Web Usage Mining: Extraction, Maintenance and Behaviour Trends

P.A Laur [1] – M. Teisseire [1] – P. Poncelet [2]

[1] Lirmm, 161 rue Ada, 34392 Montpellier cedex 5, France
69042 Heidelberg, Germany
{laur, teisseire}@lirmm.fr
[2] EMA/LGI2P, Ecole des Mines d'Alès
Site EERIE, Parc Scientifique Georges Besse, 30035 Nîmes cedex 1, France
Pascal.Poncelet@ema.fr

Abstract. With the growing popularity of the web, large volumes of data are gathered automatically by Web Servers and collected into access log files. Analysis of such files is generally called Web Usage Mining and tends to search user's behaviour patterns from one or more web servers in order to extract relationships within extracted data. Over the past few years, numerous work have focussed on this problem and some tools have been created to analyse user's behaviour on a web server. Even if it is possible to analyse user's behaviour, two important problems are not taken into account: how to handle new records and capitalize on previous knowledge and how to analyse trend user in behaviour? In this paper we present AUSMS-Web, a system which aims to extract knowledge from a user's behaviour, to maintain this knowledge when new data is added to the logs, and to analyse users' behaviour trends on a web site.

1 Introduction

With the growing popularity of the World Wide Web, large volumes of data such as the user's address or requested URLs are automatically gathered by Web Servers and stored in access log files (access log, error log …). Analysis of such files in order to understand user behaviour is called Web Usage Mining and can provide useful information to improve a network's performance, to rebuild a web site dynamically or even to target customer in an e-business environment. Even if numerous analysis tools exist to show, for example, the number of times a URL is accessed or the list of most popular URLs, the relationships between requested resources and a customer profile was poorly analysed by those tools [6,15]. During the past few years, new techniques were implemented to analyse more precisely user behaviour and path on a web site [3]. It then becomes possible to extract strong correlations between Web Site pages (for example : *50% of users who have visited the following URL: plaquette/info-f.html and labo/infos.html have also visited situation.html*) or to establish typical behaviour during a given time period (for example*: 60% of users who have visited /jdk2.1/docs/api/Package-java.io.html and /jdk2.1/docs/api/java.io.*

BufferedWriter. html, have also visited URL /jdk2.1/docs /relnotes/deprecatedlist.html in the next 30 days).

However, two problems remain. As data sources, i.e. access log files, constantly evolve; the outcome of previously extracted data is unclear. Moreover, even if it becomes possible to analyse user behaviour, user trend analysis is not being addressed. Indeed, for a system information manager, information about recurring behaviour, appearance or disappearance of new usages, or a change in user behaviour during a given time period are important elements to improve quality of service. Whereas numerous tools and techniques exist about trend analysis in the field of time series or textual documents, to the best of our knowledge, only very little exist about data trend analysis from a web server. Most tools that tackle the problem of trend analysis only address user behaviour at a given date.

In this paper we present AUSMS-Web, a knowledge extraction tool based on a knowledge repository tool and a user trend analysis tool.

The paper is organised in the following way. In section 2 we present the problem. In section 3 we describe the functional architecture of the AUSMS-Web system by detailing the various stages. In section 4, we describe some experiments undertaken with the prototype on user trend analysis. A short description of related work on maintenance and trend analysis is presented in section 5. Lastly, in section 6, we conclude by describing the advantages of the AUSMS-Web system and the potential continuations from this work.

2 Problem Statement

In this section, we present the problem statement studied by AUSMS-Web system and we focus on three aspects: (i) user behaviour analysis, (ii) maintaining extracted knowledge and (iii) the analysis of user trends over time.

In a general way, it is possible to match a user's behaviour on a web site with navigation into a graph[1]: edges are expressed by navigation through different pages and a node corresponds to a hypertext link. We consider in the following a tree as an acyclic connected graph and a forest as an acyclic graph. In our context, a cyclic graph can be transformed into an acyclic graph while replicating the divided sub nodes [13]. A forest is thus a collection of trees where each tree is a rooted component of the forest.

