. In order to quantify the features, we consider universal set of segments in a concept space as 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 and analyze 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.

Terminology

Web-site A web-site can be modeled as a set of static and/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* includes concepts such as *Books*, *Music*, *Video*, etc. The web pages within a web-site can be categorized based on the concept(s) to which they belong. A *concept space* or simply *concept* in a web-site is defined as the set of web pages that contain information about a certain concept. Note that contents of a web page may address more than one concept, therefore concept spaces of a web-site are not necessarily disjoint sets[7].

Path A *path P* in a web-site is a finite or infinite sequence of pages:

$$x_1 \to x_2 \to ... \to x_i \to ... \to x_s$$

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

[7] Determination of the concept spaces for a Web-site 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 Feature (Feature) Any spatial or temporal attribute of a path is termed a *path feature* or *feature.* 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 to be 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 the 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, hereafter we assume all sessions belong to the same concept. 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 denote the value n, as 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. $(x_1, x_2, ..., x_i, ..., x_n) \equiv x_1 \to x_2 \to ... \to x_i \to ... \to x_n$. We use tuple representation because it simplifies our discussion. Any subsequence of pages in a path can be considered as 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 1st order segment (x_1), 2nd order segment (x_3, x_2), and 4th order segment (x_3, x_2, x_5, x_2). We exploit the notion of segment as the building block of sessions in order to model their features.

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

Basics

Features We characterize sessions through the following features:

- *Hit (H):* 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 1.
- *Sequence (S):* 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 relative location of each segment in the sequence of segments that construct 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 basis to capture different features. Throughout our discussion, 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, we will explain how using higher order bases results in more complete characterization of the session by the FM model in 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 the new features. This is another option with 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 the similarities and dissimilarities among sessions appropriately.

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. n-dimensional matrix M_{r^n} is a generalization of 2-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, $M_{4\times4\times4}$ that is a cube with 4 rows in each of its 3 dimensions, is a feature matrix for a 4-page concept space with ε^3 as the basis. 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, 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 1st dimension, row x_β on the 2nd dimension, ... , and row x_ω on the n-th 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 10000 elements. On the other hand, number of segments existing in 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.

Session Model

Here, we 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. Thus, it suffices if we explain how to extract various features for a sample segment E:

- For Hit (H), we count the number of times E has occurred in the session ($H \geq 0$). Segments may partially overlap. As far as there is at least one non-overlapping page in two segments, the segments are assumed to be distinct. For example, the session $x_1 \to x_2 \to x_2 \to x_2 \to x_1$, has a total of 4 order-2 segments, including 1 occurrence of (x_1, x_2), 2 occurrences of (x_2, x_2), and 1 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 the value of 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 $x_1 \to^1 x_2 \to^2 x_2 \to^3 x_2 \to^4 x_1$, S value for the segments (x_1, x_2), (x_2, x_2), and (x_2, x_1) are 1, 2.5 $\left(= \frac{2+3}{2}\right)$, and 4, respectively.
- For View Time (T), we add up the time spent on each occurrence of E in the session ($T \geq 0$).

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. Nowadays, any popular web-site is visited by a huge number of users. In such a scale, we may employ any similarity measure and clustering algorithm to group the sessions into clusters, but mere 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 an accurate similarity measure for the purpose of clustering, and finally, in Section 4.3, we propose a variation to conventional clustering algorithms to make them real-time adaptable to varying behaviors.

With our approach of modeling a cluster, we aggregate feature values of all clustered sessions into corresponding feature values of a virtual session, called cluster centroid. The cluster centroid is considered as a representative of all the sessions in the cluster, or equally as the 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 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 $F(E)$ 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 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.

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, complexity of constructing the clusters is dependent on N, which is 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.

The price we have to pay to achieve lower space and time complexity is to sacrifice *completeness*[8]. 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[9].

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), where $n = \max(n_1, n_2, \ldots, n_m)$. In the worst case, $FM^{(n)}$ comprises m n-dimensional matrices M_{r^n}, one for each of the model features. Thus, *space* cost of $FM^{(n)}$ is $O(mr^n)$. *Time* complexity for user classification is $O(mL)$ and for updating a cluster by assigning a new session to the cluster is $O(mr^n)$. Therefore, space and time complexity of $FM^{(n)}$ model are both independent of M.

