An Effective System for Mining Web Log

Zhenglu Yang Yitong Wang Masaru Kitsuregawa

Institute of Industrial Science, The University of Tokyo
4-6-1 Komaba, Meguro-Ku, Tokyo 153-8305, Japan
{yangzl, ytwang, kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract. The WWW provides a simple yet effective media for users to search, browse, and retrieve information in the Web. Web log mining is a promising tool to study user behaviors, which could further benefit web-site designers with better organization and services. Although there are many existing systems that can be used to analyze the traversal path of web-site visitors, their performance is still far from satisfactory. In this paper, we propose our effective Web log mining system consists of data preprocessing, sequential pattern mining and visualization. In particular, we propose an efficient sequential mining algorithm (LAPIN_WEB: LAst Position INduction for WEB log), an extension of previous LAPIN algorithm to extract user access patterns from traversal path in Web logs. Our experimental results and performance studies demonstrate that LAPIN_WEB is very efficient and outperforms well-known PrefixSpan by up to an order of magnitude on real Web log datasets. Moreover, we also implement a visualization tool to help interpret mining results as well as predict users' future requests.

1 Introduction

The World Wide Web has become one of the most important media to store, share and distribute information. At present, Google is indexing more than 8 billion Web pages [1]. The rapid expansion of the Web has provided a great opportunity to study user and system behavior by exploring Web access logs. Web mining that discovers and extracts interesting knowledge/patterns from Web could be classified into three types based on different data that mining is executed: Web Structure Mining that focuses on hyperlink structure, Web Contents Mining that focuses on page contents as well as Web Usage Mining that focuses on Web logs. In this paper, we are concerned about Web Usage Mining (WUM), which also named Web log mining.

The process of WUM includes three phases: data preprocessing, pattern discovery, and pattern analysis [14]. During preprocessing phase, raw Web logs need to be cleaned, analyzed and converted before further pattern mining. The data recorded in server logs, such as the user IP address, browser, viewing time, etc, are available to identify users and sessions. However, because some page views may be cached by the user browser or by a proxy server, we should know that the data collected by server logs are not entirely reliable. This problem can be partly solved by using some other kinds of usage information such as cookies. After each user has been identified, the entry for each user must be divided into sessions. A timeout is often used to break the entry into sessions. The following are some preprocessing tasks [14]: (a) Data Cleaning: The server log is examined to remove irrelevant items. (b) User Identification: To identify different users by overcoming the difficulty produced by the presence of proxy servers and cache. (c) Session Identification: The page accesses must be divided into individual sessions according to different Web users.

The second phase of WUM is pattern mining and researches in data mining, machine learning as well as statistics are mainly focused on this phase. As for pattern mining, it could be: (a) statistical analysis, used to obtain useful statistical information such as the most frequently accessed pages; (b) association rule mining [12], used to find references to a set of pages that are accessed together with a support value exceeding some specified threshold; (c) sequential pattern mining [13], used to discover frequent sequential patterns which are lists of Web pages ordered by viewing time for predicting visit patterns; (d) clustering, used to group together users with similar characteristics; (e) classification, used to group together users into predefined classes based on their characteristics. In this paper, we focus on sequential pattern mining for finding interesting patterns based on Web logs.

Sequential pattern mining, which extracts frequent subsequences from a sequence database, has attracted a great deal of interest during the recent surge in data mining research because it is the basis of many applications, such as Web user analysis, stock trend prediction, and DNA sequence analysis. Much work has been carried out on mining frequent patterns, as for example, in [13] [16] [10] [7] [4]. However, all of these works suffer from the problems of having a large search space and the ineffectiveness in handling long patterns. In our previous work [18], we proposed a novel algorithm to reduce searching space greatly. Instead of searching the entire projected database for each item, as PrefixSpan [7] does, we only search a small portion of the database by recording the last position of item in each sequence (LAPIN: LAst Position INduction). While support counting usually is the most costly step in sequential pattern mining, the proposed LAPIN could improve the performance significantly by avoiding cost scanning and comparisons. In order to meet special features of Web data and Web log, we propose LAPIN_WEB by extending our previous work.

In pattern analysis phase, which mainly filter out uninteresting rules obtained, we implement a visualization tool to help interpret mined patterns and predict users' future request.

Our contribution in this paper could be summarized as: 1) propose an effective Web log mining system that deals with log preprocessing, sequential pattern mining, and result visualizing; 2) propose an efficient sequential pattern mining algorithm by extending previous LAPIN techniques; 3) implement a visualization tool to interpret mining results and predict users' future behavior. Experimental results on real datasets demonstrate the effectiveness of the whole system as well as the high performance of the proposed mining algorithm, which outperforms existing algorithm by up to an order of magnitude.

