MultiCraft

International Journal of Engineering, Science and Technology
Vol. 2, No. 11, 2010, pp. 17-33

INTERNATIONAL JOURNAL OF ENGINEERING, SCIENCE AND TECHNOLOGY

www.ijest-ng.com

Constraint based frequent pattern mining for generalized query templates from web log

Ramachandra. V. Pujeri[1], G.M. Karthik[2]

[1]KGiSL Institute of Technology, Coimbatore, Tamil Nadu, INDIA
e-mail:sriramu_psg@yahoo.com, Tel +91-98431-20515, +91-422-6619929
[2]Department of Computer Science and Engineering, SACS MAVMM Engineering College, Madurai, Tamil Nadu, INDIA
e-mail:gmkarthik16@gmail.com, Tel +91-94432-85673, +91-452-2389873

Abstract

The World-Wide Web provides every Internet citizen access to an abundance of information, but difficulty increases in identifying the relevant piece of information. Popular Search engine uses log for keeping track of user activities including user queries, click-through and their behavior. Research in web mining tries to address this problem by discovering knowledge from user logs. We propose an approach to discover patterns that can predict user's search, without aid of remote server. Our method analyses user's interactions by constructing FP-tree, which facilitates in producing templates from the user logs. *Consensus tree* growth is restricted and templates are obtained from leaves, which assist user's searching process with precision. We show the effectiveness of our method on realistic web logs and explore the tradeoff between prediction's accuracy and usefulness. Test results show the improved algorithm has lower complexity of time and space, and fit the capacity of memory.

Keywords: frequent pattern mining, web log mining, CBFP mining, FP-tree, data mining.

1. Introduction

As more organizations make use of the Internet and World-Wide Web (WWW) to convey information, (Berners-Lee et al, 1994) it has overwhelmed home computer users with an enormous deluge of information. For any topic one can think of, he can find pieces of information made available by internet, ranging from individual users, which post an inventory of their record collection, to major companies that do business over the web. The abundance of information, web users need, require assistance for finding, sorting, and filtering. Beyond search engines, which are already in use, research requires concentration on the development of the agents that are high-level interface to the web (Etzioni & Weld, 1994), as well as automated answering systems (Scheffer, 2004). Many of these systems are based on machine learning and data mining techniques. Just as data mining aims at discovering valuable information that is available in conventional databases, the emerging field of *web mining* aims at finding and extracting relevant information that is hidden in web-related data, particularly in (hyper-) text documents published on the web. The web as a whole appears as the largest unstructured data source, though the data on the web sites is structured.

Almost all web sites have searching function, and the search engines under use are confronting the following troubles (Lin *et al.,* 2003); *over abundance:* Most of the data on the web are of no interest to most people. In other words, even though there is a lot of data on the web, an individual query will retrieve only a very small subset of it. *Limited coverage:* Search engines often provide results from a subset of the web pages. Because of the extreme size of the web, it is impossible to search the entire web at any time a query is submitted. Instead most search engines create indices that are updated periodically. When a query is posted, only the index is directly accessed. *Limited query:* Most search engines access on simple keyword-based searching, retrieve or order pages based on popularity of pages. *Limited customization:* Query results are determined only by query itself, often dependent on the background and knowledge of the user. The focus of this paper is to provide an overall view of how to use frequent pattern mining techniques for discovering different types of patterns in a web log database.

The conventional approaches to web site evaluation need to be revised (Cooley *et al*., 1999). The web of the search engine allows users to post keyword queries to a search engine, for finding articles related to queries and to browse organized articles

under a hierarchical structure of several levels. It is well known that queries for web search engines are often too short to contain sufficient information to discriminate ambiguous documents. It was tried to infer this information from server-side web logs (Mobasher et al, 2000). The information contained in a web log includes the IP-address of the client, the page that has been retrieved, the time at which the request was initiated, the page from which the link originated, the browsing agent used, etc. The log keeps user's queries and their clicks, as well as their browsing activities. The file is quite large: one-day log would be over several hundred megabytes in size. The purpose of web-log mining is to improve web performance by utilizing the mined pattern. Unless additional information is available for informative search terms, there is no way to determine the information that a user browse is relevant or not. User queries posted directly to search engine, results in furnishing information based solely on the keywords. In this paper we tried to reduce the time of searching, unnecessary queries to remote search engine. Instead, our technique provides necessary and required information based on past previous log information obtained from local proxy server. In particular, we apply data mining to extract useful and implicit knowledge from web logs, which keep traces of information, during users' visit of web pages on web servers. If same query posted again, our technique provides direct link obtained from mined log information, otherwise query being sent to remote server side. Indeed, data mining is a process of discovering/extracting implicit and useful knowledge from large data sets. Data mining is very promising, since popular web sites get millions of hits each single day, and since traditional or humanly methods would be infeasible to analyze such logs. The log used in this paper is from a popular commercial search engine that is currently in use.

In this paper, we develop and integrate two techniques in order to find interesting frequent patterns or templates from weblog, in order to support the informative search term of the search engine. First, a novel, compact *FP tree like TRIE* data structure is constructed (Grahne *et al.,* 2000; Han *et al,* 2000; Pei *et al.,* 2001; Pei *et al.,* 2000; Zaki and Hsiao 2002; Agarwal 2001; Srikant *et al.,* 1997; White and Drucker, 2007) which is an extended prefix tree structure, storing quantitative information about the frequent hits of a web page from weblogs. Every node of *consensus tree* points in the tree describes how they are classified for easier access of the pattern or templates, and path from root to some leaf will give information about keywords, location, time, etc., for each web page or web object obtained from web logs. Leaf of the *consensus tree* contains value to a special index referencing the buckets to store the templates for each frequently accessed web object. The tree growth restriction, keywords mutation, predicting existence of a template is discussed in this paper later. The *FP growth* method ensures in finding frequent pattern or templates along with relative keyword list using least frequent item as a suffix, offering good selectivity.

Second, using the strategy of constrained based mining (Mannila and Toivonen 1997; De Raedt & Kramer, 2001; Han *et al,* 1999; Garofalakis *et al.,* 1999; Ng *et al.,* 1998; Lee and DeRaedt, 2004), we restrict growth of *FP-tree* like *TRIE* using user-specified constraints (Ng R *et al,* 1998). *Constraint Based mining* allows us to focus on restraining the growth of *consensus tree* by providing additional mining constraints like *level* (Han, *et al.*, 1999) and *rule constraints* (Lee and DeRaedt, 2004). *Level constraint* focus on the mutation of keywords for a web object along with pruning the level of least accessed. *Rule constraint* focus to prune the template available in the leaf of the FP tree (Srikant and Agrawal,, 1996; Pei *et al.,* 2001; Grahne *et al.,* 2000). Integrating two techniques, a new mechanism is developed known as web log mined using *CBFP mining algorithm (Constraint Based Frequent Pattern mining) on web log for generalized queries* with reasonable time and space complexities. *CBFP mining algorithm* constructs a *TRIE structure* which provides templates, keywords and article relating to a query, and proposes, based on the downward closure property, narrowing down the search into level-wise search strategy. *CBFP mining algorithm* is proposed for finding useful generalized templates, associating keywords of the queries with targetted articles. *CBFP mining algorithm* will process much faster than traditional search methods. *Consensus tree* path will be same for a particular article keyed in by many users with similar query, and templates summarised, will be a direct answer to future searches by similar user keying queries. *CBFP mining algorithm* will generalize keywords that can match new queries not made previously, making templates more general and precise. *CBFP mining algorithm* provides clear views of the templates and will locate the article, which users are mostly interested. *CBFP mining algorithm* is proposed based on two points. The first one is we search for frequent keywords of any length among URL sequences from a web log. The second one is that we search for all instances URL in the input logs. An implicit user-defined constraint plays a vital role in pruning the search space of the FP-tree and reduces searching time.

Previous works focused on mining mainly to change the web structure for easier browsing (Craven *et al.,* 1999; Sundaresan and Yi, 2000), predicting browsing behaviours for prefetching (Zaine *et al*. 1998; Boyan, 1996), or predicting user preference for active advertising (Pei *et al.,* 2000; Perkowitz, 1997).Some works have been done on data mining system to discover useful patterns or templates from web server log (Ling *et al,* 2001) or to describe users' interest based on semantic (Li *et al.,* 2008), as well as on consideration of length pattern with position of webpage (Ou *et al.,* 2008), or have been designed specifically for Apache module (Heung *et al.,* 2009). Few works used data mining to gather statistical information for better weighing and ranking of the document. In previous technique, results can be obtained from newly appended data into web log without details of previous / old data, offering minimum support and setting confidence value a trivial one, while handling large log data. Dependence between page accesses and calculation over the appearance of user access sequences frequency is not effective for various users. All these works suffer from the problem of requiring a large search space and from ineffectiveness in handling long patterns. In this paper, we focus on two factors, i.e., the order of dependencies between page accesses and calculation of frequencies of user access sequences, which characterize the performance of our technique. The aim and scope of this paper is to offer recommendations for developing future personalization services and to report on initial findings on a specific aspect that is highly relevant for personalization: the study of user sessions. We propose a novel data mining technique that discovers useful and

interesting patterns and/or templates based on past and recent logs that will lead to future interest/use. The term interesting/interestingness describe how and to what extent the articles or web pages contribute to a users' interest/utilize (in terms of keywords). The proposed technique improves the performance significantly to avoid time consuming scanning, comparisons and searching. While travelling through the path of tree, scanning and comparisons are minimized, filtering out unnecessary templates, and constraints, helping to interpret mined templates, predicting users' future request.

The remaining part of the paper dwells as follows: Section 2: Surveys related work. Section 3: Discussing the *CBFP mining algorithm* in detail. Section 4: It furnishes results obtained from our mining system. Finally, Section 5 delivers the conclusion of the paper.

