Knowledge-Based Systems 49 (2013) 50–62

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Effective web log mining and online navigational pattern prediction

Abdelghani Guerbas, Omar Addam, Omar Zaarour, Mohamad Nagi, Ahmad Elhajj, Mick Ridley, Reda Alhajj *

Department of Computer Science, University of Calgary, Calgary, Alberta, Canada
School of Information Technology, Bradford University, Bradford, UK
Department of Computer Science, Global University, Beirut, Lebanon

ARTICLE INFO

Article history:
Received 31 October 2012
Received in revised form 16 April 2013
Accepted 18 April 2013
Available online 3 May 2013

Keywords:
Web mining
Weblog mining
Pattern analysis
Prediction
Navigation
Indexing

ABSTRACT

Accurate web log mining results and efficient online navigational pattern prediction are undeniably crucial for tuning up websites and consequently helping in visitors' retention. Like any other data mining task, web log mining starts with data cleaning and preparation and it ends up discovering some hidden knowledge which cannot be extracted using conventional methods. In order for this process to yield good results it has to rely on some good quality input data. Therefore, more focus in this process should be on data cleaning and pre-processing. On the other hand, one of the challenges facing online prediction is scalability. As a result any improvement in the efficiency of online prediction solutions is more than necessary. As a response to the aforementioned concerns we are proposing an enhancement to the web log mining process and to the online navigational pattern prediction. Our contribution contains three different components. First, we are proposing a refined time-out based heuristic for session identification. Second, we are suggesting the usage of a specific density based algorithm for navigational pattern discovery. Finally, a new approach for efficient online prediction is also suggested. The conducted experiments demonstrate the applicability and effectiveness of the proposed approach.

1. Introduction

Organizations, companies and institutions are relying more and more on their websites to interact with clients. Retaining current clients and attracting potential ones push these organizations, companies and institutions to look for attractive ways to make their websites more useful and efficient. To achieve this goal, some auditing work needs to be done. Such an auditing task can be performed in at least two alternative ways. First, users of a specific website can be asked to evaluate their browsing experience. Then actions will be taken to improve the website's structure and/or content based on the feedback received from the clients who did provide a feedback. Second, clients' automatically recorded navigational history is analyzed and the website is tuned up accordingly. There is no doubt that the second option is the best. This is mainly because it does not rely on clients' manual voluntary input. Not only that, but also the analysis of navigational history of clients can be fully automated. This kind of analysis is referred to as web usage mining (WUM) and precisely in our case it is web log mining.

Web usage mining has many applications [10], e.g., personalization of web content, pre-fetching and caching and support to the design, recommendation systems [15], prefetching and cashing [14], among others. There are various benefits of web usage mining, especially in e-commerce. Clients can be targeted with appropriate advertisement. Also, relevant products can be suggested to clients in real-time while browsing the website. According to [7], the usage mining process can be divided into three steps. It starts first with data cleaning and pre-processing. Second, the pre-processed data is mined for some hidden and useful knowledge. Finally, the web log mining process ends by analyzing the mining results.

Like any mining task, web usage mining has also to deal with huge datasets. Therefore, issues related to space availability and running time are present and has to be faced. Besides these computational challenges, there exist some other ones that need to be dealt with as well. These issues are mainly due to the nature of the web log file itself [1]. Web server logs were not meant in the first place to be used for tracking users' navigational behavior. Their first purpose was to be used by server administrators to keep track of the server's bandwidth and capacity. The following are globally the problems a data miner faces when using server side collected data namely web access log to discover navigational patterns.

* Corresponding author at: Department of Computer Science, University of Calgary, Calgary, Alberta, Canada. Tel.: +1 403 210 9453.
E-mail addresses: alhajj@ucalgary.ca, rsalhajj@gmail.com (R. Alhajj).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.04.014

First, we need to distinguish between different visitors. The problem is that certain visitors may use proxy servers or share the same machine to browse the website. Therefore, using the IP address assigned to a user's computer as a unique identifier might lead to erroneous results. Second, users use backward and forward buttons of the browser and these actions are not recorded in the log. Therefore, we need to deal with missing information. Also, when a user requests a resource, the server will most likely log more than one entry. Many records might be added to the log for one single request. We have to get rid of the extra information the log collects. Third, we need to identify the different browsing sessions a user might have within a period of time. Forth, we need to estimate the time spent by a user on the last page he/she visited during a specific session. In addition to the aforementioned problems which are directly related to the web usage mining process itself, there are problems related of its applications such as online navigational pattern prediction. This prediction task has to be done in a timely manner with the best accuracy possible.

In this paper, we are interested in the process of web log mining and online navigational pattern prediction. The usage data we have selected to work on is the data logged by a web server in a file called access log file. Usage data can also be collected from visitors' computers by using adapted browsers or by using techniques such as cookies; we stayed away from that option simply because it might raise privacy concerns. Therefore, the first contribution of this paper is directly related to the access log data cleaning and preparation. The second contribution is the use of a density based clustering algorithm to mine for navigational patterns. The third contribution is a suggestion of a new efficient and accurate way in predicting navigational behavior online. Test results support the effectiveness of our method.

The first contribution of this paper is related to data cleaning and preparation. It starts by creating page views and ends up by generating sessions. Amongst the existing approaches for sessions' identification are the time-based heuristics. The idea of these time based heuristics is the use of a duration threshold to decide whether a session has ended or not. Our contribution in this area consists of the improvement of such a heuristic in order to get better quality results.

The second contribution is the use of a density based clustering algorithm namely DBSCAN, to mine for navigational patterns. We have opted for clustering instead of association rules discovery or sequential patterns mining because both of these two techniques require a user input parameter such as the minimum support. This is not the only issue about these two approaches but also they cannot detect low frequent and meaningful patterns unless the minimum support is too low which means too many rules and frequent patterns to generate. The clustering algorithm we have opted for requires one crucial input parameter from the user. DBSCAN is very sensitive to that input parameter. However, we have used a second algorithm that helps in finding the right value to use for that input parameter. Also, we have opted for DBSCAN because it detects outliers unlike some other clustering algorithms such as K-means for example. Outliers are closely related to erroneous results thus detecting and removing them is important [31].

The last contribution is the use of an inverted index built from all identified sessions from the log to narrow down and speed up the search for a k nearest neighbors for an online session, where k is positive integer specified by the user. We also cared about the accuracy of predicting the pattern of an online session in two ways. First, by taking into consideration not only the pages a user have viewed so far but also the order of those pages. For this we used generalized suffix trees. Second, we cared about the patterns where the order is not important and for this we used a binary matrix with pages as columns and sessions as rows. Details of these methods will follow in the related sections.

The rest of this paper is organized as follows. Section 2 is dedicated to related work. Section 3 includes a description of the proposed framework that integrates the three contributions mentioned above. Section 4 presents the experiments conducted and the results obtained. Section 5 is conclusions and future research directions.

2. Related work

In WUM in general it is not required to know about a user's identity; however, it is necessary to distinguish between different users [4,13,20,27]. If a website requires users to sign in before they can start browsing, it will be very easy not only to differentiate between users but also to identify each single user. The problem arises when a website allows visitors to anonymously browse its content, which is common place. In this case, relying only on what the web server records in the log to differentiate between visitors becomes challenging. In fact, it becomes more challenging if the web server is logging visitors' activities using common log format. The difficulty in distinguishing between visitors arises from the fact that some visitors' activities will be logged with the same IP address due to transiting by the same proxy server or by sharing an internet connection. Different ways to distinguish between users has been listed in [24].

By using cookies users can be identified. When a visitor connects for the first time to a website that uses cookies, the server sends a cookie to the web client along with the requested resource. The next time the same user requests another resource from the same website, the browser will send the cookie stored on the visitors computer along with the request if the cookie is still not expired. The server will be able to recognize the user if the request is coming with a cookie. The problem with this approach is that users can delete cookies stored on their computers and the server will not be able to recognize the coming back visitors.

A simple and straightforward approach in resolving the issue of having users with the same IP or domain name is the elimination of all requests coming from proxies and shared IPs. For example, all requests having the word proxy or cache in their domain name will be eliminated. The drawback of this approach is that navigational patterns of proxy/cache users will not be discovered. Also the dataset may become too small to conduct a valid analysis.

