Knowledge Discovery from Web Usage Data: Complete Preprocessing Methodology

G T Raju[1] and P S Satyanarayana[2]

[1]Asst. Prof. Department of CSE, [2]Professor, Department of E&C
B.M.S. College of Engineering, Bangalore
Visvesvaraya Technological University
Karnataka, INDIA

Summary

The exponential growth of the Web in terms of Web sites and their users during the last decade has generated huge amount of data related to the user's interactions with the Web sites. This data is recorded in the Web access log files of Web servers and usually referred as Web Usage Data (WUD). Knowledge Discovery from Web Usage Data (KDWUD) is that area of Web mining deals with the application of data mining techniques to extract interesting knowledge from the WUD. As Web sites continue to grow in size and complexity, the results of KDWUD have become very critical for efficient and effective management of the activities related to: e-business, e-education, e-commerce, personalization, website design & management, network traffic analysis, the cache, the proxies, great diversity of Web pages in a site, search engine's complexity, and to predict user's actions. In this paper, we propose a complete preprocessing methodology, one of the important steps in KDWUD process. Several heuristics have been proposed for cleaning the WUD which is then aggregated and recorded in the relational data model. To validate the efficiency of the proposed preprocessing methodology, several experiments were conducted and the results shows that the proposed methodology reduces the size of Web access log files down to 73-82% of the initial size and offer richer logs that are structured for further stages of KDWUD.

Keywords

Preprocessing, Knowledge Discovery, Web Usage Data, Web Usage Mining.

1. Introduction

Web log data is usually diverse and voluminous. This data must be assembled into a consistent, integrated and comprehensive view, in order to be used for pattern discovery. As in most data mining applications, data preprocessing involves removing and filtering redundant and irrelevant data, removing noise, transforming and resolving any inconsistencies. Data preprocessing has a fundamental role in KDWUD applications. A significant problem with most of the pattern discovery methods is that, their difficulty in handling very large scales of WUD. Despite the fact that, most of the KDWUD processes done off-line, the size of WUD is in the orders of magnitude larger than those met in common applications of machine learning. Rushing to analyze usage data without a proper preprocessing method will lead to poor results or even to failure. Preprocessing methodology has not received enough analysis efforts. Hence managing the quantity of data that is continuously increasing, and the great diversity of pages on Web site has become critical for KDWUD applications.

In this paper, we propose a complete preprocessing methodology that allows the analyst to transform any collection of web server log files into structured collection of tables in relational database model. The log files from different Web sites of the same organization are merged to apprehend the behaviors of the users that navigate in a transparent way. Afterwards, this file is cleaned by removing all unnecessary requests, such as implicit requests for the objects embedded in the Web pages and the requests generated by non-human clients of the Web site (i.e. Web robots). Then, the remaining requests are grouped by user, user sessions, page views, and visits. Finally, the cleaned and transformed collections of requests are saved onto a relational database model. We have provided filters to filter the unwanted, irrelevant, and unused data. Analyst can select the log files from different Web servers and decide what entries he/she is interested (HTML, PDF, and TXT). The objective is to considerably reduce the large quantity of Web usage data available and, at the same time, to increase its quality by structuring it and providing additional aggregated variables for the data mining analysis that follow.

Manuscript received January 3, 2008
Manuscript revised January 20, 2008

The rest of the paper is organized as follows. The problem of data preprocessing is formulated in section 2. Overviews of the data preprocessing steps are explained in section 3. The experimental results and observations are provided in section 4. Related work is given in section 5. Finally, in section 6, we conclude and provide some perspectives in KDWUD preprocessing.

2. Problem Formulation

Consider the set R = $\{r_1, r_2 \ldots, r_n\}$ of all Web resources from a Web site. If U = $\{u_1, u_2 \ldots, u_m\}$ is the set of all the users who have accessed that site, then the log entry is defined as $l_i = < u_i, t, s, r_i, ref_i >$, where $u_i \in U$; $r_i \in R$; t represents the access time, s represents the request status and ref_i represents the referring page. ref_i is optional as in some Web log formats like the CLF where the referring page is not recorded. Here, 's' is a three-digit code indicating the request's success or failure. In the later case, it also indicates the cause of the failure. A status with a value of s = 200 represents a succeeded request; while a status of s = 404 shows that the requested file was not found at the expected location. $L_i = \{l_{i1}, l_{i2} \ldots, l_{in}\}$ arranged in the ascending order, constitutes a Web server log. In the case of N Web servers, the set of log files is $Log = \{L_1, L_{2 \ldots,} L_N\}$. Using these notations, the preprocessing problem is formulated as follows – "Given a set of log files *Log* of Web site – extract the Users, User sessions, visits, and page views of the Web site users with a given time interval Δt".