[8] A model is more complete if it is a better approximation for the real session/cluster.

[9] A formal proof for uniqueness of the FM model for a sesson/cluster is included in [42].

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

Table 1. Parameters

From $O(mr^n)$, complexity increases exponentially with n, which is the order of the FM model. Based on Property 1, as the order n increases, the FM model becomes more complete in describing its corresponding session or cluster:

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

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. Formal proof for Property 1 is included in [42].

The other crucial parameter in $O(mr^n)$ 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 absolutely application-dependent. The FM model is an open model in a sense that its structure allows incorporating new features as the need arises for different applications. Performing comparisons based on more features result in more accurate clustering, though again the complexity is increased.

Now let us compare the performance of FM with two other conventional models, namely the Vector model and the Markov model. The Vector model can be considered as one special case of the FM model. As used in [47], the Vector model is equivalent to an $FM^{(1)}$ model with H as the only captured feature. Thus, the Vector model scales as $O(r)$, 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). 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 [19], the Markov model has demonstrated acceptable performance in the context of WUM [9]. The transition matrix of an order-n Markov model is extractable from H feature matrix of an $FM^{(n+1)}$ model. Thus, the FM model at least captures the same amount of information as with an equivalent Markov model. They also benefit from the same time complexity of $O(L)$ 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.

4.2 Similarity Measure

A *similarity measure* is a metric that quantifies the notion of "similarity". To capture behaviors of the web-site 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 indicator of dissimilarity instead of similarity. For the purpose of clustering, both approaches are applicable.

In [37], 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 a new similarity measure, $PPED$, particularly defined to alleviate the overestimation problem attributed to pure Euclidean distance measure in the context of WUM [47]. This measure satisfies the mentioned precondition. Before defining the function of this similarity measure, let us explain how the FM model is interpreted by a similarity measure.

With a similarity measure, each feature matrix of FM is considered as 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 as 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 (analogous procedure applies when the similarity measure expresses similarity instead of dissimilarity). Each session model comprises a series of feature vectors, one for each feature captured by the FM model. For each feature F_i, the similarity measure is applied on 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. 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, we report on the results of our experiments in finding the compromised set of weight factors for H and S features.

Here, we explain our new similarity measure. 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 i-th elements, respectively. Vectors are assumed to have $N = r^n$ elements, where r is the cardinality of the concept space.

Projected Pure Euclidean Distance (PPED)

$PPED$ is a variant of Pure Euclidean Distance measure (PED) to alleviate the *overestimation* problem. To illustrate the overestimation problem with PED, 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 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.

Assume $\overrightarrow{A}$ and $\overrightarrow{B}$ are two feature vectors of the same type belonging to a session and a cluster model, respectively. To estimate the dissimilarity between $\overrightarrow{A}$ and $\overrightarrow{B}$, $PPED$ computes pure Euclidean distance between $\overrightarrow{A}$ and the projection of $\overrightarrow{B}$ on those coordinate planes at which $\overrightarrow{A}$ has non-zero components:

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

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

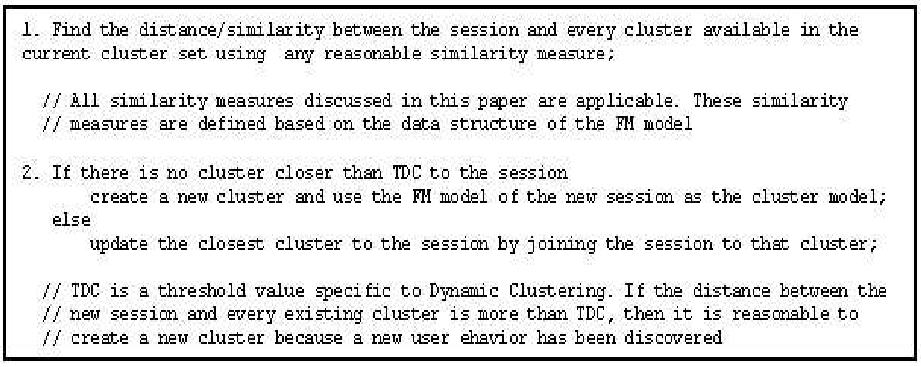
Non-zero components of $\overrightarrow{A}$ belong to those segments that exist in the session. Zero values, on the other hand, are related to the remainder of the segments in the basis universal set. By contrasting $\overrightarrow{A}$ with the projected $\overrightarrow{B}$, we compare the session 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.

