LODAP: A LOg DAta Preprocessor for mining Web browsing patterns

G. CASTELLANO, A. M. FANELLI, M. A. TORSELLO
Department of Computer Science
University of Bari
Via Orabona, 4 - 70126 Bari
ITALY

Abstract: - In this paper, we present LODAP, a log data preprocessor which is able to extract user sessions starting from the requests stored in the log file of a Web site. LODAP is composed of several modules. A data cleaning module cleans the log file by removing useless records in order to retain only relevant requests encoding the user navigational behaviour. The data structuration module groups the remained requests in user sessions, by using a time-based method. Finally, the data filtering module considerably reduces the size of data concerning the extracted user sessions by deleting the least visited pages and the uninteresting sessions. In addition, a data summarization module creates reports which represent information summaries mined from the analyzed log file and containing the results provided by each module of LODAP. The implemented tool is characterized by a wizard-based interface which guides the analyst during the preprocessing of the log data through a sequence of " panels". Each panel is a graphical window which offers a basic functionality of the processor. Tests on the log files of a specific Web site show that the LODAP tool can effectively reduce the log dataset size and identify significant user sessions.

Key-Words: - Data cleaning, data filtering, data preprocessing, user sessions identification, Web mining, Web usage mining.

1 Introduction

The explosive growth of Web applications and online services makes the Web the most important source of information today. Since Web applications are becoming more complex in their structure and content, their use may often result difficult for no-expert users. Extracting useful patterns from browsing behavior of users in order to understand their preferences is becoming a fundamental facet in the development of adaptive Web sites. Accurate identification of user browsing patterns is particularly important in Web personalization not only to help the site's owner in improving its quality but also to adapt the content/structure of the site.

In the process of discovery and analysis of Knowledge from World Wide Web, Web mining covers a crucial role. Web Usage Mining (WUM) is a recent research area devoted to mine browsing patterns from usage data typically stored in Web log files on Web servers [5].

A WUM methodology typically includes three main steps: data preprocessing, pattern discovery and pattern analysis. Among these, data preprocessing is a very crucial step since the success of the next steps depends heavily on the results of this first task [7]. Indeed, once properly preprocessed, data can be expressed in a consistent manner useful to automatically derive the users' interests through a process of pattern discovery.

Briefly speaking, the aim of data preprocessing is to identify user sessions, encoding the navigational behavior of users, starting from the information contained in the Web log files. A Web log file contains requests made to the Web server in chronological order. According to the Common Log Format [1], a line of a log file contains: the client's host name or IP address, the request's date and time, the operation type (GET, POST, HEAD, and so on), the requested resource name (URL), a code indicating the status of the request, the size of the requested page (if the request is successful). As an example, a record in a log file appears as:

```
66.249.65.243 - - [26/Mar/2006:07:10:44
+0200]
"GET/gallery2/main.php?g2_view=core.DownloadI
tem&g2_itemId=5875 HTTP/1.1" 200 1745276
```

In order to develop a WUM methodology for dynamic link suggestion, in this paper we focus on the preprocessing of log data and identify three main steps:

- Data cleaning: log data are cleaned by removing irrelevant records (e.g. accesses to multimedia objects, robots' requests, etc.) so as to retain only information concerning accesses to visited Web pages.
- Data structuration: the selected requests are structured into user sessions. Each user session contains the sequence of pages visited by the same user during a certain time period.
- Data filtering: Web requests are further selected by retaining only URLs of the most visited pages.

To perform preprocessing, we designed and implemented a software tool, called LODAP (LOg DAta

Preprocessor) that takes as input log files related to a Web site and outputs a database containing some statistics about pages visited by users and the identified user sessions. A key feature of LODAP is the wizard-based interface that guides the user during the preprocessing of the log data. Tests on log files of a specific Web site show that the developed preprocessing tool can effectively reduce the log dataset size and identify user sessions which encode the user navigational behavior in a significant manner.