Let us consider *DB* a tree database, i.e. a forest where each tree *T* is composed of a unique identifier (user's computer address) and a sub tree included in the forest (user behaviour in a web server). More formally, we denote a tree as $T = (N,B)$ where *N* is the set of labelled nodes, i.e. hypertext links, and *B* the set of branches, i.e. user's navigation. Let *supp (p)* be the support value for a sub tree corresponding to the number of its occurrences in the database *DB*. In other words, the support of a sub tree *p*, i.e. user's behaviour, is defined as the percentage of all the trees in the database which contain *p*. We say that a tree $S=(N_s, B_s)$ is contained in $T=(N,B)$ iff i) $N_s \subseteq N$, ii) $b=(n_x, n_y) \in B_s$ if and only if n_y is a parent of n_x in *T*. In order to decide whether a structure is frequent or not, a value of minimal support is specified by the user (*minSupp*) so a structure is frequent if the condition $supp(p) \geq minSupp$ holds. The problem of analysing user's behaviour consists in finding the most frequent behaviours, which are in *DB* and whose support is higher than minSupp.

Let us now consider the evolution of the data sources, i.e. incoming of new connections from a web server. That is to say *db* the database increment where new information is referenced. Let $U=DB \cup db$, be the updated database holding all data from *DB* and *db*. Let L^{DB} be the frequent behaviours set in *DB*. The problem of keeping up to date information is to seek the frequent behaviours in *U*, noted L^{U}, by respecting the same support value. Moreover, maintenance must take previously extracted knowledge into account so as to avoid restarting retrieval algorithms from scratch when the data is updated.

Finally, the trend analysis problem is complementary with the one of keeping up to date extracted knowledge. Indeed, in the last case we took an interest in a given support value, i.e. we are trying to know what happens to frequent behaviours for a given support value when new data is incoming. On the other hand, in the trend analysis case, we have to maintain a list holding all frequent behaviour during each extracted knowledge stage to see how they are evolving. The problem of trend analysis is to search which behaviours are increasing or decreasing.

3 The AUSMS-Web system

The aim of A.U.S.M.S.-Web is to propose an environment of discovery and knowledge extraction for web server data by taking into account information recovery, extracted knowledge update and the evolution of user's trend analysis on the web site. The AUSMS-Web system is an extension of A.U.S.M.S. system (Automatic Update Schema Mining System) [7] which was devoted to searching common sub-structures in a graph (Web Content Mining). AUSMS-Web has been fitted to handle data from web servers and to perform trend analysis. These general principles illustrated in figure 1 are similar to those of knowledge extraction process. It can be broken into four main phases. First, starting from rough data log files, pre-processing eliminates irrelevant data and ensures their transformation. In the second phase, a knowledge extraction algorithm is used to find the frequent behaviours. In order to maintain knowledge, the information obtained during this phase is kept in a database. The exploitation of the results is facilitated by a frequent behaviour visualisation tool in a third phase. Lastly, user trend evolution analysis is performed from stored data. It allows the extraction of recurring, increasing or decreasing trends for example.

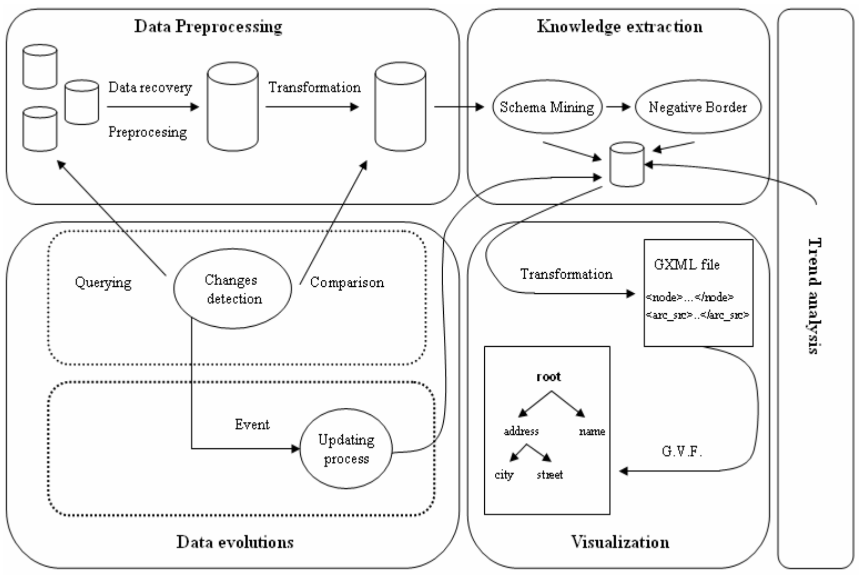


Fig. 1. General architecture

In our Web Usage Mining context, rough data is collected by web server access log files[1]. Each access log file input is automatically added every time a resource request reaches the web server. From this source, a process of extraction and transformation is carried out and the extracted data is stored in a database. Within the framework of Web data, a filtering process is carried out to eliminate irrelevant data for the analysis: image, sounds, video…

