

# Web Usage Mining: Sequential Pattern Extraction with a Very Low Support

F. Masegla, D. Tanasa, and B. Trousse

INRIA Sophia Antipolis  
2004 route des lucioles - BP 93  
06902 Sophia Antipolis, France

E-mail: {Florent.Masegla, Doru.Tanasa, Brigitte.Trousse}@sophia.inria.fr

**Abstract.** The goal of this work is to increase the relevance and the interestingness of patterns discovered by a Web Usage Mining process. Indeed, the sequential patterns extracted on web log files, unless they are found under constraints, often lack interest because of their obvious content. Our goal is to discover minority users' behaviors having a coherence which we want to be aware of (like hacking activities on the Web site or a users' activity limited to a specific part of the Web site). By means of a clustering method on the extracted sequential patterns, we propose a recursive division of the problem. The developed clustering method is based on patterns summaries and neural networks. Our experiments show that we obtain the targeted patterns whereas their extraction by means of a classical process is impossible because of a very weak support (down to 0.006%). The diversity of users' behaviors is so large that the minority ones are both numerous and difficult to locate.

**Keywords:** web usage mining, sequential patterns, clustering, patterns summary, neural networks.

## 1 Introduction

Analyzing the behavior of a Web site's users, also known as Web Usage Mining, is a research field which consists in adapting the data mining methods to the records of access log files. These files collect data such as the IP address of the connected host, the requested URL, the date and other information regarding the navigation of the user. Web Usage Mining techniques provide knowledge about the behavior of the users in order to extract relationships in the recorded data. Among available techniques, the sequential patterns are particularly well adapted to the log study. Extracting sequential patterns on a log file, is supposed to provide this kind of relationship: "*On the Inria's Web Site, 10% of users visited consecutively the homepage, the available positions page, the ET<sup>1</sup> offers, the ET missions and finally the past ET competitive selection*". This kind of behavior is just a supposition, because extracting sequential patterns on a log file also implies managing several problems:

---

<sup>1</sup> ET: Engineers, Technicians

- the cache (on the user’s computer) and the proxies (which can be cache servers) can lower the number of records in the access log.
- the great diversity of pages on the site.
- the research engines, which allow the user to directly access a specific part of the Web site (thus reducing the number of entries in the log file, and the number of navigations shared by the users).
- the representativeness of the visited part compared to the entire site (a research team can represent less than 0.7% of the site).
- the representativeness of the users who navigate through that part of the site, compared to the users of the whole site.

If caching problems can be solved [3], the representativeness requires a strong study. In order to illustrate our goal, let us consider the sequential patterns we are supposed to obtain. Due to the small size of the “job offer” part of the site, users requesting a page on that part represent only 0.5% of users on the entire site. In the same way, users navigating on the “teaching” part of the LAMBDA research project represent only 0.01% of all the users. Therefore, a Web Usage Mining study on such a Web site, has to manage this particular representativeness in order to provide satisfying results. Our goal is to show that a classical<sup>2</sup> sequential pattern mining process is not able to provide behaviors with such a weak support. Furthermore we present a method for discovering the behavior of all the users of a Web site, including the minority behaviors. We describe our experiments and then conclude the paper.

## 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.

### 2.1 Sequential Pattern Mining

In [1], the association rules mining problem is defined as follows:

**Definition 1.** *Let  $I = \{i_1, i_2, \dots, i_m\}$ , be a set of  $m$  literals (items). Let  $D = \{t_1, t_2, \dots, t_n\}$ , be a set of  $n$  transactions ; Associated with each transaction is a unique identifier called its TID and an itemset  $I$ .  $I$  is a  $k$ -itemset where  $k$  is the number of items in  $I$ . We say that a transaction  $T$  contains  $X$ , a set of some items in  $I$ , if  $X \subseteq T$ . The support of an itemset  $I$  is the fraction of transactions in  $D$  containing  $I$ :  $\text{supp}(I) = \|\{t \in D \mid I \subseteq t\}\| / \|\{t \in D\}\|$ . An association rule is an implication of the form  $I_1 \Rightarrow I_2$ , where  $I_1, I_2 \subset I$  and  $I_1 \cap I_2 = \emptyset$ . The rule  $I_1 \Rightarrow I_2$  holds in the transaction set  $D$  with confidence  $c$  if  $c\%$  of transactions in  $D$  that contain  $I_1$  also contain  $I_2$ . The rule  $r : I_1 \Rightarrow I_2$  has support  $s$  in the transaction set  $D$  if  $s\%$  of transactions in  $D$  contain  $I_1 \cup I_2$*

<sup>2</sup> our research field excludes methods with constraints and sampling methods for reasons that will be given in this paper

(i.e.  $\text{supp}(r) = \text{supp}(I_1 \cup I_2)$ ).