3. Overview of Data Preprocessing

Fig. 1 shows the overview of data preprocessing in KDWUD. It comprises of the following steps – Merging of Log files from Different Web Servers, Data cleaning, Identification of Users, Sessions, and Visits, Data formatting and Summarization.

3.1 Merging

At the beginning of the data preprocessing, the requests from all log files in *Log* put together into a joint log file '£' with the Web server name to distinguish between requests made to different Web servers and taking into account the synchronization of Web server clocks including time zone differences. Fig. 2 shows the pseudo code for this task. For privacy reasons, we anonymize the resulting log file '£' since the host names or the IP addresses need to be removed when sharing log files or publishing results.

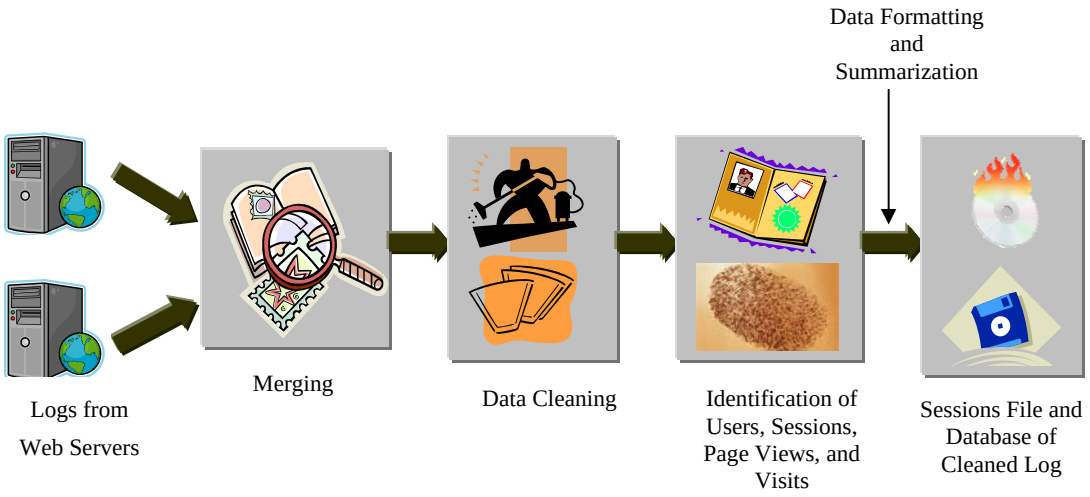

Fig.1 Overview of Data Preprocessing in KDWUD

Therefore, the original host name is replaced with an identifier that keeps the information about the domain extension (i.e. the country code or organization type, such as .com, .edu, or .org). The merging problem is formulated as "Given the set of log files *Log* = {L_1, L_2, L_3...L_n}, merge these log files into a single log file *£ (joint log file)*". Let L_i be the i^{th} log file. Let $L_{i.c}$ is a cursor on L_i's requests and $L_{i.l}$ is the current log entry from L_i indicated by $L_{i.c.}$ Let $L_{i.l.time}$ be the time t of the current log entry of $L_{i.}$ Let S=(w_1,w_2,...w_n) is an array with Web server names, where S[i] is the Web server's name for the log $L_{i.l}$.