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 in reducing the time complexity of the similarity measurement. According to Equation 2, the time complexity of $PPED$ is $O\left(mL\right)$ (refer to Table 1 for the definitions of the terms). In Section 5, we report on the superiority of $PPED$ performance as compared to two classical similarity measures, i.e. PED and cosine of the angle formed by the feature vectors (Vector Angle or VA).

4.3 Dynamic Clustering

As discussed in Section 4.1, since the FM model of a cluster is independent of the cluster cardinality, any cluster manipulation with FM has a reasonably low complexity. Leveraging on this property, we can apply the FM model in real-time applications.

One benefit of this property is that FM clusters can be updated dynamically and in real-time. Note that in most common cluster representations, complexity of adding a new session to a cluster is dependent on the cardinality of the cluster. Therefore, practically in large scale systems, they are not 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 market, 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.



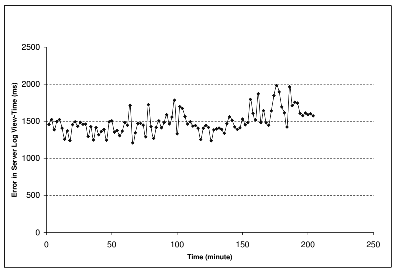
```
1. Find the distance/similarity between the session and every cluster available in the
current cluster set using  any reasonable similarity measure;

  // 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 ehavior has been discovered
```

Fig. 3. An algorithm for *dynamic clustering*

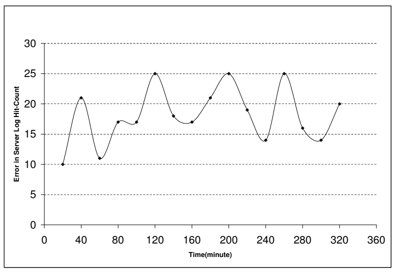
Periodical re-clustering is the typical approach in updating the clusters. This approach results in high accuracy, but it cannot be performed in real-time. According to our experiments to compare the accuracy of the dynamic clustering with that of a periodical re-clustering (see Section 5), dynamic clustering 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 required. That is, the cluster set should be updated in longer periods through periodical re-clustering to avoid divergence of the cluster set from the trends of the real user behaviors. Meanwhile, dynamic clustering can be applied in real-time to adapt the clusters and the cluster set to short-term behavioral changes.

5 Performance Evaluation

5.1 Tracking Tier



a. Server log error due to network time



b. Server log error due to cache hits

Fig. 4. Reliability of the server log usage data as compared to our system

We conducted several experiments to verify the mechanisms employed in tracking tier, and compared reliability of the usage data collected by our tracking system versus server log data reliability. We show that since with our remote agent the usage data are collected at the client side, the inaccuracy attributed to the server log data is entirely eliminated. Specifically, our data acquisition system is able to exclude the network transfer time from the recorded view-times for pages and also capture all cache hits.

To estimate the error due to the inclusion of network transfer time in page view-times, as recorded in the server log, we included a series of 10 pages within a real-world web-site directory[10]. This web-site comprises of 70 web-pages and it runs Apache web server version 1.3.12. The 10 included pages circularly call one another so that every 2 seconds the browser automatically requests the web server to upload the next page. We collected the page access entries for these pages as recorded both by our acquisitor and the server log for a period of 3.5 hours between 11:30am and 3:00pm during a working day of the week. The "NO-CACHE" option was used in the HTML pages to force the browser to retrieve the pages from the web server. Therefore, a server log entry is recorded for each page access and server log is not penalized for missing the cache hits. For each page access, we extracted the view-time of the page based on the corresponding server log entries and compared it with the exact view-time captured by our system to estimate the server log error. We computed the average error over successive time periods of 2 minutes each.