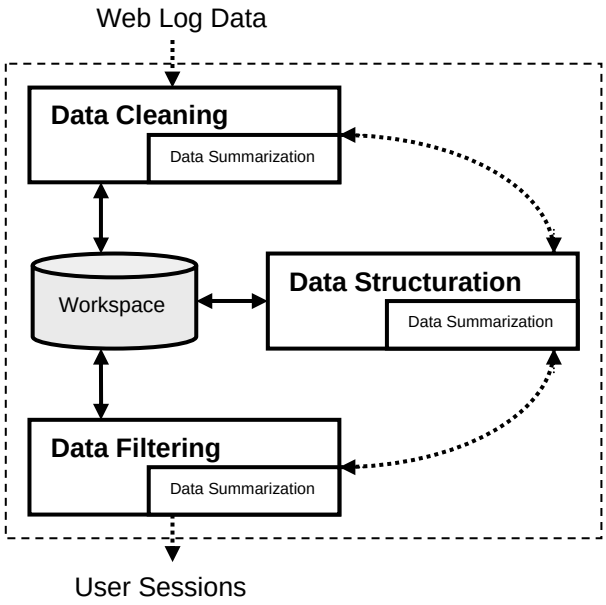

Fig. 1. Architecture of the log data preprocessing tool. Data flow (solid arrows) and control flow (dashed arrows) among LODAP panels are shown.

2 Architecture of LODAP

The architecture of the log data preprocessor (fig. 1) includes the three main modules corresponding to the steps of the data preprocessing process, namely data cleaning, data structuration and data filtering. In addition, a further module is included in the structure of LODAP, called data summarization. In particular:

- The data cleaning module removes redundant and useless records contained in the Web log file;
- The data structuration groups the remaining Web requests into user sessions;
- The data filtering module selects the most visited pages in the Web site;
- The data summarization provides reports containing information useful for the aims of the process of log file analysis. As illustrated in fig. 1, this component is employed in every module of the tool in order to illustrate the obtained information.

The tool has been designed to provide a visual framework for log data processing with a graphical "wizard"-based user interface that supports the analyst in carrying out the successive steps of log data preprocessing through a sequence of "panels". Each panel is a window that offers, through a graphical user interface, a basic functionality concerning a specific subtask of the preprocessing (e.g. data loading, data cleaning, etc.).

On the architectural level, panels are associated to procedural modules that are connected to a workspace, implemented as a Microsoft ACCESS database, wherein all transient information coming from each panel are collected and properly organized (fig. 1). The user decides when to pass from one panel to the successive, by pushing a "Next" button which is common to all panels. A "Back" button is also available in each panel, thus allowing a bi-directional control flow of the preprocessing. Information gathered into each panel is actually saved in the workspace, hence a backward step from a panel to its preceding does not cause information loss.

In the following, we describe in detail the functions of each module in LODAP.

2.1. Data cleaning

The data cleaning module is intended to clean Web log data by deleting irrelevant and useless records in order to retain only usage data that can be effectively exploited to recognize users' navigational behavior. Since Web log files record all user interactions, they represent a huge and noisy source of data, often comprising an high number of unnecessary records. By removing useless data, we can reduce the size of these files in order to use less storage space and to facilitate upcoming steps. Of course the choice of log data to be removed depends on the ultimate goal of the web personalization system. In our case, the goal is to develop a WUM system to offer personalized dynamic links to the site's visitors, hence the system has to keep only log data concerning explicit requests that actually represent users' actions. As a consequence, the data cleaning module has been developed to remove the following requests:

- *Requests with access method different from "GET".* Generally, requests containing a value different from "GET" in the field of the access method do not refer to explicit requests of users but they often concern with CGI accesses, properties of the Server, visits of robots, etc. Hence, these requests are considered non-significant and, consequently, they are removed from the log file.
- *Failed and corrupted requests.* These requests are represented by records containing a HTTP error code. A status with value of 200 represents a succeeded request. A status with value different from 200 represents a failed request (e.g. a status of 404 indicates that the requested file was not found at

the expected location). Also corrupted lines with missing values in some fields are eliminated in order to clean log files from incomplete information.

