A Reliable, Efficient, and Scalable System for Web Usage Data Acquisition

Cyrus Shahabi Farnoush Banaei-Kashani Jabed Faruque

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

Abstract

Usage data acquisition tier is a necessary component of every *Web Usage Mining (WUM)* system. For efficacious WUM, the acquisition system must be reliable, efficient, scalable. The usage data are usually obtained from either Web server log, at the server side, or Web browser, at the client side. We argue that usage data acquisition via server log is neither reliable, nor efficient. It is unreliable due to side effects of the network, i.e. missing cache hits and variable network transfer time, and inefficient because of usage data requiring extensive preprocessing before it can be utilized. Instead, we propose a client-side data acquisition mechanism that entirely eliminates the inaccuracy with data collection, by collecting the data at the client side, and inefficiency with data acquisition, by avoiding unnecessary preprocessing of the data. We also explain the scalable design of our data acquisition server. Our experimental results verify correctness of our approach and demonstrate significant improvement in efficacy of this system as compared to data acquisition via server log.

1 Introduction

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 [13, 15]. Knowledge of user access patterns is useful in numerous applications: supporting Web-site design decisions such as content and structure justifications [6, 23], optimizing systems by enhancing caching schemes and load-balancing, making Websites adaptive [16], supporting business intelligence and marketing decisions [3], testing user interfaces, monitoring for security purposes, and more importantly, in Web Personalization applications such as recommendation systems [18] and target advertising. Commercial products such as *Personify*^{TM} [25], *WebSideStory*^{TM} [26], *BlueMartini*^{TM} [27], and *WebTrends*^{TM} [28], and acquired companies such as *Matchlogic*^{TM}, *Trivida*^{TM}, *Andromedia*^{TM}, and *Rightpoint*^{TM} are all witnesses of commercial interests in WUM. A comprehensive survey of the existing efforts in WUM is provided by Srivastava et al. [24].

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. A typical WUM system consists of two tiers: 1) Acquisition, in which user interactions are acquired, and 2) Analysis, in which user access patterns are discovered and interpreted by applying typical data mining techniques to the acquired data. In this paper, we describe a reliable, efficient, and scalable Web usage data acquisition tier for real-time WUM. Our data acquisition system is 1) *reliable* 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 [29].

The usage data are usually obtained from either Web server log, at the server side, or Web browser, at the client side[1]. We argue that server log is

[1]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 paper

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 [7], 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 [5, 17]. 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 [9]. 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 [5], 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 process, 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 and are rejected by the user community [8]; 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 [17]; *cache-busting* defeats the speed up advantage gained by caching; the *hit-metering* scheme proposed by Mogul et al. [14] requires modification to the HTTP protocol; the *heuristics* proposed by Cooley et al. [5] only provide relative accuracy in the absence of additional information; and data acquisition at *application servers* [2] 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 [19], we introduced a remote agent that acquires the user interactions from the browser at the client side. In this paper, we describe a client-side Web usage data acquisition system developed based on this remote agent[2]. 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 a reliable, 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 ap-

is concerned, proxy server logs have the same characteristics as Web server logs.

[2]This system is currently operational and was demonstrated at VLDB 2000 Conference [20]. Also, WUM systems incorporating this acquisition subsystem are discussed in [21, 22].

proach. 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 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 is the safest method to superimpose state on the HTTP protocol and perform user session identification implicitly with minimum overhead on the clients. Moreover, since data to be captured are actively selected by the remote agent, there is no need for data cleansing as opposed to the server log that passively records any type of request. Finally, the acquisitor benefits a scalable architecture to satisfy requirements of large-scale WUM systems.

The remainder of this paper is organized as follows. In Section 2, we describe detailed design of our data acquisition system. The results of our experiments for verification and evaluation of the system are included in Section 3. Finally, Section 4 concludes the paper.

2 System Design

Our usage data acquisition 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[3], 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 by 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. [4] 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 the remote agent is only uploaded once to the client machine 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 data acquisition 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.

[3]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.