In order to mine for navigational patterns it is mandatory to know what visitors have looked at each time they have visited the website. Each time a visitor comes to the website is considered a session. Identifying users' sessions from the web log is not easy as it may seem. Logs may span long period of time during which visitors may come to the website more than once. Therefore, sessions' identification becomes the task of dividing the sequence of all page requests made by the same user during that period into subsequences called sessions. Many approaches have been used by researchers for sessions' identification. According to [24], the most popular session identification techniques use a time gap between requests.

It has been mentioned in [7] that many commercial products use 30 min as default time-out threshold. However, many thresholds can be found in the literature. These thresholds vary from 10 min [5,7,12,22] to 2 h. It has been also mentioned in [24] that the most widely used time gap is 25.5 min established based on empirical data. This threshold has been calculated as follows. The authors of [5] conducted a study that allowed them to calculate the mean inactivity time within a site. The value found was 9.3 min. After they added 1.5 standard deviations to the mean, the value of 25.5 min was obtained and was defined as a standard inactivity time. This was defined as a cutoff threshold to identify

sessions. Again, it has been mentioned in [24] that many commercial and free applications use the rounded 30 min time gap.

According to [6] a session must not contain a backward reference. This approach consists of converting log data into a set of traversal sub-sequences. Each traversal subsequence represents a maximal forward reference from the starting point of a user access sequence. A forward sequence of references is a sequence that does not contain repeated references. A repeated reference is called a backward reference. For example, let A, B, C, A and D be a sequence of references requested by one user. This sequence contains a repeated reference A; therefore, it will be split into two maximal forward sequences: A, B, C and A, D. Each of these two forward sequences is considered a session. This idea of identifying sessions does not seem to justify itself as also mentioned in [24]. First, a visitor might click on the back button of the browser to review a page and this action might not reach the server due to the use of cache. Second, a visitor might be forced to go back to a page during one single visit simply because that page is like a hub in the website.

Session identification by referrer has been described in [3] as follows. Let S be a session under construction and let p and q be two consecutive page requests, where p belongs to S. q is considered to belong to S if its referrer belongs to S and the difference between the timestamps of p and q does not exceed a predefined threshold. Otherwise, q is considered to be part of a new session. According to the comparative study conducted by [3], session identification by referrer exhibited very poor performance in the presence of framesets. However, time-oriented heuristics were less affected by the presence of framesets.

Different kinds of analysis can be performed on log data. For instance, log analysis tools, as mentioned in [8], take raw web data as input, process it and output some statistical results. These results might be some statistics reflecting the site activity, such as the total number of hits. It might be some diagnostic indicators like 'page not found' errors. Also it may include users' demographics such as top geographic locations. All these statistics are useful and will help administrators to fix problems. Also it will provide support for marketing decisions [24]. However, there exist other techniques that can process log data further and extract more complex observations that convey knowledge [18]. These techniques are referred to as mining techniques.

In the context of WUM, two types of clustering can be performed. Clustering users or clustering pages. Users clustering should result in groups of users visiting similar pages. Researchers have proposed clustering techniques for both types of clustering. The authors of [25] have proposed an algorithm called PageGather to cluster pages based on cliques. The authors of [21] have suggested the use of K-means clustering algorithm to cluster users. In our proposed mining approach, we have suggested a density based clustering algorithm to cluster users in significant groups. We have opted for this clustering approach because it detects outliers and because it does not require knowing the number of clusters beforehand.

In addition to the benefits of discovering users' navigational patterns from browsing history, there are also benefits in predicting online navigational behavior of a visitor. This prediction task can for example help reducing the response time of the server and can help also in personalizing the content of a page before it is delivered. According to the literature, there are two main approaches in predicting navigational patterns in real time. These two approaches are clustering based and the K-nearest-neighbors approach. There is no doubt that scalability and accuracy are the two things we look for in any prediction technique. Clustering-based online pattern prediction solutions are not as accurate as the k-Nearest-Neighbors (kNNs) approach; however kNN suffers from its non-scalability. In our proposed refined mining process, we suggest a fast kNN method to predict navigational patterns online.

3. The proposed web usage mining approach

In this section, we present our view on how we can get good quality sessions, what mining technique is better to use to get all navigational patterns and how we can perform online navigational

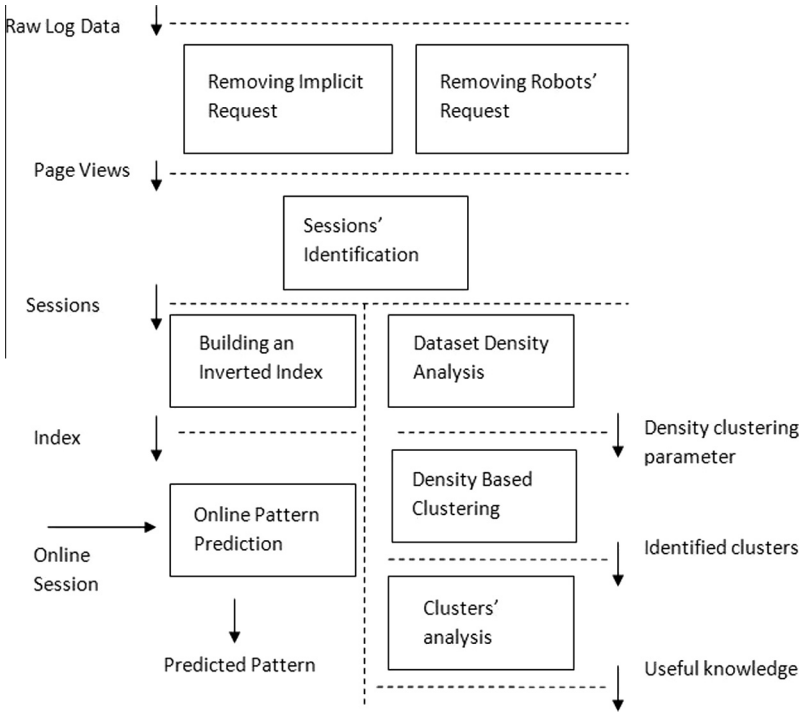

Fig. 3.1. Block diagram of the proposed framework, the results of session identification process go through a density based clustering to mine for useful knowledge. In the same time they go through an efficient online kNN to find the sessions that are mostly related to the current online session.

```
Input: a list of sequences
Output: a list S of valid sessions
begin
01 -  S ← {}
02 - Sort sequences according to user id and timestamp
03 - For all sequences do
04 -        Divide the current sequence according to page-stay threshold
05 -       If the sequence is monolithic
06 -           current sequence is a single session
07 -           S ←S ∪ {current sequence}
08 -       Else /*sequence with gaps*/
09 -           If there is a shared pattern between the sub-sequences
10 -              Each subsequence is a separate session and all sessions are from one single user
11-                  S ←S ∪ {All su-bsequences}
12 -       Else /* no shared pattern*/
13-           Skip sequence.
14- End for
15- Return
End
```

Fig. 3.2. dividing a sequence of requested pages into separate sessions.

Table 3.1
Example of a shared pattern. Three different sessions for the same user share the pattern P1 and P4.

User Id	Session ID	Session
User01	1	P1, P3, P4, P2
User01	2	P1, P2, P4
User01	3	P1, P4

pattern prediction in a timely manner. All the components of our proposed framework are depicted in Fig. 3.1.

As shown in Fig. 3.1, the cleaning and preprocessing step consists mainly of removing implicit requests and removing requests made by robots. A robot (i.e., "web crawler") has been defined in [19] as well known software program that performs automatic information retrieval from the web by taking advantage from its structure to move from page to page and from site to site. Not removing crawlers' traverses from the usage data will affect negatively the analysis results.

In our proposed framework, we consider a visitor as a robot if one of the two following criteria applies: (1) the visitor has requested the file robots.txt; (2) the visitor exists in the robots lookup list [26]. Authors of [28] proposed to add a third criterion based on the browsing speed. They think if a number of requests have been made in less than a specific threshold and the number of the requests is also above another threshold then the requestor is a robot. We did not adopt this idea simply because we think that legitimate crawlers will request the file robots.txt and/or will identify themselves in the agent field of the log and illegitimate crawlers will avoid simplistic navigational behavior to hide.