Steps:

1. *Initialize the joint log file* £ *cursor*
2. *Scan the log entries from each log file* L_i *in* Log *and append to* £
3. *Sort the* £ *entries in ascending order based on access time*
4. *Return* £

```
Merge_logs()
{
  // Takes a set of log files Log as input and returns the single log
file £
      containing the entries from all the log files.
 //Open the file £ and initialize the cursor.
   Open (£); £.c = 1;
 // Process each log file Li
   For each log file Li in Log do
      {
       Open (Li); Li.c =1;
                  // open the log file Li and initialize cursor
       While (not EOF (Li)) do
         {
          Li.l=fetch (Li.c);   // fetch an entry from Li
          Li.l' = Li.l + S[i]; // add server name to the entry l
          £=£ ∪ Li.l';        // append
               £.c = £.c + 1;      // Advance the cursor of £
           Li.c = Li.c + 1;   // adnace the cursor of Li
         }
         Close(Li)
      }
   Close(£)
   Sort the log file £ entries in ascending order by the access time t
   Return (£)
 }
```

Fig. 2 Pseudo code for Merging of log files

3.2 Data Cleaning

The second step of data preprocessing consists of removing useless requests from the log files. Since all the log entries are not valid, we need to eliminate the irrelevant entries. Usually, this process removes requests concerning non-analyzed resources such as images, multimedia files, and page style files. For example, requests for graphical page content (*.jpg & *.gif images) and requests for any other file which might be included into a web page or even navigation sessions performed by robots and web spiders. By filtering out useless data, we can reduce the log file size to use less storage space and to facilitate upcoming tasks. For example, by filtering out image requests, the size of Web server log files reduced to less than 50% of their original size. Thus, data cleaning includes the elimination of irrelevant entries like:

- Requests executed by automated programs, such as web robots, spiders and crawlers; these programs generate the traffic to web sites, can dramatically bias the site statistics, and are also not the desired category which KDWUD investigates.
- Requests for image files associated with requests for particular pages; an user's request to view a particular page often results in several log entries because that page includes other graphics, while we are only interested in what the users explicitly request, which are usually text files.
- Entries with unsuccessful HTTP status codes; HTTP status codes are used to indicate the success or failure of a requested event, and we only consider successful entries with codes between 200 and 299.
- Entries with request methods except GET and POST.

3.3 Identification

This step groups the unstructured requests of a log file by user, user session, page view and visits. At the end of this step, the log file will be a set of transactions (user sessions or visits).

3.3.1 User

In most cases, the log file provides only the computer address (name or IP) and the user agent (for the ECLF [2] log files). For Web sites requiring user registration, the log file also contains the user login (as the third record in a log entry) that can be used for the user identification. When the user login is not available, each IP is considered as a user, although it is a fact that an IP address can be used by several users. For KDWUD, to get knowledge about each user's identity is not necessary. However, a mechanism to distinguish different users is still required for analyzing user access behavior. Since users are treated as anonymous in most Web servers, in this paper, the *reactive strategy* is applied to Web logs and users are approximated in terms of IP address, type of OS and browsing software.

3.3.2 User Session

Identifying the user sessions from the log file is not a simple task due to proxy servers, dynamic addresses, and

cases where multiple users access the same computer (at a library, Internet cafe, etc.) or one user uses multiple browsers or computers. A user session is defined as a sequence of requests made by a single user over a certain navigation period and a user may have a single (or multiple) session(s) during a period of time. Session identification is the process of segmenting the access log of each user into individual access sessions [5]. Two time-oriented heuristic methods: *session-duration* based method and *page-stay-time* based method have been specifically proposed by [6, 7, and 8] for session identification. In this paper, we use the *timeout threshold* in order to define the users' sessions.

Fig. 3 summarizes the session identification process with a pseudo code. It checks to see the session has timed-out or if the referring file is not present in any of the open session histories. If so, a new session is opened. Since the *Log* has already been sorted by IP address/Agent, all of the open session histories are potential candidates for the page file access being processed. '***Session_Gen'*** function calls the '***Distance'*** function that finds the history that most recently accessed page file *f*.

3.3.3 Page View

The page view identification step determines which page file requests are part of the same page view and what content was served. This is necessary to provide meaningful results in the pattern analysis phase. If this step is not performed, the discovered patterns can be dominated by page files that make up a single popular page view. Page views are identified by using the time of the request. For requests made at the same time, only the first request (as ordered in the log file) is retained and the following ones are discarded.

Session_Gen ()

{ Let $H_i=\{f_1,f_2,\ldots, f_n\}$ denote the time-ordered session history

Let l_j, f_j, r_j, and t_j denote log entry, request, referrer, and time (at which the request was received) respectively.

