Web Usage Mining: Extracting Unexpected Periods from Web Logs

F. Masseglia, A. Marascu
INRIA Sophia Antipolis
AxIS Project-Team
2004 route des Lucioles
06902 Sophia Antipolis
France
fmassegl@sophia.inria.fr
amarascu@sophia.inria.fr

P. Poncelet
EMA-LGI2P/Site EERIE
Parc Scientifique Georges Besse
30035 Nîmes Cedex 1
France
Pascal.Poncelet@ema.fr

M. Teisseire
LIRMM UMR CNRS 5506
161 Rue Ada
34392 Montpellier Cedex 5
France
teisseire@lirmm.fr

Abstract

Existing Web Usage Mining techniques are currently based on an arbitrary division of the data (e.g. "one log per month") or guided by presumed results (e.g "what is the customers behaviour for the period of Christmas purchases?"). Those approaches have two main drawbacks. First, they depend on this arbitrary organization of the data. Second, they cannot automatically extract "seasons peaks" among the stored data. In this paper, we propose to perform a specific data mining process (and particularly to extract frequent behaviours) in order to automatically discover the densest periods. Our method extracts, among the whole set of possible combinations, the frequent sequential patterns related to the extracted periods. A period will be considered as dense if it contains at least one frequent sequential pattern for the set of users connected to the Web site in that period. Our experiments show that the extracted periods are relevant and our approach is able to extract both frequent sequential patterns and the associated dense periods.

Keywords: Sequential Patterns, Temporal Mining, Web Usage, Dense Periods.

1 Introduction

Analyzing the behaviour of a Web Site users, also known as Web Usage Mining, is a research field which consists in adapting the data mining methods to access log files records. These files collect data such as the IP address of the connected machine, the requested URL, the date and other information regarding the navigation of the user. Web Usage Mining techniques provide knowledge about the behaviour of the users in order to extract relationships in the recorded data [4, 14, 16, 20]. Among available techniques, the sequential patterns [1] are particularly well adapted to the log study. Extracting sequential patterns on a log file, is supposed to provide the following kind of relationship: "*On the INRIA's Web Site, 10% of users visited consecutively the homepage, the available positions page, the ET*[1] *offers, the ET missions and finally the past ET competitive selection*".

This kind of behaviour is only supposed to exist, because extracting sequential patterns on a log file means to manage several problems (caches and proxies, great diversity of pages on the site, search engines which allow the user to directly access a specific part of the Web site, etc.). Among those problems, let us focus on the arbitrary division of the data which is done today. This division comes either from an arbitrary decision in order to provide one log per x days (*e.g.* one log per month), or from a wish to find particular behaviours (*e.g.* the behaviour of the Web site users from November 15 to December 23, during Christmas purchases). In order to better understand our goal, let us consider student behaviours when they are connected for a working session. Let us assume that these students belong to two different groups having twenty students. The first group was connected on 31/01/05 while the other one was connected on 01/02/05, (*i.e.* the second group was connected one day later). During the working session, students have to perform the following navigation: First they access URL "www-sop.inria.fr/cr/tp_accueil.html", then "www-sop.inria.fr/cr/ tp1_accueil.html" which will be followed by "www-sop.inria. fr/cr/tp1a.html".
Let us consider, as it is usual in traditional approaches, that we analyze access logs per month. During January, we only can extract twenty similar behaviours, among 200,000 navigations on the log, sharing the working

[1] ET: Engineers, Technicians.

session. Furthermore, even when considering a range of one month or of one year, this sequence of navigation does not appear sufficiently on the logs (20/20000) and will not be easy to extract. Let us now consider that we are provided with logs for a very long period (*e.g.* several years). With the method developed in this article, we can find that it exists at least one dense period in the range [31/01-01/02]. Furthermore, we know that, during this period, 340 users were connected. We are thus provided with the new following knowledge: 11% (*i.e.* 40 on 340 connected users) of users visited consecutively the URLs "tp_accueil.html", "tp1_accueil.html", and finally "tp1a.html".

Efficient tools are proposed today [22, 9] for analyzing logs at different level of granularity (day, month, year). They allow for instance to know how many time the site is accessed or how many requests have been done on each page. Nevertheless, as they depend on the chosen granularity, they suffer the previously addressed drawback: they cannot obtain frequent patterns on a very short period because usually such patterns do not appear sufficiently on the whole log. Close to our problem, [15] propose to extract episodes rules on a long sequence as well as the the optimal window size. Nevertheless our problem is very different since we do not consider that we are provided with a unique long sequence. In our context, *i.e.* access logs, sequences correspond to different behaviours of users on a Web Server. Then we have to manage a very huge set of data sequences and we have to extract both frequent sequences and the period where these sequences appear.

The remainder of this paper is organized as follows. Section 2 goes deeper into presenting sequential patterns and how they can be used on Web Usage Mining. In Section 3, we give an overview of Web Usage Mining approaches which are based on sequential patterns. Section 4 presents our motivation for a new approach. Our solution based on a new heuristic called PERIO is presented in Section 5. Experiments are reported Section 6, and Section 7 concludes the paper with future avenues for research.

2 Definitions

In this section we define the sequential pattern mining problem in large databases and give an illustration. Then we explain the goals and techniques of Web Usage Mining with sequential patterns. The sequential pattern mining definitions are those given by [21].

2.1 Sequential Pattern Mining

The problem of mining sequential patterns from static databases is defined as follows [1]:

Definition 1 *Let $I = \{i_1, i_2, ..., i_m\}$, be a set of m literals (items). I is a k-itemset where k is the number of items in I. A sequence is an ordered list of itemsets denoted by $< s_1 s_2 \dots s_n >$ where s_j is an itemset. The data-sequence of a customer c is the sequence in D corresponding to c. A sequence $< a_1 a_2 \dots a_n >$ is a $subsequence$ of another sequence $< b_1 b_2 \dots b_m >$ if there exist integers $i_1 < i_2 < \dots < i_n$ such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$.*

Example 1 *Let C be a client and S=$< (c)\ (d\ e)\ (h) >$, be that client's purchases. S means that "C bought item c, then he bought d and e at the same moment (i.e. in the same transaction) and finally bought item h".*

Definition 2 *The $support$ of a sequence s, also called $supp(s)$, is defined as the fraction of total data-sequences that contain s. If $supp(s) \geq minsupp$, with a minimum support value $minsupp$ given by the user, s is considered as a frequent sequential pattern.*

The problem of sequential pattern mining is thus to find all the frequent sequential patterns as stated in definition 2.