2. Existing methodologies

Most of the previous approaches dwelt with mainly analysing the contents of web documents (content mining) or the graph structure of the web (structure mining) and/or user-specific browsing recommendations (Armstrong *et al,* 1995; Pazzani *et al,* 1996; Balabanovic and Shoham, 1995). A series of tools to retrieve, store, and to analyze the data extracted from log files have been designed and implemented (Maristella *et al,* 2007). WebWatcher system (Armstrong *et al,* 1995) predicts which links on the currently viewed page are most interesting to the user's search goal, that has to be specified in advance, and recommends the user to follow these links, or explicit feedback about interesting or not interesting pages (Pazzani *et al,* 1996). Based on the web users interest, web page Recommendations are done with Web Utilization Miner (Spiliopoulou 1999) is a publicly available, prototypical system that allows mining web logs using advanced association rule discovery algorithms. Special techniques have to be used to infer the browsing paths (so-called *click streams*) of individual users (Cooley *et al,* 1999). Some, individuals are rightly concerned over the deliberate sharing of their information mined and analyze learners' preferences, wishing to prevent breaches (Rosie, 2009; Man *et al.,* 2009; Kumar, 2009). For restructuring web sites (Perkowitz and Etzioni, 2000), association rule are defined for matching user queries with web using formal concept analysis (YaJun and Haiming, 2010), predicting browsing behaviors for prefetching (Zaine *et al.,* 1998; Boyan *et al*., 1996), personalizing web services (Mobasher *et al.,* 2000; Mobasher *et al.,* 2002; Pierrakos *et al.,* 2003), for describing user's interest based on semantic (Chen Li *et al.,* 2008), classification and prediction based on character N-grams (Liu and Keselji, 2007), predicting browsing behaviors for prefetching (Zaine *et al.,* 1998; Boyan 1996), optimizing search engines (Joachims, 2002), predicting user preference for active advertising (Pei *et al,* 2000),frequent traverse pattern with consideration of length of pattern, along with position of webpage (Ou *et al.,* 2008), recognizing web spiders (Tan and Kumar, 2002) and Mining focused to change the web structure for easier browsing (Craven *et al.,* 1999; Sundaresan and Yi, 2000). Double prediction-by-partial-match scheme (DPS) designed based on the memory aware request specifically for Apache module (Heung *et al.,* 2009). Web access data are made available in Doubly-linked tree, access patterns are obtained using recursive algorithms (Ratnesh *et al.,* 2009). Re-mining overhead with old and new log data is handled by incremental web traversal pattern mining algorithm (Jia-Ching *et al., 2009)*, but using static projected link matrix. A self-supervised learning handle method (Nguyen *et al.,* 2010) handle sequence labeling problem and achieve precision based on selective attributes. For more scalability and profile tracking, used unsupervised niche clustering algorithm, which suffers in handling large volume of log data (Hawwash and Nasraoui, 2010). The generalized fuzzy association rules are mined from logs considering users' interest and semantic annotated content (Rui, 2010; Panagiotis *et al.*, 2010), which cannot project future interest, unable to handle large log data, and not applicable to more complex taxonomic nature. Existing predictive prefetching algorithm that uses a graph called *Dependency Graph (DG)* (Griffioen and Appleton, 1995), text compression domain called *Prediction by Partial Match (PPM)* (Padmanabhan and Mogul, 1996), *WM* and WM_o method using association rules among user access (Yannis *et al,*2003) needs synthetic data generator. *DG* algorithm needs fixed length of the transaction and regular refreshing of cache. *DG* achieves low performance and accuracy with decreasing usefulness and network traffic. *PPM* algorithm handles maximum length of the transaction and predicts preceding document within request streams. *PPM* efficiency examined with pruning criteria; it's overhead in selecting confidence and support value. WM_o and WM algorithm outperforms *DG* and *PPM* in terms of accuracy and usefulness for same network traffic. WM_o and WM algorithm has effective prefetching with space restriction, low overhead in network traffic. WM_o and WM algorithms prediction with new transaction are handled less effectively, association rule framing based on new transaction influenced by support and confidence value not feasible. Knowledge management requirement for web log mining are expressed by David (2008). The purpose of our project is to find useful templates with improved speed and accuracy. Secondly, it is to discover the topics, in which users are mostly and clearly interested *(means related to user's interest)*. We design *CBFP mining algorithm* that provides clear views of the templates and it will discover the article which users are mostly interested, for improving search engine's performance.

3. Constraint Based Frequent Pattern Mining Algorithm

In traditional methods, in order to evaluate the measure of user's interest, a group of users would have to be selected. Certain activities that they are expected to perform would have to be identified and their actions recorded while performing these tasks. It is impractical and costly to carry out this each time. To evaluate the web site, the data is automatically recorded by the web server. All the activities that take place on a web site are recorded in a file called web server log. For the task of applying data mining

techniques, we extract information from web data in order to understand and better serve the needs of users navigating on the web (Srivastava et al. 2000). In this work, patterns are discovered by applying the frequent pattern mining methods i.e. *CBFP mining algorithm* on the log data. For this reason the log data have to be converted in the preprocessing phase such that the output of the conversion can be used as the input of the algorithm. Pattern analysis means understanding the results obtained by the algorithms and drawing conclusions. In pattern analysis, local proxy server enables our technique to analyse information retrieved are relevant to user or not, using feedback for each accessed article.

Figure 1 shows the process of mining algorithm. As can be seen, the input of the process is the log data. The data has to be preprocessed in order to have the appropriate input for the mining algorithms. The different methods need different input formats, and thus the preprocessing phase can provide three types of output data. First, the frequent patterns discovery phase needs only the web pages visited by a given user. In this case the sequences of the pages are irrelevant and the duplication of the same pages is omitted. Second, if a page was visited more than once by a given user in a user-defined time interval, then it is relevant as well. For this reason the preprocessing module of the whole system provides the sequences of web pages by users or user sessions. Third, for mining not only the sequences are needed but also the keywords used in a query for a web pages, visited by a given user. In this case the backward navigations are omitted; only the forward navigations are relevant, which form a tree for each set of keywords of a particular web page. After the discovery has been achieved, the analysis of the patterns follows. The whole mining process is an illustrative task which is depicted by the feedback in Figure 1. Depending on the results of the analysis either the parameters of the preprocessing step can be tuned (i.e. by choosing another time interval to determine the sessions of the users) or only the parameters of the mining algorithms (In this case that means the minimum support threshold). In this work the aim of mining is to discover the generalized templates from pages frequently visited at the same time, and to discover the page for corresponding keywords. The results obtained is used to allow users to view all templates (both simple and generalized) or generalized templates only, view templates sorted by different ways, and display templates in different level of details. The discovery of the templates can provide insights to the web editors as to what topics users are mostly interested in. When incorporated with the regular search engine, those templates can improve the search speed.

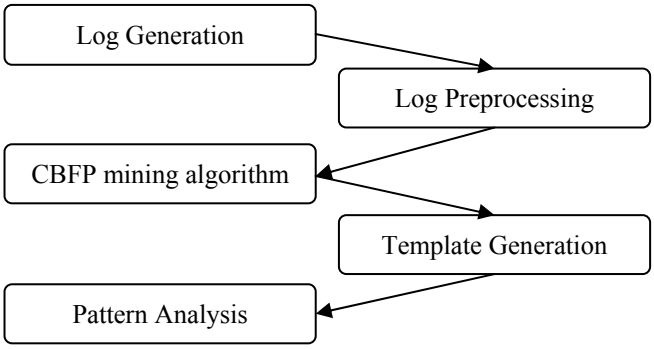

Figure 1 Overall process of mining algorithm

202.161.108.167 - - [01/Feb/2003:00:00:03 +1100] "GET /timetables/city /2003s1/ cc 4logo.gif HTTP/1.1" 206 14102 "http://www.cs. rmit.edu.au/ timetables /city/2003s1/ cover.html""Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)"

213.183.13.65 - - [01/Feb/2003:00:00:16 +1100] "GET /~winikoff/palm/dev.html HTTP/1.1" 302 244 "http:// www.google.de/search? q=sources+onboardc+ examples&ie =UTF-8 &oe=UTF-8&hl=de&meta=" "Scooter/3.3"

Figure 2 Sample Web Access Log

3.1 User activity Logs: The web of the search engine consists of a search engine, with well-versed articles (written by experts of various domains) as answers to the user queries, and a well organized hierarchy of articles for browsing (but one cannot use both interchangeably). The users' activities are recorded in the raw IIS log files, which keep a lot of information about each user's access to the web server been used. Each log entry includes an anonymous session identifier, a timestamp, and the URL of the visited web page as shown in Figure 2.

From these and similar interaction logs, user trails can be reconstructed using the methodology defined by White and Drucker (White and Drucker, 2007). We extracted such trails from the logs of users of the Windows Live Toolbar. For each user, interaction logs were grouped based on browser identifier information. Within each browser instance, user navigation was summarized as a path known as a *browser trail*, from the first to the last web page visited in that browser. Located within some of these trails are *search trails* that originated with a query submission to a commercial search engine; it is these search trails that we

use to train the algorithms described in the following sections. After originating with a query submission to a search engine, search trails proceed until a point of termination where it is assumed that the user has completed their information-seeking activity or has addressed a particular aspect of their information need. Trails must contain pages that are either search result pages, or pages connected to a search result page via a sequence of clicked hyperlinks. Extracting search trails using this methodology also goes some way toward handling multi-tasking, where users run multiple searches concurrently. Since users may open a new browser window (or tab) for each task, each task has its own browser trail, and a corresponding distinct search trail. To reduce the amount of "noise" from pages unrelated to the active search task that may pollute our data, search trails are terminated when one of the following events occurs:

- User submits a new search query;
- User navigates to their homepage, initiates a web-based email session, or visits a page that requires authentication, types a URL or visits a bookmarked page;
- A page is viewed for more than 30 minutes with no activity;
- The user closes the active browser window.

On average, there are around 5 steps per search trail. These works carried out by preprocessing phase on log entry includes an anonymous session identifier, a timestamp, and the URL of the visited web page.