Let τ denote the session time-out (Usually 30 minutes)

Sort the Log data by IP address, agent, and time.

for each unique IP/agent combinations do {

for each l_j do {

if (($(t_j-t_{j-1}) > \tau$) or ($r_j \notin \{H_0,H_1,\ldots, H_m\}$))

then increment i, add l_j to H_i

else assign = **Distance** (H, r_j), add l_j to H_{assign}

}

}}

Fig. 3 Pseudo code for Session Identification

After the page view identification, the log file will contain, normally, only one request for each user action. Each session must start with a seed page view, i.e. an initial page file or set of page files from which all subsequent page views will be derived. In the vast majority cases, the seed page view is made up of a single file, or starts with a single file that defines the frame structure and immediately causes additional page files to be requested. It is very rare for an unrelated site to link to more than one page file of a different site in a single hypertext link. However, it is possible, and for these cases all of the page files contributing to the seed page view would have to be explicitly entered into the algorithm.

3.4 Data Formatting and Summarization

This is the last step of data preprocessing. Here, the structured file containing sessions and visits are transformed to a relational database model. Then, the data generalization method is applied at the request level and aggregated for visits and user sessions to completely fill in the database. Two tables are designed in the relational database model – one for storing the log data and the other for storing the sessions. The data summarization concerns with the computation of aggregated variables at different abstraction levels (e.g. request, visit, and user session). These aggregated variables are later used in the data mining step. They represent statistical values that characterize the objects analyzed. For instance, if the object analyzed is a **user session**, in the aggregated data computation process, the following variables are calculated:

- The number of visits for that session
- The length of the session in seconds (the difference between the last and the first date of the visit) or in pages viewed (the total number of page views)
- The number of visits for the period considered, which can be a day, a week, or a month

If the object analyzed is a **visit**, then we propose to compute the following variables:

- The length of the visit in terms of time and page views
- The repetition factor of the visit
- The percentage of successful requests
- The average time spent on a page by the ser during the visit

Similarly, other aggregated variables that can be computed are:

- The percentage of the requests made to each Web server.
- Number of unique visitors/hosts per hr/day/week/month

- Number of unique user agents per hr/day/week/month

Depending on the objective of the analysis, the analyst can decide to compute additional and more complex variables.

4 Experimental Results

Several experiments have been conducted on log files collected from NASA Web site during July 1995. Through these experiments, we show that our preprocessing methodology reduces significantly the size of the initial log files by eliminating unnecessary requests and increases their quality through better structuring. This is shown in Table 1. It is observed from the Table 1 that, the size of the log file is reduced to 73-82% of the initial size. The Fig. 4 shows the GUI of our toolbox with preprocessor tab. Unique image/CSS related statistics are presented in the figures Fig. 5. Day wise unique visitors' statistics are presented in the Fig 6. Page views statistics are presented in the Fig 7. The user sessions are shown in the Table 2.

5 Related Work

After illustrating the use of our preprocessing methodology through different processes, we present in this section the main related works in this domain. In the recent years, there has been much research on Web usage mining [3,4,5,6,7,8,9,10,11,12,13,14,15,16]. However, as described below, data preprocessing in KDWUD has received far less attention than it deserves. Methods for user identification, sessionizing, page view identification, path completion, and episode identification are presented in [3]. However, some of the heuristics proposed are not appropriate for larger and more complex Web sites. For example, they propose to use the site topology in conjunction with the ECLF [2] file for what they call the user identification. The proposed heuristic aims to distinguish between users with the same IP address, OS and Browser by checking every page requested in a chronological order. If a page requested is not referred by any previous page requested, then it belongs to a new user session. The drawback of this approach is that it considers only one way of navigating in a Web site, by following links. However, in order to change the current page, the users can, for instance, type the new URL in the address bar (most browsers have an auto completion feature that facilitates this function) or they can select it from their bookmarks. In another work [15], the authors compared time-based and referrer-based heuristics for visit reconstruction. They found out that a heuristic's appropriateness depends on the design of the Web site (i.e. whether the site is frame-based or frame-free) and on the length of the visits (the referrer-based heuristic performs better for shorter visits). In [16], Marquardt et al. addressed the application of WUM in the e-learning area with a focus on the preprocessing phase. In this context, they redefined the notion of visit from the e-learning point of view. In their approach, a learning session, visit in our case, can span over several days if this period corresponds to a given learning period. As shown above, the preprocessing step is important and should be present in all WUM analysis. Therefore, we compared our preprocessing methodology with the preprocessing described in other general WUM research works [3, 9, 10, 11, 12, 15 and 16]. The results of this comparison are provided in Table 3. The comparison Table 3 focuses only on the data preprocessing step and it shows how different preprocessing features were implemented in the main related works.