2.2 Access Log Files Analysis with Sequential Patterns

The general idea is similar to the principle proposed in [6]. It relies on three main steps. First of all, starting from a rough data file, a pre-processing step is necessary to clean "useless" information. The second step starts from this pre-processed data and applies data mining algorithms to find frequent itemsets or frequent sequential patterns. Finally, the third step aims at helping the user to analyze the results by providing a visualization and request tool.
Raw data is collected in access log files by Web servers. Each input in the log file illustrates a request from a client machine to the server (*http daemon*). Access log files format can differ, depending on the system hosting the Web site. For the rest of this presentation we will focus on three fields: client address, the URL asked for by the user and the time and date for that request. We illustrate these concepts with the access log file format given by the CERN and the NCSA [3], where a log input contains records made of 7 fields, separated by spaces [18]: **host user authuser [date:time] "request" status bytes**

The access log file is then processed in two steps. First of all, the access log file is sorted by address and by transaction. Then each "uninteresting" data is pruned out from the file.

Definition 3 Let Log be a set of server access log entries. An entry g, $g \in Log$, is a tuple:
$g =< ip_g, ([l_1^g.URL, l_1^g.time]\ ...\ [l_m^g.URL, l_m^g.time]) >$ such that for $1 \leq k \leq m$, $l_k^g.URL$ is the item asked for by

Client	d1	d2	d3	d4	d5
1	a	c	d	b	c
2	a	c	b	f	c
3	a	g	c	b	c

Figure 1. File obtained after a pre-processing step

the user g at time $l_k^g.time$ and for all $1 \leq j < k$, $l_k^g.time > l_j^g.time$.

The structure of a log file, as described in Definition 3, is close to the "`Client-Time-Item`" structure used by sequential pattern algorithms. In order to extract frequent behaviour from a log file, for each g in the log file, we first have to transform ip_g into a client number and for each record k in g, $l_k^g.time$ is transformed into a time number and finally $l_k^g.URL$ is transformed into an item identifier. Table 1 gives a file example obtained after that pre-processing. To each client corresponds a series of times and the URL requested by the client at each time. For instance, the client 2 requested the URL "f" at time $d4$.
The goal is thus, according to definition 2 and by means of a data mining step, to find the sequential patterns in the file that can be considered as frequent. The result may be, for instance, `< ( a ) ( c ) ( b ) ( c ) >` (with the file illustrated in Figure 1 and a minimum support given by the user: 100%). Such a result, once mapped back into URLs, strengthens the discovery of a frequent behaviour, common to n users (with n the threshold given for the data mining process) and also gives the sequence of events composing that behaviour.

3 Related Work

Several methods for extracting sequential patterns have been applied to Web access log files [13, 20, 2, 8, 23, 17, 14]. We report in this section some studies using this temporal aspect for analyzing a Web users behaviour.

The WUM tool [20] allows to discover navigation patterns which are considered as "interesting" from a statistical point of view. WUM proposes to extract patterns depending on their threshold and a user request.
In [13], the authors propose WebTool. This system takes into account all the steps of a Web Usage Mining process, from data selection to result display, *via* data transformation and patterns extraction. WebTool is based on a prefix tree (PSP [11]) to extract sequential patterns.
In [8] the authors propose to consider the temporal aspect of Web accesses in a user clustering method. This clustering algorithm is based on a sequences alignment method in order to evaluate their distance. The main contribution is to evaluate the quality of the proposed clusters. They are compared, during the experiments, to the clusters obtained thanks to a distance based on itemsets.
Authors of [23] consider the navigation patterns as Markov chains. They propose to build a Markov model for a link prediction method taking into account the previous navigations. The paper is devoted to the problems related to Markov models and the transition matrix built for each log.
Recent work on analyzing Web usage have focused on the quality of the results, their relevance and their utility. This is also the case for work related to the temporal aspect of navigation patterns. In [17] the authors show that the characteristics of the Web site have to be considered before deciding to use frequent itemsets or frequent sequences (as well as sequential patterns). Mainly, three characteristics are proposed: topology, connectivity degree and length of potential navigations. They show that sequential patterns are adapted for Web sites having long potential navigations (including Web sites involving dynamic pages).
According to the authors of [14], the study of the result quality has to consider sequential patterns with very low support. High or average thresholds often lead to useless (obvious) patterns. Nevertheless, extracting sequential patterns with very low support is very difficult because of the number of candidates generated. The authors thus propose to split down the problem in a recursive way in order to consider each sub-problem as a specific data mining step.

Whatever the goals pursued by these Web Usage Mining approaches, they always depend on the division of the data. In the next section, we propose to understand the goal of our proposal and the general principle of our heuristic.

4 Motivation and Principle

This section is devoted to motivating our proposal regarding the relevance and utility of the tackled knowledge. It also illustrates the issues involved and the general principle of our method.

4.1 Motivation

The outline of our method is the following: enumerating the sets of periods in the log that will be analyzed and then identifying which ones contain frequent sequential patterns. In this section we will define the notions of period and frequent sequential patterns over a period. Let us consider the set of transactions in Figure 2 (upper left table). Those transactions are sorted by timestamp, as they would be in

a log file. In this table containing 9 records, the customer c_1, for instance, has connected at time 1 and requested the URL a. Let us now consider the "in" and "out" timestamps of each client, reporting their arrival and departure (upper right table in Figure 2). The first request of client c_1 has occurred at time 1, and the last one at time 4. We can thus report the periods of that log. In the example of Figure 2 there are 5 periods. During the first period (from time 1 to time 2), the client c_1 was the only one connected on the Web site. Then, clients c_1 and c_2 are connected during the same period p_2 (from time 3 to time 4), and so on.