Authors in [9] propose a level-wise algorithm for mining frequent behaviour from access log files. In AUSMS-Web system, we have extended this algorithm to include the associated negative border [11]. The negative border is the collection of all sequences that are not frequent but both of whose generating sub-sequences are frequent. In this article, we will not describe this mining algorithm. We will simply give an overview of the approach. Interested readers may refer to [7]. In the first part of the algorithm, nodes of the trees are translated into sequence elements. In order to keep the parent relationship and the node depth, additional informations are added to each element. In the second part, the recursive algorithm acts in the following way: Frequent elements (i.e. nodes of the trees) are first searched into the transformed database. Candidates 2-elements are generated (generating phase) from these ele-

[1] In order to avoid the problem of user's identification inside access log files and memory link to the proxy or web navigator, we have used during the different experiments a dynamic web server which was updating, by the way of cookies and php, a log database. However, later on in this paper, we will use the term "access log" for log files from web servers as well as for dynamic databases.

ments. This generation is done by extending a frequent element x with an other one y only if x can be a parent of y, i.e. depth(y)>depth(x)+1. Then we examine if such candidates match with trees in database (pruning phase). This process is applied until no more candidates can be generated. With an aim of improving the candidate generating procedure as well as the management of candidate elements, we use a bitmap representation inspired by [1]. This structure offers the advantage of considerably reducing the storage space and the ability to generate candidates easily. Moreover it is particularly adapted for the search of long behaviours.

In the following sub-sections, we firstly provide an algorithm for maintaining previously extracted knowledge. We secondly describe the trend analysis phase. Finally we have a look on the visualisation module.

3.1 Consideration of data sources evolution

The negative border obtained in the previous stage enables us to take into account the updates and to maintain extracted knowledge.

This is realised in the following way: the data sources are compared from a time specified by the user (delay). This operation is carried out in the AUSMS-Web system by an agent which acts either in a temporal way (fixed time difference since last update), or in a direct way (user activation). The agent then starts the incremental algorithm with the obtained results (UpdateSet).

```
Algorithm IncrementalUpdate

Input : BN+F, UpdateSet
Output: BN+F updated

1  : retrieve BN+F;
2  : $S_{change} = \varnothing$;
3  : Foreach element e $\in$BN+F do
4  :    uptade e.support with UpdateSet;
5  :  If e.state <> e.previous_state then
6  :       delete e.father successors;
7  :        $S_{change} = S_{change}$ + e.father;
8  :  EndIf
9  : EndFor
10 : Foreach node n $\in S_{change}$ do
11 :    apply mining algorithm on (n);// apply generating and pruning phase from n
12 :  EndFor
13 : return BN+F;
```

Firstly, this algorithm retrieves the stored tree structure (*BN+F*) from previous computation. This tree structure holds all frequent and negative border element generated by the mining algorithm. As previously stated in section 3, the mining algorithm is recursive and acts in a tree manner. So, it can be started on any of the saved structure node by retrieving his father's information, deleting his entire father's node successors and finally applying the generating algorithm on this father. The second step in this algorithm consists in checking whether each structure's element support has changed due to the update itself or not. After updating these supports, we go through the structure to check if the node's state is the same (i.e. if a frequent node is still frequent, if a negative border node has become frequent or if the node state does not change). The key point here is to build a set of the entire changing state nodes fathers (S_{change}) to delete their successors and finally to start the mining algorithm again on the remaining fathers (some nodes could have been deleted during the previous successors deletion step). Finally, we obtain an updated tree structures that holds frequents and negative border elements as if we started the mining algorithm directly on the updated database. The computation time is equal or often better to the computation time required by the mining from scratch. The last task of the agent is to store the newly extracted knowledge.

