webSPADE: A Parallel Sequence Mining Algorithm to Analyze Web Log Data

Ayhan Demiriz
E-Business Department, Verizon Inc.,
919 Hidden Ridge, Irving, TX 75038
E-mail:ayhan.demiriz@verizon.com

Abstract

Enterprise-class web sites receive a large amount of traffic, from both registered and anonymous users. This traffic consists of many click streams, which are defined as collections of hits (requests) from a specific user during a specific session. Data warehouses are built to store and help analyze these click streams to provide companies with valuable insights into the behavior their customers. This article proposes a parallel sequence mining algorithm, webSPADE, to analyze the click streams found in site web logs. In this process, raw web logs are first cleaned and inserted into a data warehouse. The click streams are then mined by webSPADE, the proposed algorithm that uses one full scan and several partial scans of the data. An innovative web-based front-end is used for visualizing and querying the sequence mining results. Based on relational database technology, this analysis technique enables the analysis of very large amounts of data in a short amount of time. Thus, an analysis of monthly data can be performed quickly and easily, as shown by the algorithm run-times reported in this paper. The webSPADE algorithm is currently in use at Verizon to analyze the daily traffic of the Verizon.com web site.

Keywords: Web Mining, Sequence Mining, Web Log, e-commerce, Parallel Mining, Association Mining

1 Introduction

Many traditional companies see the enormous opportunities in using e-commerce sites, especially e-stores, as ways to reach customers outside the traditional business channels. Simply running an e-commerce site will not improve customer satisfaction and retention, however. While a user-friendly e-commerce site may attract new customers and strengthen relationships with existing customers, a poorly designed or implemented site may drive away potential customers and damage relationships with current customers. Because of this, businesses should approach e-commerce sites as new markets, where companies must open and maintain the lines of communication with new, and existing, customers.

Assuming that user sessions in web logs are constructed by appropriate technology, we must first clean the web logs to remove redundant information. Parsing the cleaned web logs and inserting the data into a repository (data warehouse or relational database) is the next step in the analysis process. Data stored in a repository can easily be used for frequency analysis with proven database technologies to create excellent summary reports. However, when it comes to analyzing the sequences, even with well defined process flows, the number of nested queries required to follow the processes step by step within a relational database framework make the analysis prohibitively expensive. This expense, combined with the fact that simple database queries are unlikely to discover hidden sequences and relationships within the data, make it important to use an effective sequence mining algorithm to analyze the data contained within the web logs.

We propose a parallel sequence mining algorithm based on [20, 21]. There are several major differences in this paper compared to earlier work [20, 21]. Differences and improvements can be summarized as follows:

- webSPADE is a Wintel-based parallel implementation.
- It only requires one full scan of the data compared to three full scans in previous algorithms.
- Temporal joins are used in webSPADE contrast to the non-temporal joins in the original algorithm.
- The design of webSPADE achieves data and task parallelism simultaneously.
- The current system has been in production since Mid-October of 2001 without any major problem.
- Click stream data is analyzed daily and sequences are stored in a relational database.
- A user-friendly front-end is used to visualize and mine stored sequences for a user-determined time range and support level.
- By using front-end, it is possible to analyze click stream data from a very large time period (e.g. whole year) in a short time.

The paper is organized as follows: We briefly discuss web mining and sequence mining in Section 2. We propose an integrated solution for click stream analysis in Section 3. The solution has four components: The web log data parser, the data warehouse, the sequence mining algorithm webSPADE, and the front-end interfaced used for displaying and analyzing the mining results. The first two components, the log parser and data warehouse, are mentioned very briefly in this paper, but a more detailed discussion of these two components is outside this paper's scope. An analysis of web log data from Verizon.com during the month of January of 2002 is given in Section 4 for illustration purposes. Computational times are also reported in Section 4. The paper wraps up with a conclusion and information about our future work.

2 Related Work

When our customers visit our web sites and click on our links, their actions speak to us. How clearly we hear them depends on whether or not we perform accurate click stream analysis. The simplest analysis is to get first level statistics based on the web log data. For example, the most visited page, the longest path, average transaction time, and the proportion of registered customers to the rest are some of the most common reports and are easily generated by off-the-shelf reporting tools such as Analog [19] for web log data. While these types of simplistic report will help business managers understand the "business at a glance," additional value is found in thorough analysis of the web log data using Web Mining techniques.

Web Mining can be defined in short as the application of data mining algorithms to web related data. There are three components of web mining:

- Content Mining.
- Structure Mining.
- Web Usage Mining.

Content mining is the analysis of the information (data) such as text and graphics that are presented in the web pages. One example would be the classification of homepages. One can identify course homepages on the web by distinguishing course homepages from the rest. Clustering is also used to segment pages. Structure mining is used to understand the topology of a web site specifically the inter-page relations hidden in tree-like structures of web sites. In this paper, we specifically explore web usage mining. Web usage mining consists of three phases: Preprocessing, pattern discovery and pattern analysis [18].

A click stream is defined as the ordered sequence of pages visited (or requested) by a customer/user (both registered and anonymous) in a session. The first challenge in web log data preprocessing is to define a session and a customer. This is the main step in the preprocessing phase of web usage mining. This step is dependent on the way the web log data is collected: server level collection, client level collection, or proxy level collection. The choice of collection is dependent on the technology. Technical difficulties might prevent a valid analysis on the web log data depending on the choice of collection. Throughout this paper we assume that by using an appropriate technology, such as cookies, we can distinguish the sessions and the customers properly. In the next sub-section we will describe some of the pattern discovery techniques.

2.1 Common Methods in Pattern Discovery

This section describes some of the analytical methods used in web usage mining. As we mentioned earlier, basic level statistical analysis is the most common way of extracting knowledge from web log data. Having descriptive statistics such as the most frequently requested pages, average access time, and the most common error codes might help to improve web traffic, system performance, and security. Statistical analysis is used to monitor the site and help IT professionals evaluate its efficiency and functioning. Its merits for business usage are very limited.