Cust_Id	time	URL
c_1	1	a
c_1	2	b
c_2	3	a
c_1	4	d
c_2	5	d
c_3	6	d
c_2	7	e
c_3	8	e
c_3	9	f

Cust_Id	In	Out
c_1	1	4
c_2	3	7
c_3	6	9

Period	Begin/End	Customers
p_1	[1..2]	c_1
p_2	[3..4]	$c_1,\ c_2$
p_3	[5]	c_2
p_4	[6..7]	c_2, c_3
p_5	[8..9]	c_3

Figure 2. A log containing three sequences and the associated periods

Let us now consider the navigation sequences of the log represented in Figure 2. Those sequences are reported in Figure 3, as well as the frequent sequential patterns extracted on the whole log and on the identified periods. With a minimum support of 100 % on the whole log, the only frequent sequential pattern is merely reduced to the item d: $< (d) >$. Let us now consider the periods identified above, as well as the customers connected for each period. For the periods p_1, p_3 and p_5, reduced to a single client, there is no relevant frequent pattern. For the period p_2 a sequential patterns is extracted: $< (a)\ (d) >$. This pattern is common to both clients connected during the period p_2: c_1 and c_2. Finally, during period p_4, the pattern $< (d)\ (e) >$ is extracted.

The following part of this section is devoted to more formal definitions of period, connected clients and stable periods. Let C be the set of clients in the log and D the set of recorded timestamps.

Definition 4 *P, the set of potential periods on the log is defined as follows:*
$P = \{(p_a, p_b)/(p_a, p_b) \in D \times D \ and \ a \leq b\}$.

In the following definition, we consider that $d_{min}(c)$ and $d_{max}(c)$ are respectively the arrival and departure time for c in the log (first and last request recorded for c).

Definition 5 *Let $C_{(a,b)}$ be the set of clients connected during the period (a, b). $C_{(a,b)}$ is defined as follows:*
$C_{(a,b)} = \{c/c \in C$
$and\ [d_{min}(c)..d_{max}(c)] \cap [a..b] \neq \emptyset\}$.

Finally, we give the definitions of *stable period* and *dense period*. The first one is a maximal period p_m during which C_{p_m} does not vary. With the example given in Figure 2, the period $[6..7]$ is a stable period. This is not the case for $[3..3]$ which is included in $[3..4]$ and contains the same clients (*i.e.* $C_{(3,3)} = C_{(3,4)}$). A *dense period* is a stable period containing at least a frequent sequential pattern. In the example given in section 1, the period corresponding to January 31 (*i.e.* during the working session) should be a dense period.

Definition 6 *Let P_{stable} be the set of stable periods. P_{stable} is defined as follows:*

$$P_{stable} = \{(m_a, m_b)/(m_a, m_b) \in P \ and$$

$$1)\ \nexists\, (m'_a, m'_b)/(b-a) < (b'-a')$$
$$and\ [a'..b'] \cap [a..b] \neq \emptyset$$
$$and\ C_{(m'_a, m'_b)} = C_{(m_a, m_b)}$$

$$2)\ \forall (x,y) \in [a..b], \forall (z,t) \in [a..b]/$$
$$x \leq y, z \leq t\ then\ C_{(x,y)} = C_{(z,t)}.\}$$

Condition 1, in definition 6, ensures that no largest period includes (m_a, m_b) and contains the same clients. Condition 2 ensures that there is no arrival or departure inside any period of P_{stable}.

Definition 7 *A stable period p is dense if C_p contains at least a frequent sequential pattern with respect to the minimum support specified by the user proportionally to $|C_p|$.*

The notion of dense period (definition 7), is the core of this paper. In the following, our goal will be to extract those periods, as well as the corresponding frequent patterns, from the log file. In order to give an illustration, let us consider the period p_e containing 100 clients ($|C_{p_e}| = 100$) and a minimum support of 5 %. Any sequential pattern included in at least 5 navigations of C_{p_e} will be considered as

Cust_Id	Sequence	log	p_1, p_3, p_5	p_2	p_4
c_1	$<(a)\,(b)\,(d)>$	$<(d)>$	–	$<(a)\,(d)>$	$<(d)\,(e)>$
c_2	$<(a)\,(d)\,(e)>$		–		
c_3	$<(d)\,(e)\,(f)>$		–		

Figure 3. Frequent sequential patterns obtained for customers connected at each period

frequent for that period. If there exists at least a frequent pattern in p_e then this period has to be extracted by our method. Extracting the sequential patterns of each period by means of a traditional sequential pattern mining method is not a suitable solution for the following reasons. First, sequential pattern mining algorithms (such as PSP [11] or PrefixSpan [19] for instance) can fail if one of the patterns to extract is very long. When considering navigations on a Web site, it is usual to find numerous requests for a same URL (pdf or php files for instance). Finally, during our experiments, with a total amount of 14 months of log files, we detected approximately $3,500,000$ stable periods. We believe that mining dense period by means of a heuristic is more relevant that several millions calls to a traditional algorithm for mining sequential patterns. The outline of our approach, intended to detect dense periods in the log file, is presented in the next section.

4.2 General Principle

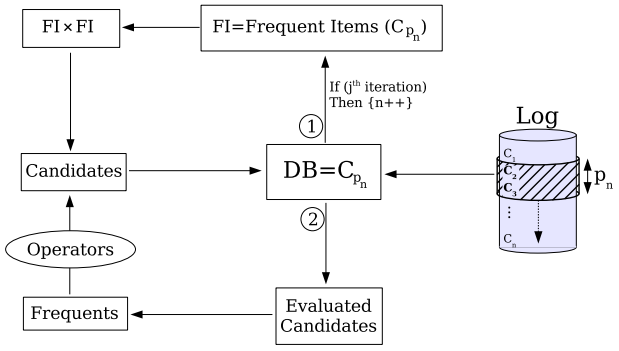

Figure 4. Overview of the operations performed by PERIO

Figure 4 gives an overview of the PERIO heuristic that we propose for solving the problem of dense period mining. First, starting from the log, the periods are detected. Those periods are then considered one by one and sorted by their "begin" timestamp. For each iteration n, the period p_n is scanned. The set of clients C_{p_n} is loaded in main memory ("DB" in Figure 4). Candidates having length 2 are generated from the frequent items detected in C_{p_n} (step "1" in Figure 4). Because of the large number of candidates generated, this operation only occurs every s steps (where s is a user defined parameter). Candidates are then compared to sequences of C_{p_n} in order to detect the frequent patterns (step "2" in Figure 4). Frequent patterns are injected in the neighborhood operators described in Section 5.1.1 and the new generated candidates are compared with the sequences of C_{p_n}. In order to obtain a result as fine as possible on each period, it is possible for the user to give the minimum number of iteration (j) on each period.