3.2 Trend analysis

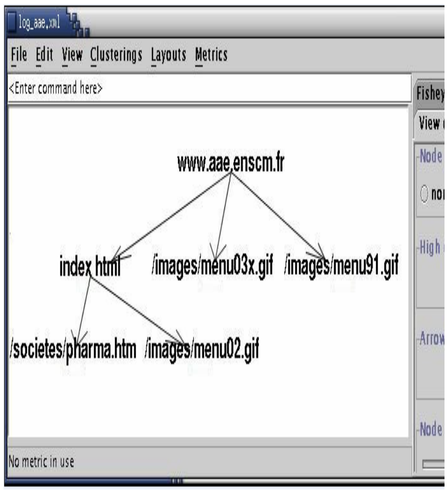
As we have seen in section 2, trend analysis consists in searching frequent structure evolutions during a given time period. One problem linked with this analyse is data storage: How can we find an efficient structure to maintain the extracted knowledge? Due to high number of intermediate results, it is compulsory to find a well suited structure. The second challenge, linked to the trends themselves, is to quickly search the same structures in order to follow their time evolutions and to suit the user's needs (increasing trends, decreasing trends, cycling trends…).

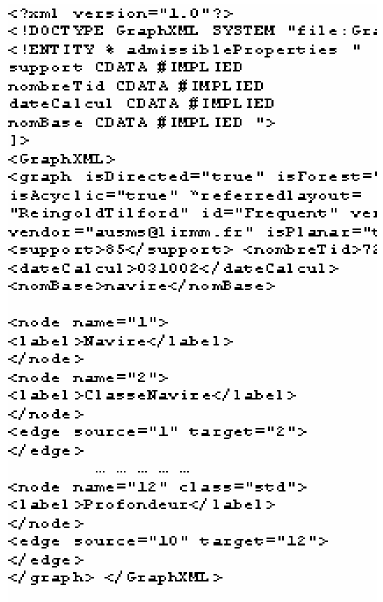
```
Algorithm TrendAnalysis
Input: ($L^{DB}_{t1}$ + $L^{DB}_{t2}$ +… + $L^{DB}_{tn}$) = $L^{DB}_{t}$, frequent structures set stored with their support value at date : $t_1$ ..... $t_n$.
Output: $H_f$, each frequent history stored in pairs (date,support).
1 : Foreach different frequent f in $L^{DB}_t$ do
2 :         $H_f = \emptyset$;
3 :         Foreach frequent e $\in L^{DB}_t$ do
4 :                 If $f = e$ then $H_f = H_f +$ (e.date, e.support); endif
5 :         enddo
6 :         Return $H_f$;
7 : enddo
```

The trend analysis principle is defined as follows: The extraction algorithm is executed for different supports values. Users can specify increments. For example, they can choose to have the support values increasing by a 5% step. The analysis will in this case start at 0.05, 0.10, 0.15 and so on. Obtained results (i.e. frequent structures) are stored using bitmaps within a structure described in [1]. Indeed, we have noticed during different experimentations that this structure was very efficient not only during computations but also for its low memory. Frequent structures history allows us to match the results with a specify trend profile. Use of bitmaps allows to quickly compare different frequent structures with the "AND" binary operator. Trends simply describe evolutions that match user's choice with additional information during time periods when the trend itself is checked.

3.3 Visualisation

Whereas previous modules are dedicated to providing and maintaining frequent behaviours, this module makes it possible to visualise these behaviours in graphs. For that, we are using GraphXML [5] which is a graph description language in XML especially designed for drawing and displaying systems. GraphXML makes it possible for the user to add a significant amount of information to the graphs. We use the Graph Visualisation Framework's (GVF) Java classes to visualise and handle the structures described by the GraphXML format.





```
<?xml version="1.0"?>
<!DOCTYPE GraphXML SYSTEM "file:Gr
<!ENTITY * admissibleProperties "
support CDATA #IMPLIED
nombreTid CDATA #IMPLIED
dateCalcul CDATA #IMPLIED
nomBase CDATA #IMPLIED ">
]>
<GraphXML>
<graph isDirected="true" isForest='
isAcyclic="true" "referredlayout=
"ReingoldTilford" id="Frequent" ve
vendor="ausms@lirmm.fr" isPlanar="t
<support>85</support> <nombreTid>7
<dateCalcul>031002</dateCalcul>
<nomBase>navire</nomBase>

<node name="1">
<label>Navire</label>
</node>
<node name="2">
<label>ClasseNavire</label>
</node>
<edge source="1" target="2">
</edge>
          … … … … …
<node name="12" class="std">
<label>Profondeur</label>
</node>
<edge source="10" target="12">
</edge>
</graph> </GraphXML>
```

Fig. 2. A behaviour example in GraphXML and its XML representation

Figure 2 represents screenshots of visualised trends. We find at the left a frequent trend structure resulting from the frequent trends search on the AAE Web Logs displayed via GVF. On the right-hand side we have the same description within the GraphXML format.