According to the literature, there exist different ways to detect sessions in web log data. Time-out oriented techniques are widely used by commercial. In this technique a session is identified by the set of requested pages during a predefined threshold time-out interval. The most used time-out threshold to start a new session is 30 min. In this work, we also stick with this technique, but we will investigate the effect of two different thresholds. Our reasons for opting for a time-out based sessionizing approach is the fact that the technique can handle a log following the common web log format and the extended format. Also, it can be used whether a website is utilizing cookies or not.

Our contribution at this stage came from the following observation. We are interested in navigational patterns. In other words, we are interested in repetition in visits. Consequently, we should be interested in returning visitors with repeated requests. If a returning visitor requests different pages each visit, this will not help in discovering a navigational pattern. In fact, a visitor who returns to the website is more than likely to view some or all pages viewed in previous visits. Therefore, instead of just cutting a sequence of pages belonging to a user using the 30 min threshold and accepting the results, we double check whether there exist a shared pattern between these sessions or not. A shared pattern between sessions is simply having some specified pages in common between these sessions. For example let's suppose that session1 corresponds to eight visited pages; say P1, P3, P6, P8, P12, P122, P123 and P600. On the other hand session 2 corresponds to 7 visited pages; say P3, P21, P122, P600, P706 and P900 then a shared pattern would be P3, P122 and P600. If we found out that there exists some shared pattern between all identified sessions for the same visitor we accept the resulted sessions. However, founding out that there is nothing in common between the identified sessions of the visitor might be a good indicator that we are not dealing with one single visitor but with overlapping sessions belonging to different users.

To implement our idea we have modified the original time-out sessionizing heuristic to make sure that the identified sessions from the same sequence must share a pattern. This will increase the quality of the sessions generated as demonstrated in the test results reported in Section 4. The pseudo code of our algorithm is given next in Fig. 3.2.

The different visits of the same user share some or all pages viewed. The order of the pages viewed is not important and the shared patterns do not have to be contiguous. As an example, let's assume that we have the sessions shown in Table 3.1 belonging to the same user. It is obvious that P1 and P4 are shared between all three sessions. Therefore, we say that P1, P4 is a shared pattern between all sessions.

Testing whether a list of sessions share a pattern or not needs to be performed in an efficient manner. For the case where the patterns care about the page order, we have opted for generalized suffix trees. Suffix trees are very efficient data structures that often provide linear time solutions to challenging string problems. In WUM context, they have been used by [29] to mine for navigational patterns. A suffix tree can be used to represent all suffixes of a given string. A generalized suffix tree can be used to represent all suffixes of a set of strings. Fig. 3.3 shows the algorithm used to look for a shared pattern between sessions where the page order is important.

Detecting patterns where the page order is not important is easier. We propose a simple method where a binary matrix with pages as columns and sessions as rows is used as a data structure. Performing a logical AND operation on all columns will yield the

```
Input: a list of strings S, minimum length of the shared pattern l, minimum number of strings
sharing the pattern m.
Output: Answer is yes, if shared pattern exists and No otherwise.
01- Answer ← no
02- Build Generalized suffix tree for all strings
03- For each leaf node f
04-    current node ← f
05-    While root is not reached
06-       p ← parent(current node)
07-       add string id of the leaf node f to the node p if this id has not been added to it yet
07-       if the number of string ids added to p >= m and ||path(p, root)||>l then
08-            Answer← yes
09-       end if
10-       current node ← p
11-    end While
12- end For
13- Return Answer
```

Fig. 3.3. Using generalized suffix trees to detect patterns where page order is required.

Table 3.2
Detecting patterns where the order of the pages is not required.

	P1	P2	P3	P4	P5	P6	P7
S1	1	1	1	0	1	1	0
S2	1	0	0	1	1	0	0
S3	1	1	1	0	1	1	0

patterns detected. As shown in Table 3.2 if we perform an AND operation on all columns then P1 and P5 will be the pattern needed.

We need to mention that we have noticed that the amount of research work published with regards to data cleaning and pre-processing in WUM is very little compared to what has been published with regards to the actual mining. Among related papers to data pre-processing, we could find two papers that have caught our attention. The authors of the first paper [16] have made an effort to come up with one comprehensive algorithm that cleans the data, identifies users and sessions. The problem with their approach is that beside the work they have done in terms of data cleaning and users' identifications, they have proposed nothing new in terms of sessions' identifications. They have simply adopted the use of a time-out based technique using the 30 min for sessions' identification. The authors of the second paper [30] have proposed a complex system to reconstruct sessions. The proposed system cannot reconstruct sessions unless a model is built from real sessions. According to the authors real sessions can be obtained using referrer information available in the log. The problem with this approach is that first it works only for logs following extended format. Second, using referrer information does not guarantee 100% accurate results; otherwise, we just adopt this approach to identify all sessions at once and no need to add one more layer. Our proposed approach is a real enhancement to the timeout-based technique and it does work for both log formats.

3.1. Refined pattern mining process

The algorithm we suggest for the refined pattern mining step is DBSCAN [9]. This algorithm falls into the category of density based clustering algorithms. We have opted for this algorithm mainly because it detects outliers. In addition, it does not require the number of clusters to be known by the user beforehand unlike other algorithms such as K-means. However, it requires another input parameter and it is very hard for the user to guess a good value for that parameter. Not only that but also DBSCAN is very sensitive to that input parameter. For that reason, we are recommending the use of another algorithm called Optics which helps in selecting an appropriate value to use as input for DBSCAN.

3.2. Efficient online navigational behavior prediction

To predict online patterns, we propose a scalable kNN-based approach. A classical kNN-based approach would look for k nearest neighbors of a point in a dataset by computing distances between that specific point and all other points in the dataset. Then these points are sorted according to their distances to the original point in an ascending order and only the first k points are outputted. In the context of an online application, this might be acceptable only when the size of the dataset is relatively small. However; if the dataset size is huge this becomes unacceptable. Our suggested kNN scalable solution is based on the idea of using indexes. We view sessions as text documents and we consider an online session as a query. Therefore, looking for the k nearest sessions to an online session will be translated to looking for the documents most relevant to the current query. Our suggested approach consists of the adoption of the well known technique used in information retrieval systems, namely TF-IDF combined with the cosine similarity measure to find the closest sessions to a current online session. On the other hand and in order to speed up the search process, we built an inverted index from all sessions.

We view online pattern detection as the problem of finding the most relevant documents to a query made of set of keywords from a repository of text documents based on the following observations. A text document is made of a collection of some of the words available in the vocabulary of a specific language. Similarly, a session is made of a collection of the references of some of the pages of a website. Text documents may contain repeated terms. Also, a session may contain one page or more many times. This can happen simply when a user reload a page or comes back to one of the pages viewed earlier. Therefore, based on the aforementioned observations, we decide to treat sessions as text documents and benefit from the well developed techniques in the informational retrieval domain to increase results accuracy. This method is depicted in Fig. 3.1 where identified sessions go through an inverted index process then they are compared with an online session.

When a visitor is browsing the website, we keep track of the last few requested pages only. We use the concept of sliding window. Each time the visitor requests a new page we slide the window by one. Consequently, the newly requested page will be added and the oldest page in the window will be dropped. Let's assume that the size of the sliding window is w. Before we can make any prediction about the navigation pattern of the current visitor, we

```
Input: the newest w pages requested
Output: a predicted pattern
01- calculate the TF-IDF values of the query vector based on the w recent requested pages
02-find all sessions that contain these w pages using the inverted index
03-calculate the cosine between the query vector and vectors of all relevant session
04-Sort cosine values in a descending order
05-pick the top k sessions on the top of the list
06-compute the centroid of picked sessions
07-Return the calculated centroid
```

Fig. 3.4. Predicting a pattern for an online session.

```
Input: a set S of sessions, w size of the key in terms of number of terms the key has to contain
Output: an inverted index
01- For all sessions in S Do
02-   If session length is larger than w Then
03-     Create all possible keys of size w
04-     If a created key have not been added yet to the index Then
05-       Add the pair (Key, session ID) to the index
06-     Else
07-     Update the list of the sessions associated with that key by adding the current session ID
08-     End IF
09-   End IF
10- End For
```

Fig. 3.5. Inverted index creation given a set of sessions.