4.3 Limits of Sequential Pattern Mining

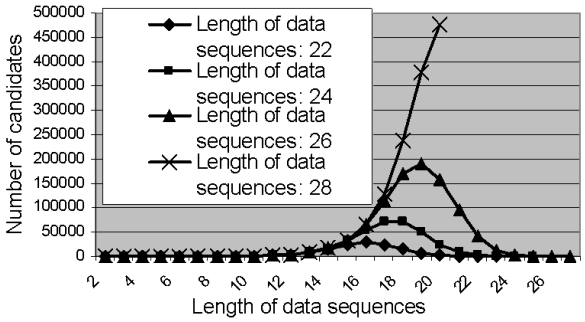

Figure 5. Limits of a framework involving PSP

Our method will process the log file by considering millions of periods (each period corresponds to a sub-log). The principle of our method will be to extract frequent sequential patterns from each period. Let us consider that the frequent sequences are extracted with a traditional exhaustive method (designed for a static transaction database). We argue that such a method will have at least one drawback leading to a blocking operator. Let us consider the example of the PSP [12] algorithm. We have tested this algorithm on databases containing only two sequences (s_1 and s_2). Both sequences are equal and contain repetitions of itemsets having length one. The first database contains 11 repetitions of the itemsets $(1)(2)$ (*i.e.* $s_1 =< (1)(2)(1)(2)...(1)(2) >$, lentgh($s_1$)=22 and $s_2 = s_1$). The number of candidates generated at each scan is reported in Figure 5. Figure 5 also reports the number of candidates for databases of sequences

having length 24, 26 and 28. For the base of sequences having length 28, the memory was exceeded and the process could not succeed. We made the same observation for PrefixSpan[2] [19] where the number of intermediate sequences was similar to that of PSP with the same mere databases. If this phenomenon is not blocking for methods extracting the whole exact result (one can select the appropriate method depending on the dataset), the integration of such a method in our process for extracting dense periods is impossible because the worst case can appear in any batch[3].

5 Extracting Dense Periods

In this section, we describe the steps allowing to obtain the dense periods of a Web access log. We also describe the neighborhood operators designed for PERIO, the heuristic presented in this paper.

5.1 Heuristic

Since our proposal is a heuristic-based miner, our goal is to provide a result having the following characteristics: For each period p in the history of the log, let $realResult$ be the set of frequent behavioural patterns embedded in the navigation sequences of the users belonging to p. $realResult$ is the result to obtain (*i.e.* the result that would be exhibited by a sequential pattern mining algorithm which would explore the whole set of solutions by working on the clients of C_p). Let us now consider $perioResult$ the result obtained by running the method presented in this paper. We want to minimize $\sum_{i=0}^{size(perioResult)} S_i/S_i \notin realResult$ (with S_i standing for a frequent sequence in $perioResult$), as well as maximize $\sum_{i=0}^{size(realResult)} R_i/R_i \in perioResult$ (with R_i standing for a frequent sequence in $realResult$). In other words, we want to find most of the sequences occurring in $realResult$ while preventing the proposed result becoming larger than it should (otherwise the set of all client navigations would be considered as a good solution, which is obviously wrong).

This heuristic is inspired from genetic algorithms and their neighborhood operators. Those operators are provided with properties of frequent sequential patterns in order to produce optimal candidates. The main idea of the PERIO algorithm is to scan P_{stable} the set of stable periods and, for each p in P_{stable} to propose candidates population thanks to previous frequent patterns and neighborhood operators. These candidates are then compared to the sequences of C_p in order to know their threshold (or at least their distance from a frequent sequence). These two phases (neighborhood operators and candidate valuation) are explained in this section.

5.1.1 Neighborhood Operators

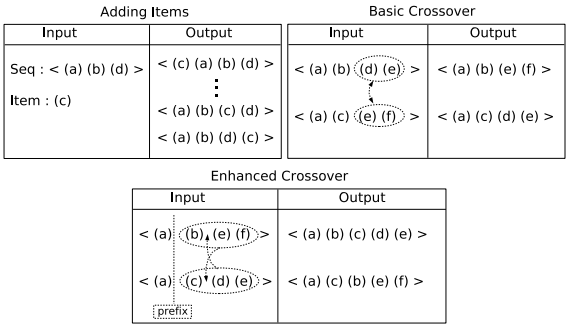

Figure 6. Some operators designed for extracting frequent navigation patterns

The neighborhood operators we used were validated by experiments performed on the Web logs of Inria Sophia Antipolis (see section 6). We chose "Genetic-like" operators as well as operators based on sequential pattern properties. We present here some of the most efficient operators for the problem presented in this paper. When we talk about random sequence, we use a biased random such that sequences having a high threshold may be chosen before sequences having a low threshold.

Finally, we evaluated the success rates for each of our operators thanks to the average number of frequent sequences compared to the proposed candidates. An operator having a success rate of 20 % is an operator for which 20 % of the proposed candidates are detected has frequent.

New frequent items: When a new frequent item occurs (after being requested by one or more users) it is used to generate all possible 2-candidate sequences with other frequent items. The candidate set generated is thus added to the global candidate set. Due to the number of candidate sequences to test, this operator only has a 15% ratio of accepted (*i.e.* frequent) sequences. This operator however remains essential since the frequent 2-sequences obtained are essential for other operators.

Adding items: This operator aims at choosing a random item among frequent items and adding this item to a random sequence s, after each item in s. This operator generates $length(s)+1$ candidate sequences. For instance,

[2] http://www-sal.cs.uiuc.edu/~hanj/software/prefixspan.htm

[3] In a web usage pattern, numerous repetitions of requests for pdf or php files, for instance, are usual.

with the sequence $<$ (a) (b) (d) $>$ and the frequent item c, we will generate the candidate sequences $<$ (c) (a) (b) (d) $>$, $<$ (a) (c) (b) (d) $>$, $<$ (a) (b) (c) (d) $>$ and finally $<$ (a) (b) (d) (c) $>$. This operator has a 20% ratio of accepted sequences, but the sequences found are necessary for the following operators.

Basic crossover: This operator (largely inspired by genetic algorithms operators) uses two different random sequences and proposes two new candidates coming from their amalgamation. For instance, with the sequences $<$ (a) (b) (d) (e) $>$ and $<$ (a) (c) (e) (f) $>$, we propose the candidates $<$ (a) (b) (e) (f) $>$ and $<$ (a) (c) (d) (e) $>$. This operator has a good ratio (50%) thanks to frequent sequences embedded in the candidates generated by previous operators.