Given two parameters specified by the user, *minsupp* and *minconfidence*, the problem of association rule mining in a database  $D$  aims at providing the set of frequent itemsets in  $D$ , i.e. all the itemsets having support greater or equal to *minsupp*. Association rules with confidence greater than *minconfidence* are thus generated.

As this definition does not take time into consideration, the sequential patterns are defined in [10]:

**Definition 2.** A sequence is an ordered list of itemsets denoted by  $\langle s_1 s_2 \dots s_n \rangle$  where  $s_j$  is an itemset. The data-sequence of a customer  $c$  is the sequence in  $D$  corresponding to customer  $c$ . A sequence  $\langle a_1 a_2 \dots a_n \rangle$  is a subsequence of another sequence  $\langle b_1 b_2 \dots b_m \rangle$  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  $S = \langle (3) (4\ 5) (8) \rangle$ , be that client's purchases.  $S$  means that "C bought item 3, then he or she bought 4 and 5 at the same moment (i.e. in the same transaction) and finally bought item 8".

**Definition 3.** The support for a sequence  $s$ , also called  $\text{supp}(s)$ , is defined as the fraction of total data-sequences that contain  $s$ . If  $\text{supp}(s) \geq \text{minsupp}$ , with a minimum support value *minsupp* given by the user,  $s$  is considered as a frequent sequential pattern.

## 2.2 Access Log Files Analysis With Sequential Patterns

The general idea is similar to the principle proposed in [4, 9, 11]. 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 requested 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 [12], where a log input contains records made of 7 fields, separated by spaces:

**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. Afterwards each "uninteresting" data is pruned out from the file. During the sorting process, in order to allow the knowledge discovery process to be more efficient, URLs and clients are mapped into integers. Each time and date is also translated into relative time, compared to the earliest time in the log file.

**Definition 4.** Let  $Log$  be a set of server access log entries. An entry  $g$ ,  $g \in Log$ , is a tuple  $g = \langle ip_g, ([l_1^g.URL, l_1^g.time] \dots [l_m^g.URL, l_m^g.time]) \rangle$  such that for  $1 \leq k \leq m$ ,  $l_k^g.URL$  is the item asked for by 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 4, is close to the “Client-Time-Item” structure used by sequential pattern algorithms. In order to extract frequent behaviors 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  $l_k^g.URL$  is transformed into an item number. 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 “60” at time  $d4$ . The goal is thus, according to definition 3 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, for instance, be  $\langle ( 10 ) ( 30 ) ( 20 ) ( 30 ) \rangle$  (with the file illustrated in table 1 and a minimum support given by the user: 100%). Such a result, once mapped back into URLs, strengthens the discovery of a frequent behavior, common to  $n$  users (with  $n$  the threshold given for the data mining process) and also gives the sequence of events composing that behavior.

Client	d1	d2	d3	d4	d5
1	10	30	40	20	30
2	10	30	20	60	30
3	10	70	30	20	30

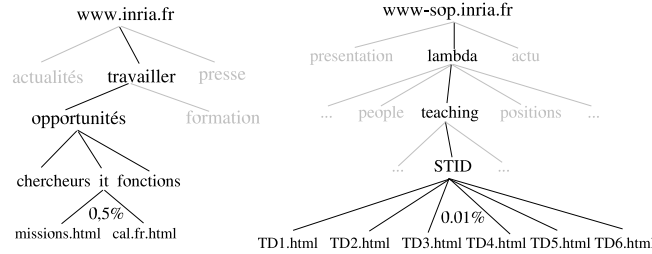
**Table 1.** File obtained after a pre-processing step

### 3 Divide and Discover: Motivations and Principle

#### 3.1 Motivations

Let us consider Inria’s web sites. The main site is *www.inria.fr*, the site of Sophia Antipolis is *www-sop.inria.fr*, and so on. These sites can be represented as shown in Figure 1 and the pages contents can be about jobs, research, teaching... From such a log analysis, we can provide some lessons:

- Usually, the sequential patterns coming from such a log can be disappointing. In fact their relevance is weak and they can be obvious and not so useful (e.g. “0.1% of users arrive at the homepage and then go to the contents page”).
- The interesting behaviors are contained in a specific part of the log. For example, in Figure 1, the part corresponding to the teaching activities of J. Smith (STID) will be requested by **0.01%** of users recorded in the log. The users interested in job opportunities will represent 0.5% of all requests on the site.