4 Experimentation

In this section, we mainly address trend analysis experiments carried out with the AUSMS-Web system. We used two data sets to study various existing trends. We do not take into account in the experiments the user's choice on the specification of a particular trend but we describe some increasing or decreasing trends.

The first data set comes from the LIRMM laboratory and regroups different connections realised on the laboratory web site from September 1996 to March 2000. The relevance of using this system over such a long time period is justified by the fact that it allowed us to note new types of usage (various organised conferences, new people…).

The second data set is from the former students' association of the Montpellier National Superior School of Chemistry and only includes two months of connections. The relevance of this data set compared to the previous one is to localise strong changes in user behaviour. To build this data set, we have joined each weekly log file. By regularly obtaining information from the server, we were able to carry out different experiments. In the remainder of this paper, we will consider that the file named AAE (1) corresponds to a new log file for each week whereas the file named AAE (2) corresponds to the accumulation of the information (i.e. accumulation of all AAE (1) during that time period). When analysing regular file, a new behaviour could become frequent whereas it could be drowned out by other behaviours when we analyse the cumulative file.

Figure 3 shows the difference between these two types of files. The behaviour pattern `<(/societes/pharma.htm,/images/menu02.gif)(/images/menu03x.gif)(/images/menu91.gif)>` which represents the fact that users accessed the web site at the same time (i.e. in a very short period of time) `/societes/pharma.htm` and `/image/menu02.gif` and then they have been on `/images/menu03x.gif` and finally on `/images/menu91.gif`.

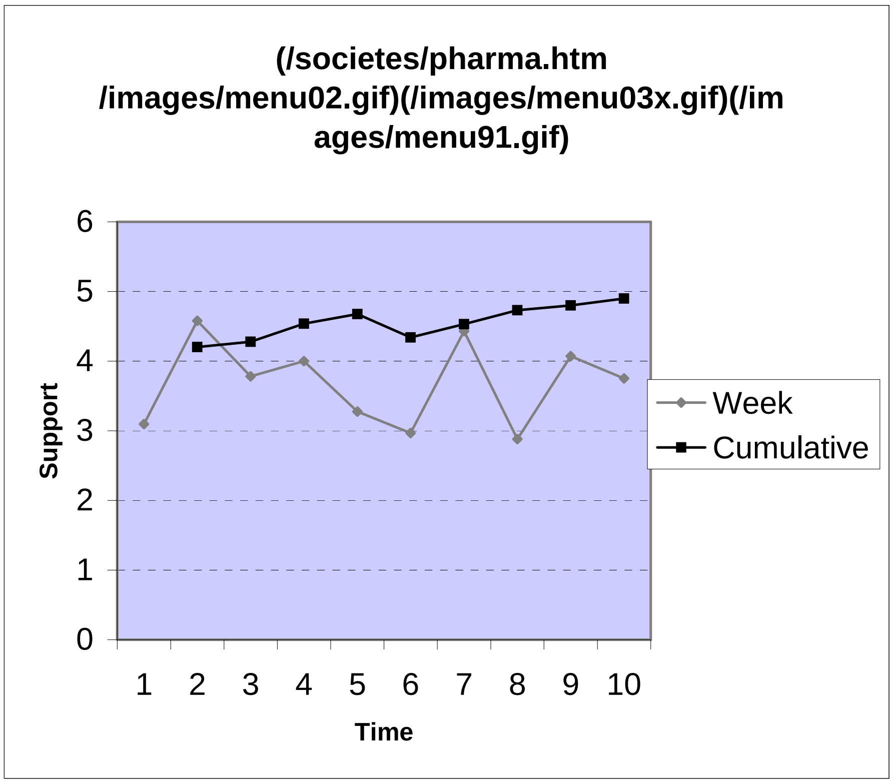


Fig. 3. A trend example

In the case of accumulated data, we can note that the behaviour pattern is increasing. Indeed, we note more and more user behaviour match with this pattern. On the other hand, if the analysis is done week after week we note that this behaviour is not increasing. We note that between week 2 and week 6 it was decreasing. As expected, we can notice that in the cumulative file, some behaviours are not well considered since they are not sufficiently frequent.