Clustering is also used for extracting knowledge from the web log data. It is very useful, especially when the customer data is merged with the demographic data to generate the customer segmentations. In addition to customer segmentation, some web personalization methods also use the clustering approach. Classification can also be used to categorize customers based on the properties extracted from the web log data and related demographic information. To classify such data, the task must be defined very carefully and the required categorization should be monitored carefully. The difficulty, in classification methods such as decision trees, is to prepare proper training set(s) prior to extracting rules.

Another useful way of pattern discovery is dependency modelling. The aim here is to model user behavior during the various stages of navigation. This approach uses the probabilistic learning techniques, such as Hidden Markov Models and Bayesian Belief Networks, during the learning process. An example for such an approach is the visualization of user navigation patterns by using a Markov model based on clustering [4]. Since a click stream is an ordered sequence, sequence mining plays an important role in knowledge extraction from the web log data. In the following section, we will describe the sequence mining and its predecessor, association mining, used to discover frequent sequences for both temporal and non-temporal data.

2.2 Sequence Mining

Sequence mining is an extension of association mining that only finds non-temporal patterns. Association mining is used to find the related pages that are accessed together in the click stream. Association mining was originally used to perform Market Basket Analysis to determine which items were purchased together. The goal of Market Basket Analysis is to find the related items that are purchased together most frequently.

In this context association rules refer to rules used to identify pages which are accessed together above a threshold level (support). Many algorithms have been successfully developed by various researchers for sequence analysis. One of the earliest is Apriori [2]. The Apriori algorithm can reveal the relationships between pages that are not directly connected (that is pages are not linked through hyperlinks). This is desirable, because it generates hidden rules along with the known relations (frequent hyperlinks). Businesses can benefit greatly from the knowledge extracted from these rules. As we mentioned earlier, simple statistical analysis has limited value for generating business rules.

The Apriori algorithm makes several passes over the data to find frequent items (pages in our context). In the kth pass, it finds all the item sets having k items. Each pass consists of two phases: candidate generation and support counting. The item sets from $(k-1)$th pass is used to generate the candidate list in kth pass. Then the data is scanned in the support counting phase to find the support of the items that are in the candidate list. The items in the candidate list that satisfy the minimum support are kept for the next pass. The algorithm continues until no item set supports the minimum threshold.

Although many variations of the Apriori algorithm have been developed, they still require multiple passes over the data. This is of questionable utility, especially in the case of very large datasets. To speed up the process SQL variations of Apriori algorithm have been also developed such as in [14, 15]. Special SQL operators were proposed for such implementations. On the other hand a SQL based language also was used in [16].

One of the drawbacks of the Apriori-like algorithms is generation of redundant rules. Depending on the support level, association mining may generate an excessive number of rules. On the other hand, since Apriori algorithm does not take into consideration the order which items are selected (i.e. pages are viewed), some of the rules found are invalid or misleading for click stream analysis. The last phase of the web mining, pattern analysis, is designed to filter out these unnecessary rules.

The problem of mining sequential patterns was introduced in [3]. They also presented three algorithms for solving this problem. The *AprioriAll* algorithm was shown to perform better than the other two approaches. In subsequent work [17], the same authors proposed the GSP algorithm, which is 20 times faster than *AprioriAll*. They also introduced maximum gap, minimum gap, and sliding window constraints on the discovered sequences.

Independently, [11] proposed mining for *frequent episodes*, which are essentially frequent sequences in a single long input-sequence (typically, with single items events, though they can handle set events). However, our formulation is geared towards finding frequent sequences across many different input-sequences. They further extended their framework in [10] to discover *generalized episodes*, which allows one to express arbitrary unary conditions on individual sequence events, or binary conditions on event pairs. The MEDD and MSDD algorithms [12] discover patterns in multiple event sequences; they explore the rule space directly instead of the sequence space. PrefixSpan is the most recent algorithm for sequence mining, it uses repeated database projections to mine frequent sequences [13].

Sequence discovery can essentially be thought of as association discovery [1] over a temporal database. While association rules discover only intra-event patterns (called itemsets), we now also have to discover inter-event patterns (sequences). The set of all frequent sequences is a superset of the set of frequent itemsets. Due to this similarity sequence mining algorithms like *AprioriAll*, GSP, etc., utilize some of the ideas initially proposed for the discovery of association rules.

We chose to modify SPADE [20, 21], since it is a very efficient algorithm for general sequence mining. Unlike previous approaches, which make multiple database scans and use complex hash-tree structures that

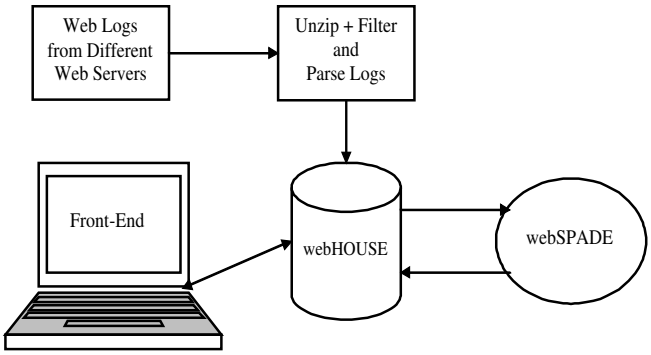


Figure 1: System Architecture

tend to have sub-optimal locality, SPADE partitions the original problem into smaller sub-problems using equivalence classes on frequent sequences. Not only can each equivalence class be solved independently, but it is also very likely that it can be processed in main-memory. Thus SPADE usually makes only three database scans - one for frequent 1-sequences, another for frequent 2-sequences, and one more for generating all other frequent sequences. If the supports of 2-sequences is available then only one scan is required. SPADE uses only simple temporal join operations, and is thus ideally suited for direct integration with a DBMS.