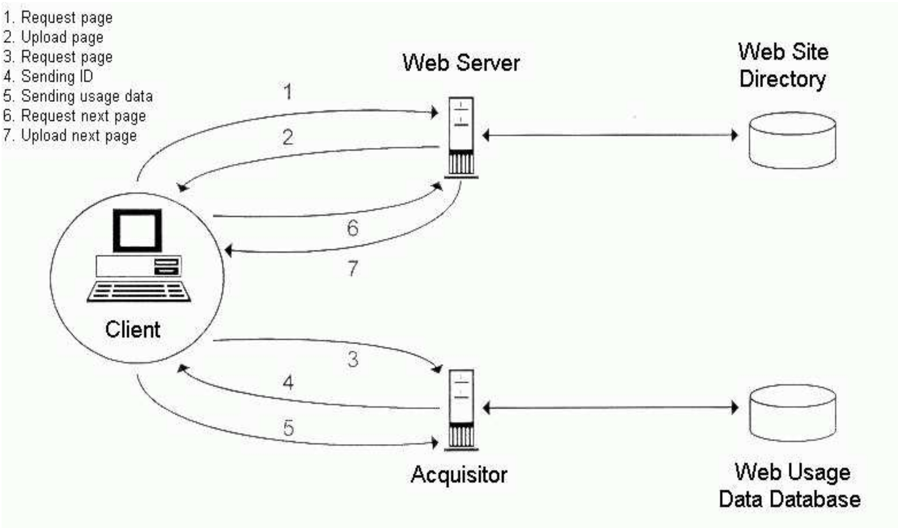


Figure 1: Mechanism of data acquisition

2.1 Remote Agent

The existence and activities of the remote acquisition agent should be transparent to users because users usually consider usage data acquisition process as an overhead to their actual Web browsing purpose. Specifically, the agent should be implemented as a light-weight piece of code[4] 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 [10], 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. [12] 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) [30], we prefer the Java applet technology because while it is well supported by all common browsers, it is designed to be secure and hence satisfies users' expectations of privacy [31]. 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* as we have described in [21] in details.

The complete code for the remote agent is included in the appendix of the paper. 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. 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

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

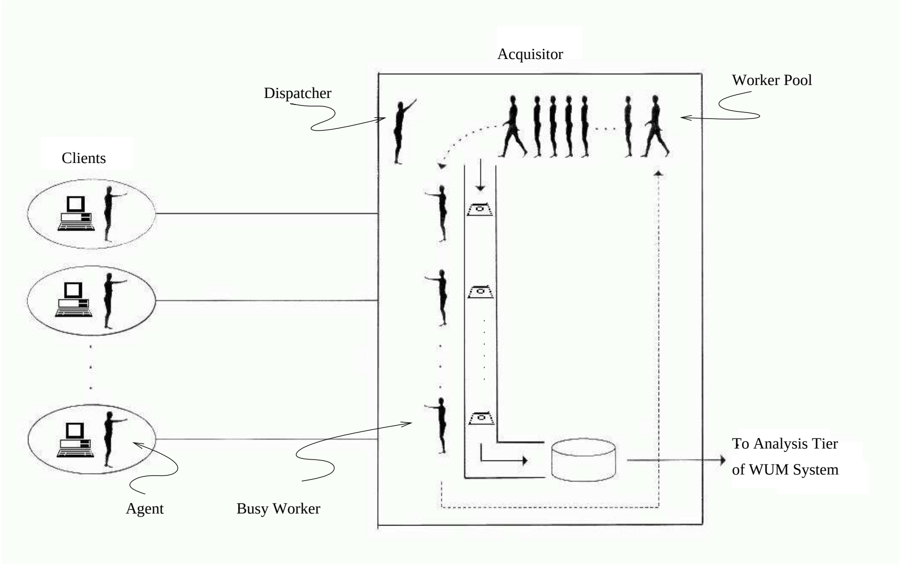


Figure 2: Architecture of the acquisitor (data acquisition server)

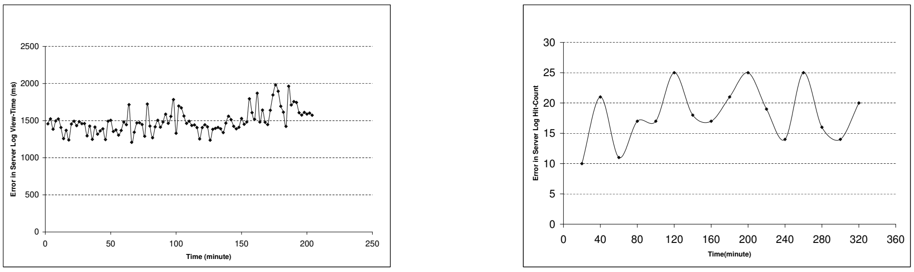
view-time and hit-count features of user navigation. However, the same procedure is applicable in acquiring other navigational features.