Enhanced crossover: Encouraged by the result obtained when running the previous operator, we developed a new operator, designed to be an enhancement of the basic crossover, and based on the frequent sequences properties. This operator aims at choosing two random sequences, and the crossover is not performed in the middle of each sequence, but at the end of the longest prefix common to the considered sequences. Let us consider two sequences $<$ (a) (b) (e) (f) $>$ and $<$ (a) (c) (d) (e) $>$ coming from the previous crossover operator. The longest prefix common to these two sequences is $<$ (a) $>$. The crossover therefore starts after the item following a, for each sequence. In our example, the two resulting candidate sequences are, $<$ (a) (b) (c) (d) (e) $>$ and $<$ (a) (c) (b) (e) (f) $>$. This operator has a success ratio of 35%.

Final crossover: An ultimate crossover operator was designed in order to improve the previous ones. This operator is based on the same principle as the enhanced crossover operator, but the second sequence is not randomly chosen. Indeed, the second sequence is chosen as being the one having the longest common prefix with the first one. This operator has a ratio of 30%.

Sequence extension: This operator is based on the following idea: frequent sequences are extended with new pages requested. The basic idea aims at adding new frequent items at the end of several random frequent sequences. This operator has a success ratio of 60%.

Figure 6 gives an illustration of some operators described in this section.

5.1.2 Candidate Evaluation

The PERIO heuristic is described by the following algorithm:

Algorithm PERIO
In: P_{stable} the set of stable periods.
Out: SP The sequential patterns corresponding to the most frequent behaviours.
For ($p \in P_{stable}$) {
  // Update the items thresholds
  $itemsSupports$=getItemsSupports(C_p);
  // Generate candidates from frequent
  // items and patterns
  $candidates$=neighborhood(SP, $itemsSupport$);
  For ($c \in candidates$) {
    For ($s \in C_p$) {
      CandidateValuation(c, s);
    }
  }
  For ($c \in candidates$) {
    If (support(c) $> minSupport$ OR $criteria$){
      insert(c, SP);
    }
  }
}
End algorithm PERIO

Algorithm CANDIDATEEVALUATION
In: c a candidate to evaluate and s the navigation sequence of the client.
Out: $p[c]$ the percentage given to c.
  // If c is included in s, c is rewarded
  If ($c \subseteq s$) $p[c]$=100+length(c);
  // If c, having length 2, is not included then
  // give c the lowest mark.
  If (length$c$)$\leq$ 2) $p[c]$=0;
  // Else, give s a mark and give
  // largest distances a penalty
  $p[c]$=$\frac{length(LCS(c,s))*100}{length(c)} - length(c)$;
End algorithm CANDIDATEEVALUATION

For each stable period of P_{stable}, PERIO will generate new candidates and then compare each candidate to the sequence of C_p. The comparison aims at returning a percentage, representing the distance between the candidate and the navigation sequence. If the candidate is included in the sequence, the percentage should be 100% and this percentage decreases when the amount of interferences (differences between the candidate and the navigation sequence) increases. To evaluate this distance, the percentage is obtained by the fraction of the length of the longest com-

mon subsequence (LCS) [5] between s and c, on the length of s: $|LCS(s,c)|/|s|$. Furthermore, in order to obtain frequent sequences that are as long as possible, we use an algorithm that rewards long sequences if they are included in the navigation sequence. On the other hand, the algorithm has to avoid long, not included, sequences (in order for the clients not to give a good mark to any long sequence). To cover all these parameters, the calculation performed by the client machine is described in the algorithm CANDIDATEEVALUATION. Finally evaluated candidates having either their support greater than or equal to the minimal support value or corresponding to a "natural selective criteria" are stored into SP. This last criteria, which is user-defined, is a threshold corresponding to the distance between the candidate support and the minimal support. In our case, this criteria is used in order to avoid than the PERIOD heuristic leads towards a local optima.

5.2 Result Summary and Visualization

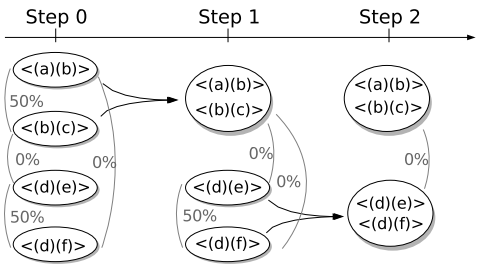

Figure 7. Clustering of sequential patterns before their alignment

Due to the number of candidates proposed by such a heuristic, the number of resulting sequences is very large. For instance, if the patterns <(a)(b)> and <(a)(b)(c)> are extracted by PERIO, then they will be both inserted in the result. In fact this problem cannot be reduced to the inclusion problem. As the size of extracted patterns is very long and as the delay of processing period has to be as short as possible, we could obtain patterns which are very close. Furthermore, extracted patterns could be very different since they represent different kind of behaviours. In order to facilitate the visualization of the issued result, we propose to extend the work of [10].

Our method performs as follows. We cluster together similar sequences. This operation is based on a hierarchical clustering algorithm [7] where the similarity is defined as follows:

Step 1:				
S_1:	<(a,c)	(e)	()	(m,n)>
S_2:	<(a,d)	(e)	(h)	(m,n)>
SA_{12}:	(a:2, c:1, d:1):2	(e:2):2	(h:1):1	(m:2, n:2):2
Step 2:				
SA_{12}:	(a:2, c:1, d:1):2	(e:2):2	(h:1):1	(m:2, n:2):2
S_3:	<(a,b)	(e)	(i,j)	(m)>
SA_{13}:	(a:3, b:1, c:1, d:1):3	(e:3):3	(h:1, i:1, j:1):2	(m:3, n:2):3
Step 3:				
SA_{13}:	(a:3, b:1, c:1, d:1):3	(e:3):3	(h:1, i:1, j:1):2	(m:3, n:2):3
S_4:	<(b)	(e)	(h,i)	(m)>
SA_{14}:	(a:3, b:2, c:1, d:1):4	(e:4):4	(h:2, i:2, j:1):3	(m:4, n:2):4

Figure 8. Different steps of the alignment method with the sequences from example 2