The remainder of this paper is organized as follows. We present the related work in Section 2. In Section 3, we introduce our Web log mining system, including preprocessing, pattern discovery and pattern analysis parts. Experimental results and performance analysis are reported in Section 4. We conclude the paper and provide suggestions for future work in Section 5.

2 Related Work

Commonly, a mining system includes three parts, as mentioned in Section 1, data preprocessing, pattern discovery and pattern analysis. In this section, we first introduce some related work in data preprocessing and then we focus on pattern mining and pattern analysis.

Data preprocessing: Because of the proxy servers and Web browser cache existing, to get an accurate picture of the web-site access is difficult. Web browsers store pages that have been visited and if the same page is requested, the Web browser will directly displays the page rather than sending another request to the Web server, which makes the user access stream incomplete. By using the same proxy server, different users leave the same IP address in the server log, which makes the user identification rather difficult. [14] presented the solution to these problems by using Cookies or Remote Agents. Moreover, in the same paper, the authors have presented several data preparation techniques to identify Web users, i.e., the path completion and the use of site topology. To identify the user sessions, a fixed time period, say thirty minutes [14] [3], is used to be the threshold between two sessions.

Sequential pattern mining: Srikant and Agrawal proposed the GSP algorithm [16], which iteratively generates candidate k-sequences from frequent (k-1)-sequences based on the anti-monotone property that all the subsequences of a frequent sequence must be frequent. Zaki proposed SPADE [10] to elucidate frequent sequences using efficient lattice search techniques and simple join operations. SPADE divides the candidate sequences into groups by items, and transforms the original sequence database into a vertical ID-List database format. SPADE counts the support of a candidate k-sequence generated by merging the ID-Lists of any two frequent (k-1)-sequences with the same (k-2)-prefix in each iteration. Ayres et al. [4] proposed the SPAM algorithm, which uses SPADE's lattice concept, but represents each ID-List as a vertical bitmap. SPADE and SPAM can be grouped as candidate-generation-and-test method. On the other hand, Pei et al. proposed a pattern growth algorithm, PrefixSpan [7], which adopts a projection strategy to project sequences into different groups called $projected\ databases$. The PrefixSpan algorithm recursively generates a projected database for each frequent k-sequence to find the frequent (k+1)-sequences. A comprehensive performance study showed that PrefixSpan, in most cases, outperforms former apriori-based algorithms [8]. However, PrefixSpan still needs to scan large projected database and it does not work well for dense datasets, i.e. DNA sequences, which is an very important application. These observations motivate our work in [18], which proposed an efficient sequential pattern mining algorithm LAPIN by the idea of using the last position of each item to judge whether a k-length pattern could grow to a (k+1)-length pattern. LAPIN could improve the performance significantly by largely reduce the search space required. In this paper, we propose another pattern mining algorithm by combining the merits of both LAPIN and PrefixSpan to meet the special requirement of Web logs, which is very sparse.

Visualization tools: Pitkow et. al. proposed WebViz [9] as a tool for Web log analysis, which provides graphical view of web-site local documents and access patterns. By incorporating the Web-Path paradigm, Web masters can see the documents in their web-site as well as the hyperlinks travelled by visitors. Spiliopoulou et. al. presented Web Utilization Miner (WUM) [11] as a mining system for the discovery of interesting navigation patterns. One of the most important features of WUM is that using WUM mining language MINT, human expert can dynamically specify the interestingness criteria for navigation patterns. To discover the navigation patterns satisfying the expert criteria, WUM exploits Aggregation Service that extracts information on Web access log and retains aggregated statistical information. Hong et. al. proposed WebQuilt [5] as a Web logging and visualization system that helps Web design teams run usability tests and analyze the collected data. To overcome many of the problems with server-side and client-side logging, WebQuilt uses a proxy to log the activity. It aggregates logged usage traces and visualize in a zooming interface that shows the Web pages viewed. Also it shows the most common paths taken through the web-site for a given task, as well as the optimal path for that task.