**Fig. 1.** Parts of the Inria's Web sites

- In order to get interesting patterns on that log, we thus have to specify a really low support.

Let us study the question of a low support. In our research framework, we exclude methods with constraints and sampling methods. Without denying the efficiency of such methods, we argue that techniques based on constraints do not allow to find all the patterns (which are still to discover, thus unknown from the user and his constraints). The second technique is sampling. We consider that the representativeness we are working with is so weak that the size of the sample will be almost the same as that of the log. Let us now imagine that we specify a very low support. Two problems will then appear:

- The response time will be too long (in most cases, the result won't even be obtained due to the complexity of the process).
- The amount of frequent patterns generated by this process (in the case the process ends) would be very large.

Nevertheless, the behaviors we want to discover have a really low support. These behaviors correspond to minorities, but we aim at discovering this kind of patterns since we consider that they are highly relevant. For instance, among these behaviors, we can notice the hacking activities or the navigation (from students) on some teaching pages. Our goal is thus to provide patterns, revealing behaviors such as:

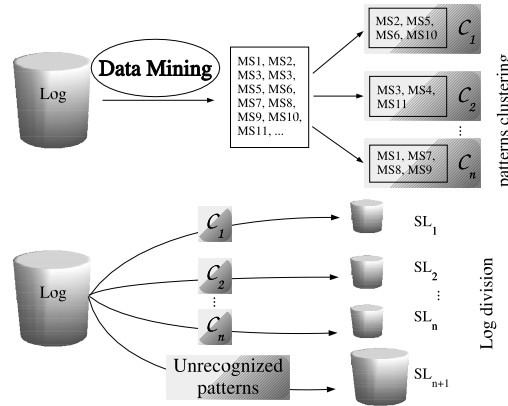
- 0.08% of users have a navigation similar to hacking activities. Among them 90% respected a typical hacking navigation.
- 0.007% of users have a navigation related to the teaching pages of J. Smith. Among them, 15% requested consecutively the 6 pages of his course on data mining.

The very weak support of these patterns is mainly due to the great diversity of the behaviors on the analyzed logs and to the large number of URLs contained in that site. In order to solve the problems described above, we developed the “Divide and Discover” method.

### 3.2 Principle

The outline of our method is the following: discovering clusters of users (grouped by behavior) and then analyzing their navigations by means of a sequential pattern mining process. Our method thus relies on two steps. The first step aims at dividing the log into sub-logs, supposed to represent separated activities. The second step aims at analyzing the behavior of users recorded in each sub-log. The principle of our method is thus the following:

1. Extracting sequential patterns on the original log.
2. Clustering these sequential patterns.
3. Dividing the log according to the clusters obtained above. Each sub-log contains sessions from the original log, corresponding to at least one behavior of the cluster which enabled to create this sub-log. A special sub-log is then created to collect the sessions from the original sub-log which do not correspond to a cluster from the previous step.
4. For each sub-log, apply this whole process (recursively).



**Fig. 2.** Divide & Discover principle

Figure 2 illustrates this method. First sequential patterns are obtained and clustered ( $C_1$  to  $C_n$ ). Then the log is divided ( $SL_1$  to  $SL_n$ ) upon these clusters. Finally a special sub-log ( $SL_{n+1}$ ) is created for the sessions which can not be matched with a behavior from the original log. The quality of the results produced by our approach will rely on this sub-log. In fact, the first sub-logs contain the most represented categories of users. They are thus interesting, but the most interesting patterns will come from the study of the unclustered sessions of the sub-log  $SL_{n+1}$ . Considering this sub-log as a new original log, and repeating the process (as described in Figure 2) will allow us to discover behavior with a

low representativeness. In order to provide reliable results, our method depends on a specific factor: the quality of the division proposed for a log. This division relies on the clustering performed on the discovered sequential patterns in the original log. We describe in the next section the method we employed to cluster the sequential patterns.

## 4 Clustering Based on Pattern Generalisation

We studied several clustering methods for sequential patterns. We describe here the most efficient method that we used for sequential pattern clustering. The clustering method used in this research is based on a method developed in 2000 by [2] for indexing web sequences in the context of Web-based recommender systems. The efficiency of such a method is based on the neural approach and its effectiveness relies on the use of summarized descriptions for sequential patterns: such descriptions are based on a generalization of Web access sequences (cf. section 4.2).