An efficient click stream analysis requires both a robust parser for the web log data and an efficient sequence mining algorithm to analyze the cleaned data. In the next section, we propose an integrated solution to analyze the web log data. First web log data is parsed efficiently and stored in a data warehouse. We then run the webSPADE sequence mining algorithm, a special application of SPADE [20, 21], to generate the rules. Since click streams are ordered by time, rules found by webSPADE have temporal relations e.g. $A \rightarrow B$ means that if A happens then later B happens i.e. if page A is visited then later page B is also visited.

3 Click Stream Analysis: An Integrated Solution

Click stream analysis plays an important role in the decision-making process of an e-commerce site. The knowledge obtained from such analysis can be deployed in restructuring the web site, personalizing the homepages and approaching the customer in a better way i.e. enhanced Customer Relationship Management (CRM).

We propose an integrated solution for performing click stream analysis in this section. A simplified system architecture is shown in Figure 1. The parser feeds the data into webHOUSE, a data warehouse. The sequence mining algorithm, webSPADE, reads daily data from webHOUSE and inserts the daily sequences into webHOUSE. The front-end is used to query the webHOUSE to analyze sequences. In this section, we first explain parsing the web log data and creating the database. Then we explain the modifications to the sequence mining algorithm SPADE introduced in [20]. Finally, we explain the usage of front-end in Section 3.4.

3.1 Parsing The Web Log Data

Logs recorded by web servers provided the data for analysis. The web-servers record each page request and related information such as specifications of the requestor's browser type, IP address, user name, time of request, action and page requested, protocol, etc. Different web-servers generate different types of web logs and some are capable of keeping more than one type of log, e.g. Microsoft Internet Information Server. The two most common log formats are NCSA and W3C-extended. Even though there are other types of logs, some of them do not provide sufficient information for analysis. Therefore, our web log processor only supports NCSA and W3C-extended format. The W3C-extended format allows the option of only capturing the desired fields and thus we expect the web-server using the W3C-extended format to be configured to capture at least the required information fields.

The web log processor is designed to be intelligent and does not require the user to know the type and specifications of the web log. It only requires the user to provide the location of the log file. It automatically determines the type of the log and whether it contains the required fields. The current parser runs in a parallel mode and is extremely efficient. On average it takes less than an hour and a half to parse and insert the cleaned data into the data warehouse for daily hits at Verizon.com. This cleaned data consists of both anonymous and registered requests and contains more than two million hit records each day. Note that we do not include all the hits to Verizon.com and it should be noted that certain pages on Verizon.com use cookies to keep track of the user sessions and the application process flows. This means that session information and some other operational data is also kept in the data warehouse.

A sophisticated data warehouse is used for storing daily request data and related information. Due to the proprietary nature of the application, we will only mention a very early version of data warehouse in the next sub-section. In practice, we only need three data fields -session id, time stamp and page id- for click stream analysis.

3.2 Web Log Data Warehouse

Since there is a huge amount of data to be stored and analyzed, a sophisticated database system is needed, which not only prevents the data redundancy but also provides readily available analysis. Considering the constraints and available tools, Microsoft SQL Server is utilized as database engine. A star schema such as in Figure 2 is used in this data warehouse. It should be noted that the star schema given in Figure 2 is a very early version of the current data warehouse. It is only mentioned here to illustrate the system. In early versions of the data warehouse, we used IP address as session id for simplicity. OLAP reporting tools are also available with Microsoft SQL Server.

Utilizing the star schema depicted in Figure 2 reduces the storage requirements approximately to 5% of the raw data size. For example, if a web-page contains a mixture of 9 image and script files, the web server will record it as 10 hits: 1 for the page itself and 9 for the images and script files. If only the page hit is kept and the rest are discarded, the log would be reduced to 1/10th of its size. Not only do we reduce the storage size, but also bring the ability to run both static and ad hoc queries to the web log data by creating the database. The fact table is the main table in the star schema. Several OLAP reports are also created or updated when this table is updated. In the next sub-section we explain the webSPADE algorithm used in this paper to perform the sequence mining.

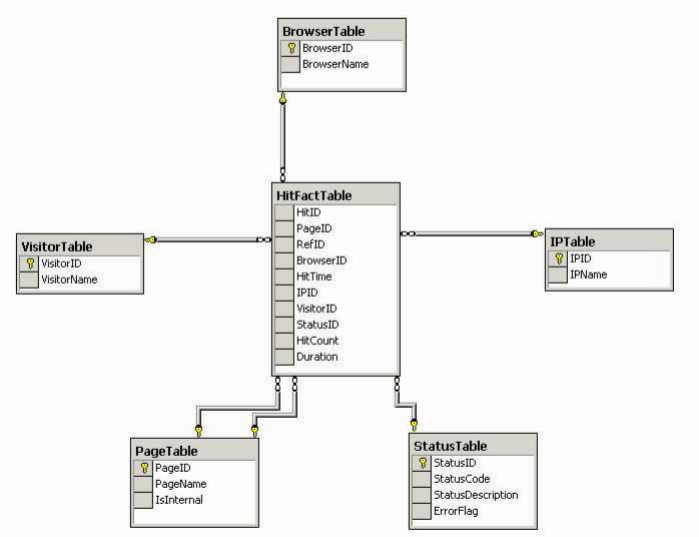


Figure 2: Star Schema of the Web Log Data Warehouse

3.3 webSPADE: A Parallel Sequence Mining Algorithm

The nature of the sequence mining problem makes massive computation unavoidable. This situation is the ideal application for parallel programming. In this section we present a parallel sequence mining algorithm that differs from previous work [21, 6, 7] in many ways. Parallel sequence mining algorithms are generally derived from sequential ones by introducing load balancing schemes for multiple processors and distributed memory. Since the data warehouse is built on MS SQL Server in our implementation, webSPADE has been developed in the Wintel environment, simplifying the parallelization of the serial programs.