Figure 4 represents the trend analysis over a period of time on the LIRMM logs.

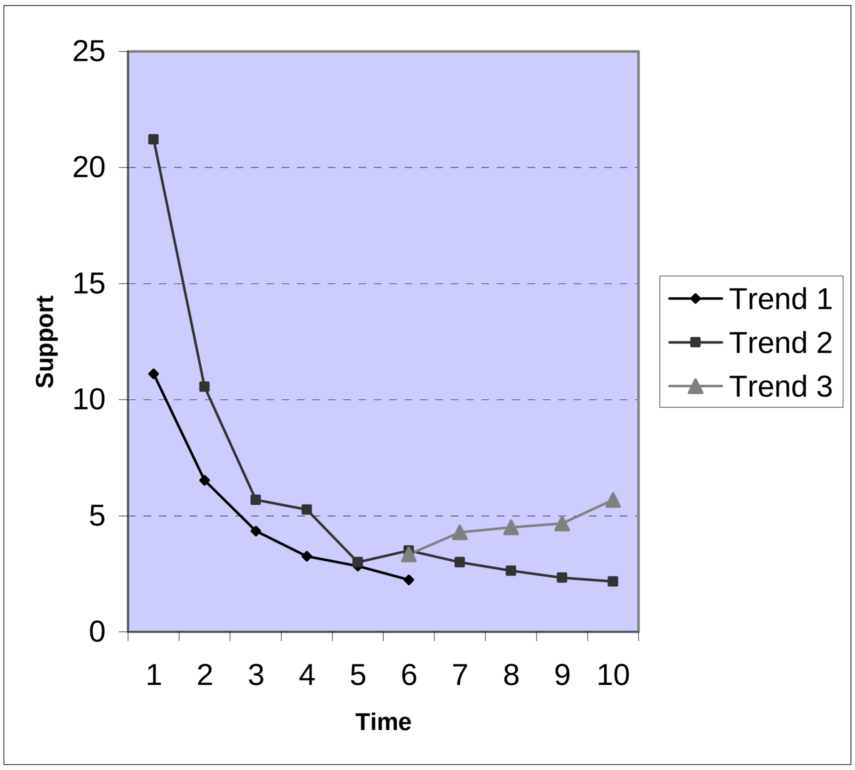


Fig. 4. Trend analysis from LIRMM web logs

We can notice that trend 1 which matches with `< (/images /calvinGrey.gif,/images/html.gif)(/images/mail.gif)(/images/calvinGrey.gif)>` decreases regularly over time. We also note that this trend completely disappears at period 6. Due to the support being too low, our algorithms are not able to provide a result after that point. The decreasing trend is also confirmed for trend 2 which matches with : `</images/memoire.gif, images/molecule.gif)(/images/emploi.gif)(/images/mail.gif)>`.

Finally trend 3 (`</robots.txt) (/robots.txt)>`) evolves during the time period and corresponds to the number of times those pages are accessed by a search engine.

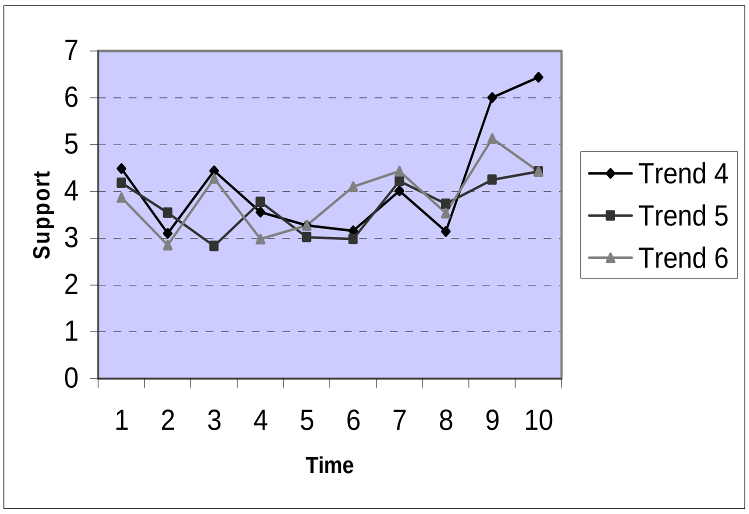


Fig. 5. Trend analysis from AAE (1)