Definition 8 *Let s_1 and s_2 be two sequences. Let $|LCS(s_1,s_2)|$ be the size of the longest common subsequence between s_1 and s_2. The degree of similarity between s_1 and s_2 is defined as: $d = \frac{2\times|LCS(s_1,s_2)|}{|s_1|+|s_2|}$.*

The clustering algorithm performs as follows. Each sequential pattern is first considered as a cluster (C.f. Step 0, Figure 7). At each step the matrix of similarities between clusters is processed. For instance, sequences <(a)(b)> and <(b)(c)> are similar at 50% since they share the same itemset (b). If we now consider the two following sequences <(a)(b)> and <(d)(e)>, their similarity is 0%. The two close clusters are either {<(a)(b)>, <(b)(c)>} or {<(d)(e)>, <(d)(f)>} since they have a same distance. They are grouped together into a unique cluster. Step "2" of Figure 7 shows the three clusters: {<(a)(b)>, <(b)(c)>}, {<(d)(e)>} and {<(d)(f)>}. This process is repeated until there is no more cluster having a similarity greater than 0 with an existing cluster. The last step of Figure 7 gives the result of the clustering phase: {<(a)(b)>, <(b)(c)>} et {<(d)(e)>, <(d)(f)>}.

The clustering algorithm ends with clusters of similar sequences, which is a key element for sequences alignment. The alignment of sequences leads to a weighted sequence represented as follows: $SA =< I_1 : n_1, I_2 : n_2, ..., I_r, n_r >: m$. In this representation, m stands for the total number of sequences involved in the alignment. I_p $(1 \leq p \leq r)$ is an itemset represented as $(x_{i_1} : m_{i_1}, ...x_{i_t} : m_{i_t})$, where m_{i_t} is the number of sequences containing the item x_i at the n_p^{th} position in the aligned sequences. Finally, n_p is the number of occurrences of itemset I_p in the alignment. Example 2 describes the alignment process on 4 sequences. Starting from two sequences, the alignment begins with the insertion of empty items (at the beginning, the end or inside the sequence) until both sequences contain the same number of itemsets.

Example 2 *Let us consider the following sequences: $S_1 =<$ (a,c) (e) (m,n) $>$, $S_2 =<$ (a,d) (e) (h) (m,n) $>$, $S_3 =<$ (a,b) (e) (i,j) (m) $>$, $S_4 =<$ (b) (e) (h,i) (m) $>$. The steps leading to the alignment of those sequences are*

detailed in Figure 8. First, an empty itemset is inserted in S_1. Then S_1 and S_2 are aligned in order to provide SA_{12}. The alignment process is then applied to SA_{12} and S_3. The alignment method goes on processing two sequences at each step.

At the end of the alignment process, the aligned sequence (SA_{14} in Figure 8) is a summary of the corresponding cluster. The approximate sequential pattern can be obtained by specifying k: the number of occurrences of an item in order for it to be displayed. For instance, with the sequence SA_{14} from Figure 8 and $k = 2$, the filtered aligned sequence will be: <(a,b)(e)(h,i)(m,n)> (corresponding to items having a number of occurrences greater than or equal to k).

6 Experiments

PERIO was written in C++ and compiled using gcc without any optimizations flags. All the experiments were performed on a PC computer with Pentium 2,1 Ghz running Linux (RedHat). They were applied on Inria Sophia Antipolis logs. These logs are daily obtained. At the end of a month, all daily log are merged together in a monthly log. During experiments we worked on 14 monthly logs. They were merged together in order to be provided with a unique log for a 14 months period (from January 2004 to March 2005). Its size is 14 Go of records. There are 3.5 millions of sequences (users), the average length of these sequences is 2.68 and the maximal size is 174 requests.

6.1 Extracted Behaviours

We report here some of the extracted behaviours. Those behaviours show that an analysis based on multiple division of the log (as described in this paper) allows to obtain behavioural patterns embedded in short or long periods. Execution time of PERIO on this log with a minimal support value of 2% is nearly 6 hours. The support of 2% was the best setting for obtaining interesting patterns and limiting the size of the output. We have found 1981 frequent behaviours which were grouped together on 400 clusters with techniques described in Section 5.2.

Figure 10 focuses on the evolution of the following behaviours:

- $C1$ =<(semir/restaurant) (semir/restaurant/consult.php) (semir/restaurant/index.php) (semir/restaurant/index.php)>
- $C2$ =<(eg06) (eg06/dureve_040702.pdf) (eg06/fer_040701.pdf) (eg06)>
- $C3$ =<(requete.php3) (requete.php3) (requete.php3)>
- $C4$ =<(Hello.java) (HelloClient.java) (HelloServer.java)>
- $C5$ =<(mimosa/fp/Skribe) (mimosa/fp/Skribe/skribehp.css) (mimosa/fp/Skribe/index-5.html)>
- $C6$ =<(sgp2004) (navbar.css) (submission.html)>

All itemsets of behaviour $C4$ are prefixed by "oasis/anonym2/Prog Rpt/TD03-04/hello/". For $C3$ the prefix is "mascotte/ anonym3/web/td1/" and for $C6$ the prefix is "geometrica/events/".

The first behaviour ($C1$) corresponds to a typically periodic behaviour. Actually, the Inria's restaurant has been closed for a few weeks and people had to order a cold meal through a dedicated web site. This web site was located at "semir/restaurant". $C2$ is representative of behaviours related to the recent "general assembly" of French researchers, hosted in Grenoble (France, Oct 2004).

Behaviours $C3$ and $C4$ correspond to navigation performed by students on pages about computer science courses and stored on some Inria researcher pages.

When we have noticed the $C5$ behaviours, we asked the reasons of such behaviours to the pages owner. His interpretation is that such behaviours are due to the large number of exchanged mails on March 2004 through the mailing list of Skribe (generating numerous navigations on the web pages of this project). Two different peaks appear, (begin of April and middle of April) for the behaviour $C6$. Those peaks correspond in fact to the submission steps (respectively abstract and full papers) of articles for the SGP2004 Conference.

Some of the extracted behaviours do not occur on short periods only. Their occurrences are frequent on several weeks or even several months. Their support on the global log is related to the number of customers connected for each period. This is the case, for instance, of:

- $C7$ =<(css/inria_sophia.css) (commun/votre_profil_en.shtml) (presentation/ chiffres_en.shtml) (actu/actu_scient_colloque_ encours_fr.shtml)>

The evolution of $C7$ is reported in Figure 9. We can observe that this behaviour occurs for 5 consecutive months (from May to September).