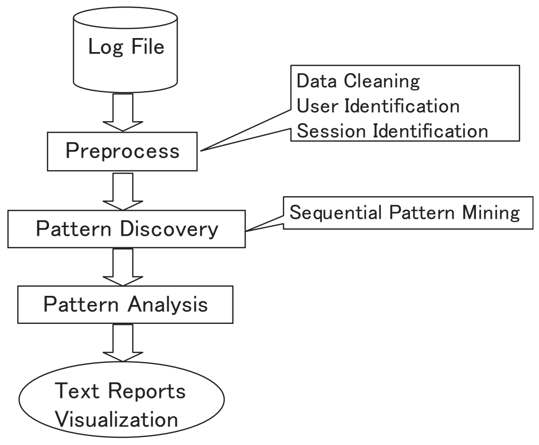


Fig. 1. Web Log Mining System Structure

3 Web Log Mining System

We designed a Web log mining system for sequential pattern mining and its visualization, as shown in Fig. 1. The input and output of the system is Web log files as well as visualized patterns or text reports. As mentioned in Section 1, the whole system includes:

- *Data Preprocessing.* This is the phase where data are cleaned from noise by overcoming the difficulty of recognizing different users and sessions, in order to be used as input to the next phase of pattern discovery. Data preprocessing phase always involves data cleaning, user identification and session identification.
- *Pattern Mining.* While various mining algorithms could be incorporated into the system to mine different types of patterns, currently, we only implemented sequential pattern mining on Web log data. We plan to add other part in future work.
- *Pattern Analysis.* In this phase, the mined patterns which in great numbers need to be evaluated by end users in an easy and interactive way: text report and and visualization tool.

We will discuss each part in more detail in following subsections.

3.1 Data Preprocessing

The raw Web log data is usually diverse and incomplete and difficult to be used directly for further pattern mining. In order to process it, we need to:

1) Data Cleaning. In our system, we use server logs in Common Log Format. We examine Web logs and remove irrelevant or redundant items like image, sound, video files which could be downloaded without an explicit user request. Other removal items include HTTP errors, records created by crawlers, etc., which can not truly reflect users' behavior.

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 an 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 [14]. 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) Session Identification. To identify 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 [2] [6], 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.

3.2 Sequential Pattern Mining

Problem Definition. A $Web\ access\ sequence$, s, is denoted as $\langle i_1, i_2, \ldots, i_k\rangle$, where i_j is a $page\ item$ for 1 ≤ j ≤ k. The number of page items in a Web access sequence is called the $length$ of the sequence. A Web access sequence with length l is called an l-$sequence$. A sequence, $s_a = \langle a_1, a_2, \ldots, a_n\rangle$, is contained in another sequence, $s_b = \langle b_1, b_2, \ldots, b_m\rangle$, if there exists integers $1 \leq i_1 < i_2 < \ldots < i_n \leq m$, such that $a_1 = b_{i_1}$, $a_2 = b_{i_2}$,..., $a_n = b_{i_n}$. We denote s_a a $subsequence$ of s_b, and s_b a $supersequence$ of s_a. Given a Web access sequence $s = \langle i_1, i_2, \ldots, i_l\rangle$, and an page item α, $s \diamond \alpha$ denotes that s concatenates with α, as $Sequence\ Extension$ (SE), $s \diamond \alpha$=$\langle i_1, i_2, \ldots, i_l, \alpha\rangle$. If $s' = p \diamond s$, then p is a $prefix$ of s' and s is a $suffix$ of s'.

A $Web\ access\ sequence\ database$, S, is a set of tuples $\langle uid, s\rangle$, where uid is a user_id and s is a Web access sequence. A tuple $\langle uid, s\rangle$ is said to contain a sequence β, if β is a $subsequence$ of s. The support of a sequence, β, in a sequence database, S, is the number of tuples in the database containing β, denoted as $support(\beta)$. Given a user specified positive integer, ε, a sequence, β, is called a frequent Web access sequential pattern if $support(\beta) \geq \varepsilon$. For sequential pattern mining in the pattern discovery phase, the objective is to find the complete set of Web access sequential patterns of database S in an efficient manner. Let our running database be the sequence database S shown in Table 1 with min_support = 2. We will use this sample database throughout the paper.

Here, we propose an efficient sequential pattern mining algorithm to mine Web logs by extending our previous work LAPIN [18]. Let us first briefly introduce the idea of LAPIN:

LAPIN Algorithm. For any time series database, the last position of an item is the key used to judge whether or not the item can be appended to a given prefix (k-length)

Table 1. Sequence Database

UID	Sequence
10	$acbcdadc$
20	$bcbcbab$
30	$dbcbadca$

Table 2. Item Last Position List

UID	Last Position of Different Item
10	$b_{last} = 3$ $a_{last} = 6$ $d_{last} = 7$ $c_{last} = 8$
20	$c_{last} = 4$ $a_{last} = 6$ $b_{last} = 7$
30	$b_{last} = 4$ $d_{last} = 6$ $c_{last} = 7$ $a_{last} = 8$

sequence (assumed to be s). For example, in a sequence, if the last position of item α is smaller than, or equal to, the position of the last item in s, then item α cannot be appended to s as a (k+1)-length sequence extension in the same sequence.