Table 3.3
List of sessions – example.

Session ID	Session
01	P1, P2, P3
02	P1, P4, P1, P6
03	P2, P3, P4

Table 3.4
Inverted index – example.

Key	List of sessions containing the key
P1,P2,P3	01
P1,P4,P1	02
P4,P1,P6	02
P2,P3,P4	03

Table 4.1
Testing environment.

Main memory	1 GB
CPU	3.4 GHz
Operating system	Windows XP professional SP3

have to wait till the visitor requests at least w pages. Once it is the case, a query vector is created. The size of the vector is the number of pages in the website. The values in the query vector are the TF-IDF values of each requested page existing in the current sliding window. If a page does not exist in the current sliding window then its TF-IDF value is zero. The next step is computing the cosine between the query vector representing the online session of the current visitor and a selected sub-list of vectors representing relevant sessions to the current online session. We can get this relevant sub-list of vectors by using an inverted index. Therefore, instead of computing the cosine between the current online session and all sessions we do it only for a subset of the sessions. Thus, a huge computational time is being saved. How to build the inverted index and how to get a subset of relevant sessions to an online session is detailed in the next section. A TF-IDF value of a specific term in a document d is calculated as the following:

$$TF(d,t) = \log\left(1 + \frac{n(d,t)}{n(d)}\right) \quad \text{And} \quad TF.IDF(d,t) = TF(d,t)/n(t)$$

where $n(d,t)$ is the number of occurrences of term t in document d; $n(d)$ is the number of terms in document d. $n(t)$ is the number of documents containing term t.

The cosine between two vectors A, that represents the online query vector and B that represents a vector from the chosen subset of session vectors is calculated as: $\cos(A,B) = \frac{A \cdot B}{||A|| \cdot ||B||}$. The elements of each vector are the TF-IDF values mentioned before. The algorithm in Fig. 3.4 shows how to predict the pattern of an online session.

We have used an inverted index in order to get only sessions that contain the current pages viewed by an online user. Only these sessions will be involved in cosine computations. This way, all irrelevant sessions will not be taken into consideration in the prediction process. The inverted index we are using does not map one single term to a list of documents that contain that term and to be more precise it does not map one page to a list of sessions containing that page. It maps an ordered sequence of w pages to a list of sessions containing that contiguous sequence of pages. As an example let's assume that we have the following list of sessions. The algorithm in Fig. 3.5 shows how the inverted index is built.

For example, the inverted index for the list of sessions in Table 3.3 assuming $w = 3$ is shown in Table 3.4.

4. Experiments and results

This section describes the experiments conducted and evaluates the results obtained. We describe briefly the datasets we have used to conduct the experiments also we describe the testing environment. We present some statistics about the datasets used. We describe the work conducted to enhance the classical time-out based heuristic to identify sessions. We comment on the patterns discov-

Table 4.2
Datasets used and number of page views identified.

Data set	Period	Number of entries	Number of page-views
NASA web log	Month of July 1995	1,891,715	523,160
UNB web log	4 days in Jan 2003	168,942	85,557

Table 4.3
A sample of robots detected in both datasets.

NASA dataset	UNB dataset
hydra.wcupa.edu	pcp067869pcs.glst3401.nj.comcast.net
bang.engin.umich.edu	129.175.169.7
wfr-20–1.rz.uni-frankfurt.de	alfredo.wise-guys.nl
query2.lycos.cs.cmu.edu	218.20.189.12

ery results obtained using DBSCAN. We show the gain earned in terms of running time using our proposed kNN scalable version to predict online navigational patterns.

Two different real datasets have been used to conduct our series of tests. The first dataset is from the NASA space center web server log. As it has been confirmed by [11] this is a well known dataset that has been used by many researchers. The log data was recorded according the common format and it spans the whole month of July 1995. It is available on the internet for the public [23]. The second dataset we have used has been provided to us by the authors of [17]. We have been given the log data plus the whole list of pseudo real sessions. Their paper [17] has a detailed explanation how pseudo real sessions have been constructed. The dataset has been extracted from the University of Brunswick web server log file. The computer we have used to conduct all experiments has the characteristics shown in Table 4.1. All algorithms have been implemented using Java programming language.

We start the process by filtering out all noisy data, implicit requests, entries related to errors and any entries related to requests made by robots. Tables 4.2 and 4.3 provide an overview of the work done at this stage.

4.1. Sessions' identification

The improvement we are suggesting at this step of the WUM process is based on the idea that returning visitors without any repeated requests will not help in discovering navigational patterns. Therefore, after detecting sessions using a time-out threshold such as 30 min we proceed to check whether the detected sessions of the same user share a pattern or not. If a shared pattern existed we approve the identified sessions. Otherwise, the sequence we are trying to divide into sessions is skipped.

We have conducted the following investigation to see the effect of imposing our suggested additional constraint on the quality and quantity of the identified sessions. In our investigation we have used two different time-out thresholds: 30 min and 10 min. We have conducted equal number of experiments for both thresholds. For each threshold we tried three different values of the required minimum length of a shared pattern: 1, 2 and 3. Also for each threshold and for each value of the required minimum length of a shared pattern, we introduced another variable R which varies from 10% to 100%. A 100% value of R means that all concerned sessions must share a pattern. A 50% value means that only half of the concerned sessions must share a pattern and so on. For instance, the first line of Table 4.5 corresponds to the results obtained by our enhanced heuristic using a 10 min time-out threshold where all (100%) concerned sub-sequences must share a pattern of length at least 3.

Each time we run our sessionizer, we record the total number of correctly identified sessions, the total number of identified sessions and the number of false positives. In order to be able to compare the quality of the results returned by our proposed heuristic and the results returned by the classical time-out based heuristic, first we have run the classical time-out based heuristic for both thresholds and the results are as shown in Fig. 4.4. Second, we have run our proposed sessionizer using the 10 min time-out threshold and the results are also as shown in Tables 4.5, 4.6 and 4.7. Finally, we have repeated the same steps using the 30 min time-out threshold as illustrated by Tables 4.8, 4.9 and 4.10.

Before we start commenting on the results obtained by using our proposed heuristic, we have to mention that the total number of real sessions in the dataset is 17,975. Also, we need to notice that even though the classical heuristic was able (for both thresholds) to correctly identify a high number of sessions it has also included a very high number of false positives. 30.59% false positives have been added when the 10 min threshold has been used and 27.41% when the 30 min has been used; see Table 4.4. This is due to blindly dividing a sequence into a series of sessions based on a predefined threshold only. These high numbers of false positives are nothing but a source of misleading results. To the contrary, the results obtained by our proposed heuristic have a very low number of false positives especially when we require a lengthier shared pattern between most of the concerned sub-sequences. The lowest ratio we have got is 1.6% as shown in Table 4.5. We have noticed that using the 10 min threshold in our proposed heuristic yield better results than using the 30 min threshold. This might be mainly due to the nature of the website. We have to mention also that when we are reducing the number of false positives to its minimum, the number of correctly identified sessions is reduced as well. Therefore, the correctly identified session might not be representative of all the real navigational patterns. This concern is similar to the one triggered by the elimination of all requests coming from hosts having the term proxy.

To conclude, the experiments conducted showed clearly that we can get good quality results by taking into consideration not only the time gap between two consecutive requests of the same user but also by checking whether there exist any shared pattern between the concerned sub-sequences. The idea behind our proposed sessionizer can be plugged to any non-time-based heuristics as well, such as the one based on the referrer if the log is following the extended format.

4.2. Clustering

We have used the two different datasets at this stage of the WUM process to show the benefits of the algorithms we are suggesting to use for navigational patterns mining. In this section we first describe the experiments conducted and the results obtained using a 1 month web usage data from NASA web log. Then, we describe what we have done with the University of Brunswick web log data.