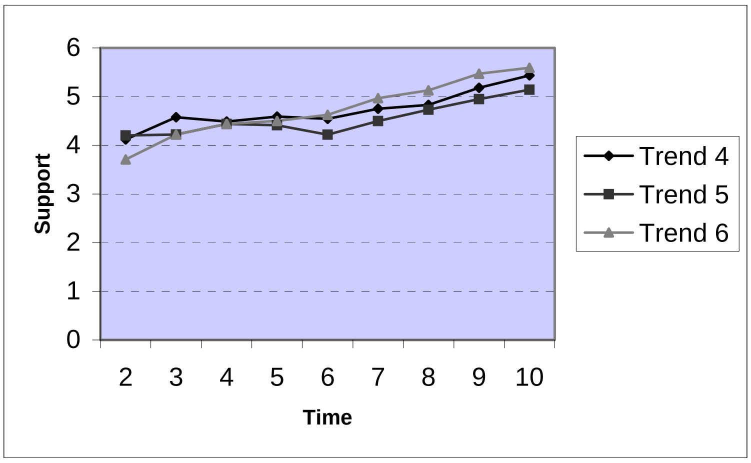


Fig. 6. Trend analysis from accumulated data AAE (2)

Figures 5 and 6 illustrate some trends extracted from the former students' web server. Theses trends match with:

Trend 4: `<(/, /css/00.css, /css01.css) (/images/meter150.gif)>`

Trend 5: `< (/images/backgrd3.gif) (/images/menu02.gif) (/images/creuset.gif)(/images/menu91.gif)(/images/menu90.gif,/images/menu92.gif)>`

Trend 6: `<(/css/00.css)(/images/backgrd3.gif) (/images/menu004a.gif)/images/meter050.gif)(/images/menu003d.gif)>`

Here again, we can notice that in figure 6 the different trends are increasing over time. However, a closer analysis of figure 5 shows that the user's behaviour is not increasing as much as it seems. There are even some weeks for which user behaviour decreases.

5 Related work

In this section, we address the issues of maintaining knowledge as well as trend analysis. A complete overview of Web Usage Mining systems and approaches is proposed in [3].

To our knowledge, there is no previous work on the maintenance of frequent behaviours extraction. Nevertheless, the search for behaviours could approach that of sequential patterns. In this section, we will thus examine the work carried out around this field. Around the sequential patterns and the basis of many approaches, [2] proposes an algorithm called FUP, for maintaining knowledge obtained from association rules. However, the problems of incremental updates within sequential patterns framework are much more complex than that of the association rules due to the size of the search space (i.e. the number of combinations is much larger). In [12], the authors propose an algorithm called ISM (Incremental Sequence Mining) based on SPADE [14] which allows an update of the frequent sequences when new customers and new transactions are added to the data base. The suggested approach builds a lattice of sequence which contains all the frequent and negative border elements [11]. When new information arrives, it is added to this lattice. The problem within this approach is obviously the increasing size of the negative border which in our case is minimised. In [10], the ISE (Incremental Sequence Extraction) algorithm searches for frequent patterns and generates candidates in the entire database by attaching the sequences of the incremental database to those of the original one. This approach avoids keeping the sequences contained in the negative border and the recalculation of these sequences when the initial data base has been updated. However, by eliminating the negative border, it is necessary to traverse more often the base to seek the candidates. In [16] the algorithm proposed uses both the concepts of negative border of the original data base and the concepts of suffixes and prefixes as proposed in ISE. To control the size of this negative border, they introduce a minimum support for these elements thus reducing its size. Moreover this algorithm realises an extension by prefix and suffix (using the negative border). The problem within this algorithm lies in the choice of the value of the minimum support for the negative border.