Table 1. Results after Preprocessing

Website	Duration	Original Size	Size after Preprocessing	% Reduction in Size	No. of Sessions	No. of Users
NASA	1-10th Aug 95	75361 bytes (7.6MB)	20362 bytes	72.98%	6821	5421
NASA	20-24th July 95	205532 bytes (20.6MB)	57092 bytes	72.22%	16810	12525
Academic Site	12-28th May 2001	28972 bytes (2.9MB)	5043 bytes	82.5%	1645	936

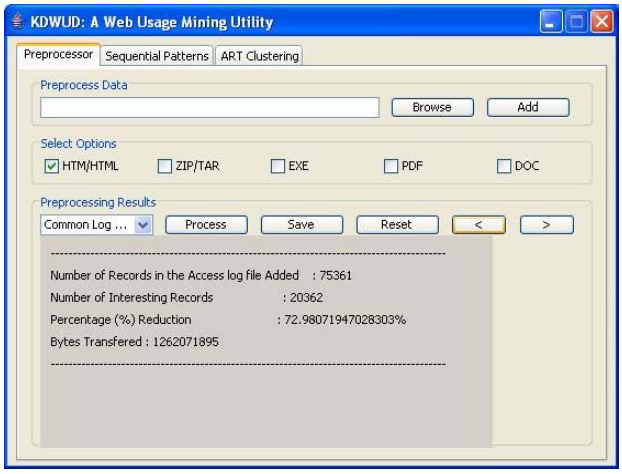

Fig. 4 KDWUD toolbox: Results after preprocessing (NASA Log file, Aug 1995)

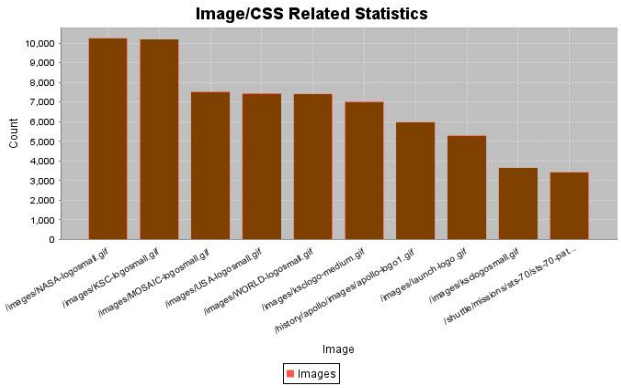

Fig. 5 Unique Image/CSS related statistics (NASA Log file, July 1995)

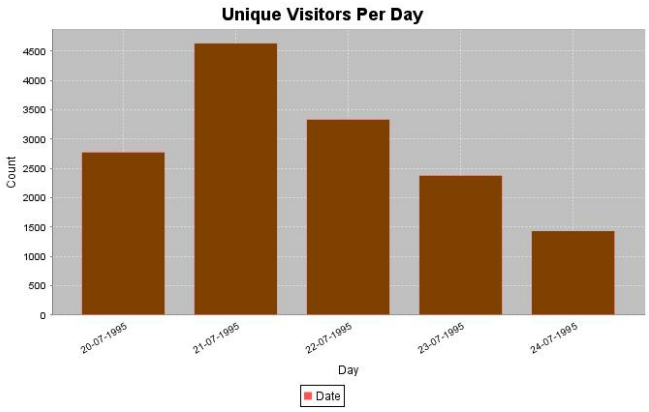

Fig. 6 Unique Visitors (NASA Log file, July 1995)