3.2 Data Preprocessing: The raw web log data is usually diverse and incomplete and difficult to be used directly for further pattern mining. The users' activities are recorded in the raw IIS log files, which keep a lot of information about each user's access to the web server. Preprocessing helps our mining algorithm to produce a generalized one by pairing one-keyword queries on generalized template. In order to process it, we need to:

3.2.1 Data Cleaning: To learn from the web logs, the first task is to perform data cleaning by breaking apart a long sequence of visits by the users into user sessions. Each user session consists of only the pages visited by a user in a row. Since we are only dealing with web server logs, the best we could do is to take an intelligent guess as to which pages in a long sequence belong to a single user session. As we will see, a strong indicator is the time interval between two successive visits. When the time exceeds the average time it takes a person to read a web page, there is a strong indication that the user has left the browser to do other things. In the web logs, a user can be identified by an individual IP address, although using the IP address to identify the users is not reliable. Several users may share the same IP address, and the same user may be assigned different IP address at different times. In addition, the same user may make different sequences of visits at different times. Thus, data cleaning means to separate the visiting sequence of pages into visiting sessions. Most work in mining employ a predefined time interval to find the visiting sessions. For example, one can use a two-hour time limit as the separating time interval between two consecutive visiting sessions, because people usually do not spend two hours on a single web page. Sometimes it is possible to obtain more information about user sessions than using a fixed time interval. By learning the grouping of web pages and web sites, one can find more meaningful session separator. The work by Lou et al. (2002) provides a method to use clustering analysis to find the group of related web servers visited by users from a web proxy server. If a user jumps from one group of related server to another, then it is highly likely that the user ends one session and starts another. Using this information to cut and pick the web pages, more accurate user session knowledge can be obtained.

3.3.2 User Identification: To identify the users, one simple method is requiring the users to identify themselves, by logging in before using the web-site or system. Another approach is to use cookies for identifying the visitors of a web-site by storing a unique ID. However, these two methods are not general enough because they depend on the application domain and the quality of the source data, thus in our system we only set them as an option. More detail should be implemented according to different application domains. We have implemented a more general method to identify user based on Cooley et al. (1999). We have three criteria:

1. A new IP indicates a new user.
2. The same IP but different web browsers, or different operating systems, in terms of type and version, means a new user.
3. Suppose the topology of a site is available, if a request for a page originates from the same IP address as other already visited pages, and no direct hyperlink exists between these pages, it indicates a new user (Option).

3.2.3 Session Identification: Identification of the user sessions is also very important because it will largely affects the quality of pattern discovery result. A user session can be defined as a set of pages visited by the same user within the duration of one particular visit to a web-site. According to Wu et al. (1998), Pei et al. (2000) and Perkowitz (1997), a set of pages visited by a specific user is considered as a single user session if the pages are requested at a time interval not larger than a specified time period. In our system, we set this period to 30 minutes. We filter out documents that are not requested directly by users. These are image requests or CSS requests in the log that are retrieved automatically after accessing requests to a document page containing links to these files and some half-baked requests. Their existence will not help us to do the comparison among all the different methods. We consider web log data as a sequence of distinct web pages, where subsequences, such as user sessions can be observed by unusually long gaps between consecutive requests.

Table 1 Combined Log File Format

Field Name	Description
host	IP/DNS address of the http client that made the request
rfc931	Client identification according to rfc 931.
authuser	Login name used by the client for authentication.
datetime	The date and time stamp of the http request, and the time-zone of the server.
request	The http request contains the requested resource (e. g. "index.html"), the http method (e. g. "GET") and the http protocol version (e. g. "1.1").
statuscode	This field indicates the success or failure of the http request.
bytes	Number of bytes transferred
referrer	The URL the client visited before coming to the website.
useragent	Web browser and operating system used by the client.

The description of the attributes shown in Table 1 reveals that log file data primarily focuses on technical issues and is weakly structured. Therefore, it is merely possible to analyze log files instantaneously to gain insight into visitor behavior. Consequently, it is necessary to carry out different steps to prepare log file data. These steps are formalized within the web log mining process model, which is discussed in the following section.

The patterns have to be evaluated by domain experts and might be deployed in order to support decisions at the operational level. All these steps together build the super ordinate concept of *data mining* which is an integrated process of gaining information from data. In the context of log file, the concept of *web log mining* applies. The results of the log pre-processing phase are shown in Table 2.

3.3 CBFP mining algorithm: Previous experiments show that existing algorithm does not produce satisfactory results: only a few useful templates are produced. The first reason is that the generalization step of introducing [any] into templates is often too bold: such templates cover too many queries incorrectly, and thus have a low accuracy. Second, many keywords have slightly different spellings (sometimes with a minor spelling error) and thus are not regarded as the same, preventing possible generalization. Third, many keywords are synonyms; but since they are spelled differently, they cannot be treated as the same for possible generalization. Fourth, one-keyword queries are not paired for generalization, since it would inevitably produce an overly generalized template "article ID ← [any]". Tree constructed by *CBFP mining algorithm* address the first problem using level constraints. Rests, are handled by rule constraints used by *CBFP mining algorithm*. The algorithm uses the basic idea and techniques of *Apriori* algorithm and *FP tree* method. Algorithm employs level-wise and explore pattern based on downward traversal.

Table 2 Cleansed Log Format

Field Name	Description
Date	The date on which the activity occurred
Time	The time the activity occurred.
IP Address	The IP address of the client that accessed the Server.
URL	The resource accessed: for example, an HTML page, a CGI program, or a script.
User Agent	The browser used on the client.

The log obtained from preprocessing phase is given to *CBFP mining algorithm* for constructing *consensus tree*. In the *FP tree* technique, each frequent pattern obtained from the leave of tree. Each piece of user log information is meant for constructing or updating *consensus tree*. The path of the tree, from root to some leaf represents information about the article, used for level constraint and rule constraint. Each node of the tree represents the classification of all articles, which help in updating the article. A node can have path to different other node of the level next to it. A single path covers too many queries correctly since their article and keywords posted by user is different, hence accuracy of accessing article based on the number of keywords and their meaning results in better accuracy. A constraint on particular node is applicable to its entire descending node, so tree also has downward closure property. Level-wise search strategy used for finding frequent articles and corresponding keywords. *FP tree* like *TRIE* structure (called *consensus tree*) for shared representation obtaining generalized templates and level-wise search strategy for predicting the future user's interestingness of an article. In *consensus tree* constructed by *CBFP mining algorithm* is shown in Figure 3. The *consensus* generalize one-keyword queries pairing, since it would inevitably produce an overly generalized template The level-wise search of article along with their keywords starts from the root of the tree. Each node in level-1 of the tree represents the different type of search engine (we can have another level to represents the time-zone), and level-2 shows the number of keywords used in a query. Each leaf contains pointer to all articles belonged to it. Each leaf is organized and contains pointer to the bucket where the article resides. Each leaf may split and combine based on extendible hashing technique. Extendible hashing technique is used for storing article, each article being connected to a link list. The link list contains keyword of user query, corresponding to user click through on that article. Depth of *consensus tree* is same for all nodes. The growth of the

consensus tree is constrained using level-wise and rule constraint. *CBFP mining algorithm* has a reasonable time complexity and space complexity.

An important class of web data mining problem is mining of path traversal patterns, which can be used to decide the next likely web page requests based on significant statistical correlations. If such a sequence appears frequently enough, then this sequence indicates a frequent traversal pattern. However, most previous studies of path traversal pattern mining are based on the model of a uniform support threshold without taking into consideration important factors such as the length of the pattern, the positions of web pages, etc. *Level-wise constraint* decides existence of a node in same level for *consensus tree*. *Consensus tree* is initially constructed with preprocessed log information with above said levels. The article traverses through the path of the *consensus tree* based on the information like time, type of search engine and keywords used in user query. Each node of *consensus tree* will check the *level constraint* when information traverses based on adaptive threshold. While an article travels through the path of the *consensus tree*, each node's $\theta_{i,j}$ value in consensus path will be incremented. $\theta_{i,j}$ is number of articles visited i^{th} node in j^{th} level of the *consensus tree* (sometimes referred as *support value* of node). A i^{th} node in j^{th} level of *consensus tree* is deleted when $conf(\Theta_{i,j}) < L_i^{\delta}$, referred as *Level-wise constraint*. The *confidence value* ($conf(\Theta_{i,j})$) obtained using $\Theta i,j$ value of the current node i^{th} in j^{th} level of the *consensus tree*, and sum of all Θ value of the $j+1^{th}$ level nodes such as

$$conf(\theta_{ij}) = \theta_{ij} / \sum_k \theta, k = 1,2,..j+1 \quad (1)$$

Adaptive Threshold value L^{δ} for i^{th} node is calculated as

$$L_i^{\delta} \approx (conf\ (\theta_{i,j})) / \sum (conf\ (\theta_{(k,j+1)})) \quad (2)$$

Based on the *support* and *confidence value* a node in the *consensus tree* is removed. Deletion of node happens when there is no user event with particular web server or search engine.