Results of this study are reported in Figure 4-a. In this figure, the X-axis is the time period of the experiment (in minutes) and the Y-axis is the average

[10] USC Annenberg School of Communications (http://www.ascusc.org/jcmc)

server log error (in milliseconds) in capturing the view-times of pages. As illustrated, the error can be as large as 2 seconds in each page view-time. Also, we extracted the average page view-time in this web-site from a server log containing real user access entries. The average page view-time for the web-site is 15 seconds; therefore, our system can improve the view-time accuracy up to 13%.

It is important to note that due to the large variance of network transfer time, as measured 1) at different times of the day and week and 2) at the same time but for different users dispersed in the Internet, we cannot simply eliminate the network transfer time by deducting a fixed value from all view-times captured by server log. Since our experiment is performed in a fairly short period of time, the variation of the network transfer time is not quite obvious in our results. However, as reported by Leighton et al. [22] variance of network transfer time can be as large as 7-10 seconds.

Finally, to measure the number of cache hits missing from the server log, we tracked real users access to the same web-site for a period of 5.5 hours. We counted number of page access entries existing in the acquisitor log but missing from the server log. These are the pages that are retrieved either from the browser cache or the proxy cache. In Figure 4-b, we demonstrate average number of cache hits missing from the server log (Y-axis) as a function of time (X-axis). The average is computed over successive time periods of 20 minutes each. The total average number of missing hits from the server log amounts to 0.9 pages per minute, whereas average number of total page accesses for this site is 2.23 pages per minute. Thus, our system can improve the hit-count accuracy up to 40%.

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 the dynamic clustering, and 5) 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. Here, we summarize the results of these experiments. The detailed description of the results, and also our experimental methodology is included in [42].

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 Equation 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 the similarities. Our experiments were intended to find the compromised set of weight factors w_S (weight factor for S) and w_H that results in the optimum accuracy in capturing the similarities.

The experiment results show that regardless of the weight factors applied, the accuracy is always above 94%. Thus, both features (Hit and Sequence) are equally successful in identifying the spatial similarities. Depending on distinguishability of the dataset, the optimum accuracy is achieved by employing a compromised combination of the similarities detected in *Hit* and *Sequence*. In brief, when similarity among users of the same cluster decreases, it is more important to track which pages they visit (*Hit*) rather than where in the session they visit each page (*Sequence*).

Accuracy of the Similarity Measures In Section 4.2, we introduced $PPED$ as an accurate similarity measure. Here, we compare accuracy of $PPED$ with two classical similarity measures, i.e. PED and cosine (VA).

The experiment results demonstrate that for real data, which assumes low distinguishability, $PPED$ outperforms VA with a wide margin. The results also show that since $PPED$ can measure the similarity between a user and a cluster based on user characteristics rather than cluster characteristics, overestimation of the distance between the session and its intended cluster is avoided by disregarding unnecessary cluster characteristics in distance estimation. Therefore, $PPED$ can achieve up to 30% improvement in accuracy as compared to PED.

Performance of the FM 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 demonstrate that accuracy of the Vector model decreases as the user sessions become less distinguishable, while the FM model can reasonably maintain its accuracy even with highly indistinguishable datasets. This superiority is because of: 1) incorporating *Sequence* into the model, and 2) capturing features based on order-2 segments.

Performance 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.

The experiment results show that as expected accuracy of dynamic clustering is less than that of K-Means but the degradation of the accuracy is tolerable. Thus, the dynamic clustering can be applied to achieve adaptability but it should be complemented by long-term periodical re-clustering. The results also demonstrate that the performance of the dynamic clustering is much better in updating the existing clusters as compared to creating new clusters.