### 4.1 Neural Method

We propose a neural clustering method based on [2] (integrated in the object-oriented framework called CBR\*Tools<sup>3</sup> [6] for supporting the reuse of past experiences). It was successfully applied in the context of browsing advisors in a Web thematic repertory, for enterprises such as France Telecom. This method relies on a hybrid model of connectionist memory inspired from [7] and composed from a connexionist part [5] and a flat memory compound of patterns' groups.

A threshold  $s_i$  is associated to each prototype, which will be modified during the learning step. Such a threshold determines an *influence region* in the input space. If a pattern introduced in the network falls in the influence region of a prototype, then this prototype will be activated. Such a region is determined by the set of input vectors satisfying a distance measure lower than the threshold. If there is no activated prototype, a new one is created.

So the structure of a prototype-based network such as ARN2 is evolutionary in the sense that the number of prototypes at the hidden level is not a priori fixed and might be increased during the learning step. A prototype is characterized by its reference vector, an influence region and a set of representing patterns.

### 4.2 Summarizing Sequential Patterns

To characterize a sequential pattern, we use four attributes based on a *generalization of web pages belonging to it*: we use 1) the *multi-site* aspect and 2) the *first-level category* aspect (for all sites). Currently such a category identification is done at the syntactic level on URLs. For example, page (a) belongs to WWW-SOP.INRIA.FR and has TEACHING as a second-level category value. So

---

<sup>3</sup> CBR\*Tools URL=<http://www-sop.inria.fr/axis/software.html>

page (a) will be considered as representing a document and a second-level category both for WWW-SOP.INRIA.FR and for the LAMBDA first-level category. The four attributes calculated from the Web pages of the sequential pattern are: 1) the number of second-level categories for the pattern per site, 2) the number of documents for the pattern per site, 3) the number of second-level categories for the pattern per first-level category, for all sites and finally 4) the number of documents per first-level category (for all sites). The dimension of the description vector is equal to  $2 \times (\text{number of considered sites} + \text{number of first-level categories from Web pages of patterns i.e. the union of first-level categories independently of site})$ . Each attribute is normalized between 0..1 and has an importance weight assigned related to the chosen context.

*Example 2.* Let us consider a group of students requesting the pages of a course about data mining: `lambda/teaching/STID/`. Several pages can thus be requested on that part of the site: `annee02-03.html`, `TD1.html`, `TD2.html`, `TD3.html`, `accesslog.html` and `errorlog.html`. Each of these pages will be named a, b, c, d, e and f (i.e. `a=lambda/teaching/STID/annee02-03.html`, `b=lambda/teaching/STID/TD1.html`, ...). The goal is of course to detect that these six pages will be grouped in the same cluster. In fact, once this cluster is detected, a sequential pattern mining process on the sub-log file corresponding to that cluster will allow to find patterns with high support on this sub-log and very low representativeness on the entire log.

Let us suppose that 1) we join logs from two sites WWW.INRIA.FR and WWW-SOP.INRIA.FR, 2) we structure such logs into sessions and finally 3) the extracted patterns use Web pages from only six first-level categories (such as "LAMBDA"). Let us consider that we extracted four sequential patterns concerning these pages:  $\langle (a)(b) \rangle$ ,  $\langle (b)(c) \rangle$ ,  $\langle (d)(e) \rangle$  and  $\langle (d)(f) \rangle$ . The Web pages "a...f" belonging to these patterns belong to the same site "WWW-SOP.INRIA.FR", with "LAMBDA" as the first-level category and "TEACHING" as the second-level category. So they have the same summary on the four following attributes.

<code>oCategory2PerSite</code>	0, 2	<code>oCategory2PerCategory1</code>	0,0,0,0,2,0
<code>oDocsPerSite</code>	0, 2	<code>oDocsPerCategory1</code>	0,0,0,0,2,0

The prototype network built on such an example has an input space dimension equal to 16 (related to the description of a pattern). Since the four patterns have an equal description, we obtain only one class:  $C = \{ \langle (a)(b) \rangle, \langle (b)(c) \rangle, \langle (d)(e) \rangle, \langle (d)(f) \rangle \}$ . Hence, our objective of grouping together sessions containing Web pages that are related, is now fulfilled.