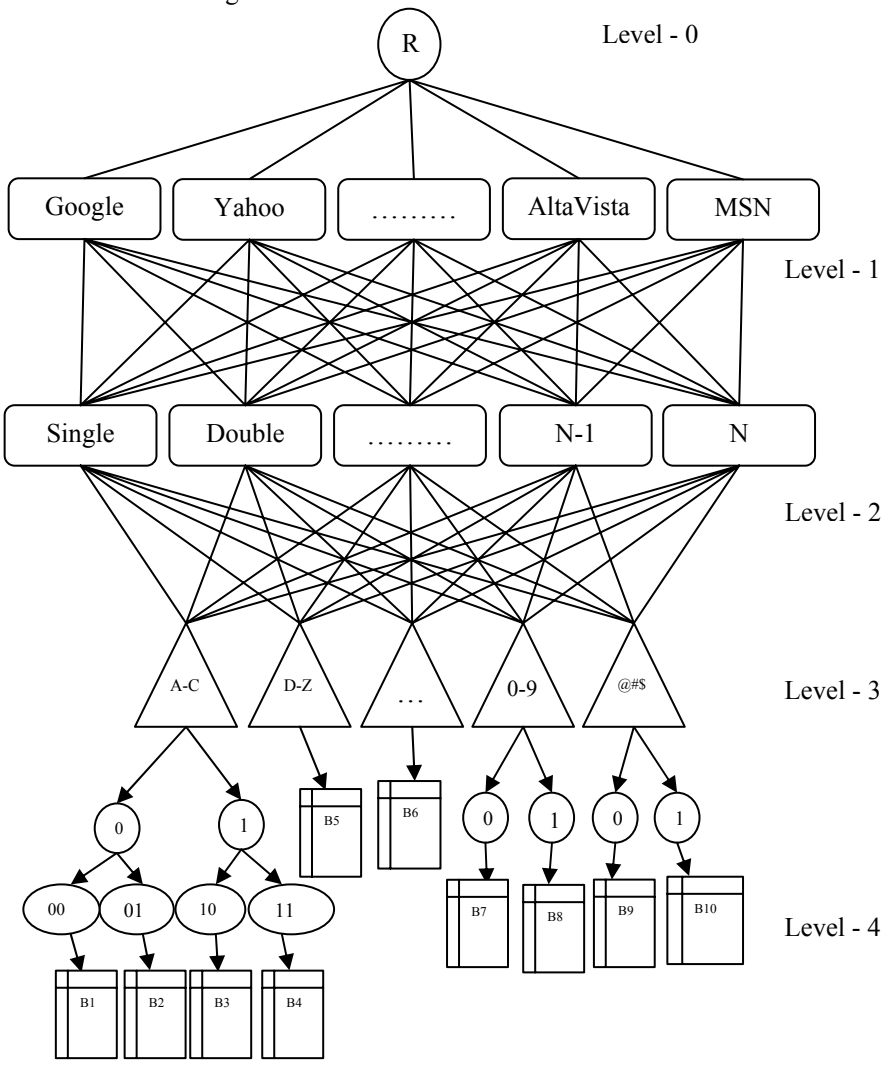

Figure 3 FP-tree constructed by *CBFP mining algorithm*

Rule constraint handles two problems; firstly, many keywords have slightly different spellings (sometimes with a minor spelling error) and thus are not regarded as the same, preventing possible generalization. Secondly, many keywords are synonyms;

however, since they are spelled differently, they cannot be treated the same for possible generalization. Rule constraint handles both issues using mutation factor along with WordNet (Miller, 1995). Rule constraint are applied on each article whether they exist or not, based on the articles accessed information obtained from preprocessed log. Rule constraint is check on leaf node of the *consensus tree* only. Each leaf node contain pointer, points to which bucket (B1, B2, B3 …), article's detail to be stored. Leaf reference is the index structure of extendible hashing technique. Article's name alone fetched from the URL link, and given to the hash function will specify bucket allocated for that article. Bucket size is decided in such way that a single access load into memory leads to faster access of template. Extendible hashing technique decides when bucket is to be estranged and merged. Bucket is estranged or merged when numbers of articles in it are greater or lesser than some threshold value. Rule constraints are applied on the article x in the i^{th} bucket for their existence e_{xi} in the *consensus tree*, based on the values of following factors, hit ratio h_{xi} time (include user time + session time), mutation value m_{xi}, semantic factor s_{xi}, grace period g_{xi} and alive factor a_{xi}. Rule constraints and their factor are checked while updating *consensus tree* along with all factor. The hit ratio h_{xi} is total number of time article accessed by user, obtained from preprocessed log. Each article is given a grace period g_{xi} helps in predicting the user interestingness, based on the h_{xi} and when they accessed lastly. Alive factor a_{xi} contain flag value, 0 represent article deletion, when updating *consensus tree* and 1 represent article existence in bucket till g_{xi} expires. Alive factor a_{xi} initially 1 for new article, changed when g_{xi} expires in next updating process and article deleted when no new information available in next updating process. Article $(a_{xi}=0)$ exist during next update based on h_{xi} and g_{xi}. Existence of an article x in i^{th} bucket calculated as

$$e_{xi} \approx \begin{cases} \dfrac{(\alpha h_{xi} + \beta h_{xi})}{\gamma}, if \quad (a_{xi} = 1) \wedge g_x \\ \dfrac{(\alpha h_{xi} + \beta h_{xi})}{(\alpha + \beta)}, if \quad (a_{xi} = 0) \wedge g_x \end{cases} \tag{3}$$

α, β, γ are adaptive threshold values given based on the user's interestingness. The definition of adaptive threshold has been given above. However, without specific knowledge, it is very difficult and intricate for users to set adaptive threshold for every single web page. If the support threshold is set too high, users cannot obtain enough rules. Therefore, users have to set a lower threshold and conduct the mining again, which may or may not lead to results of better quality. If the threshold is too small, there may be an excessive number of rules for the users and the runtime may be unacceptably long. To overcome this problem, we need an automatic and reasonable methodology to provide the determination of support threshold of web documents. Therefore, we introduce the general probabilistic framework based on the Markov chains which can be used to determine the adaptive threshold of each web page. A Markov chain is a discrete-time stochastic process defined over a set of states S in terms of a matrix P of transition probabilities. The entry $P_{i,j}$ in the transition probability matrix P is the probability that the next state will be j, given that the current state is i. Thus, for all $i, j \in S$, we have $0 \le P_{i,j} \le 1$, and $\sum_j P_{i,j}=1$. The adaptive threshold of the newly identified mining model as the following formulas:

$$Threshold \approx \begin{cases} apt_{\sup(P)} \approx m * |D| / \mu_P, if \quad m * |D| / \mu_P \ge S^{th} \\ apt_{\sup(P)} \approx S^{th}, if \quad m * |D| / \mu_P \le S^{th} \end{cases} \tag{4}$$

where $|D|$ is the size of the log, μ_p is the mean recurrence time of page P, and $m \in [0, 1]$ is a parameter to determine the relationship between the interestingness of web pages and the mean recurrence time. S^{th} is employed for pruning some obsolete rules whose web pages have very low expected occurrence frequencies. Hence the values of α, β and γ are approximately equal to Markov chain *Threshold* value, which leads to results in better quality. We resemble user's interest with adaptive threshold values (α, β, γ), without specific knowledge and user's involvement in each and every single web page.

Each article contains a pointer to header of the link list, which contains keyword of user query. *CBFP mining algorithm* considers both syntactic constraints and semantic constraints. The syntactic constraints are handled by mutation factor, whereas for semantic constraints using WordNet (Miller 1995), a value is calculated for semantic factor. Keyword is subjected to mutation process. The mutation process carried out with keyword exchange is referred to as external mutation. Keyword (may be one or more) are compared with the existing ones for number of character vary, referred as internal mutation. The two processes handle the problem of many keywords having slightly different spellings (sometimes with a minor spelling error) and thus are not regarded as the same, preventing possible generalization. Based on the internal and external mutation, mutation factor m_{xi} value is obtained. Sometimes, many keywords are synonyms; but since they are spelled differently, they cannot be treated as the same for possible generalization, where mutation process fails. We used WordNet to generate such a hierarchy automatically. A problem occurred is that most keywords have different senses or meanings, which in turn, have different parents in the hierarchy. We adopt the first, or the most frequently used meaning of each keyword in the WordNet. WordNet useful generalization of the two queries is that minor differences in the spelling of the same keyword are regarded as different keywords and keywords having very similar meanings but with different spellings. WordNet is used as analytical tool which generalize keywords give semantic value. Keyword created or removed forms a node in link list based on the m_{xi} and s_{xi}. The preprocessed log contains information shown in Table 2. For each article i, number of keywords and Keyword list are separated, given to mutation process. In mutation process, keywords are first subjected to internal mutation and find semantic factor s_{xi} using WordNet (Miller 1995). If number of keywords is more than one, they are subjected to external mutation. In external mutation the order of the keywords are changed and their semantic factor s_{xi}' are calculated with WordNet. For each possibilities (in ordering of keywords) the semantic factor $s_{xi}{}^{'1}$, $s_{xi}{}^{'2}$, $s_{xi}{}^{'3}$, ... are calculated and used in calculating mutation factor m_{xi} for an article i. The m_{xi} value is calculated as

$$m_{xi} \cong \begin{cases} \dfrac{((0.6)\times s_{xi})+((0.4)\times S_{xi}')}{\sum_j s_{xi}'^{j}}, if\ (s_{xi} > S_{xi}') \\ \dfrac{((0.4)\times s_{xi})+((0.6)\times S_{xi}')}{\sum_j s_{xi}'^{j}}, if\ (s_{xi} < S_{xi}') \\ \dfrac{s_{xi}}{\sum_j s_{xi}'^{j}}, if\ (s_{xi} = S_{xi}') \vee (S_{xi}' < 0) \\ s_{xi}, if\ (S_{xi}' = 0) \end{cases} \quad (5)$$

where S_{xi}' is the highest semantic value among $s_{xi}'^{1}$, $s_{xi}'^{2}$, $s_{xi}'^{3}$,… values. The value of $S_{xi}'=0$ means single keyword or external mutation of keywords has no semantic. The mutation factor m_{xi} assigned to particular order of keywords that resulted after mutation process. With m_{xi} value keywords differences in the spelling of the same keyword are regarded as different keywords and keywords having very similar meanings but with different spellings are resolved, which also helps in retrieving process.

The consensus FP tree is constructed based on the information obtained from the preprocessing phase. From preprocessed log keyword list of each articles are given to mutation process, and obtain m_{xi} value and sequence order of the keywords. For each article's number of keywords, m_{xi}, and keyword list are given to hash function of extensible hashing technique. The hash value is nothing but a pointer to a specific bucket where article along with keywords are stored. We bind the hash value, user agent, date, time, and number of keywords along with articles traverse through the *consensus tree*. When an article traverses through each node in the path of the tree, node will be updated with the hash value of each article, and incremented $\Theta i,j$ value. $\Theta i, j$ value will be increment for a particular node based on level of the tree, since each level deals with separate information such as first level deal with user agent, second level deal with number of keywords and so on. Finally article is stored in bucket based on the hash value. Here article can be directly stored in the buckets but lacks in different level of abstraction of information regarding article. We can obtain information from *consensus tree* such as all article accessed using a particular user agent, single or multiple search keyword articles, article starting with particular strings. Each node of *consensus tree* contains hash value of all articles that has visited it. Based on stored article's hash value, we can abstract all articles that belong to particular level/node of *consensus tree* (which has specific information). Each node/level/article's existence is constrained using above said factors.