Performance of the FM Model with Real Data We conducted several experiments to study the performance of the FM model in handling the real data. Particularly, we investigated FM capabilities in capturing meaningful clusters in real data. To collect the real data, we tracked the web-site of a journal, Journal of Computer-Mediated Communication (JCMC) at University of Southern California[11], for a 15-day period in Summer 2001. We used our remote agent described in Section 3.1 for tracking. The JCMC web-site provides on-line access 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 the categorization of the journal issues).

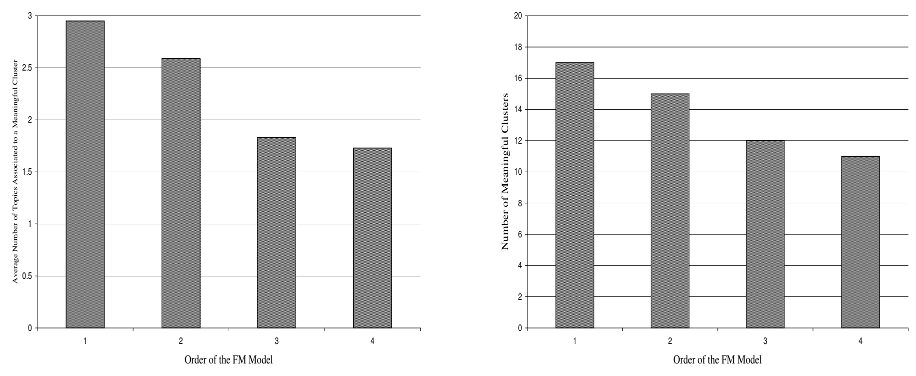
During the tracking period, we collected a total of 502 sessions with the maximum length of 56. The entire web-site is dedicated to Computer-Mediated Communication issues; hence, we consider all 554 pages of the web-site in the same concept space. As mentioned in Section 3, when a remote agent observes an idle period of at least 25.5 minutes during which user stops interaction with the web-site, 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.

Category	**Topic of the Issue**	**Associated Web-Page IDs**[12]
(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

Table 2. Journal issues and their categorization based on topic

[11] http://www.ascusc.org/jcmc/

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 web-site, 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[13]. Finally, to identify the "meaningful" clusters, we manually reviewed the topics associated to each cluster. Assuming users of the web-site are often seeking information about a certain topic at each session (a logical assumption for a journal site), we consider a meaningful cluster as a cluster that its associated topics are related to each other (according to expert human view).



(a) (b)

Fig. 5. Performance of the FM model with real data

Figure 5 depicts the results of our experiments with the real data. Figure 5-a shows that as the order of the FM used to model the sessions increases, the

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

[13] 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.

average number of topics associated to each meaningful cluster decreases. On the other hand, Figure 5-b illustrates that with increase in the FM order, the total number of the meaningful clusters generated decreases. Since higher order FM is more complete, and since with higher FM orders number of the dimensions 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, however, 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 web-site. Then, as a new session arrives, first it can be classified using the 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 the low-order clusters to estimate the general characteristics of the user behavior as classified to the more general clusters.

6 Conclusions

In this chapter, we defined a framework for Web Usage Mining (WUM) that satisfies requirements of the web personalization applications. The framework comprises 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 (40% improvement),
– Superiority of FM over the Vector model (at least by 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), and
– Capabilities of the FM model in identifying meaningful clusters with empirical datasets.

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 chapter [41]. Running a real web personalization application in this framework will allow us to further enhance 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).