To the best of our knowledge, there is little research concerning the different user trends analysis on log servers. However, numerous work exist on trend analysis notably in the case of time series (long term or short term move, cyclic move, random move…) or in textual data[2]. We can mention the work of [8] which largely inspired this work. The authors propose a system to identify trends in textual documents. The principle is as follows. After a data pre-processing, they use a sequential pattern algorithm to establish some sentences and keep the historical link to each extracted pattern. They then search sentences that match a trend with the help of a form definition language. Experiments led by the authors consider trend analysis on a patent database.

6 Conclusion

In this article, we proposed AUSMS-Web system to analyse, maintain and extract user's behaviours trend on a Web Site. The advantage of our system lies in a unified approach to respond to a problem that has not been adequately taken into consideration by existing systems. For example, user trend analysis and the maintenance of extracted knowledge provide very relevant information to maintain and dynamically modify a Web server. During different experiments conducted in trend analysis, we have noted that the results were relevant to our data sets. We are currently working on user clusters analysis. Indeed, even if it is now possible to analyse precisely user trends on a Web site, it becomes necessary to improve the research by taking into account not only the user's class but also time periods of behaviour modification. Retrieving information on "phase changes", i.e. a web page suddenly becomes very popular with a certain class of users over a week and is never requested by that same class again, allows the Web site manager to better understand users' behaviour and to provide new information more suited to each user class.

References

1. Ares J., Gehrke J., Yiu T. and Flannick J.: Sequential Pattern Using Bitmap Representation. In Proceedings of Principles and Practice of Knowledge Discovery in Data (PKDD'02), Edmonton, Canada, July 2002.
2. Cheung D.W., Han J., Ng V. and Wong C.Y.: Maintenance of Discovered Association Rules in Large Databases: an Incremental Update Technique. In Proceedings of the International Conference on Data Engineering (ICDE'96), pp. 116-114, New Orleans, USA, February 1996.
3. Cooley R.W.: Web Usage Mining: Discovering and Application of Interesting Patterns from Web Data. PHD Dissertation, University of Minnesota, 2000.

[2] A detailed introduction to trends in temporal series is proposed in [4].

4. Han J. and Kamber M.: Data Mining – Concepts and Techniques. In Morgan Kaufmann Publishers, 2001.
5. Herman I. and Marshall M.S.: GraphXML An XML based graph interchange format. Centre for Mathematics and Computer Sciences (CWI), Technical Report INS-R0009, pp. 52-62, Amsterdam, 2000.
6. Kdnuggets. www.kdnuggets.com/.
7. Laur P.A., Teisseire M. and Poncelet P.: AUSMS: an environment for frequent sub-structures extraction in a semi-structured object collection. In proceedings of 14th DEXA03 Conference, LNCS, Prague, Czech Republic, pp. 38-45, September 2003.
8. Lent B., Agrawal R. and Srikant R.: Discovering Trends in Text Databases. In Proceedings of the 3rd International Conference on Knowledge, Newport Beach, California, August 1997.
9. Masseglia F., Poncelet P. and Cicchetti R.: An efficient algorithm for Web Usage Mining. In Networking and Information Systems Journal, Vol. 2, N°5-6, pp. 571-603, 1999.
10. Masseglia F., Poncelet P. and Teisseire M.: Incremental Mining of Sequential Patterns in Large Database. In Data and Knowledge Engineering, Vol. 46, N°1, pp.97-121, July 2003.
11. Mannila H. and Toivonen H.: On an Algorithm for Finding all Interesting Sequences . In Proceedings of the 13th European Meeting on Cybernetics and Systems Research, Vienna, Austria, April 1996.
12. Parthasarathy S. and Zaki M. J.: Incremental and Interactive Sequence Mining. In Proceedings of the Conference on Information and Knowledge Management (CIKM'99), pp. 251-258, Kansas City, USA, November 1999.
13. Wang K. and Liu H.: Discovering Structural Association of Semistructured Data. In IEEE Transactions on Knowledge and Data Engineering, pp. 353-371, January 1999.
14. Zaki M.: Scalable Data Mining for rules. PHD Dissertation, University of Rochester-NewYork, 1998.
15. Zaïane O., Xin M. and Han J.: Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In Proceedings on Advances in Digital Libraries Conference (ADL'98), Santa Barbara, CA, April 1998.
16. Zheng Q., Xu K., Ma S. and Lu W.: The Algorithms of Updating Sequential Patterns. In Proceedings of the International Conference on Data Mining (ICDM'02), Washington DC, USA, April 2002.