The generalized template generated by *CBFP mining algorithm* acquired using sequential scan on all leaf nodes of the *consensus tree* helps in improving the performance of the search engine. Pattern analysis phase reveals precision and coverage of the generalized templates. Accuracy and coverage are two numbers for evaluating such a template. Pattern analysis uses accuracy as a factor to evaluate template or pattern obtained. Accuracy of a template means covering user queries by correctly generated template, along with incorrectly covered templates. Coverage (C_x) is the number of original user queries that are correctly covered (or predicted) by this template, and accuracy is the percentage of queries that correctly covered. Let Δ_x is the number of queries incorrectly covered by a template (when users clicked a different article) then $accuracy \approx \sum_x C_x / \sum_x (C_x+\Delta_x)$ (6). The templates represent a set of user queries with the same or similar intention, and keywords associated with an article as an answer. *CBFP* mining algorithm use level constraint as monotonic and rule constraint as anti-monotonic property (Lee & De Raedt, 2004). *A template with the maximum coverage while the accuracy is above a threshold is chosen*. All queries covered by the new template are removed, and the process repeats until no new templates can be generated. The patterns obtained from the *consensus tree* are beneficial, and can provide insights into the web editors so as to what topics users are mostly interested in. When incorporated with regular search engine, these templates can improve the search speed, as well as the recall and precision of the search engine.

4. Experiment and results

The log data can be collected at the server-side, client-side, proxy servers, or obtained from an organization's database (which contains business data or consolidated web data). A web proxy acts as an intermediate level of caching and can be used to reduce the loading time of a web page faced by users at work, as network traffic load at the server and client side (Cohen *et al*., 1998). The performance of proxy caches depends on their ability to predict future page requests correctly. Proxy traces may reveals the actual HTTP requests correctly. Proxy traces may reveal the actual HTTP requests from multiple clients to multiple web servers. We used IIS log data source for characterizing the browsing behavior of a group of anonymous users sharing a common proxy server. In this work, we hold that user requested is retrieved entirely from the original web servers or from a web cache before being sent to the users. Web documents or objects have different sizes, while a cache only has a finite size. In cache requested documents appears, and user can be satisfied immediately, which is called a hit; otherwise, the documents fetched from the original server, which is termed as *miss*. The hit rate is ratio between the number of requests that hit in the proxy cache and the total number of requests. The hit ratio is an even more realistic measure of performance for our algorithm; it is the ratio between the number of bytes that hit in the proxy cache and total number of bytes requested. Experiments are conducted with different cache sizes. The size of the cache is expressed in terms of the percentage of the total number of bytes of all objects in a web log. The algorithm was constructed with an offline mining system of two days raw IIS log files obtained from local proxy server, and time was calculated for log pre-process, along with tree construction. *CBFP* technique discovers templates in different level of abstraction from the *consensus tree* based on the number of keywords. Templates obtained from user log, are more reliable and interesting templates are found, since more queries will be asked for each article, and over generalized templates would be

rejected. Pattern evaluation tests how effectively those templates help to improve the search engine performance. We test with template matching mechanism into actual search engine. We run the algorithm with generated synthetic logs referred as training log data, to check the effectiveness of the templates mined from the training log. We measure the testing queries that are matched with the templates, benefited from using templates. *CBFP mining algorithm* predicted the templates those queries are the same as what templates predicted with optimal speed. The running time of mining algorithm is calculated with tree hierarchy as well as with syntax and semantic prediction using Word Net. We obtained better result tuning the threshold values, which allow us to find out which template produces good results. The *CBFP mining algorithm* runs with optimal precision with better overall performance, as they have much higher recall. Considering that we need to predict whether the templates last for future user queries, done with *consensus tree* information. The templates generated produce good results in predicting the users' interest measure based on the threshold values used in construction of the *consensus tree*. Our algorithm differs much larger and more general, and it is automatically generate templates over keywords. *CBFP mining algorithm* guide the rule induction process, since the templates build from *consensus tree* which extract the rules, are pruned by deleting with care that they will not affect the predictive accuracy.