- *Requests for multimedia objects.* Due to the model underlying the HTTP protocol, a separate access request is executed for every file, image, multimedia object embedded in the requested Web page. As a consequence, a single request for a Web page may often produce several log entries that correspond to files automatically downloaded without an explicit request of the same user. Requests for such type of file can be easily identified since they contain a particular URL name suffix, such as gif, jpeg, jpg, and so on. Whether to keep or remove requests for multimedia objects depends on the kind of Web site to be personalized and on the purpose of the WUM system. In general, these requests do not represent the effective browser activity of the user visiting the site, hence they are deemed redundant and are removed. In other cases, eliminating requests for multimedia objects may cause a loss of useful information. The decision upon retaining or removing these entries is left to the analyst, who can select the suffixes to be removed in a panel of the data cleaning module (see fig. 2).
- *Requests originated by Web robots.* Log files may contain a number of records corresponding to requests originated by Web robots. Web robots (also known as Web crawlers or Web spiders) are programs that automatically download complete Web sites by following every hyperlink on every page within the site in order to update the index of search engine. Requests created by Web robots are not considered usage data and, consequently, have to be removed. To identify web robots' requests, the data cleaning module implements two different heuristics. Firstly, all records containing the name "robots.txt" in the requested resource name (URL) are identified and straightly removed. The second heuristic is based on the fact that the crawlers retrieve pages in an automatic and exhaustive manner, so they are characterized by a very high browsing speed (intended as total number of pages visited/total time spent to visit those pages). Hence, for each different IP address we calculate the browsing speed and all requests with this value exceeding a threshold (pages/second) are regarded as made by robots and are consequently removed. The value of the threshold is established by analyzing the browser behaviour arising from the considered log files.

After data cleaning, only requests for relevant resources are kept in the database. We formally define $R = \{r_1, r_2, ..., r_{n_R}\}$ as the set of all distinct resources requested from the Web site under analysis.

2.2 Data structuration

This module groups the unstructured requests remaining in the log data into user sessions. A user session is defined as a limited set of resources accessed by the same user within a particular visit. Identifying user sessions from the log data is a difficult task because many users may use the same computer and the same user may use different computers. Hence, one main problem is how to identify the user. For Web sites requiring user registration, the log file contains the user login that can be used for user identification. When the user login is not available, we simply identify a user from the IP address, i.e. we consider each IP address as a different user (being aware that an IP address might be used by several users). We define $U = \{u_1, u_2, ..., u_{n_U}\}$ as the set of all the users (IP) that have accessed that web site.

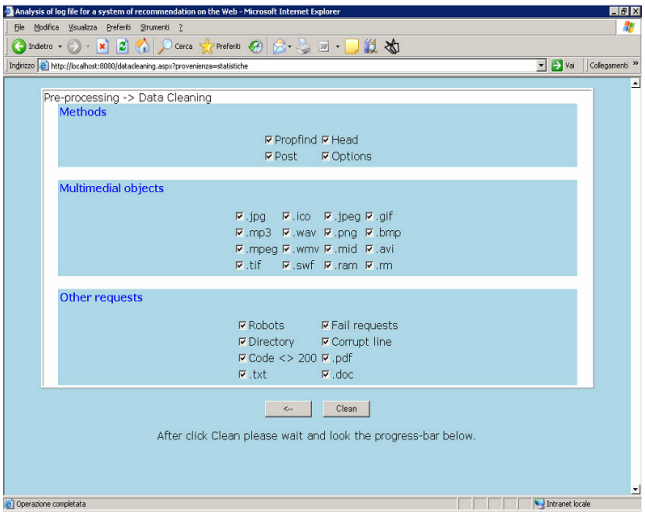

Fig. 2. A panel of the Data Cleaning module.

According to other approaches proposed in the literature [3], [4], we exploit a time-based method to identify sessions. Precisely, as in [2] and [6], we consider a user session as the set of accesses originating from the same user within a predefined time period. Such time period is defined by considering a maximum elapsed time $\Delta t_{\max}$ between two consecutive accesses. Moreover, to better handle particular situations which might occur (such as users accessing several times to the same page due to slow connections or intense network traffic), a minimum elapsed time $\Delta t_{\min}$ between two consecutive accesses is also fixed. We define a user session as a triple $\mathbf{s}^{(i)} = \langle u^{(i)}, t^{(i)}, \mathbf{r}^{(i)} \rangle$ where $u^{(i)} \in U$ represents the user identifier, $t^{(i)}$ is the access time of the whole session, $\mathbf{r}^{(i)}$ is the set of all resources (with corresponding access time) requested during the *i*-th session, namely $\mathbf{r}^{(i)} = \langle (t_1^i, r_1^i), (t_2^i, r_2^i), ..., (t_{n_i}^i, r_{n_i}^i) \rangle$ with $r_j^i \in R$, where the

access time t_k^i to a single resource satisfies the following:

$$t_{k+1}^i \geq t_k^i \text{ and } \Delta t_{\min} < t_{k+1}^i - t_k^i < \Delta t_{\max} .$$

Summarizing, after the data structuration phase, a collection of n_S sessions $\mathbf{s}^{(i)}$ is identified from the log data. We denote the set of all identified sessions by $S = \left\langle \mathbf{s}^{(1)}, \mathbf{s}^{(2)}, ..., \mathbf{s}^{(n_S)} \right\rangle$.