4.2.1. NASA web log data

The raw data has gone through all steps of cleaning, pre-processing and sessions' identification before it was ready for our clustering algorithm. In fact, it has gone into another step before clustering. It has been analyzed by OPTICS [2] first. OPTICS has taken the identified sessions in a form of a matrix as input and outputted them in a certain order according to their closeness to each other. The number of columns of the input matrix is the number of pages in the website and its number of rows is the number of sessions. In other words, each session is represented by a vector.

Table 4.4
Results of the classical time-out based sessionizer.

Threshold	Correctly identified sessions	False positives	Total	Ratio of false positives/total (%)
10 min	16,173	7129	23,302	30.59
30 min	15,345	5793	21,138	27.41

Table 4.5
Enhanced sessionizer results; time-out = 10 mn, length shared pattern $\geqslant 3$.

R (%)	Correctly identified sessions	False positives	Total identified sessions	Ratio of false positives/total (%)
100	6995	111	7106	1.6
90	6995	111	7106	1.6
80	6995	111	7106	1.6
70	6996	114	7110	1.6
60	7098	130	7228	1.8
50	7189	139	7328	1.9
40	7272	158	7430	2.1
30	7445	202	7647	2.6
20	7804	338	8142	4.2
10	8309	490	8799	5.6

Table 4.6
Enhanced sessionizer results; time-out = 10 mn, length shared pattern $\geqslant 2$.

R (%)	Correctly identified sessions	False positives	Total identified sessions	Ratio of false positives/total (%)
100	7309	128	7437	1.7
90	7347	128	7475	1.7
80	7488	147	7635	1.9
70	7814	189	8003	2.4
60	8508	314	8822	3.6
50	9303	496	9799	5.1
40	9990	693	10,683	6.5
30	10,616	924	11,540	8.0
20	11,234	1301	12,535	10.4
10	11,565	1599	13,164	12.1

Table 4.7
Enhanced sessionizer results; time-out = 10 mn, length shared pattern $\geqslant 1$.

R (%)	Correctly identified sessions	False positives	Total identified sessions	Ratio of false positives/total (%)
100	10,602	2625	13,227	19.8
90	10,989	2882	13,871	20.8
80	12,491	3602	16,093	22.4
70	13,352	4107	17,459	23.5
60	14,236	4817	19,053	25.3
50	14,790	5337	20,127	26.5
40	15,024	5537	20,561	26.9
30	15,176	5705	20,881	27.3
20	15,201	5738	20,939	27.4
10	15,208	5742	20,950	27.4

Table 4.8
Enhanced sessionizer results; time-out = 30 mn, length shared pattern $\geqslant 3$.

R (%)	Correctly identified sessions	False positives	Total identified sessions	Ratio of false positives/total (%)
100	7390	323	7713	4.2
90	7390	323	7713	4.2
80	7390	323	7713	4.2
70	7394	327	7721	4.2
60	7478	342	7820	4.4
50	7570	366	7936	4.6
40	7666	404	8070	5.0
30	7882	403	8285	4.9
20	8217	648	8865	7.3
10	8720	917	9637	9.5

We have fed OPTICS with two different kinds of matrices, a binary matrix and a non-binary matrix. Each cell in the binary matrix represents whether a specific page has been requested during a specific session or not, 1 for yes and 0 for no as shown in Fig. 4.1.

Table 4.9
Enhanced sessionizer results; time-out = 30 mn, length shared pattern ⩾ 2.

R (%)	Correctly identified sessions	False positives	Total identified sessions	Ratio of false positives/total (%)
100	7709	367	8076	4.5
90	7737	367	8104	4.5
80	7945	417	8362	5.0
70	8236	482	8718	5.5
60	8900	665	9565	7.0
50	9616	929	10,545	8.8
40	10,142	1183	11,325	10.4
30	10,670	1474	12,144	12.1
20	11,155	1911	13,066	14.6
10	11,311	2070	13,381	15.5

Table 4.10
Enhanced sessionizer results; time-out = 30 mn, length shared pattern ⩾ 1.

R (%)	Correctly identified sessions	False positives	Total identified sessions	Ratio of false positives/total (%)
100	10,980	2935	13,915	21.1
90	11,316	3154	14,470	21.8
80	12,416	3754	16,170	23.2
70	13,049	4118	17,167	24.0
60	13,822	4642	18,464	25.1
50	14,197	4902	19,099	25.7
40	14,355	5049	19,404	26.0
30	14,476	5146	19,622	26.2
20	14,496	5160	19,656	26.3
10	14,496	5160	19,656	26.3

P1	P2	P3	P4
0	1	0	1
1	1	1	0
0	1	1	1

Fig. 4.1. An example of a binary matrix that can be used as input for OPTICS.

P1	P2	P3	P4
0	5	0	10
20	2	5	0
0	10	6	7

Fig. 4.2. An example of a non-binary matrix that can be used as input for OPTICS.

On the other hand, each cell in the non-binary matrix represents how long in total a specific page has been viewed in a specific session as shown in Fig. 4.2.

The results of OPTICS have been plotted as shown in Figs. 4.3 and 4.4. In Fig. 4.3 the calculated reach-ability distances of all points (sessions) translated into a binary matrix are plotted according to the order OPTICS has put them in. When we pick the value 1.2 from the *Y*-axis and we draw a horizontal line we can see that this value will result in four different clusters. One large cluster starting from the right that contains many (dense regions) sub-clusters and then three smaller clusters to the left. OPTICS gives a clear picture of the dense regions within the dataset. This makes it very easy for us to select the appropriate density parameter epsilon for DBSCAN to cluster the dataset. In Fig. 4.4 the graph is obtained by plotting OPTICS results using our non-binary matrix. It is very clear that the graph we have got this time is more precise than the one we have got using binary values. This is simply due to the fact that we are feeding OPTICS with more precise values rather that zeros and ones. Also when we pick the value 0.0022 from the *Y*-axis and we draw a horizontal line we can see that this value will result in two different clusters as shown in the same figure. As an example we have run DBSCAN with epsilon equals to 0.0022 and minimum point's equals to 5. We have got exactly two clusters as OPTICS claimed and we have analyzed the smaller cluster to see what this pattern really means. The only purpose for choosing 1.2 in Fig. 4.3 and 0.0022 in Fig. 4.4 is to show how OPTICS helps in detecting dense regions. It is up to the final user to choose a reachability distance. The value to be selected is the one that helps in getting clear and separate dense areas.

After we have identified the members of the smaller cluster using DBSCAN, we have computed the centroid of the cluster. Then we have all pages of the centroid that have a weight above zero. The list of these pages is as shown in Table 4.11.

The centroid of a cluster is the mean vector of all cluster members. The dimension value for each page in the mean vector is calculated by finding the ratio of the sum of the page's weights across sessions to the total number of sessions in the cluster. As we can see from Table 4.11, all pages are under the same category/shuttle/ and P63 "/shuttle/missions/sts-48/sts-48-info.html" has the highest weight.

4.2.2. University of Brunswick web log data

The main purpose of doing the following set of experiments on the UNB web log data is not to show again how to use OPTICS and DBSCAN to mine for navigational patterns but to show how our refined way of identifying sessions can help in eliminating some possible fictional patterns that can be created using a classical time-out based sessionizer heuristic. We have not used the whole dataset to perform our experiments. Part of the dataset is enough to show the effect of our proposed sessionizer. Also there is no need to run DBSCAN to cluster the dataset and calculate the centroids of the identified clusters and analyze them. Running OPTICS and plotting the calculated reachability distances will do the job.