Example 1. When scanning the database in Table 1 for the first time, we obtain Table 2, which is a list of the last positions of the 1-length frequent sequences in ascending order. Suppose that we have a prefix frequent sequence $\langle a \rangle$, and its positions in Table 1 are 10:1, 20:6, 30:5, where uid:pid represents the sequence ID and the position ID. Then, we check Table 2 to obtain the first indices whose positions are larger than $\langle a \rangle$'s, resulting in 10:1, 20:3, 30:2, i.e., (10:b$_{last}$ = 3, 20:b$_{last}$ = 7, and 30:d$_{last}$ = 6). We start from these indices to the end of each sequence, and increment the support of each passed item, resulting in $\langle a \rangle : 2, \langle b \rangle : 2, \langle c \rangle : 2$, and $\langle d \rangle : 2$, from which, we can determine that $\langle aa \rangle$, $\langle ab \rangle$, $\langle ac \rangle$ and $\langle ad \rangle$ are the frequent patterns.

From the above example, we can show that the main difference between LAPIN and most existing algorithms is the searching space. PrefixSpan scans the entire projected database to find the frequent pattern. SPADE temporally joins the entire ID-List of the candidates to obtain the frequent pattern of next layer. LAPIN can obtain the same result by scanning only part of the search space of PrefixSpan and SPADE, which indeed, are the last positions of the items. The full justification and more detail about LAPIN can be found in [18].

However, we can not get the best performance by directly applying LAPIN to Web log ming because of the different properties between datasets. Comparing with general transaction data sequences that are commonly used, Web logs have following characteristics:

(a) no two items/pages are accessed at the same time by the same user.
(b) very sparse, which means that there are huge unique items and few item repetition in one user sequence.
(c) user preference should be considered during mining process.

Based on above points, we extended LAPIN to LAPIN_WEB with:

(1) dealing with only Sequence Extension (SE) case, no Itemset Extension (IE) case.
(2) using sequential search instead of binary search. In more detail, LAPIN_WEB does not use binary search in the item position list, but use pointer+offset sequential search strategy, which is similar to that used in PrefixSpan.
(3) incorporating user preference into mining process to make the final extracted pattern more reasonable.

```
**LAPIN_WEB Algorithm** :
**Input** : A sequence database, and the minimum support threshold, $\varepsilon$
**Output** : The complete set of sequential patterns

**Function** : Gen_Pattern($\alpha$, $S$, $CanI_s$)
**Parameters** : $\alpha$ = length k frequent sequential pattern; $S$ = prefix border position set of (k-1)-length sequential pattern; $CanI_s$ = candidate sequence extension item list of length k+1 sequential pattern
**Goal** : Generate (k+1)-length frequent sequential pattern

**Main():**
1. Scan DB once to do:
    1.1 $B_s \leftarrow$ Find the frequent 1-length $SE$ sequences
    1.2 $L_s \leftarrow$ Obtain the item-last-position list of the 1-length $SE$ sequences
2. For each frequent $SE$ sequence $\alpha_s$ in $B_s$
    2.1 Call Gen_Pattern ($\alpha_s$, 0, $B_s$)

**Function Gen_Pattern**($\alpha, S, CanI_s$)
3. $S_\alpha \leftarrow$ Find the prefix border position set of $\alpha$ based on $S$
4. $FreItem_{s,\alpha} \leftarrow$ Obtain the $SE$ item list of $\alpha$ based on $CanI_s$ and $S_\alpha$
5. For each item $\gamma_s$ in $FreItem_{s,\alpha}$
    5.1 Combine $\alpha$ and $\gamma_s$ as $SE$, results in $\theta$ and output
    5.2 Call Gen_Pattern ($\theta$, $S_\alpha$, $FreItem_{s,\alpha}$)
```

Fig. 2. LAPIN_WEB Algorithm pseudo code

LAPIN_WEB: Design and Implementation. We used a lexicographic tree [4] as the search path of our algorithm. Furthermore, we adopted a lexicographic order, which was defined in the same way as in [17]. This used the Depth First Search (DFS) strategy.