The core point of our implementation is to modify the SPADE algorithm for the purpose of performing click stream analysis. The details of the original SPADE algorithm can be found in [20]. SPADE is a generic algorithm used for performing sequence mining for both time-dependent and time-independent data. The main advantage of using this algorithm is the use of join operations instead of scanning all the data to count certain item sets. The original SPADE algorithm proposed in [20] requires three full scans of the data. The advantage of this algorithm compared to the Apriori algorithm is the ability to use the time information. Our implementation is even better since it requires only one full data scan which is performed as the data is retrieved from the database. Note that both SPADE and webSPADE may require large number of scans on intermediate partial data.

The reason behind using join operation in SPADE algorithm is that, as mentioned in Corollary 1 of [20], any sequence X can be derived by joining of its first two lexicographical subsequences. For example in order to find support for the rule $A \rightarrow B \rightarrow C$, it is sufficient to join the sets of the rules $A \rightarrow B$ and $A \rightarrow C$. It is obvious that the rule $A \rightarrow C \rightarrow B$ is also obtained by joining the same two subsequences. This is the most crucial step of the SPADE algorithm that gives it superiority over the Apriori algorithm. In this case, A is called as an "equivalence class".

Before going into details of webSPADE, we want to briefly discuss the differences between webSPADE and SPADE introduced in [20]. Since click streams are strictly ordered by time and no two items (page views

Table 1: Sample Set for Item A

SID	Time
1	10
1	15
2	12

for the same session) occur at the same time, we only use temporal joins in contrast to the existence of the non-temporal joins in the original algorithm. Another difference is that support counting is not performed at a session level i.e. if $A \rightarrow B$ occurs twice in a given session, support of the sequence (rule) increments by two. In the original algorithm, it would count only one instead. We consider repeating subsequences in a given session as significant for click stream analysis. Therefore, our algorithm is designed to handle repetition within a session, though a counter argument can be made regarding the misleading nature of the repeating sequences. For example: Assume that there are several independent hits on page A and B and all page B hits occur after page A hits. In this case, the support of the sequence $A \rightarrow B$ will be counted as many as the lowest number of hits on pages A and B. However, in reality, user goes from page A to B only once. This situation might cause the sequence mining algorithm to find unnecessary or illogical sequences. Our analysis shows that webSPADE does not encounter this problem and the resulting sequences reflect the true frequencies after confirming with SQL queries. Sequences make sense from the perspective of both business and real traffic flow. Nevertheless, taking the repeating sequences into consideration is important to understand the true traffic load.

As we mentioned earlier, due to the design of our implementation, the algorithm requires only one full scan of the database. Since we do not have vertical to horizontal database recovery in this implementation, we skip the second and the third scan in the original implementation, creating a significant performance improvement. The main data structure is the map of multisets in our implementation. A sample set is depicted in Table 1. SID stands for session id. This gives us the ability to join tables efficiently and reduce the number of full scans.

In terms of parallelization of the serial programs, the memory type of the computer system is an important factor to consider when designing the algorithm. Algorithms proposed in [21, 6, 7] are designed to run on distributed shared-memory due to the selection of parallel computer systems (SGI Origin 2000 and IBM SP2). Since webSPADE runs on a single machine with multiple CPUs, it is designed simply to utilize a single memory. Indeed, this particular server has 8 CPUs at 700 MHz clock speed with 8GB of total memory. The system used in [21] is composed of 12 processors SGI Origin 2000 with 195 MHz R10000 MIPS processors and 2GB main memory. Algorithms discussed in [6, 7] are run on an IBM SP cluster that consists of 79 four-processor and 3 two-processor machines with a total of 391 GB of memory. These machines have 222 MHz Power3 processors. When we compare the three different systems, a Wintel based system is the simplest and certainly the cheapest.

Sequence mining can be considered as an irregular tree search algorithm [21], with each node corresponding to an equivalence class. According to argument in [21], parallelization of the sequence mining can be achieved either by data or task parallelism. In data parallelism, processors work on distinct partitions of the database but process the global tree structure concurrently. Task parallelism, on the other hand, requires each processor to have a separate copy of the database and run on different branches of the global tree. A

static and two dynamic load balancing schemes are examined in [21] as part of task parallelism. Experiments in [21] show that task parallelism is more favorable than data parallelism, with the best task parallelism approach using recursive dynamic load balancing [21].

A parallel version of the tree projection algorithm is proposed in [6]. Each node in the projection tree corresponds to k-itemset. This is indeed similar to the equivalence class representation of SPADE algorithm. The projection tree grows in a breadth-first manner. Data and task parallelism are compared again in [6]. Bipartite graph partitioning and bin packing are used as task parallelism approaches. These are not considered as dynamic load balancing schemes. Similar to the results in [21], task parallelism results are more favorable in terms of work load and computation time. An extension to [6] is introduced in [7] by implementing a dynamic load balancing scheme. The dynamic load balancing scheme in [7] performs similar to or better than the static load balancing schemes used in [6].

With the design of webSPADE, data and task parallelism are achieved simultaneously. Since webSPADE is developed in a Wintel-based environment, it has been coded as a multi-threaded program. A high level pseudo-code of the algorithm is given in Algorithm 3.1. Finding frequent sequences is a recursive depth-first search step to reduce the required memory for searching new sequences. In this step, all the item-pairs with the same equivalence class in the item-pair set are joined pairwise to form the next item-pair set. This recursive step is repeated for this new item-pair set until no two-item exists in the following item-pair set which has minimum support. The current rules are printed at the proper places within the recursive search. A thread is created for each recursive branch in the search space. Required data is passed to the child process and immediately deleted from the parent process. Thus while task parallelism is done during depth-first search, data parallelism is also achieved by passing the required data to the child processes. Load balancing is left to the operating system (Windows 2000), which reduces the coding efforts drastically. Hence development time was relatively short. There is no limitation on number of threads in our application as they are created as needed and monitored by the operating system.