Appendix

```
import java.io.*;
import java.net.*;
import java.applet.*;
import java.lang.*;
import java.util.Date;

public class RemoteAgent extends Applet {
    // ID initialization
    private static long ID = -1 ;
    private static long startSession = System.currentTimeMillis() ;
    // Load and unload time for the page
    private long LoadTime, UnloadTime ;
    // "view-time" to be estimated
    private long Time ;
    // Acquisitor's port number
    private static final int Port = 21000

    // Setup the connection to the acquisitor, and
    // Get the unique ID (if agent is uploaded for the
    //  first time to the client machine)
    public void init(){
    Socket sock ;
    PrintWriter pwOut = null ;
    BufferedReader brIn = null ;

    try {
        if (ID < 0 || ((System.currentTimeMillis() - startSession) > 1530000)) {
        sock = new Socket(this.getCodeBase().getHost(), Port) ;
        pwOut = new PrintWriter(sock.getOutputStream(), true) ;
        brIn = new BufferedReader(new InputStreamReader(sock.getInputStream())) ;
        ID = -1 ;
        pwOut.println(ID) ;
        ID = Long.valueOf (brIn.readLine()).longValue() ;
        pwOut.close() ;
        brIn.close() ;
        sock.close() ;
        startSession = System.currentTimeMillis() ;
        }
    } catch (UnknownHostException uhe){
        return ;
    } catch (IOException ioe) {
        return ;
    }

    Time = 0;
    }

    // Record the load time for the page
    public void start() {
    LoadTime = System.currentTimeMillis() ;
    }

    // Record the unload time for the page,
    // Compute the total view-time for the page, and
    // Transfer the captured data to the acquisitor
    public void stop() {
    Socket sock ;
    PrintWriter pwOut = null ;

    UnloadTime = System.currentTimeMillis() ;
    Time =  UnloadTime - LoadTime;

    // Prepares information string about page.
    String agentID = "<UID>" + ID + "<UID>" ;
    String pageID = "<PID>" + this.getParameter("pageID") + "<PID>" ;
    String T = "<T>" + java.lang.Long.toString(Time) + "<T>" ;
    String outString = agentID + pageID + T ;

    // Trying to send browsing information to Data Server.
    try {
        if (ID > 0) {
        System.out.println(outString) ;
        sock = new Socket(this.getCodeBase().getHost(), Port) ;
        pwOut = new PrintWriter(sock.getOutputStream(), true) ;

        // Sends data to Data Server.
        pwOut.println(outString) ;
        pwOut.close() ;
        sock.close() ;
        }
    } catch (UnknownHostException uhe){
        return ;
    } catch (IOException ioe) {
        return ;
    }
    }
}
```

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 ITR-0082826, NIH-NLM grant nr. R01-LM07061, NASA/JPL contract nr. 961518, DARPA and USAF under agreement nr. F30602-99-1-0524, and unrestricted cash/equipment gifts from NCR, Microsoft, Intel and SUN.

References

1. 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, and P. Yap. 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(2) 47-56, 1997.
2. Agrawal, R., and R. Srikant. 1994. *Fast algorithms for mining association rules.* Proceedings of the 20th VLDB conference, p.p 487-499, Santiago, Chile, 1994.
3. Ansari S., R. Kohavi, L. Mason, Z. Zheng. 2000. *Integrating E-Commerce and Data Mining: Architecture and Challenges.* Second International Conference on Electronic Commerce and Web Technologies, EC-Web 2000.