For Web log, because it is impossible that a user clicks two pages at the same time, Itemset Extension (IE) case in common sequential pattern mining does not exist in Web log mining. Hence, we only deal with Sequence Extension (SE) case. The pseudo code of LAPIN_WEB is shown in Fig. 2. In Step 1, by scanning the DB once, we can obtain all the 1-length frequent patterns. Then we sort and construct the SE item-last-position list in ascending order based on each 1-length frequent pattern' last position, as shown in Table 2.

Definition 1 (Prefix border position set) *Given two sequences, A=$\langle A_1 A_2 \ldots A_m \rangle$ and B=$\langle B_1 B_2 \ldots B_n \rangle$, suppose that there exists C=$\langle C_1 C_2 \ldots C_l \rangle$ for $l \leq m$ and $l \leq n$, and that C is a common prefix for A and B. We record both positions of the last item C_l in A and B, respectively, e.g., C_l=A_i and C_l=B_j. The position set, (i, j), is called the* prefix border position set *of the common prefix C, denoted as $\mathcal{S}_c$. Furthermore, we denote $\mathcal{S}_{c,i}$ as the prefix border position of the sequence, i. For example, if A=$\langle abc \rangle$ and B=$\langle acde \rangle$, then we can deduce that one common prefix of these two sequences is $\langle ac \rangle$, whose prefix border position set is (3,2), which is the last item C's positions in A and B.*

In function $Gen\_Pattern$, to find the prefix border position set of k-length α (Step 3), we first obtain the sequence pointer and offset of the last item of α, and then perform a sequential search in the corresponding sequence for the (k-1)-length prefix border position. This method is similar to pseudo-projection in PrefixSpan, which is efficient for sparse datasets.

Definition 2 (Local candidate item list) *Given two sequences, A=$\langle A_1 A_2 \ldots A_m \rangle$ and B=$\langle B_1 B_2 \ldots B_n \rangle$, suppose that there exists C=$\langle C_1 C_2 \ldots C_l \rangle$ for $l \leq m$ and $l \leq n$, and*

```
Input : $S_\alpha$ = prefix border position set of length k frequent sequential pattern $\alpha$;
BV$_s$ = bit vectors of the ITEM_IS_EXIST_TABLE; $L_s$ = $SE$ item-last-position list;
$CanI_s$ = candidate sequence extension items; $\varepsilon$ = user specified minimum support
Output : $FreItem_s$ = local frequent $SE$ item list

1. For each sequence, F, according to its priority (descending)
2.     $S_{\alpha,F} \longleftarrow$ obtain prefix border position of F in $S_\alpha$
3.     if ($Size_{local\_cand\_item\_list} > Size_{suffix\_sequence}$)
4.         bitV $\longleftarrow$ obtain the bit vector of the $S_{\alpha,F}$ indexed from BV$_s$
5.         For each item $\beta$ in $CanI_s$
6.             Suplist[$\beta$] = Suplist[$\beta$] + bitV[$\beta$];
7.             $CanI_{s,p} \longleftarrow$ obtain the candidate items based on prior sequence
8.     else
9.         $L_{s,F} \longleftarrow$ obtain $SE$ item-last-position list of F in $L_s$
10.          M = Find the corresponding index for $S_{\alpha,F}$
11.         while ( M < L$_{s,F}$.size)
12.             Suplist[M.item]++;
13.             M++;
14.             $CanI_{s,p} \longleftarrow$ obtain the candidate items based on prior sequence
15. For each item $\gamma$ in $CanI_{s,p}$
16.     if (Suplist[$\gamma$] $\geq \varepsilon$)
17.         $FreItem_s$.insert($\gamma$);
```

Fig. 3. Finding the SE frequent patterns

that C is a common prefix for A and B. Let $D = (D_1 D_2 \ldots D_k)$ be a list of items, such as those appended to C, and $C' = C \diamond D_j$ $(1 \leq j \leq k)$ is the common sequence for A and B. The list D is called the $local\ candidate\ item\ list$ of the prefix C'. For example, if A=$\langle abce \rangle$ and B=$\langle abcde \rangle$, we can deduce that one common prefix of these two sequences is $\langle ab \rangle$, and $\langle abc \rangle$, $\langle abe \rangle$ are the common sequences for A and B. Therefore, the item list (c,e) is called the $local\ candidate\ item\ list$ of the prefixes $\langle abc \rangle$ and $\langle abe \rangle$.