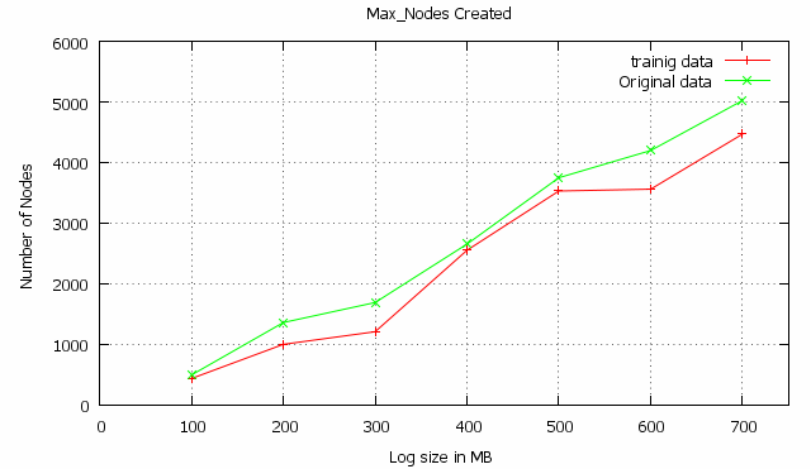

Figure 4 Maximum number of node created by *consensus tree*

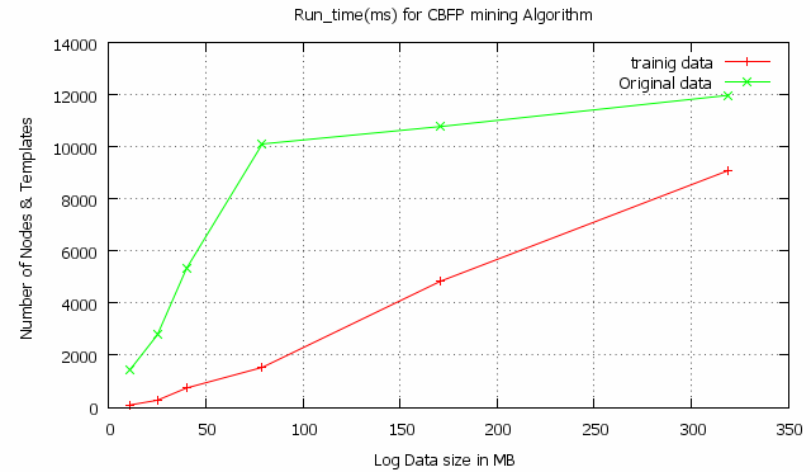

Figure 5 Running time of *CBFP mining algorithm* in milliseconds

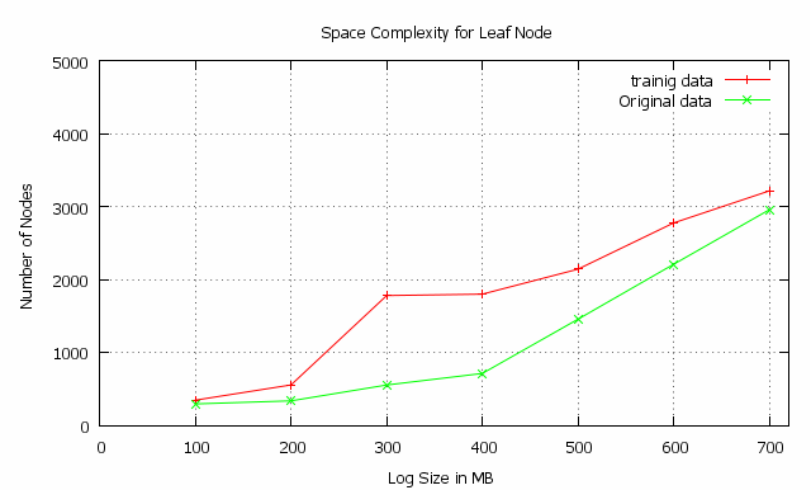

Figure 6 Space requriment for leaf node of *consensus tree*

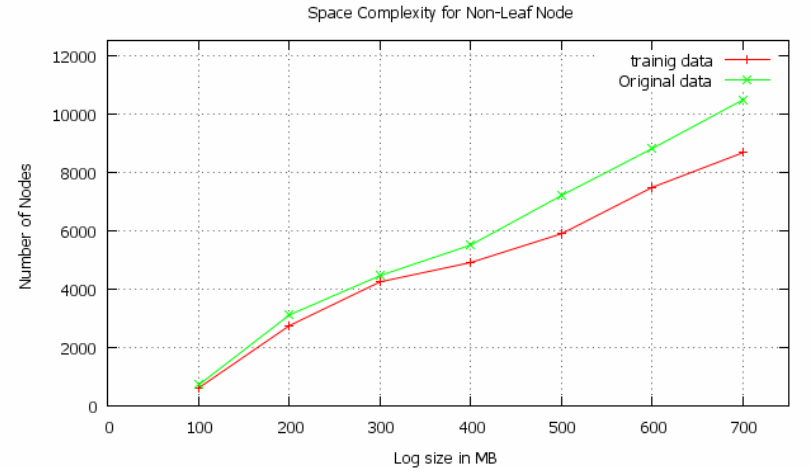

Figure 7 Space requirement for non-leaf node of *consensus tree*

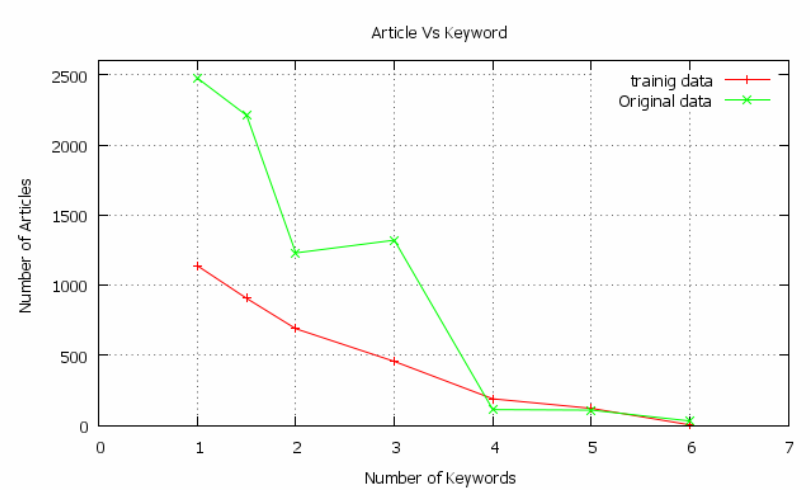

Figure 8 Number of article vs number of keywords

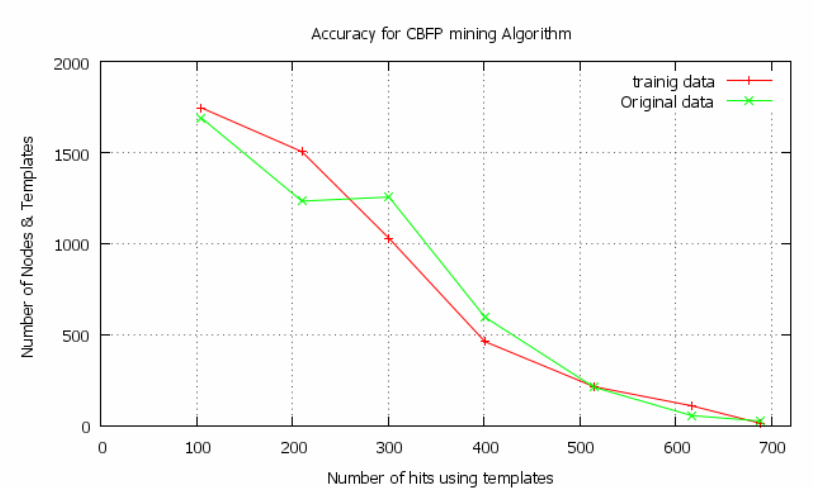

Figure 9 Accuracy of templates maximum number of node created by *consensus tree*

Figure 4 shows maximum number of nodes created including leaf and non leaf node. Figure 5 reveals running time (millisecond) of *CBFP mining algorithm* for raw log file slower than training log data, since raw log contains wide range of article, the time consumed by mutation and semantic process of the *CBFP technique*. Space complexities of *CBFP* mining algorithm are shown in Figure 6 & Figure 7. In *consensus tree* non leaf node creation are constrained based on level constraints shown in Figure 7, where in Figure 6 leaf node creations highly influenced by m_{xi} and s_{xi}. Figure 8 shows how templates are more generalized using *CBFP mining algorithm*. The generalized templates evaluated using Accuracy, means covering user queries correctly by generated template shown in Figure 9 with number of hits. Based on results, algorithm show apt result for training log data; with raw log data it shows optimal results.

Existing predictive prefetching algorithm uses a graph called *Dependency Graph (DG)* (Griffioen *et al*, 1995), text compression domain called *Prediction by Partial Match (PPM)* (Padmanabhan *et al,* 1996),*WM* and WM_o method using association rules among user access (Yannis *et al,*2003) needs synthetic data generator. *DG* algorithm was set equal to length of the transaction; and for the need to flush cache after completion of each transaction. *DG* achieves low performance, shows asymptotically increasing confidence and accuracy with decreasing usefulness and network traffic. WM_o and WM algorithm outperforms *DG* and *PPM* in terms of accuracy and usefulness for same network traffic. WM_o and WM algorithm has effective prefetching with space restriction, low overhead in network traffic. WM_o and WM algorithms prediction with new transaction are handle less effectively, association rule framing based on new transaction influenced by support and confidence value are feasible. Our algorithm are efficient than WM_o and WM (Yannis *et al,* 2003) in terms of predicting support and confidence value, and less space requirement. *CBFP mining algorithm* is efficient in terms of reducing number of candidates and network traffic. To evaluate the performance of the algorithm, we generated synthetic log files with a set of transactions. Generated synthetic log contain all site documents that have links to other documents. *CBFP mining* algorithm construct a *consensus tree* containing random variable fan-out uniformly for each node. When updating with raw log data some node has no incoming links, some has greatest fan-out, constrained with *Rule Constraints w*ith respect to document sizes of synthetic log, tree size comparable equal or less, not greater. Each path of the *consensus tree* from one article to other is constrained by *Level constraints*. The paths are created in groups stored in the bucket. The paths are actually the full length and the dependencies between nodes found in *consensus tree*, controlled by *Level constraint*. Test results shown in above graph states that *level* and *rule* constraints combination of values does not change the templates with respect to their pattern paths. *CBFP mining* algorithm compare data values obtained (Yannis *et al,* 2003) based on hit ratio and cache utilization. The usage of *CBFP mining algorithm* for an LRU cache is evaluated by assigning few hundred kilobytes to Client side cache for handling web page documents. The results of this experiment with standard data values (Yannis *et al., 2003)* compared with *CBFP* mining values are reported in Figure 10. From this Figure, it is clear that *CBFP mining algorithm* is beneficial, helping a cache to improve its hit rate and by increasing cache size will satisfy future reference. The same Figure shows relative performance of the prefetching algorithm. For small cache size, all algorithms have similar performance, and for very large cache size, the performance of all algorithms converges since almost all the sited documents are cached. Figure 11 and 12 show the results of experiments by comparing our algorithm data with prefetching algorithm's standard data, thus clearly outperforming all algorithms in terms of accuracy and usefulness. We have conducted experiment to show working of our algorithm with different set of confidence values, shown in Figure 13. As per this Figure, the accuracy of our algorithm depends on the selection of the confidence value. It clearly depicts that confidence value (0.29) provided by Adaptive threshold and shows better accuracy and more beneficial result than manual values. From all these results, our algorithm presents clearly best performance during workload increase and out performs other technique, with no compromise on accuracy and interest factor of template.

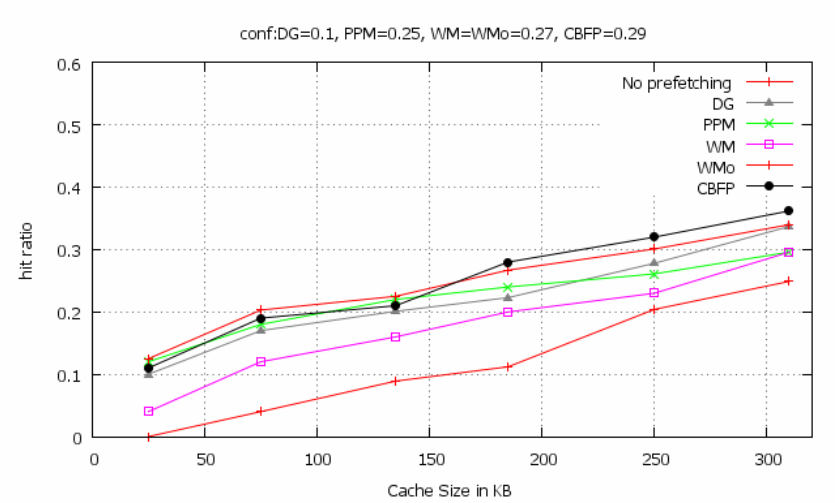

Figure 10 Cache hits as a function of the cache size.

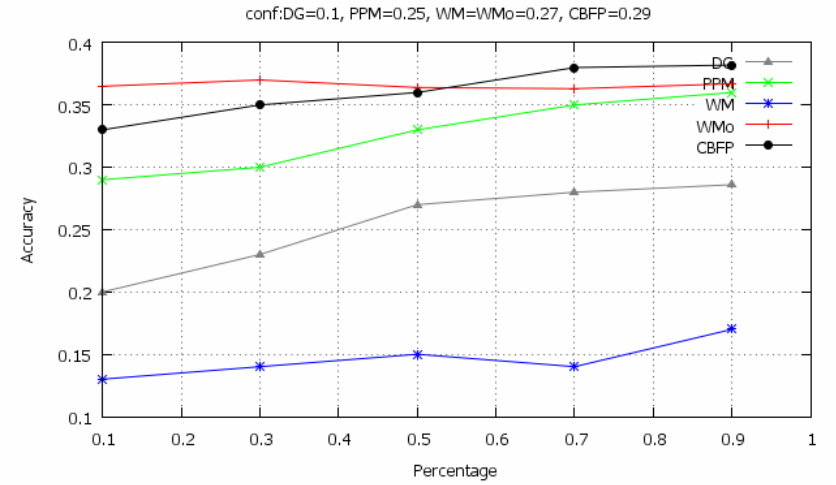

Figure 11 Comparison of accuracy with *CBFP mining algorithm*.

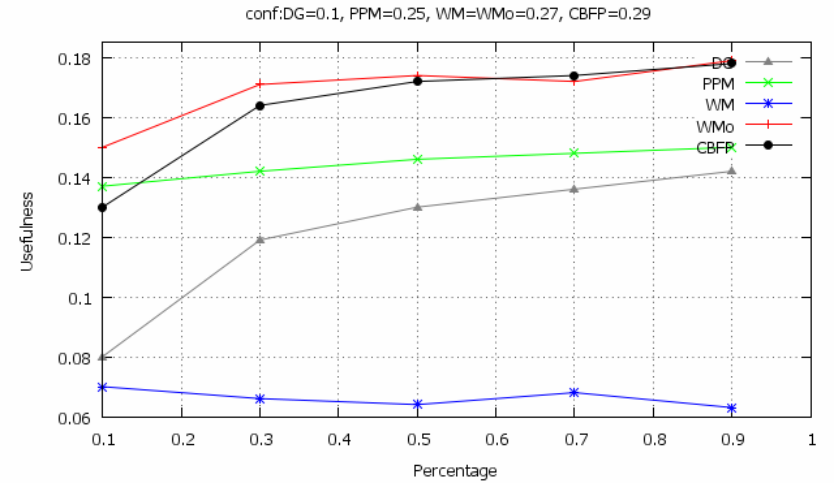

Figure 12 Comparison of *CBFP* technique based on usefulness.