Algorithm 3.1 (webSpade).

$$
\begin{aligned}
&\textit{Given min\_support and database } \mathcal{D} \\
&\mathcal{F}_1 \;=\; \{\textit{Frequent items or 1-sequences }\} \\
&\mathcal{F}_2 \;=\; \{\textit{Frequent 2-sequences (item-pairs) }\} \\
&\textit{Enumerate} - \textit{Freq} - \textit{Seq}(\mathcal{F}_2);
\end{aligned}
$$

Frequent items (pages) ($\mathcal{F}_1$) are found during the creation of the required data structures (data retrieval). This step is linear and does not require parallelization. Data is directly imported from the data warehouse for a specified time window by running a SQL query. Finding frequent 2-sequences ($\mathcal{F}_2$) is also parallelized in our implementation. This step requires exhaustive search by two nested for-loops. For each step in the main for-loop, a thread is created to find $\mathcal{F}_2$. `Enumerate-Freq-Seq` is the main function in webSPADE and each time this function is called, except in the main process, a new thread is also created. Potentially, hundreds of threads are created during the run time of webSPADE. The details of `Enumerate-Freq-Seq` function is given in Algorithm 3.2. σ represents the support count of the corresponding data structure. T is the set of item-pairs to be passed to the next branch. There are two join operations as mentioned above to find sequences - for example to find both $A \rightarrow B \rightarrow C$ and $A \rightarrow C \rightarrow B$ sequences. $\mathcal{L}$ represents the intermediate item-pair sets after join operations. Sequences that have frequency below the support level are removed from further analysis. $\mathcal{F}$ again corresponds to the frequent sequences to be printed later. This part

of webSPADE does not differ much from the original SPADE algorithm and it is suggested that interested readers to see [20] for further details.

Algorithm 3.2 (Enumerate-Freq-Seq).

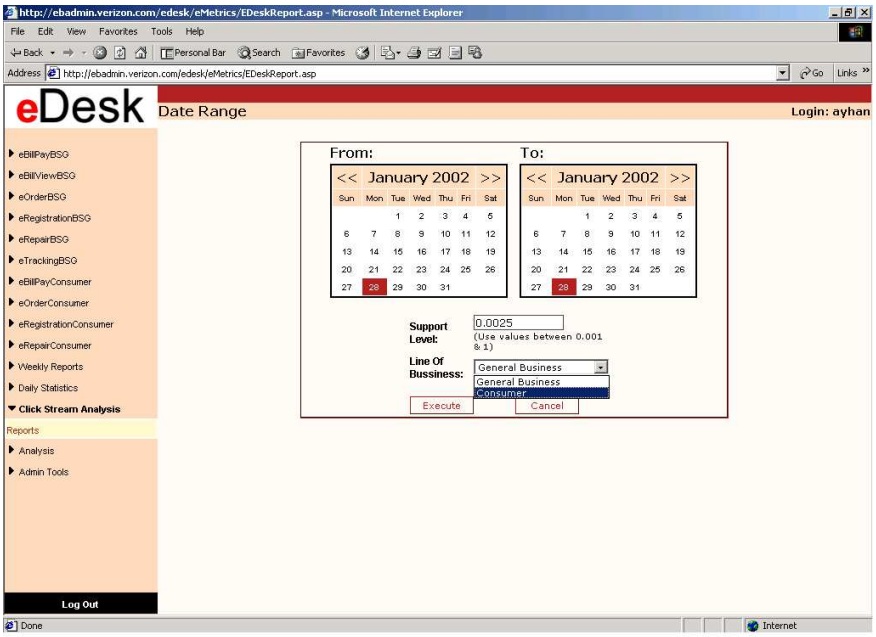
Given Set S
for all item-pairs $A_i \in S$ **do:**
{ **print the sequence**
$T_i = \emptyset$;
if antecedent (equivalence class) changed then
{**if** $size(T_i) > 1$ **then**
$Enumerate - Freq - Seq(T_i)$;
if $size(T_i) = 1$ **then**
print the sequence }
while A_j **and** A_i **have same antecedent do:**
{$R = A_i \bigvee A_j$;
$\mathcal{L}(R) = \mathcal{L}(A_i) \cap \mathcal{L}(A_j)$; ***Join operation***
if $\sigma(R) \geq min\_support$ **then**
{$T_i = T_i \bigcup \{R\}$;
$\mathcal{F}_{|R|} = \mathcal{F}_{|R|} \bigcup \{R\}$; }
$R = A_j \bigvee A_i$;
$\mathcal{L}(R) = \mathcal{L}(A_j) \cap \mathcal{L}(A_i)$; ***Join operation***
if $\sigma(R) \geq min\_support$ **then**
{$T_i = T_i \bigcup \{R\}$;
$\mathcal{F}_{|R|} = \mathcal{F}_{|R|} \bigcup \{R\}$; }
}
}

One important benefit of sequence mining is to automatically find all the possible sequences, because it is not humanly possible to think all possible sequence combinations. Therefore it is not possible to write SQL queries for every possible combination, yet it takes quite some time to run such queries for even very small set of items (pages). There are commercially available data warehouses, such as the one found in Microsoft Commerce Server 2000, which allow users to query the related databases but it is practically impossible to come up with every single combination of SQL queries to replace sequence mining.

The strength of our application comes from utilizing database technology to store and aggregate the sequence mining results and present them to the end user in a very short time. While running sequence mining on-the-fly for large datasets is generally regarded as practically impossible, in this paper we introduce an unprecedented analysis technique in the next sub-section to do just that. Our methodology can be considered as a new way to perform a mine and explore approach, i.e. mine the partial data now and then aggregate the results to mine larger data later.

3.4 An Innovative Way of Scaling up Sequence Mining Algorithms

We present a very innovative way of scaling up any sequence mining algorithm by utilizing the database technology. This allows users to analyze very large datasets spanning a large time window e.g. a quarter.