Step 4, shown in Fig. 2, is used to find the frequent SE (k+1)-length pattern based on the frequent k-length pattern and the 1-length candidate items. Commonly, support counting is the most time consuming part in the entire mining process. In [18], we have found that LCI-$oriented$ and $Suffix$-$oriented$ have their own advantages for different types of datasets. Based on this discovery, in this paper, during the mining process, we dynamically compare the suffix sequence length with the local candidate item list size and select the appropriate search space to build a single general framework. In other words, we combine the two approaches, LAPIN_LCI and LAPIN_Suffix, together to improve efficiency at the price of low memory consuming. The pseudo code of the frequent pattern finding process is shown in Fig. 3.

From a system administrator's view, the logs of special users (i.e. domain experts) are more important than other logs and thus, should be always considered more prior, as shown in Fig. 3 (Step 1). The appended candidate items are also judged based on this criteria (Step 7 and Step 14).

3.3 Pattern Visualization

We could see from pattern mining process that given a support, usually there are great number of patterns produced and effective method to filter out and visualize mined pattern is necessary. In addition, web-site developers, designers, and maintainers also need to understand their efficiency as what kind of visitors are trying to do and how they are doing it. Towards this end, we developed a navigational behavior visualization tool based on Graphviz [1].

At present, our prototype system has only implemented the basic sequential pattern discovery as the main mining task, which requires relevant simple user-computer interface and visualization. As more functions are added and experiment done, we will make the tool more convenient to the users.

4 Performance Study

In this section, we will describe our experiments and evaluations conducted on the real-world datasets. We performed the experiments using a 1.6 GHz Intel Pentium(R)M PC machine with a 1 G memory, running Microsoft Windows XP. The core of LAPIN_WEB algorithm is written in C++ software. When comparing the efficiency between LAPIN_WEB and PrefixSpan, we turned off the output of the programs to make the comparison equitable.

4.1 Real Data

We consider that results from real data will be more convincing in demonstrating the efficiency of our Web log mining system. There are two datasets used in our experiments, DMResearch and MSNBC. DMResearch was collected from the web-site of China Data Mining Research Institute [2], from Oct. 17, 2004 to Dec. 12, 2004. The log is large, about 56.9M, which includes 342,592 entries and 8,846 distinct pages. After applying data preprocessing described in Section 2.1, we identified 12,193 unique users and average length of the sessions for each user is 28. The second dataset, MSNBC, was obtained from the UCI KDD Archive [3]. This dataset comes from Web server logs for msnbc.com and news-related portions of msn.com on Sep. 28, 1999. There are 989,818 users and only 17 distinct items, because these items are recorded at the level of URL category, not at page level, which greatly reduces the dimensionality. The 17 categories are ”frontpage”, ”news”, ”tech”, ”local”, ”opinion”, ”on-air”, ”misc”, ”weather”, ”health”, ”living”, ”business”, ”sports”, ”summary”, ”bbs”, ”travel”, ”msn-news”, and ”msn-sports”. Each category is associated with a category number using an integer starting from ”1”. The statistics of these datasets is given in Table 3.

Table 3. Real Dataset Characteristics

Dataset	# Users	# Items	Min. len.	Max. len.	Avg. len.	Total size
DMResearch	12193	8846	1	10745	28	56.9M
MSNBC	989818	17	1	14795	5.7	12.3M

4.2 Comparing PrefixSpan with LAPIN_WEB

Fig. 4 shows the running time and the searched space comparison between PrefixSpan and LAPIN_WEB. Fig. 4 (a) shows the performance comparison between PrefixSpan and LAPIN_WEB for DMResearch data set. From Fig. 4 (a), we can see that LAPIN_WEB is much more efficient than PrefixSpan. For example, at support 1.3%,

[1] http://www.research.att.com/sw/tools/graphviz
[2] http://www.dmresearch.net
[3] http://kdd.ics.uci.edu/databases/msnbc/msnbc.html