We picked the web log entries corresponding to several hundreds of real sessions. We sorted these log entries according to the visitor IP/Domain and time stamp. OPTICS results for real sessions are shown in the reachability distance graph in Fig. 4.5. Then we sessionized the resulted sequences applying the classical time-out based heuristic of 30 min and 10 min time-out thresholds. Also we have sessionized the same sequences using our enhanced time-out heuristic using both mentioned thresholds. Our goal is not to

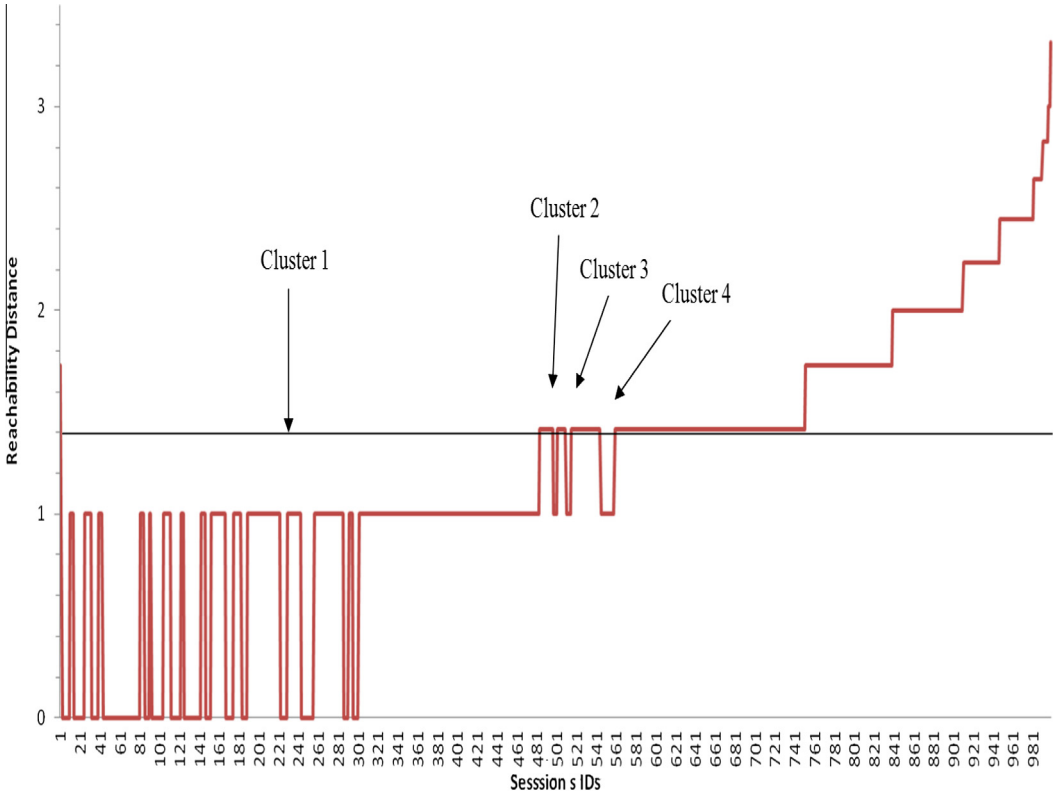

Fig. 4.3. Reachability distances graph where a binary matrix has been used as input for OPTICS.

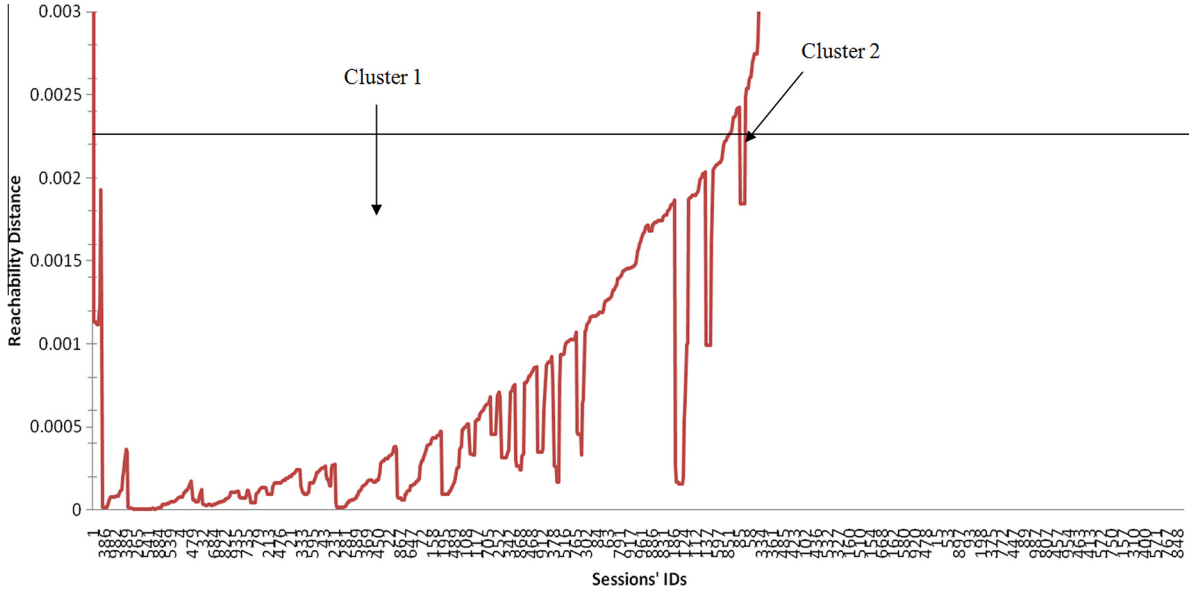

Fig. 4.4. Reachability distances graph where a non-binary matrix has been used as input for OPTICS.

count the number of false positives of each sessionizer but to show how a time-out based sessionizer can add fictional patterns and lead to misleading results. A non-binary matrix is built for the identified sessions each time we run a sessionizer. These matrices represent the time spent on each page during each session. After normalizing the data in these matrices, we run OPTICS five times as shown in Table 4.12. OPTICS results for both the classical and the enhanced time-out using 30 min thresholds are shown in Figs. 4.6 and 4.7 respectively. OPITCS results for the 10 min time-out thresholds are shown in Figs. 4.8 and 4.9 respectively. We can notice that the classical time-out threshold adds more fictional patters compared to the real sessions.

The obtained results demonstrate that plotting the reachability distances based on the sessions identified by our enhanced sessionizer gives the closest picture to reality. This also means that the 10 min threshold is the most appropriate for UNB dataset.

To conclude, in this step of our suggested refined WUM approach we used a density based clustering algorithm to get all

Table 4.11
Ordered weights of pages present in cluster 2 centroid.

Page number	Page	Page weight
P690	"/shuttle/countdown/"	7.63E−06
P310	"/shuttle/missions/sts-69/mission-sts-69.html"	2.33E−05
P522	"/shuttle/countdown/countdown.html"	3.85E−05
P906	"/shuttle/missions/sts-70/movies/movies.html"	8.43E−05
P165	"/shuttle/missions/sts-70/images/images.html"	1.02E−04
P1439	"/shuttle/resources/orbiters/endeavour.html"	2.29E−04
P946	"/shuttle/technology/sts-ewsref/sts_asm.html"	4.71E−04
P75	"/shuttle/technology/sts-newsref/stsref-toc.html"	5.24E−04
P63	"/shuttle/missions/sts-48/sts-48-info.html"	0.003048

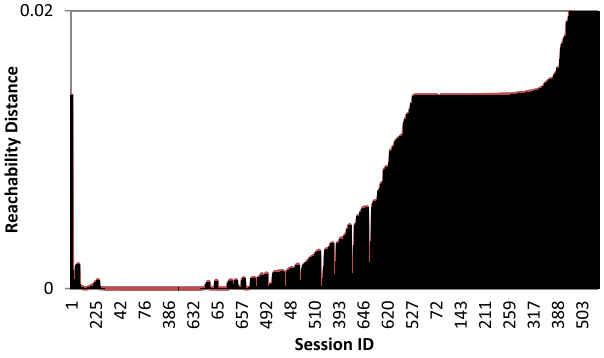

Fig. 4.5. Real sessions' reachability distances graph (referrer).

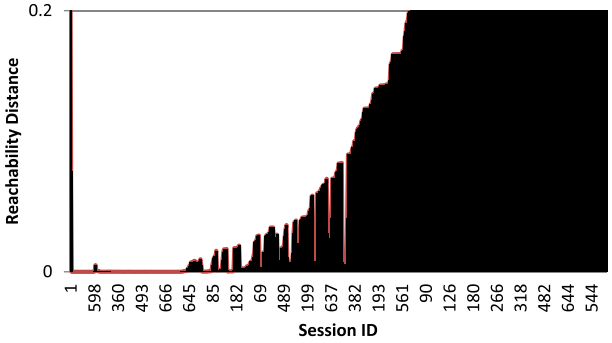

Fig. 4.6. Classical time-out based heuristic (30 mn).