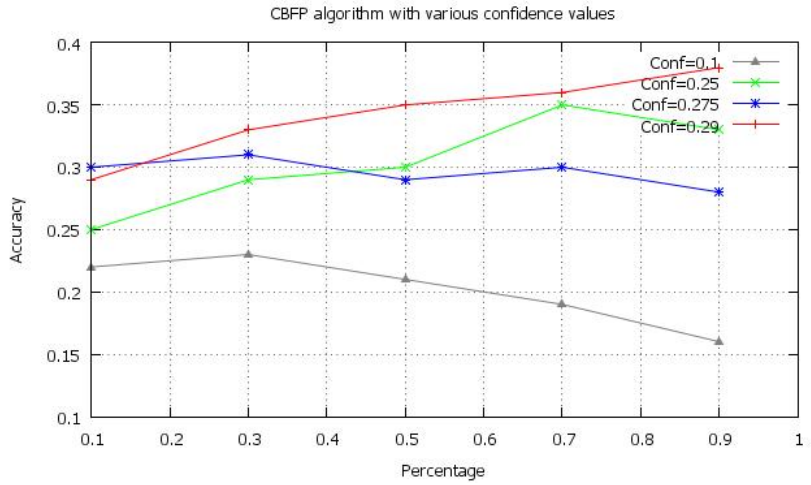

Figure 13 Accuracy of *CBFP* technique for various confidence values.

CBFP mining algorithm makes it more specific by adding more conditions; hence we differ from Quinlan (1993); Cohen (1995). Without incorporating background knowledge, as in Han et al. (1993); David (2008), using the multiple level abstractions from the *consensus tree* mining the templates is more general than Kamber (1997). Using the WordNet for calculating the semantic factor, we reduce the dimensionality, by reducing the number of ways in expressing the concept in terms of replacement, stemming and conflation (Frakes 1992), clustering or automatic thesauri (Rasmussen 1992; Srinivadsan 1992), and Latent Semantic Indexing (Deerwester et al. 1990). Our generated synthetic data, which do not contain complete and necessary information, is based on co-occurrence estimation. *CBFP mining algorithm* is a better memory utilization compared to Heung *et al*. (2009). HowNet and WordNet produced same result for representing user interestingness (Maristella *et al*. 2007). Running time for our algorithm is pruned than using classification and prediction (Liu and Keselji, 2007). *CBFP mining algorithm* is tested with the proxy log data as training log data, which contains web accessed information of students surfing Anna University web site. We preferred to concentrate on Computer science web site of Anna University of different zones currently containing several files and providing services average of 35,000 requests per day. Around 700MB of web access data are generated in a month. A wide range of information is provided such as general information about the department and the various programs and course offered. Some information, such as programs and courses, is cogged towards perspective students, other information, such as timetable, exam results and assignment details, is geared towards current students. Experiments were designed to compare access patterns of visitors from Anna University but within a state. Students search for their relevant information from client side and post their queries. If searched article is available in *consensus tree*, a direct link will be provided to user's otherwise query sent to remote server. We find relevance of the article submitted to the query, by allowing the user to rate whether it's relied to their search or not. The articles provided by our technique without aid of remote server reduce latency. The feedbacks obtained from each accessed/visited web page of the students assist in calculating precision and coverage factor. The generalized patterns obtained from *consensus tree* helps in different level of abstraction for user. The major new finding was that visitors spend more time on the pages compared to students with the website structure. We noticed that there was quite a large number of long transaction and thought that they were worth analyzing. Based on the user interest our work predicts the order of web objects based on the submitted queries. Our algorithm suits the user's nature and significantly reduces the perceived latency.

5. Conclusion

In this paper we describe a new algorithm which discovers templates or query patterns from large, raw user logs of local proxy cache server that provides the topics in which users are mostly interested with an eye for their future interest. Unnecessary submission of query to search engine is avoided and in turn searching time is minimized. *CBFP mining* Technique suits the web hyper textual nature, considerably reduces the perceived latency, and reveals the important factors that affect the performance of web prefetching algorithms. It reveals the order of dependencies between the web document accesses. Our technique is designed to address specific limitations and its characteristics includes issue of solving the interleaving of requests belonging to random patterns within user's request, their dependencies between web document accesses and ordering of request. Templates with similar semantic can be generalized and their necessity over longer period of time is predicted with reliability. Keyword misplacement and character replacement within keywords are handled by our technique. Our technique outperforms the previous technique, which will improve search engine performance by producing reliable generalized templates without incorporating knowledge with optimal precision of the search engine. Template provides the insight to the web editors as to what topics users are mostly interested in and needed in the near future. Templates obtained from our algorithm are useful to improve the performance of the search engine. The results were validated by experiments using real web traces. Our experiments demonstrated that the *CBFP*

mining algorithm improves hit rate. Our technique offers effective predictive prefetching, achieves larger accuracy in prediction for heavy workloads, with low overhead in network traffic proved with hit ratio. The process of transforming the original web logs into templates requires the use of heuristics as several steps. Thus high accuracy from the data mining algorithms cannot be expected. However, the evidence supporting the piece comes from a different algorithms and log data. Most web sites perform only a rudimentary analysis of web log based on hits in some period. Our work has shown that, it is possible to use our algorithm to find golden nuggets in the web logs. In the near future, we plan to study how to deal for dynamically changing database-driven web sites and query–level prediction.

References

Agarwal.R., Aggarwal.C., Prasad.V.V.V., 2001. A tree projection algorithm for generation of frequent item-sets. *Journal of Parallel and Distributed Computing*. Vol. 61, No. 3, pp. 350-371.

Armstrong.R., Freitag.D., Joachims.T., Mitchell.T., 1995. WebWatcher: A Learning Apprentice for the World Wide Web. *In proceedings of the 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*.

Balabanovic.M., Shoham.Y., 1995. Learning information retrieval agents: Experiments with automated web browsing. *AAAI Spring Symposium on Information Gathering From Heterogeneous, Distributed Resources,* March 1995.

Berners-Lee.T., Cailliau.R., Luotonen.A., Nielsen.H., Secret.A., 1994. The World-Wide Web. *Communications of the ACM*. Vol.37, No.8, pp.76-82.

Boyan.J., Freitag.D., Joachims.T., 1996. A Machine Learning Architecture for Optimizing Web Search Engines. *In Proceedings of the AAAI Workshop on Internet Based Information Systems*.

Chen Li, Jiayin Qi, Huaying Shu, 2008. A HowNet Based Web Log Mining Algorithm. *In Proceedings of Research and Practical Issues of Enterprise Information Systems II, Springer*. Vol.255, pp.923-931, DOI: 10.1007/978-0-387-76312-5.

Cohen.E., Krishnamurthy.B., Rexford.J., 1998. Improving end-to-end performance of the web using server volumes and proxy filters. *In Proceedings of ACM SIGCOMM*. pp.241-253.

Cohen.W.W., 1995. Fast Effective Rule Induction. *In Proceedings of 12th International Conference on Machine Learning*.

Cooley.R., Mobasher.B., Srivastava.J., 1999. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*. Vol.1, No.1, pp.5-32.

Craven.M., DiPasquo.D., et al,1999. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*. Elsevier. Vol.118, No.1, pp.69-113.

David. L.Olson., 2008. Ethical aspects of web log data mining. *International Journal of Information Technology and Management*. Vol.7, No.2, pp.190-200.

DeRaedt.L., Kramer.S., 2001. The levelwise version space algorithm and its application to molecular fragment finding. *In Proceedings of the 17th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. Vol.2, pp.853-859.

Deerwester.S., Dumais.S.T., Landauer.T., Harshman., 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*. Vol.41, pp.391-407.

Etzioni.O., Weld.D., 1994. A softbot-based interface to the internet. *Communications of the ACM*, *Special Issue on Intelligent Agents.* Vol.37, No.7, pp.72–76.

Frakes.W.B., 1992. Stemming Algorithm. *Information retrieval: data structures and algorithms*. pp.131-160

Garofalakis.M.N., Rastogi.R., Shim.K., 1999. SPIRIT: Sequential pattern mining with regular expression constraints. *In Proceedings of 25th International Conference on Very Large Data Bases*. pp.223-234.

Grahne.G., Lakshmanan.L.V.S., Wang.X., 2000. Efficient mining of constrained correlated sets. *In Proceedings of 16th International Conference on Data Engineering (ICDE'00), IEEE Computer Society*. pp.512-521.

Griffioen.J., Appleton.R., 1995. Reducing File System Latency Using a Predictive Approach. *In Proceedings of 1994 USENIX Annual Technical Conference (USENIX '95)*. pp.197-207.

Han.J., Cai.Y.,Cercone.N., 1993. Data-Driven Discovery of Quantitative Rules in Relational Databases. *IEEE Transaction on Knowledge and Data Engineering*. Vol.5, No.1, pp.29-40.

Han.J., Lakshmanan.L.V.S., Ng.R.T., 1999. Constraint-Based Multidimensional Data Mining. *IEEE Computer Society Press*. Vol.32, No.8, pp.46-50.

Han.J., Pei.J., Yin.Y., 2000. Mining frequent patterns without candidate generation. *ACM SIGMOD Record*. Vol.29, No.2, pp.1-12.

Hawwash.B., Nasraoui.O., 2010. Mining and tracking web user trends from Large Web server logs. *Statistical Analysis and Data Mining*. Vol.3, No.2, pp.106-125.

Heung Ki Lee, Baik Song An, Eun Jung Kim, 2009. Adaptive Prefetching Scheme Using Web Log Mining in Cluster-Based Web Systems. *In Proceedings of the 2009 IEEE International Conference on Web Services (ICWS '09)*. *IEEE Computer Society*. pp903-910.