There are two drawbacks with this implementation. 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™ [32]. 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, including Apache™. We are currently studying other technologies that can be used to eliminate this problem.

2.2 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 [33]). This greatly facilitates usage data collection in such systems.



a. Server log error due to network time b. Server log error due to cache hits

Figure 3: Reliability of the server log usage data as compared to our system

3 Performance Evaluation

We conducted several experiments to verify the mechanisms employed in our system, and compared reliability of the usage data collected by our data acquisition 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. We performed some experiments to verify the reliability advantage of the data collected by our data acquisition system versus server log data.

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[5]. 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.

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

Results of this study are reported in Figure 3-a. In this figure, the X-axis is the time period of the experiment (in minutes) and the Y-axis is the average 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. [11] 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 3-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%.

4 Conclusions

Although we were among the first research groups advocating client-side web usage data acquisition [19], due to some contractual and logistic restrictions we could not release the details of our design and its performance evaluation results until now. Most of our publications only briefly discussed the advantages of client-side usage data acquisition in general. In this paper, we describe the detailed design of our reliable, efficient, and scalable Web usage data acquisition system for real-time Web Usage Mining (WUM). Our data acquisition system is:

1. *Reliable*, because it collects accurate data that do not incorporate network side-effects. We demonstrated that our system outperforms server log by improving the accuracy of the captured view-time and hit-count up to 13% and 40%, respectively.

2. *Efficient*, because it eliminates unnecessary preprocessing of the data such as user session identification and data cleansing; hence, allows real-time and anonymous WUM.

3. *Scalable*, because it implements a standard scalable architecture at the server side to be applicable with large-scale applications where volume of the data acquired by the WUM system is very large.

We hope that this paper with its detailed description of our design (including source code) as well as a performance comparison be useful to the community for efficient acquisition of accurate web usage data for efficacious WUM.

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

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and ITR-0082826, NASA/JPL contract nr. 961518, DARPA and USAF under agreement nr. F30602-99-1-0524, and unrestricted cash/equipment gifts from NCR, IBM, 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] Ansari S., R. Kohavi, L. Mason, Z. Zheng. 2000. *Integrating E-Commerce and Data Mining: Architecture and Challenges.* EC-Web 2000.

[3] Buchner A.G. and 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.

[4] Catledge L. and J. Pitkow. 1995. *Characterizing Browsing Behaviors on the World Wide Web.* Computer Networks and ISDN Systems, 27(6), 1995.

[5] 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.

[6] 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.

[7] 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.

[8] Greenspun P. 1999. *Philip and Alex's Guide to Web Publishing.* Chapter 9, User Tracking; ISBN: 1-55860-534-7.

[9] 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.

[10] Lieberman H. 1995. *Letizia: An Agent that Assists Web Browsing.* Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, August 1995.

[11] Leighton T. 2001. *The Challenges of Delivering Content on the Internet.* Keynote address in ACM SIGMETRICS 2001 Conference, Massachusetts, June 2001.

[12] 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.

[13] 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.

[14] 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.

[15] Mulvenna M.D., S.S. Anand, and A.G. Bchner. 2000. *Personalization on the Net using Web mining: Introduction.* CACM 43(8): 122-125, 2000.

[16] Perkowitz M., and O. Etzioni. 2000. *Toward adaptive Web sites: Conceptual framework and case study.* Artificial Intelligence 118, p.p 245-275, 2000.

[17] Pitkow J.E. 1997. *In Search of Reliable Usage Data on the WWW.* The Sixth International World Wide Web Conference, Santa Clara, California, 1997.

[18] Sarwar, B.M., G. Karypis, J.A. Konstan, and J. Riedl. 2000. *Analysis of Recommender Algorithms for E-Commerce.* ACM E-Commerce'00 Conference. October, 2000.

[19] Shahabi C., A. Zarkesh, J. Adibi, V. Shah. 1997.*Knowledge Discovery from Users Web-Page Navigation.* Proceedings of the IEEE RIDE97 Workshop, April, 1997.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] *http://www.personify.com*

[26] *http://www.websidestory.com*

[27] *http://www.bluemartini.com*

[28] *http://www.webtrends.com*

[29] *http://docs.yahoo.com/release634.html*

[30] *http://www.javascript.com*

[31] *http://java.sun.com/sfaq*

[32] *http://msdn.microsoft.com/remotescripting*

[33] *http://www.akamai.com*