Once all the sessions have been identified, the data structuration module presents a panel that lists the extracted sessions and enables the analyst to visualize and eventually save the details (IP address, requested resources in the session, date and time of the requests) of each user session (see fig. 3).

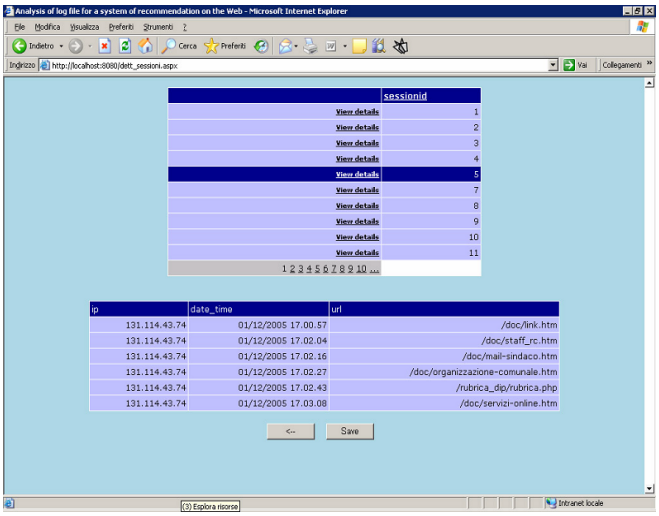

Fig. 3. A panel of the Data Structuration module.

2.3 Data filtering

After the identification of user sessions, LODAP performs a data filtering step to leave out less requested resources and retain only the most requested ones. More precisely, the aim of the data filtering module is to remove the least requested resources in order to filter only the resources requested in most of the identified user sessions. For each resource r_i, we consider the number NS_i of different sessions that required r_i and compute the quantity $NS = \max_{i=1...n_R} NS_i$. Then, we define a threshold ε whose value is a low percentage of NS and remove each request with $NS_i < \varepsilon$. In this way, the data filtering module can considerably reduce the number of relevant requested resources, thus providing a volume of data that can be easily managed in the next steps of usage mining.

Besides removing the entries corresponding to the least requested resources, the data filtering module eliminates all the user sessions that comprise only less requested resources. In this way, the size of data is even more reduced.

2.3 Data summarization

Each module of the LODAP tool provides a set of statistics that summarize the results of the performed pre-processing step. To do this, each module makes use of a sub-module, called data summarization, that generates reports summarizing the information obtained after the application of pre-processing step. These statistical information permit to obtain a schematic and concise description of the usage data mined from the analyzed log file. Precisely, the created statistical reports provide the necessary information to detect some particular aspects related to the user browsing behavior or to the traffic of the considered site (such as how many images, videos, etc are downloaded; the volume of the requests made; how many requests are generated by robots, etc).

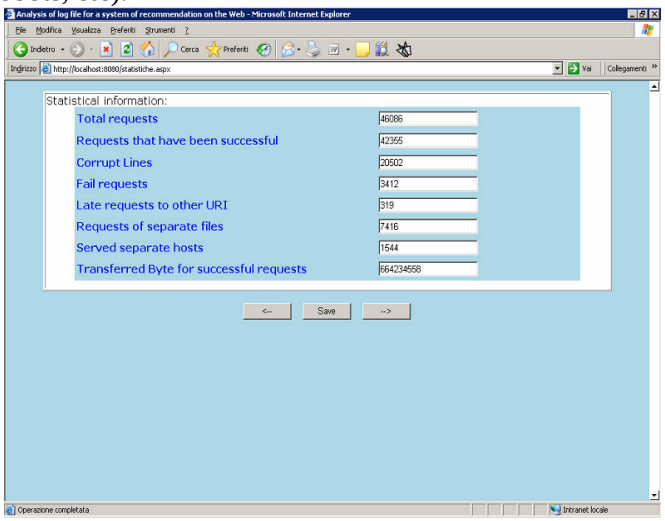

Fig. 4. The first information summary.

Each time the data summarization sub-module is invoked by other modules, it accesses to the data stored in the database representing the workspace of LODAP and creates a specific summary. A preliminary summary is generated as soon as the log data are loaded into the pre-processor and contains information about the total number of requests of the analyzed log file, the number of the satisfied requests, the number of failed or corrupt requests, the volume of transferred bytes, ecc. (fig. 4).