Hu R., Chen W., Bai P., Lu Y., Chen Z., Yang Q., 2008. Web query transaltion via web log mining. *In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. pp.749-750.

Jia-Ching.Y., Vincent.S.T., Philip.S.Y., 2009. Efficient incremental mining of qualified web traversal patterns without scanning original databases. *International Conference on Data Mining Workshops.* pp.338-343.

Joachims.T., 2002. Optimizing search engines using click through data. *In Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02), ACM Press*. pp133–142.

Kamber.M., et al, 1997. Generalization and decision tree induction: efficient classification in data mining. *In Proceedings of 7th International Workshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications.*

Kumar R., 2009. Mining web logs: applications and challenges. *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. pp.3-4.

Lee S., DeRaedt L., 2004. Constraint based mining of first order sequences in SeqLog. *Database Support for Data Mining Applications, Springer*. Vol.2682, pp.154-173.

Ling.C., Gao.J., Zhang.H., Qian.W., 2001. Mining Generalized Query Patterns from Web Logs. *In Proceedings of the 34th Annual Hawaii International Conference on System Sciences(HICSS-34)*. Vol.5, pp.5020.

Lin Jie Bin, Liu Ming De, Chen Xiang, 2003. Data mining and OLAP Theory & Practice. *Beijing: Tsinghua University Press*, pp.194-244.

Liu H., Keselj V., 2007. Combined mining of Web server logs and web contents for classifying user navigation patterns and predicting users' future requests. *Data & Knowledge Engineering*. Vol.61, No.2, pp.304-330.

Lou.W., Liu.G., Lu.H., Yang.Q., 2002. Cut-and-pick transactions for proxy log mining. *In Proceedings of the 8th International Conference on Extending Database Technology (EDBT 2002)*. pp.88-105.

Man.W.L., Sherry.L.C., Kyriacos.C., Xiaohui.L., 2009. Mining students' behavior in web-based learning programs. *Expert systems with Applications.* Vol.36, No.2-2, pp.3459-3464.

Mannila.H., Toivonen.H., 1997. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery, Springer*. Netherlands. Vol.1, No.3, pp.241-258.

Maristella Agosti, Giorgio Maria Di Nunzio, 2007. Gathering and Mining Information from Web Log Files. *Lecture Notes in Computer Science, Springer*. Vol.4877, pp.104-113.

Miller.G., 1995. WordNet: a lexical database for English. *Communications of the ACM.* Vol.38, No.11, pp.39-41.

Mobasher.B., Cooley.R., Srivastava.J., 2000. Automatic personalization based on web usage mining. *Communications of the ACM, ACM*. Vol.43, No.8, pp.142–151.

Mobasher.B., Dai.H., Luo.T., Nakagawa.M., 2002. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery, Springer*.

Ng.R.T., Lakshmanan.L.V.S., Han.J., Pang.A., 1998. Exploratory mining and pruning optimizations of constrained associations rules. *In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*. Vol.27, No.2, pp.13-24.

Nguyen.M.T., Kawamura.T., Nakagawa.H., Tahara.Y., Ohsuga.A., 2010. Automatic mining of human activity attributes from weblogs. *IEEE/ACIS 9th International Conference on Computer and Information Science (ICIS). pp.633-638.*

Padmanabhan.V., Mogul.J., 1996. Using Predictive Prefetching to Improve World Wide Web Latency, *ACM SIGCOMM Computer Comm. Rev.,* Vol.26, No.3.

Panagiotis.G., Iraklis.V., Magdalini.E., 2010. Mining frequent generalized patterns for web personalization in the presence of taxonomies. *International Journal of Data Warehousing and Mining(JDWM).* Vol.6, No.1, pp.58-76.

Pazzani.M., Muramatsu.J., Billsus.D., 1996. Syskill & Webert: Identifying interesting web sites. *In Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96), AAAI Press*. pp54–61.

Pei.J., Han.J., Lakshmanan.L.V.S., 2001. Mining frequent itemsets with convertible constraints. *In Proceedings of 17th International Conference on Data Engineering (ICDE'01)*.

Pei.J., Han.J., Mao.R., 2000. CLOSET: An efficient algorithm for mining frequent closed itemsets. *In Proceedings of ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'00)*. pp21-30.

Pei.J., Han.J., Behzad Mortazavi-Asl, Zhu.H., 2000. Mining access pattern efficiently from web logs. *Knowledge Discovery and Data Mining, Current Issues and New Applications, Springer*. Vol.1805, pp.394-407.

Perkowitz.M., Etzioni.O., 2000. Towards adaptive web sites: Conceptual framework and case study. *Artificial* Intelligence. Vol.118, No.1, pp.245–275.

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

Quinlan.J.R., 1993. C4.5: Programs for Machine Learning. *San Mateo, Morgan Kaufmann*.

Ou J.-C., Lee C.-H., Chen M.-S., 2008. Efficient algorithms for incremental Web log mining with dynamic thresholds. *The VLDB Journal-The International Journal on Very Large Data Bases, Springer*. Vol.17, No.4, pp.827-845.

Rasmussen, 1992. Clustering Algorithm. *W.B. Frakes and R. Baeza-Yates, Eds., Information Retrieval Data structures and algorithms*. pp419-442.

Ratnesh.K.J., Kasana.R.S., Suresh Jain, 2009. Efficient web log mining using doubly linked tree. *International Journal of Computer Science and Information Security (IJCSIS).* Vol.3, No.1, arXiv:0907.5433v1 [cs.IR].

Rosie Jones, 2009. Privacy in web search query log mining. *Machine Learning and Knowledge Discovery in Databases, Springer*. pp.4.

Rui.W., 2010. Mining generalized fuzzy association rules from Web logs. *2010 Seventh International Conference on Fuzzy systems and Knowledge Discovery (FSKD)*.pp.2474-2477.

Scheffer.T., 2004. Email answering assistance by semi-supervised text classification. *Intelligent Data Analysis*. Vol.8, No.5.

Spiliopoulou.M., 1999. The laborious way from data mining to web log mining. *International Journal of Computer Systems Science and Engineering, Special Issues on Semantics of the Web*. Vol.14, pp.113–126.

Srikant.R., Agrawal.R., 1996. Mining sequential patterns: Generalizations and performance improvements. *In Proceedings of 5th International Conference on Extending Database Technology (EDBT'96)*. pp3–17.

Srikant.R., Vu.Q., Agrawal.R., 1997. Mining association rules with item constraints. *In Proceedings of 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*. pp.67–73.

Srivastava.J., Cooley.R., Deshpande.M., Tan.P.N., 2000. Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations Newsletter*. Vol.1, No.2, pp.12-23.

Sundaresan.N.,Yi.J., 2000. Mining the Web for Relations. *Computer Networks-The International Journal of Computer and Telecommunications Networking*. Vol.33, No.1, pp.699-711.

Tan.P.N., Kumar.V., 2002. Discovery of web robot sessions based on their navigational patterns. *Data Mining and Knowledge Discovery. Springer*.Vol.6, No.1, pp.9-35.

White.R.W., Drucker.S.M., 2007. Investigating behavioral variability in web search. *In Proceedings of the 16th International Conference on World Wide Web (WWW-2007)*. pp.21-30.

Wu.K.L., Yu.P.S., Ballman.A., 1998. Speedtracer: A Web usage mining and analysis tool. *IBM Systems Journal*. Vol.37, No.1, pp.89-105.

YaJun.D., HaiMing.L., 2010. Strategy for mining association rules for web pages based on formal concept analysis. *Applied Soft Computing.* Vol. 100, No.3, pp.772-783.

Yannis.M., Alexandros.N., Dimitrios.K., 2003. A data mining algorithm for generalized web prefetching, *IEEE Trsnasaction on Knowledge and Data Engineering*. Vol.15, No.5, pp.1155-1169.

Zaine.O.R., Xin M., Han J., 1998. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. *In Proceedings of Fifth International forum on Research and Technology Advances in Digital Libraries (ADL'98)*.pp.19-29.

Zaki.M.J., Hsiao.C.J., 2002. CHARM: An efficient algorithm for closed itemset mining. *In Proceedings of 2nd SIAM International Conference on Data Mining*. pp.457–473.

Biographical notes

Ramachandra.V. Pujeri, Born in Bijapur, Karnataka state in India, in 1973, received the B E in Electronics and Communication Engineering from Karnataka University, Dharwad, ME in Computer Science and Engg from PSG College of Technology, Coimbatore, Ph.D in Information and Communication Engineering from Anna University, Chennai, MBA in Human Resource Management, from Pondicherry University, Pondicherry, in 1996, 2002, 2007 and 2008 respectively. He is active life member of ISTE, SSI, MIE, ACS and IEE. His has written three textbooks. He is having around 16 years of teaching experience in the various top ten engineering colleges in India. He is an active expert committee member of AICTE, NBA, DoEACC, NACC and various Universities in India. Currently, under him fifteen research scholars pursuing their Ph.D. His research interests lie in the areas of Computer Networking, Operating System, Software Engineering, Software Reliability, Modeling and Simulation, Quality of Services and Data Mining. Currently, he is working as Vice-Principal of KGiSL Institute of Technology, Coimbatore.

G.M. Karthik, Born in Madurai, Tamil Nadu state in India, in 1981, received the B.E. in Computer Science and Engineering from SACS MAVMM Engineering College, Madurai, M.E. in Computer Science and Engineering from PSNA College of Engineering and Technology, Dindugal, in 2003 and 2005 respectively. He is having 6 years of teaching experience in more than three engineering colleges in India. This paper was written while he was working on the project on Data Mining using intelligent techniques as a Research scholar at Anna University, Coimbatore, India. His primary research interests are related to Data Mining and Web Mining. Currently, he is working as Assistant Professor of Computer Science Engineering Department of SACS MAVMM Engineering College, Madurai, India.

Received September 2010
Accepted December 2010
Final acceptance in revised form December 2010