## 5 Experiments

The extraction methods are written in C++ on a Pentium (2.1 Ghz) PC running a Red-Hat system. We used the PSP algorithm [8] for sequential pattern extraction. The neural method and the GUI are realized in Java. For the Inria's main site, the data was collected over a period of one month, while for the Inria

Sophia Antipolis site, over a period of two months. Their sizes are 2.1 Go and respectively 3 Go. The description of the characteristics (cf. Figure 3) is the following:  $N$  stands for the number of lines in the log,  $S$  for the number of sessions,  $U$  for the number of filtered URLs,  $AL$  for the average length of the sessions,  $AU$  for the average number of URLs in the sessions. During our experiments, we could bring into relief frequent behaviors, with a relative representativeness getting weaker and weaker, depending on the depth of the sub-log.

	www.inria.fr	www-sop.inria.fr
$N$	11 637 62	15 158 076
$S$	432 396	564 870
$U$	68 732	82 372
$AL$	6.3	4.4
$AU$	7.2	6.3

**Fig. 3.** Characteristics of the log files

**C1:**  $\langle$ (travailler/opportunites/chercheurs.fr.html) (travailler/opportunites/chercheurs/concours2.fr.html) (recherche/equipes/index.fr.html) (recherche/equipes/listes/index.fr.html) $\rangle$  (support: 0.28%). This behavior is related to the career opportunities offered by the Inria. The users read the job opportunities page, then the page describing the competitive selection and finally the pages describing the research teams.

**C2:**  $\langle$ (scripts/root.exe) (c/winnt/system32/cmd.exe) (..\%255c..\..\%255c../winnt/system32/cmd.exe) (..\%255c..\..\%255c..\%c1%1c..\..\%c1%1c..\..\%c1%1c../winnt/system32/cmd.exe) (winnt/system32/cmd.exe) (winnt/system32/cmd.exe) (winnt/system32/cmd.exe) $\rangle$  (support: 0.04%). This behavior is typical of a search for a security hole in the system. Usually, these attacks are programmed once and then shared and used by different individuals.

**C3:** with the prefix **epidaure/Demonstrations/foie3d/**:  
 $\langle$ (endo4.html) (endo5.html) (endo6.html) (endo8.html) (endo9.html) (endo10.html) (endo11.html) (endo12.html) $\rangle$  (support: 0.01%)

The behaviors discovered with our method cover more than 50 surfing goals on Inria’s main site and more that 100 goals on the site of Inria Sophia Antipolis. We reported here three goals, from job opportunities requests to hacking activities. Thus, these behaviors illustrate the success of our method in discovering minority behaviors, which we couldn’t have discovered given their very weak representativeness.

## 6 Conclusion

In this paper, we proposed a method for extracting of the behavior of all the users of a Web site. Our approach has the characteristic to recursively divide the

log file in order to locate behaviors and to represent them as clusters. For this purpose, we had to provide a specific clustering method, dedicated to sequential patterns. The main advantage of our method is to consider the Web Usage Mining with very low support as a problem that can be solved by successive divisions. The problem thus shifts from *one open problem* to *n* problems we can solve and *one* problem that will have to be divided recursively. By proceeding this way, we could establish that the frontier between the quantity of data and the quality of the results can sometimes be pushed back by extracting behaviors with a very low representativeness.

## References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207–216, Washington DC, USA, May 1993.
2. A. Benedek and B. Trousse. Adaptation of Self-Organizing Maps for CBR case indexing. In *27th Annual Conference of the Gesellschaft für Klassifikation*, Cottbus, Germany, March 2003.
3. Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.
4. U.M. Fayad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.
5. A. Giacometti. Modèles hybrides de l'expertise, novembre 1992. PhD Thesis (in french), ENST Paris.
6. M. Jaczynski. Modèle et plate-forme à objets pour l'indexation des cas par situation comportementales: application à l'assistance à la navigation sur le web, décembre 1998. PhD thesis (in french), Université de Nice Sophia-Antipolis.
7. M. Malek. Un modèle hybride de mémoire pour le raisonnement à partir de cas, octobre 1996. PhD thesis (in french), Université Joseph Fourier.
8. 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), LNAI, Vol. 1510*, pages 176–184, Nantes, France, September 1998.
9. F. Masseglia, P. Poncelet, and R. Cicchetti. An efficient algorithm for web usage mining. *Networking and Information Systems Journal (NIS)*, April 2000.
10. R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96)*, pages 3–17, Avignon, France, September 1996.
11. Doru Tanasa and Brigitte Trousse. Web access pattern discovery and analysis based on page classification and on indexing sessions with a generalised suffix tree. In *Proceedings of the 3<sup>rd</sup> International Workshop on Symbolic and Numeric Algorithms for Scientific Computing*, pages 62–72, Timisoara, Romania, October 2001.
12. W3C. httpd-log files. In <http://www.w3.org/Daemon/User/Config/Logging.html>, 1995.