Another report (fig. 5), generated when the data cleaning is completed, contains information as the number of requests having methods different from GET, the number of requests corresponding to multimedia objects (images, videos, sounds, ecc), the number of visits made by robots, ecc, are included.

When the data structuration phase is concluded, another report is created that contains the information about the

number of the extracted user sessions and the details of each session.
Finally, a conclusive report is generated when the data filtering is performed and the entire process of log file analysis ends. This report contains information about the number of requests contained in log file before the data filtering step, the number of requests contained in log file after the data filtering step, the number of the deleted requests through the data filtering step, the percentage of deleted requests referred to the initial number, etc.

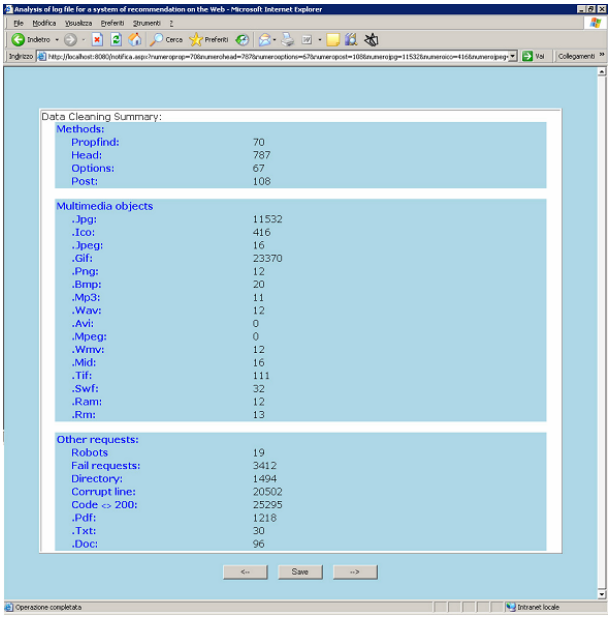

Fig. 5. The Data Cleaning summary.

3 Simulation results

The developed log data preprocessor was tested on log files stored by the Web site Server of the town of Pisa (Italy) available at the URL www.comune.pisa.it. In particular, we considered portions of log files containing the requests received by the server during the same time interval of three hours (from 4:00 pm to 7:00 pm) of three different days, as indicated in table 1.

Table 1. The considered Web log files.

Log file	Date	Size (KB)
A	12/01/2005	5745
B	12/02/2005	5089
C	12/03/2005	3554

The results of the preliminary analysis of these log files are reported in table 2, which shows the information provided by the data summarization module as soon as the log data are loaded into the database. In the following sections we describe the information mined from the examined log files by the application of each module of LODAP.

Table 2. The first statistical report.

Statistics	Log file A	Log file B	Log file C
Total Requests	59062	53543	36968
Satisfied Requests	53839	49467	33200
Corrupt Requests	24151	27171	6377
Failed Requests	4820	3870	3657
Requests to files	9220	6695	5399
Requests to other resources	403	206	111
Served hosts	2134	1699	1112
Transferred KBytes	903196	735518	484314

Table 3. Information extracted in the data cleaning module

Request category	Log file A	Log file B	Log file C
Methods			
Propfind	111	106	48
Head	965	926	899
Options	102	109	34
Post	118	91	13
Multimedia Objects			
.jpg	14633	13061	8990
.ico	669	351	1947
.jpeg	18	13	13
.gif	29338	29419	19602
.png	14	16	11
.bmp	21	16	7
.mp3	17	15	7
.wav	12	4	0
.avi	0	0	0
.mpeg	0	0	0
.wmv	24	5	8
.mid	25	11	5
.tif	127	51	8
.swf	53	78	8
.ram	17	6	10
.rm	16	6	6
Other Requests			
Robots	23	46	44
Failed requests	4820	3870	3657
Directory	1877	1028	524
Corrupt lines	24151	27171	6377
Status Code <> 200	30757	32446	10600
.pdf	1757	1160	659
.txt	42	49	49
.doc	135	111	49
Summary			
Cleaned log requests	1629	1263	2537
% initial size	2.76%	2.36%	6.86%

3.1 Data cleaning

The data cleaning module analyzes the log file in order to individuate the number of requests deemed irrelevant because don't represent explicit requests of users (such as requests for multimedia objects, accesses with methods different from "GET", requests made by robots, etc). As explained before, the results of data cleaning depend on the choices made by the analyst when using

the wizard-based tool. In our experiments, we decided to remove all possible categories of irrelevant requests (described in section 2.1).