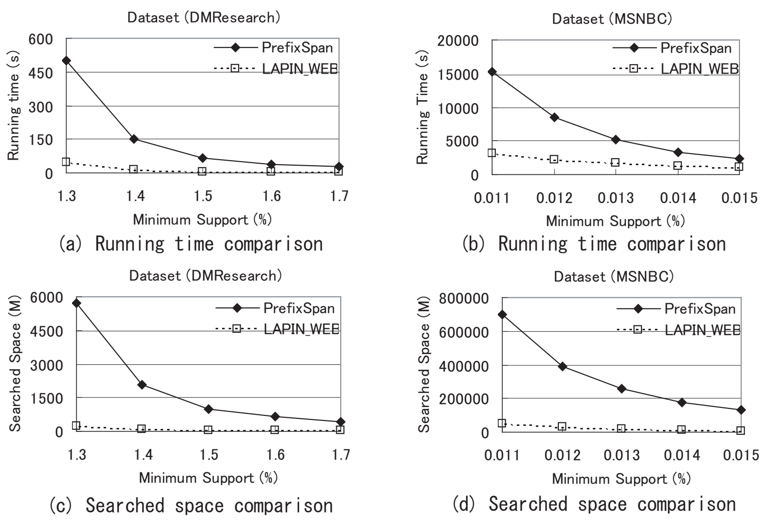


Fig. 4. Real datasets comparison

LAPIN_WEB (runtime = 47 seconds) is more than an order of magnitude faster than PrefixSpan (runtime = 501 seconds). This is because the searched space of Prefixspan (space = 5,707M) was much larger than that in LAPIN_WEB (space = 214M), as shown in Fig. 4 (c).

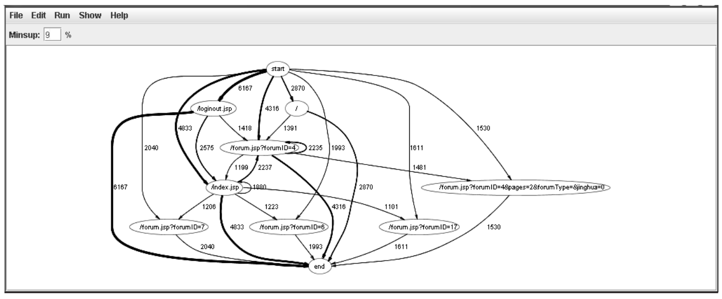
Fig. 4 (b) shows the performance comparison between PrefixSpan and LAPIN_WEB for MSNBC data set. From Fig. 4 (b), we can see that LAPIN_WEB is much more efficient than PrefixSpan. For example, at support 0.011%, LAPIN_WEB (runtime = 3,215 seconds) is about five times faster than PrefixSpan (runtime = 15,322 seconds). This is because the searched space of Prefixspan (space = 701,781M) was much larger than that in LAPIN_WEB (space = 49,883M), as shown in Fig. 4 (d).

We have not compared PrefixSpan and LAPIN_WEB on user's preference, because the former one has no such function.

4.3 Visualization Result

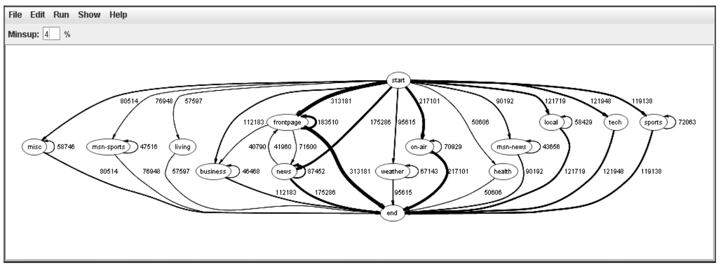
To help web-site developers, and Web administrators analyze the efficiency of their web-site by understanding what and how visitors are doing on a web-site, we developed a navigational behavior visualization tool. Fig. 5 and Fig. 6 show the visualization result of traversal pathes for the two real datasets, respectively. Here, we set minimum support to 9% for DMResearch and 4% for MSNBC. The thickness of edge represents the support value of the corresponding traversal path. The number value, which is right of the traversal path, is the support value of the corresponding path. The "start" and "end" are not actual pages belong to the site, they are actually another sites placed somewhere on the internet, and indicate the entry and exit door to and from the site.

From the figures, We can easily know that the most traversed edges, the thick ones, are connecting pages "start" $\longrightarrow$ "\loginout.jsp" $\longrightarrow$ "end" in Fig. 5, and "start" $\longrightarrow$ "frontpage" $\longrightarrow$ "end" in Fig. 6. Similar interesting traversal path can also be understood, and used by web-site designers to make improvement on link structure as well as document content to maximize efficiency of visitor path.