4. Armstrong R., D. Freitag, T. Joachims, and T. Mitchell. 1995. *WebWatcher: A Learning Apprentice for the World Wide Web.* AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, March 1995.
5. 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, Vol. 1836, Springer-Verlag, ISBN: 3-540-67818-2, July 2000.
6. Borges J., M. Levene. 1999. *Data mining of user navigation patterns.* Proceedings of Workshop on Web Usage Analysis and User Profiling (WEBKDD), in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p.p 31-36, San Diego, California, August, 1999.
7. Breese J.S., D. Heckerman, C. Kadie. 1998. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering.* Proceedings of Uncertainty in Artificial Intelligence, Madison, WI, July 1998. Morgan Kaufmann Publisher.
8. Büchner A.G., M.D. Mulvenna. 1998. *Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining.* ACM SIGMOD Record, ISSN 0163-5808, Vol. 27, No. 4, p.p 54-61, 1998.
9. Cadez I., Heckerman D., Meek C, Smyth P., and White S.: Visualization of Navigation Patterns on Web-Site Using Model Based Clustering. Technical Report MSR-TR-00-18, Microsoft Research, Microsoft Corporation, Redmond, WA,(2000)
10. Catledge L. and J. Pitkow. 1995. *Characterizing Browsing Behaviors on the World Wide Web.* Computer Networks and ISDN Systems, 27(6), 1995.
11. Chen M.S., J.S. Park, and P.S. Yu. 1998. *Efficient Data Mining for Path Traversal Patterns.* IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 2, p.p 209-221, April, 1998.
12. Cohen W., A. McCallum, D. Quass. 2000. IEEE Data Engineering Bulletin, Vol. 23, No. 3. p.p 17-24, September 2000.
13. Cooley R., B. Mobasher, and J. Srivastava. 1999. *Data Preparation for Mining World Wide Web Browsing Patterns.* Journal of Knowledge and Information Systems, 1(1):5-32, Springer-Verlag, February, 1999.
14. Drott M.C. 1998. *Using Web server logs to improve site design.* Proceedings on the sixteenth annual international conference on Computer documentation, p.p 43-50, Quebec Canada, September, 1998.
15. Fu Y., K. Sandhu, and 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, 1999.
16. Greenberg S. and 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, NIST, Gaithersburg, Maryland, June, 1999.
17. Greenspun P. 1999. *Philip and Alex's Guide to Web Publishing.* Chapter 9, User Tracking; ISBN: 1-55860-534-7.
18. Henzinger M. 2000. *Link Analysis in Web Information Retrieval.* IEEE Computer Society, Vol. 23 No. 3, September, 2000.
19. Huberman B., Pirolli P., Pitkow J., and Lukos R.: Strong Regularities in World Wide Web Surfing. Science, 280, p.p 95-97 (1997)
20. Konstan J., B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. 1997. *Applying Collaborative Filtering to Usenet News.* Communications of the ACM (40) 3, 1997.
21. 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, p.p 44-51, August 2000.

22. Leighton T. 2001. *The Challenges of Delivering Content on the Internet.* Keynote address in ACM SIGMETRICS 2001 Conference, Massachusetts, June 2001.
23. Levene L., and G. Loizou. 2000. *Zipf's law for web surfers.* Knowledge and Information Systems an International Journal, 2000.
24. Lieberman H. 1995. *Letizia: An Agent that Assists Web Browsing.* Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, August 1995.
25. Lin I.Y., X.M. Huang, and 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, November 7-9, 1999.
26. Mobasher B., R. Cooley, and 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), November 1997.
27. Mobasher B., H. Dai, T. Luo, M. Nakagawa, Y. Sun, J. Wiltshire. 2000. *Discovery of Aggregate Usage Profiles for Web Personalization.* Proceedings of the Web Mining for E-Commerce Workshop WebKDD'2000, held in conjunction with the ACM-SIGKDD Conference on Knowledge Discovery in Databases KDD'2000), Boston, August 2000.
28. Mobasher B., R. Cooley, and J. Srivastava. 2000. *Automatic Personalization Based on Web Usage Mining.* Special Section of the Communications of ACM on "Personalization Technologies with Data Mining", 43(8):142-151, August, 2000.
29. Mogul J, and P.J. leach. 1997. *Simple Hit-Metering for HTTP.* Internet draft-IETF-http-hit-metering-00.txt; HTTP Working Group. January, 1997.
30. Nasraoui O., R. Krishnapuram, A. Joshi. 1999. *Mining Web Access Logs Using a Fuzzy Relational Clustering Algorithm based on a Robust Estimator.* Proceedings of 8th World Wide Web Conference (WWW8), Torronto, May, 1999
31. Paliouras G., C. Papatheodorou, V. Karkaletsis, and C.D. Spyropoulos. 2000. *Clustering the Users of Large Web Sites into Communities.* Proceedings International Conference on Machine Learning (ICML), p.p 719-726, Stanford, California, 2000.
32. Pazzani M., L. Nguyen, and S. Mantik. 1995. *Learning from hotlists and coldists: Towards a WWW information filtering and seeking agent.* Proceedings of IEEE Intl.Conference on Tools with AI, 1995.
33. Perkowitz M., O. Etzioni. 1998. *Adaptive Web sites: Automatically Synthesizing Web Pages.* Fifth National Conference in Artificial Intelligence, p.p 727-732, Cambridge, MA, 2000.
34. Perkowitz M., and O. Etzioni. 2000. *Toward adaptive Web sites: Conceptual framework and case study.* Artificial Intelligence 118, p.p 245-275, 2000.
35. Pitkow J.E. 1997. *In Search of Reliable Usage Data on the WWW.* The Sixth International World Wide Web Conference, Santa Clara, California, 1997.
36. Schafer, J.B., J. Konstan, and J. Riedl. *Electronic commerce recommender applications.* Journal of Data Mining and Knowledge Discovery, 5:115–152, 2001.
37. Shahabi C., A. Zarkesh, J. Adibi, V. Shah. 1997.*Knowledge Discovery from Users Web-Page Navigation.* Proceedings of the IEEE RIDE97 Workshop, April, 1997.
38. Shahabi C., A. Faisal, F. Banaei-Kashani, J. Faruque. 2000. *INSITE: A Tool for Real-Time Knowledge Discovery from Users Web Navigation.* Proceedings of Very Large Databases (VLDB'2000), Cairo, Egypt, September, 2000.
39. Shahabi C., F. Banaei-Kashani, and J. Faruque. 2001. *A Reliable, Efficient, and Scalable System for Web Usage Data Acquisition.* WebKDD'01 Workshop in conjunction with the ACM-SIGKDD 2001, San Francisco, CA, August, 2001.