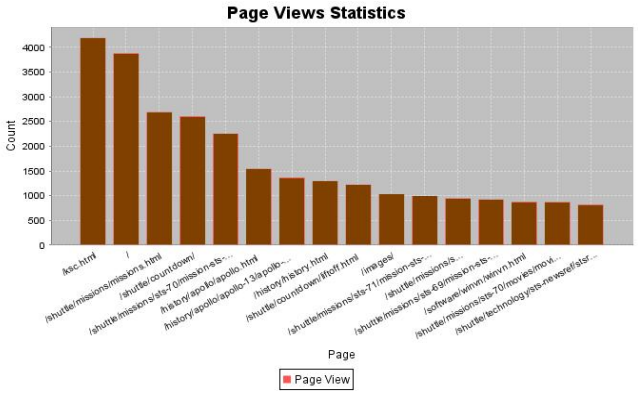

Fig. 7 Page Views statistics (NASA Log file, July 1995)

Table 2. User Sessions

Session Id	IP Address	Date &Time	URL Accessed
9	128.102.204.243	1995-07-22 01:16:58	/shuttle/missions/sts-73/mission-sts-73.html
9	128.102.204.243	1995-07-22 01:17:25	/shuttle/missions/sts-74/mission-sts-74.html
9	128.102.204.243	1995-07-22 01:17:38	/shuttle/missions/sts-72/mission-sts-72.html
9	128.102.204.243	1995-07-22 01:17:45	/shuttle/missions/sts-75/mission-sts-75.html
9	128.102.204.243	1995-07-22 01:17:52	/shuttle/missions/sts-76/mission-sts-76.html
9	128.102.204.243	1995-07-22 01:17:58	/shuttle/missions/sts-77/mission-sts-77.html
9	128.102.204.243	1995-07-22 01:18:05	/shuttle/missions/sts-78/mission-sts-78.html
10	128.102.210.40	1995-07-20 23:27:49	/shuttle/countdown/countdown.html
10	128.102.210.40	1995-07-20 23:28:11	/shuttle/technology/sts-newsref/stsref-toc.html
10	128.102.210.40	1995-07-20 23:28:57	/shuttle/technology/sts-newsref/sts_mes.html
10	128.102.210.40	1995-07-20 23:29:11	/shuttle/countdown/liftoff.html
10	128.102.210.40	1995-07-20 23:30:18	/shuttle/missions/sts-69/mission-sts-69.html
11	128.102.210.40	1995-07-21 01:58:47	/shuttle/countdown/countdown.html
11	128.102.210.40	1995-07-21 01:59:12	/shuttle/countdown/liftoff.html

Table 3. Comparison of Preprocessing Methodologies

Related work	Data Source			Data Cleaning				Data Formatting & Structuring			
	Site Map	Site Semantic	Multiple Servers	Merging	Anonym -ization	Removing Images	Removing Web Robots	User ID	Session ID	Visit ID	Generalization & Aggregation
Berendt [15]						√			IP	H_{Visit}	
Chen [12]	√										
Cooley [3]	√				√	√	√	Login	IP, Agent	H_{Visit} H_{Ref}	
Fu [11]						√			IP	H_{Visit}	√
Krishnapuram [9]						√			IP		
Shahabi [10]	√				√				IP, Session ID	H_{Visit} H_{Page}	
Becker[16]						√		Login	Login	H_{Ref}	
Our Method			√	√	√	√	√	IP,OS Agent	IP, Agent	H_{Visit} H_{Page}	√

6 Conclusion

In this paper, we have presented an overview of the data preprocessing for KDWUD process and proposed a complete methodology. The experimental results presented in section 4, illustrates the importance of the data preprocessing step and the effectiveness of our methodology, by reducing not only the size of the log file but also increasing the quality of the data available through the new data structures that we obtained. Although the preprocessing methodology presented allows us to reassemble most of the initial visits, the process itself does not fully guarantee that we identify correctly all the transactions (i.e. user sessions & visits). This can be due to the poor quality of the initial log file as well as to other factors involved in the log collection process (e.g. different browsers, Web servers, cache servers, proxies, etc.). These misidentification errors will affect the data mining, resulting in erroneous Web access patterns. Therefore, we need a solid procedure that guarantees the quality and the accuracy of the data obtained at the end of data preprocessing. In conclusion, our methodology is more complete because:

- It offers the possibility of analyzing jointly multiple Web server logs;
- It employs effective heuristics for detecting and eliminating Web robot requests;
- It proposes a complete relational database model for storing the structured information about the Web site, its usage and its users;

7 REFERENCES

[1] Configuration files of W3C httpd, http://www.w3.org/Daemon/User/Config/ (1995).