Minsup: 9 %

Fig. 5. DMResearch visualization result



Minsup: 4 %

Fig. 6. MSNBC visualization result

5 Conclusions

In this paper, we have proposed an effective framework for Web log mining system to benefit web-site designer to understand user behaviors by mining Web log data. In particular, we propose an efficient sequential pattern mining algorithm LAPIN_WEB by extending previous work LAPIN with special consideration of Web log data. The proposed algorithm could improve the mining performance significantly by scanning only a small portion of projected database and extract more reasonable web usage patterns. Experimental results and evaluations performed on real data demonstrate that LAPIN_WEB is very effective and outperforms existing algorithms by up to an order of magnitude. The visualization tool could be further used to make final patterns easy to interpret and thus improve the presentation and organization of web-site. Our framework of Web log mining system is designed in such a way that it could be easily extended by incorporating other new methods or algorithms to make it more functional and adaptive. We are now considering other pattern mining algorithms as we mentioned earlier such as clustering and association rule. Moreover, we are planning to build more sophisticated visualization tools to interpret the final results.

References

1. Google Website. http://www.google.com.
2. K. Wu, P.S. Yu. and A. Ballman, "Speedtracer: A Web usage mining and analysis tool," In *IBM Systems Journal*, 37(1), pp. 89-105, 1998.

3. H. Ishikawa, M. Ohta, S. Yokoyama, J. Nakayama, and K. Katayama, “On the Effectiveness of Web Usage Mining for Page Recommendation and Restructuring,” In *2nd Annual International Workshop of the Working Group ”Web and Databases” of the German Informatics Society*, Oct. 2002.
4. J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, “Sequential Pattern Mining using A Bitmap Representation,” In *8th ACM SIGKDD Int’l Conf. Knowledge Discovery in Databases (KDD’02)*, pp. 429-435, Alberta, Canada, Jul. 2002.
5. J.I. Hong and J.A. Landay, “WebQuilt: A Framework for Capturing and Visualizing the Web Experience,” In *10th Int’l Conf. on the World Wide Web (WWW’01)*, pp. 717-724, Hong Kong, China, May 2001.
6. J. Pei, J. Han, B. Mortazavi-Asl and H. Zhu, “Mining access pattern efficiently from web logs,” In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD’00)*, Kyoto, Japan, pp. 396-407, 2000.
7. J. Pei, J. Han, M. A. Behzad, and H. Pinto, “PrefixSpan:Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth,” In *17th Int’l Conf. of Data Engineering (ICDE’01)*, Heidelberg, Germany, Apr. 2001.
8. J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.C. Hsu, “Mining Sequential Patterns by Pattern-growth: The PrefixSpan Approach,” In *IEEE Transactions on Knowledge and Data Engineering*, Volume 16, Number 11, pp. 1424-1440, Nov. 2004.
9. J. Pitkow and K. Bharat, “WebViz: A Tool for World-Wide Web Access Log Analysis,” In *1st Int’l Conf. on the World Wide Web (WWW’94)*, Geneva, Switzerland, May 1994.
10. M. J. Zaki, “SPADE: An Efficient Algorithm for Mining Frequent Sequences,” In *Machine Learning*, Vol. 40, pp. 31-60, 2001.
11. M. Spiliopoulou and L.C. Faulstich, “WUM : A Web Utilization Miner,” In *EDBT Workshop on the Web and Data Bases (WebDB’98)*, Valencia, Spain, 1998. Springer Verlag.
12. R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” In *20th Int’l Conf. on Very Large Databases (VLDB’94)*, pp. 487-499, Santiago, Chile, Sep. 1994.
13. R. Agrawal and R. Srikant, “Mining sequential patterns,” In *11th Int’l Conf. of Data Engineering (ICDE’95)*, pp. 3-14, Taipei, Taiwan, Mar. 1995.
14. R. Cooley, B. Mobasher, and J. Srivastava, “Data Preparation for Mining World Wide Web Browsing Patterns,” In *J. Knowledge and Information Systems*, pp. 5.32, vol. 1, no. 1, 1999.
15. R. Kosala, H. Blockeel, “Web Mining Research: A Survey,” In *SIGKDD Explorations*, ACM press, 2(1): 1-15, 2000.
16. R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements,” In *5th Int’l Conf. Extending Database Technology (EDBT’96)*, pp. 13-17, Avignon, France, Mar. 1996.
17. X. Yan, J. Han, and R. Afshar, “CloSpan: Mining closed sequential patterns in large datasets,” In *3rd SIAM Int’l Conf. Data Mining (SDM’03)*, pp. 166-177, San Francisco, CA, May 2003.
18. Z. Yang, Y. Wang, and M. Kitsuregawa, “LAPIN: Effective Sequential Pattern Mining Algorithms by Last Position Induction,” *Technical Report (TR050617)*, Info. and Comm. Eng. Dept., Tokyo University, Japan, Jun. 2005.
http://www.tkl.iis.u-tokyo.ac.jp/∼yangzl/Document/LAPIN.pdf