Table 4.12
List of conducted experiments.

Sessions identified using	Threshold
Pseudo real sessions (referrer)	None
Time-out based heuristic	10 min
Time-out based heuristic	30 min
Our enhanced time-out based heuristic	10 min
Our enhanced time-out based heuristic	30 min

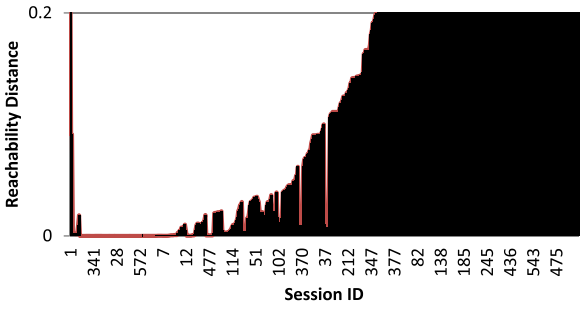

Fig. 4.7. Enhanced time-out based heuristic (30 mn).

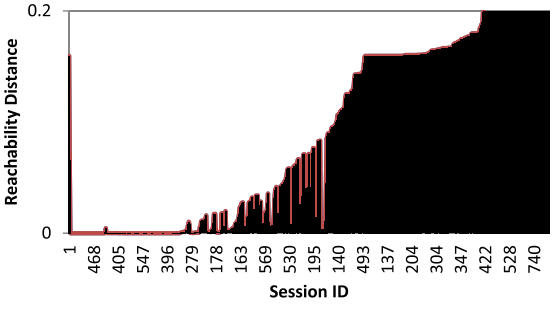

Fig. 4.8. Classical time-out based heuristic (10 mn).

navigational patterns. We can target a specific pattern (cluster) no matter how small it is. This cannot be the case if association rules mining or sequential patterns are used unless with very small support, and we mentioned what are the consequences of opting for very low support.

4.3. Online prediction

At this final phase and as we have mentioned in the previous section, each session is represented by a vector. Each single value in the vector is the TF-IDF value of the corresponding page if it does exist in the session or it is zero if the page does not exist in the session. The goal of the experiments conducted at this stage is to show how much time can be saved by the usage of an inverted index. Also, the goal is to show that the way the inverted index is built helps in getting only sessions that contain exactly the pages contained in the sliding window, in the exact same order and as a contiguous sequence. Both datasets we have used in the navigational patterns mining phase, have been used also to test the efficiency and accuracy of our proposed kNN-based online pattern prediction.

4.3.1. NASA web log dataset

As an example, we have considered the following sequence as the actual content of our sliding window, p690, p166, p1087. The results obtained in terms of running time and in terms of the most relevant sessions are shown in Tables 4.13 and 4.14.

To check how much we will save in running time by using an inverted index, we conducted ten experiments as shown in Table 4.13. First, we took the first 1000 sessions from our original dataset and we measured how long it took to get the five nearest neighbors by using an index and without using an index. Then we took the first 2000 sessions and measured also how long it took to get results in both cases. We kept adding each time 1000 sessions till we reached the size of ten 10,000 sessions. The results shown in Table 4.13 speak for themselves and there is a very huge

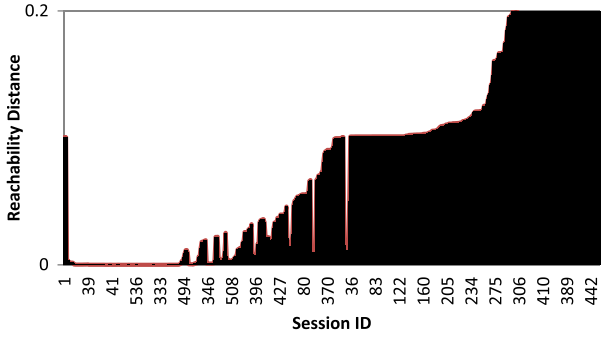

Fig. 4.9. Enhanced time-out based heuristic (10 mn).

Table 4.13
kNN response time in milliseconds for NASA dataset.

Number of sessions	No index	With index
1000	922	15
2000	3609	16
3000	8251	15
4000	14,594	16
5000	22,329	16
6000	34,531	15
7000	49,829	15
8000	62,250	15
9000	76,360	16
10,000	93,923	16

Table 4.14
kNN Results for NASA dataset.

Without use of index	With the use of inverted index
p166, p690, p1087	p110, p690, p166, p1087
p690, p166, p690, p1087	p110, p1190, p690, p166, p1087
p110, p166, p690, p1087	p690, p166, p1087, p522, p1494, p48, p75, p1087,
p110, p690, p166, p1087	p318, p1092, p1327, p110, p1020, p855, p33, p110, p1397,
p110, p1190, p166, p690, p1087	p317, p75, p690, p166, p1087, p1120

difference in running time. Also when it comes to the quality of the results returned in both cases and as we can see in Table 4.14, the usage of the inverted index helps in finding the closest sessions that contain exactly the query. For example, the highly ranked answer when the index is not used is: "p166, p690, p1087". However, the highly ranked answer when the inverted index is used is: "p110, p690, p166, p1087". The second answer is more precise because it has all pages contained in the query and they are in same order and as contiguous sequence. The First answer has all query pages but they are not in the same order.

4.3.2. The UNB web log dataset

For the UNB dataset, we picked the following three consecutive pages as the content of the sliding window "p20, p1682, p3156". Again for this dataset we took the first 1000 sessions from our original dataset and we measured how long it takes to get the five nearest neighbors by using an index and without using an index. Then we repeated the same steps as we did with the first dataset. The ten experiments conducted and their results are listed in Table 4.15.

With regards to the accuracy of the results, this time we have got similar results for both cases. All these returned results, as

Table 4.15
kNN response time in milliseconds for UNB web log dataset.

Number of sessions	No index	With index
1000	1813	15
2000	8016	15
3000	18,562	32
4000	34,797	15
5000	54,890	32
6000	78,093	31
7000	104,422	31
8000	162,516	32
9000	348,453	47
10,000	905,550	46

Table 4.16
kNN results for UNB web log dataset.

Without use of index	With the use of inverted index
p20, p1682, p3156	p20, p1682, p3156,
p20, p1682, p3156	p20, p1682, p3156,
p20, p1682, p3156	p20, p1682, p3156,
p20, p1682, p3156, p3786	p20, p1682, p3156, p3786,
p20, p1682, p3156, p885	p20, p1682, p3156, p885,

shown in Table 4.16, contain the three pages existing in the query "sliding window" and in the exact same order. However, when the inverted index is not used this kind of results is not guaranteed.

To summarize, our proposed approach at this stage helps in reducing the running time and returns more precise results. We cannot claim that we are the first to think about and use indexes in this kind of application but we have not come across any previous work that uses inverted indexes and especially in the way we build the index.

We have opted for TF-IDF as originally defined. TF-IDF focuses on the frequency of a term in a document and in the collection of documents. It might yield better results if it is modified in a way that takes into consideration a specific weight for each term, which in our case can be the time spent on a page.

5. Summary, conclusions and future work

In this paper, we proposed an efficient framework for web log mining and for online navigational behavior prediction. We have reviewed all steps of this process and we have analyzed existing approaches and made an effort to make a contribution at each step. The first step in web log mining consists of different phases. The first phase is data cleaning and preparation. Our framework accepts the raw log data and cleans it to produce page views as Fig. 3.1 shows. The page views enter a second phase of session identification as clarified in Fig. 3.1. The main problem at this level is how the sessions' identification must be done. By checking the web log we can tell when a visitor had started browsing a website but it is very hard to tell when the visitor had left. Several approaches to identify sessions such as time-out based techniques have been proposed by researchers and have been used in commercial products. At this stage of the web log mining process, we opted for this kind of heuristics mainly because it can be applied for web logs following common log format or following extended log format. Our proposed time-out based heuristic has an additional constraint compared to the original one. We have conducted two sets of experiments using two different time-out thresholds. The results obtained from these series of experiments showed that the longer a shared pattern is the lesser false positives are. Also, the bigger the number of sessions of the same user sharing a pattern is,

the smaller false positives are. The conclusion we derive from our investigation at this level is that using our suggested additional constraint in identifying sessions will help in getting less false positives which means good quality input data for mining algorithms.