[2] *W3C Extended Log File Format*, http://www.w3.org/TR/WD-logfile.html (1996).

[3] J. Srivastava, R. Cooley, M. Deshpande, P.-N. Tan, Web usage mining: discovery and applications of usage patterns from web data, SIGKDD Explorations, 1(2), 2000, 12–23

[4] R. Kosala, H. Blockeel, Web mining research: a survey, SIGKDD: SIGKDD explorations: newsletter of the special interest group (SIG) on knowledge discovery & data mining, ACM 2 (1), 2000, 1–15

[5] R. Kohavi, R. Parekh, Ten supplementary analyses to improve e-commerce web sites, in: Proceedings of the Fifth WEBKDD workshop, 2003.

[6] B. Mobasher R. Cooley, and J. Srivastava, Creating Adaptive Web Sites through usage based clustering of URLs, in IEEE knowledge & Data Engg work shop (KDEX'99), 1999

[7] Bettina Berendt, Web usage mining, site semantics, and the support of navigation, in Proceedings of the Workshop "WEBKDD'2000 - Web Mining for E-Commerce - Challenges and Opportunities", 6th ACM SIGKDD Int.

Conf. on Knowledge Discovery and Data Mining, 2000, Boston, MA

[8] B. Berendt and M. Spiliopoulou. Analysis of Navigation Behaviour in Web Sites Integrating Multiple Information Systems. VLDB, 9(1), 2000, 56-75

[9] A. Joshi and R. Krishnapuram. On Mining Web Access Logs. In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pages 2000, 63- 69

[10] C. Shahabi and F. B. Kashani. A Framework for Efficient and Anonymous Web Usage Mining Based on Client-Side Tracking. In WEBKDD 2001 - Mining Web Log Data Across All Customers Touch Points, Third International Workshop, San Francisco, CA, USA, August 26, 2001, Revised Papers, volume 2356 of LNCS, Springer, 2002, 113-144

[11] Y. Fu, K. Sandhu, and M. Shih. A Generalization-Based Approach to Clustering ofWeb Usage Sessions. In Proceedings of the 1999 KDD Workshop on Web Mining, San Diego, CA, vol. 1836 of LNAI,. Springer, 2000, 21-38

[12] M. S. Chen, J. S. Park, and P. S. Yu. Efficient Data Mining for Path Traversal Patterns. Knowledge and Data Engineering, 10(2), 1998, 209-221

[13] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. Data Mining and Knowledge Discovery, 6(1), 2002, 61-82,

[14] M. El-Sayed, C. Ruiz, and E. A. Rundensteiner. FS-Miner: Efficient and Incremental Mining of Frequent Sequence Patterns in Web Logs. In Proceedings of the Sixth Annual ACM International Workshop on Web Information and Data Management (WIDM '04), ACM Press, 2004, 128-135

[15] B. Berendt, B. Mobasher, M. Nakagawa, and M. Spiliopoulou. The Impactof Site Structure and User Environment on Session reconstruction in Web Usage Analysis. In Proceedings of the Forth WebKDD 2002 Workshop, at the ACM-SIGKDD Conference on Knowledge Discovery in Databases (KDD'2002), Edmonton, Alberta, Canada, 2002

[16] C. Marquardt, K. Becker, and D. Ruiz. A Pre-Processing Tool for Web Usage Mining in the Distance Education Domain. In Proceedings of the International Database Engineering and Applications Symposium (IDEAS'04), 2004, 78-87

G T Raju is a doctoral student at Visvesvaraya Technological University, Karnataka, INDIA. He has received his M.Tech degree in Computer Science & Engg from Bangalore University, and B.E degree in Computer Science & Engg from Bangalore University. He is currently working as an Assistant Professor in Computer Science & Engineering Department., BMS college of Engineering, Bangalore. His research interest includes Knowledge Discovery from Web Usage Data, Data mining, Clustering, and Intelligent Agents.

Dr. P S Satyanarayana is currently working as Professor and Head of Electronics & Communication department, BMS College of Engineering, Bangalore. He has received his PhD from Indian Institute of Science, Bangalore INDIA. His research interest includes Information and Coding theory Control systems, Pattern Recognition and Knowledge Discovery & Management.