40. Shahabi C., F. Banaei-Kashani, J. Faruque, and A. Faisal. 2001. *Feature Matrices: A Model for Efficient and Anonymous Web Usage Mining.* EC-Web 2001, Germany, September, 2001.
41. Shahabi C., F. Banaei-Kashani, Y. Chen, D. McLeod. 2001. *Yoda: An Accurate and Scalable Web-based Recommendation System.* Sixth International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy, September, 2001.
42. Shahabi C., F. Banaei-Kashani, J. Faruque. 2001. *Efficient and Anonymous Web Usage Mining for Web Personalization.* To appear at INFORMS Journal on Computing - Special Issue on Mining Web-based Data for e-Business Applications.
43. Spiliopoulou M., and L.C. Faulstich. 1999. *WUM: A Tool for Web Utilization Analysis.* In extended version of Proceedings of EDBT Workshop WebDB'98, LNCS 1590. Springer-Verlag, 1999.
44. Spiliopoulou M. 2000. *Web usage mining for site evaluation: Making a site better fit its users.* Special Section of the Communications of ACM on "Personalization Technologies with Data Mining", 43(8):127-134, August, 2000.
45. Srivastava J., R. Cooley, M. Deshpande, and P.N. Tan. 2000. *Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data.* SIGKDD Explorations, Vol. 1, Issue 2, 2000.
46. VanderMeer D., K. Dutta, A. Datta, K. Ramamritham and S.B. Navanthe . 2000. *Enabling Scalable Online Personalization on the Web* . Proceedings of the 2nd ACM conference on Electronic commerce, p.p 185-196, 2000.
47. Yan T.W., Jacobsen M., Garcia-Molina H., Dayal U.: From User Access Patterns to Dynamic Hypertext Linking. Fifth International World Wide Web Conference, Paris, France, (1996)
48. Zhang T., R. Ramakrishnan, and M. Livny. 1996. *BIRCH: An Efficient Data Clustering Method for Very Large Databases.* SIGMOD '96, p.p 103-114, Montreal, Canada, June, 1996.
49. Zukerman I., D.W. Albrecht, and A.E. Nicholson. 1999. *Predicting users' requests on the WWW.* Proceedings of the Seventh International Conference on User Modeling (UM-99), Banff, Canada, p.p 275-284, June, 1999.
50. *http://www.personify.com*
51. *http://www.websidestory.com*
52. *http://www.bluemartini.com*
53. *http://www.webtrends.com*
54. *http://docs.yahoo.com/docs/pr/release634.html*
55. *http://www.javascript.com*
56. *http://java.sun.com/sfaq*
57. *http://msdn.microsoft.com/remotescripting*
58. *http://www.akamai.com*