6.2 Comparison to Sequential Pattern Mining

Section 6.1 is devoted to showing some extracted behaviours and their content. In this section we aim at show-

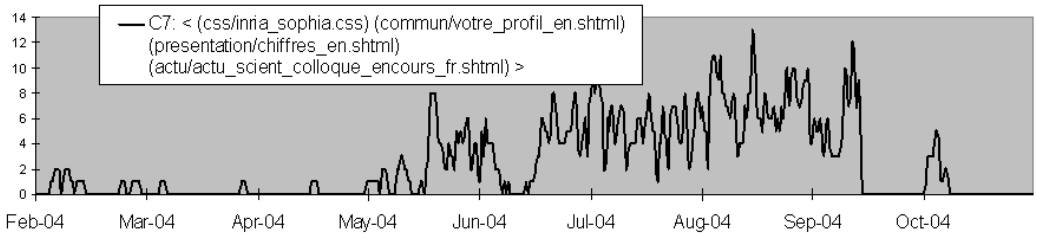

Figure 9. Peaks of frequency for a behaviour on a long period

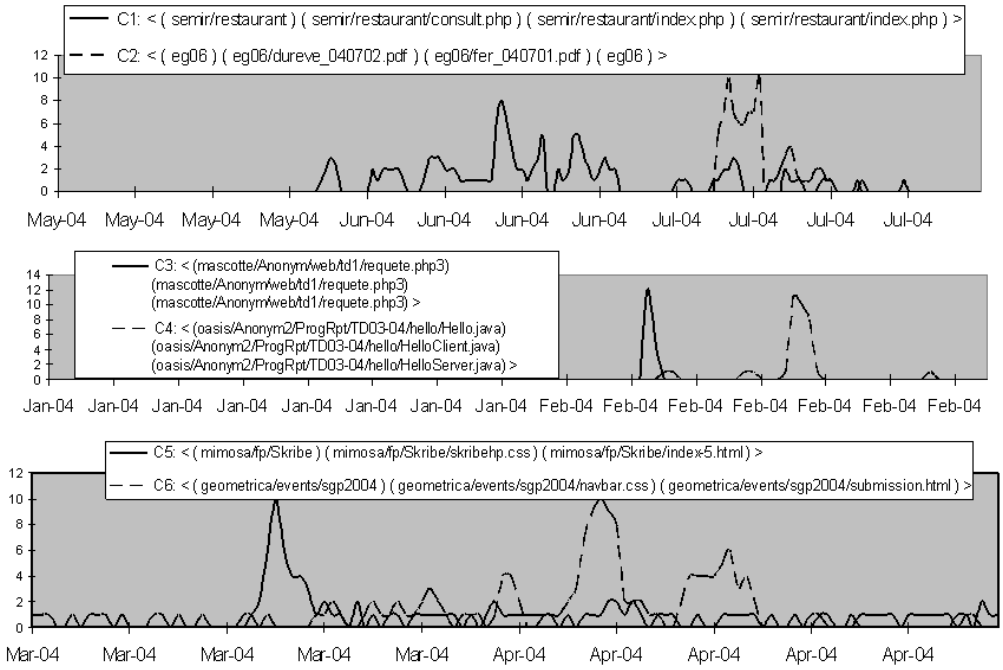

Figure 10. Peaks of frequency for $C1$, $C2$, $C3$, $C4$, $C5$ and $C6$

ing a comparison between our method on the one hand, and traditional method for sequential patterns on the other hand. We will show that the behaviours obtained by PERIO have such a low support that:

1. They cannot be extracted by a traditional sequential pattern mining algorithm.

2. The period they belong to cannot be identified by a traditional sequential pattern mining algorithm.

In Figure 11 we report several informations about the behaviours presented in section 6.1. The meaning of each value is given in Figure 12. We give those informations at three granularities (year, month and day). First of all, we give the maximum number of simultaneous occurrences of each behaviour in a stable period (column "Max"). Then we report the global support of this behaviour: the number of sequences containing the behaviour in the whole log file is given in column "Global" whereas the ratio is given in column "$\%_{Global}$".

A first comparison is given with PSP on the whole log file for each behaviour. We report in PSP_{Global} the execution time of PSP on the whole log file with a support of $\%_{Global}$. We can observe that for each behaviour, PSP is unable to extract the patterns corresponding to the given support. The main reason is that this support is much lower than any traditional method for mining sequential patterns would accept. The number of frequent items for $C6$ with a support of 0.0364% (bold "–") is 935. In this case, the number of candidates having length 2 is 1,311,805 so the main memory was rapidly overloaded and PSP could not

	Max	Global	$\%_{Global}$	PSP_{Global}	Month	$\%_{Month}$	PSP_{Month}	Day	$\%_{Day}$	PSP_{Day}
C_1	13	507	0.0197%	–	08–2004	0.031%	–	Aug–09	0.095%	20s
C_2	8	69	0.0027%	–	07–2004	0.004%	–	Jun–10	0.2%	–
C_3	10	59	0.0023%	–	07–2004	0.004 %	–	Jul–02	0.33%	10s
C_4	12	19	0.0007%	–	02–2004	0.006%	–	Feb–06	0.35%	18s
C_5	10	32	0.0012%	–	02–2004	0.01%	–	Feb–16	0.33%	21s
C_6	10	935	0.0364	–	02–2004	0.09%	–	Mar–15	0.35%	12s
C_7	10	226	0.0088%	–	04–2004	0.01%	–	Apr–03	0.23s	8s

Figure 11. Supports of the extracted behaviours at 3 granularities (Global, Month & Day)

Max	The maximum number of simultaneous occurrences of this behaviour in a stable period
Global	The support (total number of occurrences) of this behaviour in the global (14 months) log file
$\%_{Global}$	The support (percentage) corresponding to *Global* w.r.t the number of data sequences in the global log file
PSP_{Global}	The execution time of PSP on the global log file with a minimum support of $\%_{Global}$
Month	The month having the highest number of simultaneous occurrences of this behaviour in a stable period
$\%_{Month}$	The support (percentage) of this behaviour on $Month$
PSP_{Month}	The execution time of PSP on the log file corresponding to $Month$ with a minimum support of $\%_{Month}$
Day	The day having the highest number of simultaneous occurrences of this behaviour in a stable period
$\%_{Day}$	The support (percentage) of this behaviour on Day
PSP_{Day}	The execution time of PSP on the log file corresponding to Day with a minimum support of $\%_{Day}$

Figure 12. Legend for the table of Figure 11

succeed.