The second step of the web log mining process is where the real mining job is done. Also at this step of the mining process we have gone through the main web usage mining techniques and tried to find the technique that is efficient and has lesser shortcomings. We found ourselves attracted to clustering; we opted for the one that does not require a prior knowledge about the number of clusters and that detects outliers. The Algorithm we suggested for mining navigational patterns is DBSCAN. We implemented this clustering algorithm and we also implemented OPTICS and we showed how OPTICS can help in selecting the appropriate input for DBSCAN. The conclusion at this step is clustering and especially the combination of the two algorithms help in detecting all patterns and not only the most frequent ones. Fig. 3.1 shows the steps of mining the resulted sessions from the previous step.

The last part of our suggested framework is an efficient online navigational behavior prediction component. Online pattern prediction is very useful. For example, it helps in reducing the server's response time by caching pages that are more than likely to be requested by a visitor. Also, it helps in recommending products, links, online services and so on. We proposed an efficient kNN based approach to do online pattern prediction. Our approach is based on the idea of looking at sessions as documents, pages' references as terms and an online session as a query document. Therefore, making a prediction about the pattern of an online session can be translated to looking for the most relevant documents to a query. We also used the well-known TF-IDF technique to get a meaningful weight for each page in a session. In addition, we thought of the use of an inverted index that reduces the search scope for relevant sessions. We conducted experiments and measured the time spent to get the top k relevant sessions using an inverted index and without the usage of inverted index. The gain in running time is very huge when we use an inverted index. Also the sessions returned as answers when using an inverted index contain all pages existing in the online session and in the same order.

Even though, we made an enormous effort and spent a lot of time in order to come up with a refined WUM solution, there is still room for improvement in all WUM stages. First, with regards to sessions' identification we think that it is a good idea to investigate the effect of adding our suggested constraint to other kinds of sessionizers such the one that uses the referrer. To be able to conduct this kind of investigation, we have to have a log that follows the extended format and we have to have the site structure available as well. Second, we also think that it would be better if we can mine for navigational patterns by using just one combined density based clustering algorithm instead of two steps process, OPTICS then DBSCAN. The idea behind this is that we want to fully automate the clustering process. Finally, to increase the prediction accuracy, it might be a very good idea to modify how we are calculating TF-IDF values to include somehow the time spent on a page and not only how many times the page has been requested in a session.

References

[1] M. Agosti, G.M.D. Nunzio, Web log mining: a study of user sessions, in: Proceedings of the 10th DELOS Thematic Workshop on Personalized, June 2007.

[2] M. Ankerst, M. Breunig, H. Kriegel, J. Sander, OPTICS: ordering points to identify the clustering structure, Sigmod Record 28 (2) (1999) 49–60.

[3] B. Berendt, B. Mobasher, M. Nakagawa, M. Spiliopoulou, The impact of site structure and user environment on session reconstruction in web usage analysis, Webkdd 2002 – Mining Web Data For Discovering Usage Patterns and Profiles (2003) 159–179.

[4] B. Berendt, B. Mobasher, M. Spiliopoulou, J. Wiltshire, Measuring the accuracy of sessionizers for web usage analysis, in: Paper presented at the Workshop on Web Mining, First SIAM Internat. Conf. on Data Mining, Chicago, IL, 2001.

[5] L. Catledge, J. Pitkow, Characterizing browsing strategies in the world-wide-web, Computer Networks and Isdn Systems (1995) 1065–1073.

[6] M. Chen, J. Park, P. Yu, Data mining for path traversal patterns in a web environment, in: Proceedings of the 16th International Conference on Distributed Computing Systems, 1996, pp. 385–392.

[7] R. Cooley, B. Mobasher, J. Srivastava, Data preparation for mining world wide web browsing patterns, Knowledge and Information Systems. 1 (1) (1999).

[8] M. Eirinaki, M. Vazirgiannis, Web mining for web personalization, ACM Trans Inter Tech. (3) (2003) 1–27.

[9] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial database with noise, in: Proceedings of the 2nd Int. Conference on Knowledge Discovery in Databases and Data Mining, Portland, Oregon, August 1996.

[10] F. Facca, P. Lanzi, Recent developments in Web Usage Mining research, Data Warehousing and Knowledge Discovery Proceedings (2003) 140–150.

[11] S. Gunduz, M. Ozsu, A web page prediction model based on click-stream tree representation of user behavior, in: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[12] D. He, A. Göker, Detecting session boundaries from web user logs, in: Proceedings of the 22nd Annual Colloquium of IR Research, Cambridge UK, April 2000.

[13] J.M. Hernansaez, J.A. Botıa, A.F. Skarmeta, A java technology based distributed software architecture for web usage mining, in: A. Scime (Ed.), Web Mining: Applications and Techniques, Idea Group Publishing, 2005, pp. 355–372.

[14] Y.-F. Huang, J.-M. Hsu, Mining web logs to improve hit ratios of prefetching and caching, Knowledge-Based Systems 21 (1) (2008) 62–69.

[15] Y.-M. Huang, Y.-H. Kuo, J.-N. Chen, Y.-L. Jeng, NP-miner: a real-time recommendation algorithm by using web usage mining, Knowledge-Based Systems 19 (4) (2006) 272–286.

[16] Z. Huiying, L. Wei, An intelligent algorithm of data pre-processing in web usage mining, in: Proceedings of the World Congress on Intelligent Control and Automation, 2004.

[17] J. Lei, A. Ghorbani, The reconstruction of the interleaved sessions from a server log, Advances in Artificial Intelligence, 2004, pp. 133–145.

[18] Y. Li, N. Zhong, Web mining model and its applications for information gathering, Knowledge-Based Systems 17 (5–6) (2004) 207–217.

[19] A.G. Lourenço, O.O. Belo, Catching web crawlers in the act, in: Proceedings of the 6th International Conference on Web Engineering, Palo Alto, California, USA, 2006.

[20] B. Mobasher, Data mining for web personalization, in: P. Brusilovsky, A. Kobsa, W. Neidl (Eds.), The Adaptive Web, vol. 4321, Springer Berlin/Heidelberg, New York, 2007, pp. 90–135.

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

[22] A. Montgomery, C. Faloutsos, Identifying web browsing trends and patterns, Computer (2001) 94–95.

[23] NASA (95), HTTP requests to the NASA WWW server in Florida, <http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html> (retrieved 25.05.10).

[24] Z. Pabarskaite, A. Raudys, A process of knowledge discovery from web log data: systematization and critical review, Journal of Intelligent Information Systems (2007) 79–104.

[25] M. Perkowitz, O. Etzioni, Adaptive websites: automatically synthesizing web pages, in: Fifteenth National Conference on Artificial Intelligence (AAAI-98) and Tenth Conference on Innovative Applications of Artifical Intelligence (IAAI-98) – Proceedings, 1998, pp. 727–732.

[26] robotstxt.org, Robots Pages. <http://www.robotstxt.org/> (retrieved 24.05.10).

[27] J. Srivastava, R. Cooley, M. Deshpande, P.-N. Tan, Web usage mining: discovery and applications of usage patterns from web data, SIGKDD Explorations 1 (2) (2000) 12–23.

[28] D. Tanasa, Web Usage Mining: Contributions to Intersites Logs Preprocessing and Sequential Pattern Extraction with Low Support. Universite de Nice Sophia Antipolis – UFR Sciences, 2005.

[29] Y. Xiao, M. Dunham, Efficient mining of traversal patterns, Data and Knowledge Engineering (2001) 191–214.

[30] H. Zhang, A. Ghorbani, The reconstruction of user sessions from a server log using improved time-oriented heuristics, in: Second Annual Conference on Communication Networks and Services Research, Proceedings, 2004, pp. 315–322.

[31] W. Zhuang, Y. Zhang, J.F. Grassle, Identifying erroneous data using outlier detection techniques, in: E. Vanden Berghe, et al. (Eds.), Proceedings Ocean Biodiversity Informatics: International Conference on Marine Biodiversity Data Management, Hamburg, Germany 29 November to 1 December, 2004. VLIZ Special Publication, vol. 37, 2007, pp. 187–192.