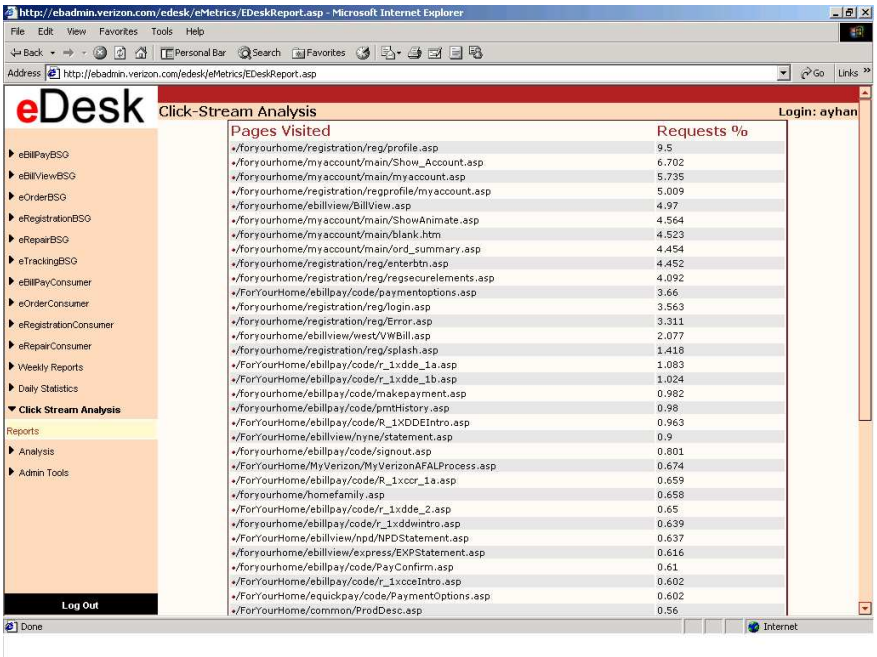


Figure 3: (A)- Parameter Selection Screen, (B)- Frequent Pages

A user friendly front-end visualizes the sequences in a very effective way and enables users to choose three different parameters: support levels, time window(start and end date of analysis) and line of business (LOB) to query the sequences. Similar visual pattern analysis techniques are introduced in [8, 9] in the context of the "mine and explore" paradigm using episode and association mining. Rules are found at a predetermined support and confidence level, then a user-interface is used to query the rule base to narrow down the selection of important rules.

Our approach is very simple. webSPADE runs daily with a predetermined support level to find sequences based on different lines of business; Support level is set to 0.1% for General Business and 0.25% for Consumer in our application. Daily sequences are then stored in a relational table. By design, webSPADE limits the length of sequences and, in this particular application, the maximum length of sequences is set to ten. Depending on the application domain, this parameter can be chosen accordingly. Since important processes in e-commerce sites, such as registration, should be kept as brief as possible, we believe that ten-level sequences are adequate for click stream analysis. Thus the relational table is composed of ten fields to determine the sequences, analysis date, line of business, frequency of sequence and total hits on analysis date. Since web log data is collected daily, we use a day as our atomic time unit. Different atomic time units may be used on other domains. For example, we may store financial sequence data hourly.

As in the case of association mining, sequence mining might results in excessive number of sequences. Finding potentially valuable sequences among numerous repetitive sequences might become impossible. Our front-end enables user to see and query the sequences in a user-friendly manner. Stored sequences can be used to analyze click streams between user specified dates. As seen from Figure 3(A), users can specify any support level and dates for a given line of business allowing a great degree of flexibility. **It should be noted that webSPADE is run daily and results are stored; there is no on-the-fly computation when the user selects parameters using the parameter selection screen.** In fact, stored results are aggregated and presented to the user. This point is the major difference from the previous work introduced in [8, 9]. Because mining algorithm in [8, 9] is run on all the available data and then resulting patterns are analyzed and searched by a user-interface. On the other hand, our methodology requires mining efforts on a portion of data (for one day). Resulting patterns are then aggregated and presented to the user for visual analysis and pattern search. Moreover, our approach scales up the underlying mining algorithm and visualizes the results simultaneously. This is a significant improvement in terms of computation efficiency.

After selecting the parameters, frequent pages are shown to the user as seen in Figure 3(B). Further analysis can be deployed by clicking on any page name in frequent page list (see Figure 3(B)). Once clicked, a pop-up window appears as seen in Figure 4(A). There are five options to choose on this screen. Users can select to see sequences that contain a selected page, start with a selected page, or end with a selected page. Users can list all the sequences as well. There is another option to list all the sequences starting with the selected page and ending with another page. Each option corresponds to a different stored procedure in the database. Having results stored in a relational table gives tremendous flexibility to report and query the results. Once an option is chosen, the resulting sequences are listed in the browser window as seen in Figure 4(B).

webSPADE and the web-based front-end, depicted in Figure 3 and 4, can be used for other time dependent sequential data. To illustrate the practical usage of sequence mining and to assess the performance of webSPADE we analyze all the data from January 2002 in next section.