We also identified (by comparing months between each others) for each behaviour the month having the highest number of simultaneous occurrences of this behaviour in a stable period. In fact, the column "Month" corresponds to the month where this behaviour has the best support compared to other months. We report in column $\%_{Month}$ the support of each behaviour on the corresponding month and in column PSP_{Month} the execution time of PSP on the corresponding month with a support of $\%_{Month}$. We can observe that PSP is unable to extract the sequential patterns corresponding to each month.

Finally, we identified for each behaviour the day having the highest number of simultaneous occurrences of this behaviour in a stable period (column "Day"). We report in column $\%_{Day}$ the support of each behaviour on the corresponding day and in column PSP_{Day} the execution time of PSP on the corresponding day with a support of $\%_{Day}$. We can observe that, at this granularity, PSP is able to extract most of the behaviours. Furthermore, PSP is even so fast that it could be applied on each day of the log and the total time would be around 70 minutes (420 days and an average execution time of approximately 10 seconds per day). Nevertheless, we have to keep in mind that with such an approach:

1. Undiscovered periods will remain (for instance a period of two consecutive days or a period of one hour embedded in one of the considered days).
2. Undiscovered behaviours will remain (embedded in the undiscovered periods).
3. The method would be based on an arbitrary division of the data (why working on each day and not on each hour or each week or each half day?).

Finally, in order to avoid the drawbacks enumerated above, the only solution would be to work on each stable period and apply a traditional sequential pattern algorithm. However this would require several millions of calls to the mining algorithm and the total execution time would be around 20 days (3,500,000 periods and an average execution time of approximately 0.5 seconds per period). Furthermore (as stated in section 4.3) this solution is not satisfying because of the long repetitive sequences that may be embedded in the data.

7 Conclusion

The proposition developed in this paper has shown that considering a log at large, *i.e.* without any division according to different values of granularity like traditional approaches, could provide the end user with a new kind of

knowledge cutting: periods where behaviours are particularly significant and distinct. In fact, our approach aims at rebuilding all the different periods the log is made up with. Nevertheless, by considering the log at large (several month, several years, ...) we have to deal with a large number of problems: too many periods, too low frequency of behaviours, inability of traditional algorithms to mine sequences on one of these periods, etc. We have shown that a heuristic-based approach is very useful in that context and by indexing the log, period by period, we can extract frequent behaviours if they exist. Those behaviours could be very limited on time, or frequently repeated but their main particularity is that they are very few on the logs and they are representative of a dense period. Conducted experiments have shown different kind of behaviours concerning for instance either students, conferences, or restaurants. These behaviours were completely hidden on the log files and cannot be extracted by traditional approaches since they are frequent on particular periods rather than frequent on the whole log.

References

[1] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proceedings of the 11th Int. Conf. on Data Engineering (ICDE'95)*, Tapei, Taiwan, March 1995.

[2] F. Bonchi, F. Giannotti, C. Gozzi, G. Manco, M. Nanni, D. Pedreschi, C. Renso, and S. Ruggieri. Web log data warehousing and mining for intelligent web caching. *Data Knowledge Engineering*, 39(2):165–189, 2001.

[3] W. W. W. Consortium. httpd-log files. In *http://lists.w3.org/Archives*, 1998.

[4] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[5] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press.

[6] U. Fayad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[7] J. Han and M. Kamber. *Data Mining, concepts and techniques*. Morgan Kaufmann, 2001.

[8] B. Hay, G. Wets, and K. Vanhoof. Mining Navigation Patterns Using a Sequence Alignment Method. *Knowl. Inf. Syst.*, 6(2):150–163, 2004.

[9] http Analyze. http://www.http-analyze.org/.

[10] H. Kum, J. Pei, W. Wang, and D. Duncan. ApproxMAP: Approximate mining of consensus sequential patterns. In *Proceedings of SIAM Int. Conf. on Data Mining*, San Francisco, CA, 2003.

[11] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*, pages 176–184, Nantes, France, September 1998.

[12] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*, Nantes, France, September 1998.

[13] F. Masseglia, P. Poncelet, and R. Cicchetti. An efficient algorithm for web usage mining. *Networking and Information Systems Journal (NIS)*, April 2000.

[14] F. Masseglia, D. Tanasa, and B. Trousse. Web usage mining: Sequential pattern extraction with a very low support. In *Advanced Web Technologies and Applications: 6th Asia-Pacific Web Conference, APWeb 2004, Hangzhou, China.*, 14-17 April 2004.

[15] N. Meger and C. Rigotti. Constraint-Based Mining of Episode Rules and Optimal Window Sizes. In *Proc. of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 313–324, Pisa, Italy, September 2004.

[16] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6(1):61–82, January 2002.

[17] M. Nakagawa and B. Mobasher. Impact of Site Characteristics on Recommendation Models Based On Association Rules and Sequential Patterns. In *Proceedings of the IJCAI'03 Workshop on Intelligent Techniques for Web Personalization*, Acapulco, Mexico, August 2003.

[18] C. Neuss and J. Vromas. *Applications CGI en Perl pour les Webmasters*. Thomson Publishing, 1996.

[19] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *17th International Conference on Data Engineering (ICDE)*, 2001.

[20] M. Spiliopoulou, L. C. Faulstich, and K. Winkler. A data miner analyzing the navigational behaviour of web users. In *Proceedings of the Workshop on Machine Learning in User Modelling of the ACAI'99 Int. Conf.*, Creta, Greece, July 1999.

[21] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th Int. Conf. on Extending Database Technology (EDBT'96)*, pages 3–17, Avignon, France, September 1996.

[22] Webalizer. http://www.mrunix.net/webalizer/.

[23] J. Zhu, J. Hong, and J. G. Hughes. Using Markov Chains for Link Prediction in Adaptive Web Sites. In *Proceedings of Soft-Ware 2002: First Int. Conf. on Computing in an Imperfect World*, pages 60–73, Belfast, UK, April 2002.