To identify requests made by web crawlers, we set the threshold related to the browsing speed to 0.5 (which means an average access time to a resource equal to 2sec). The information obtained after the cleaning of the three considered log files are illustrated in table 3. We point out that an overlap may occurs between two categories of identified requests. For example, a visit with method "Head" may be also a request for a multimedia object. In these cases, the data summarization module counts twice the removal of the request, even though only one record is deleted from the log file.

The last two lines of the table 3 show, respectively, the number of remained requests after data cleaning and the size of the cleaned log file expressed in terms of percentage referred to the initial size. It can be seen that for all the considered log file, the data cleaning module can effectively reduce the size of the log data to a small percentage of the initial size. This is especially true for the log file A and B, which contain a higher number of corrupt requests and requests with code <>200 in comparison to log file C.

3.2 Data structuration and data filtering

After data cleaning, the three log files were processed by the data structuration module in order to identify user sessions. For the considered log files, we did not have user login information, so we used only the IP address to identify distinct users. For user session identification, we set the maximum value $\Delta t_{\max}$ for the elapsed time between two consecutive requests to 30 minutes and the minimum value $\Delta t_{\min}$ to 2 seconds. The number of extracted user sessions for each examined log file can be seen in table 4.

After data structuration, the application of the data filtering module reduced further the size of Web log data by deleting the requests related to the less accessed resources. The elimination of these requests was made by setting the threshold ε to 10%. The results of this process are summarized in table 4. It can be seen that from the three analyzed log files a number of 96, 56 and 112 significant requests was identified, from what we determined 198, 136 and 118 user sessions, respectively.

Table 4. Data structuration and data filtering summary

	Log file A	**Log file B**	**Log file C**
User sessions	198	136	118
Requests before data cleaning	143	105	188
Requests after data cleaning	96	56	112
Deleted requests	47	49	76
% Deleted requests	32.87%	46.67%	40.43%

4 Conclusion

The information obtained with our experiments show the effectiveness of the LODAP tool, not only in reducing considerably the size of Web log files but also in grouping Web requests into a number of user sessions which can encode the user browsing behavior in a significant manner.

How to describe the preferences of users on the base of their navigational behavior is a problem that we are going to face in future works. Indeed, once user sessions have been identified, they can be used to understand the preferences of each user and derive the degree of interest that each user shows for a Web resource. Several measures and/or heuristics can be applied to obtain the degree of interest for a Web resource.

A possibility is to consider the degree of interest to a resource as strictly related to the frequency of accesses to that resource (number of accesses to that resource / total number of accesses during the session) and to the time the user spends on the same one. We defer this facet in future works.

References:

[1] http://www.w3.org/Daemon/User/Config/Loggin.htm#common-logfile-format.

[2] O. Nasraoui, World Wide Web Personalization, In J. Wang (ed), *Encyclopedia of Data Mining and Data Warehousing*, Idea Group, 2005.

[3] G. Paliouras, C. Papatheodoru, V. Karkaletsis, P. Tzitziras, C. D. Spyropoulos, Large-scale mining of usage data on Web sites, *AAAI Spring Symposium on Adaptive User Interface*, Stanford, California, pp. 92-97, 2000.

[4] J. Pei, J. Han, B. Motazavi-Asl, H. Zhu, Mining access patterns efficiently from web logs, *in: Paci.c-Asia Conference on Knowledge Discovery and Data Mining*, pp. 396–407, 2000.

[5] D. Pierrakos, G. Paliouras, C. Papatheodorou, and C. D. Spyropoulos, Web usage mining as a tool for personalization: a survey. *User Modeling and User-Adapted Interaction*, Vol. 13, No. 4, pp. 311-372, 2003.

[6] B.S. Suryavanshi, N. Shiri, and S.P. Mudur, A Fuzzy Hybrid Collaborative Filtering Technique for Web Personalization, in *Proc. of 3rd Workshop on Intelligent Techniques for Web Personalisation (ITWP'05)*, in conjunction with the *19th International Joint Conference on Artificial Intelligence (IJCAI05)*, Edinburg, Scotland, UK, 2005.

[7] Tanasa D. and Trousse B., Advanced Data Preprocessing for Intersites Web Usage Mining. *In IEEE Intelligent Systems*, 19(2), pp. 59-65, 2004.