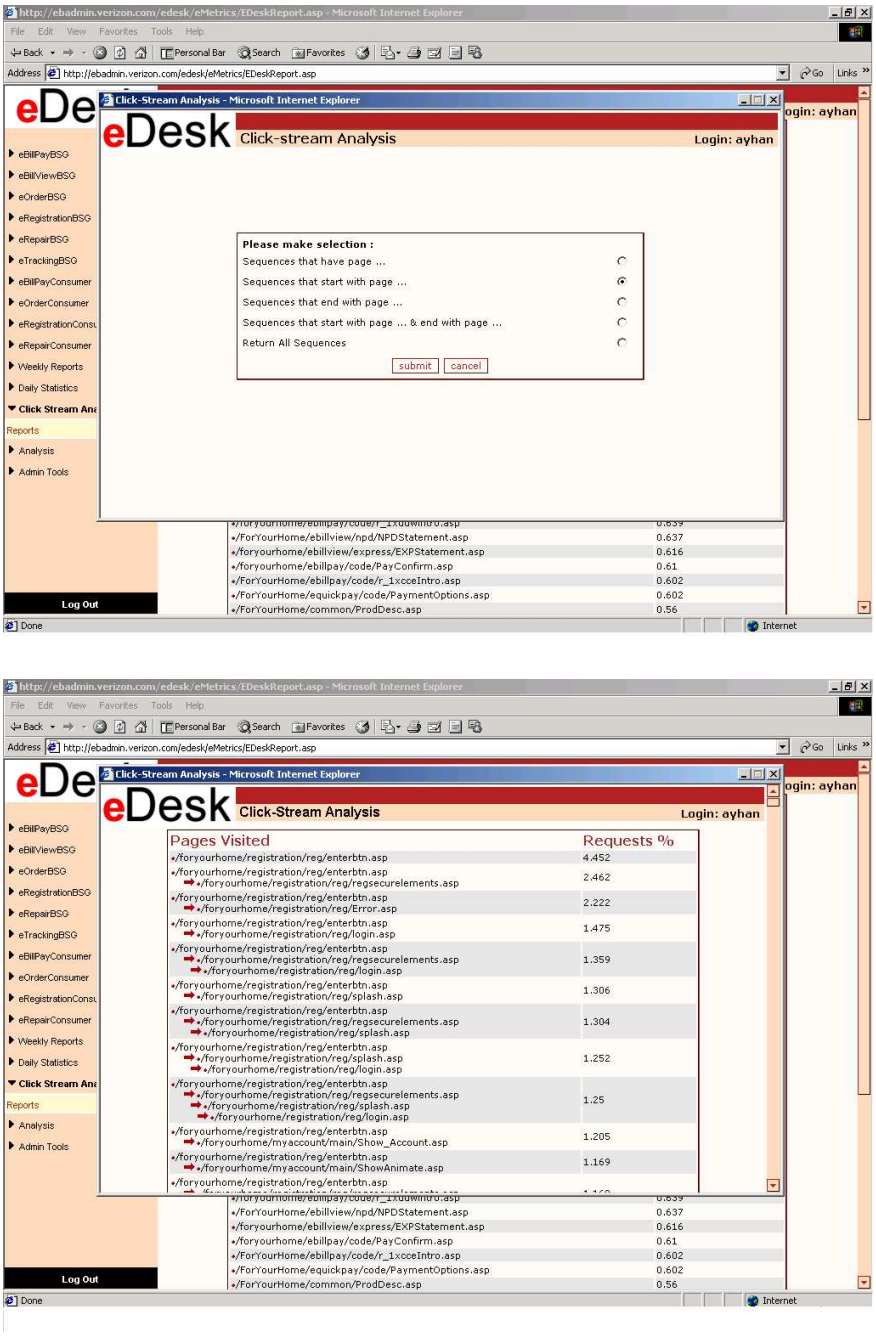


Figure 4: (A)- Analysis Selection, (B)- Sequences

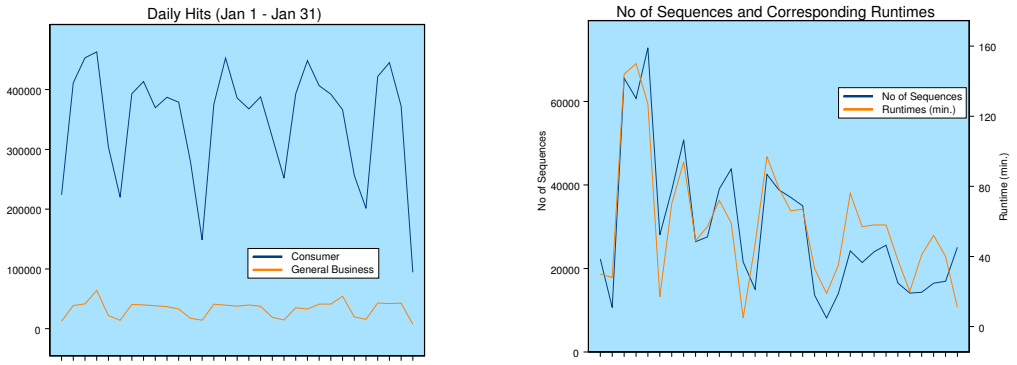


Figure 5: (A) Daily Hits, (B) Number of Sequences and Runtimes

4 Performance and Business Analysis: An Illustrative Example

webSPADE has been used for analyzing web log data since mid-October of 2001. We can now analyze virtually all the frequent sequences since then by using the front-end reporting tool mentioned in the previous section. So far there are approximately 480M cleaned hits in the datawarehouse. Based on this data, webSPADE has found approximately 6M frequent sequences. To illustrate the usage of the webSPADE algorithm, we consider the data from January 2002. As we mentioned above certain pages on Verizon.com are tagged to collect session information and our analysis covers only these pages. As we also mentioned above, the pages of two different lines of businesses are analyzed separately. Total daily hits during January are depicted in Figure 5(A). When we consider the whole month, there are approximately 12 million hits (requests) from both lines of business. This is a relatively large dataset.

The number of sequences and daily runtime of webSPADE in minutes is depicted in Figure 5(B). Reported times are the sum of runtimes for both lines of business. As expected, the number of sequences depends on number of daily hits (requests). Although webSPADE is not run on a dedicated server, the example runtimes of webSPADE can be considered close to reality. Note that runtimes also include both database access time and insertion time of sequences into the relational table.

The operational value of sequence mining is undeniable. For example, it is easy to monitor the traffic to understand whether a web page is functioning well or not. More specifically, certain pages might experience heavy traffic but the following pages may experience very low traffic. Sequence mining can easily catch such patterns. Some design problems might cause such patterns e.g. a misplaced next button at the bottom of the page; many people may not be able to see the next button because of smaller monitors. Such changes are requested in the light of sequences mining findings.

Sequence mining can also be used to find out who comes to a certain page and where they go after that page. An analysis of a General Business page (Bill View) is depicted in Figure 6. Such analysis can be done easily with SQL queries at a micro level but it is not possible to cover all the relations at a macro level (whole web site). Note that bar charts in Figure 6 do not correspond to probability distributions.

The analysis such as given in Figure 6 can result in very important insights. For example, although bill view and bill pay processes are independent from each other in general business pages, a significant portion of customers first view their bill and then pay it in the same session. It is also found that the bill view

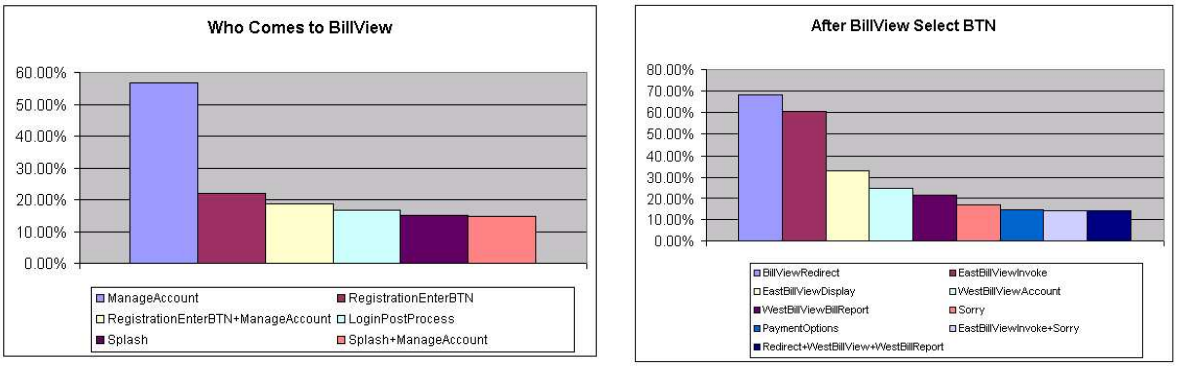


Figure 6: Analysis on Bill View Page

process sometimes fails to show bills due to the database access timeout. So, if we can increase the reliability of the bill view process, some of our customers will pay their bills in the same session. This is a very simple conclusion but it might take some time to come up with plain SQL analysis. Large scale sequence mining enables us to come to a conclusion on this matter more rapidly.

5 Future Work and Conclusion

We successfully applied a parallel sequence mining algorithm to perform click stream analysis. webSPADE requires only one full scan of the database, but several partial scans of the database. Data and task parallelism are easily achieved using multi-threaded programming. Load balancing is left to the operating system. Post-implementation tunings are still ongoing to speed up the process even more. An innovative analysis technique to scale up sequence mining algorithm is also used for value-added business analysis.

The current implementation can be adapted to other domains as well. One immediate usage of the click stream analysis is to predict future pages through which a user may navigate. This has a potential application in proxy server management and, obviously web personalization also depends on such prediction.

Market basket analysis (MBA) is the main application domain for the association mining in which rules are extracted based on purchased items in customer transactions. Rule and sequence mining algorithms such as Apriori and SPADE have superiority over other approaches such as clustering and dependency modelling. One extension to MBA is to couple its results with predictive modelling. Rules found by association mining can be used in creating new indicator variables to perform predictive modelling [5]. Such predictive models will help to categorize or segment customers based on their purchase (or navigation) patterns. In this manner, click stream analysis can be utilized to improve predictive abilities of product recommender systems by better segmenting the customers based on their navigational patterns.

Acknowledgments

I would like to thank to my colleagues in E-Business Department. Especially Nooman Karim for his help in multi-threaded programming, Mofassir Haque for his work on front-end design. Dr. Mohammed Akra should be commended for his great work on the web log parser and data warehouse design. Many thanks to DBAs Mohammed Lahlou and Wahib Omran. This project has come to the life under Walid Nouh's patient

and challenging management style. I am also very much thankful to Dr. Mohammed J. Zaki for his helpful comments and directions on improving this paper and work.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. Fayyad and et al, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, CA, 1996.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference,Santiago, Chile*, pages 487–499, 1994.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *11th Intl. Conf. on Data Engg.*, 1995.

[4] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model based clustering. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 280–284, Boston, MA, 2000.

[5] S. Gayle. The marriage of market analysis to predictive modeling. Technical report, SAS Institute Inc., Cary, NC, 2000.

[6] V. Guralnik, N. Garg, and G. Karypis. Parallel tree projection algorithm for sequence mining. In R. Sakellariou, J. Keane, J. Gurd, and L. Freeman, editors, *Proceedings of Seventh European Conference on Parallel Computing (Euro-Par)*, pages 310–320. Springer, 2001.

[7] V. Guralnik and G. Karypis. Dynamic load balancing algorithms for sequence mining. Technical Report 00-056, Department of Computer Science, University of Minnesota, 2001.

[8] M. Klemettinen, H. Mannila, and H. Toivonen. Interactive exploration of discovered knowledge: A methodology for interaction, and usability studies. Technical Report C-1996-3, Department of Computer Science, University of Helsinki, 1996.

[9] M. Klemettinen, H. Mannila, and H. Toivonen. Interactive exploration of interesting patterns in the telecommunication network alarm sequence analyzer tasa. *Information and Software Technology*, 41:557–567, 1999.

[10] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurences. In *2nd Intl. Conf. Knowledge Discovery and Data Mining*, 1996.

[11] H. Mannila, H. Toivonen, and I. Verkamo. Discovering frequent episodes in sequences. In *1st Intl. Conf. Knowledge Discovery and Data Mining*, 1995.

[12] T. Oates, M. D. Schmill, D. Jensen, and P. R. Cohen. A family of algorithms for finding temporal structure in data. In *6th Intl. Workshop on AI and Statistics*, Mar. 1997.

[13] J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of International Conference on Data Engineering (ICDE'01)*, 2001. Heidelberg, Germany.

[14] K. Rajamani, A. Cox, B. Iyer, and A. Chadha. Efficient mining for association rules with relational database systems. In *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'99)*, 1999.

[15] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 343–354. ACM, 1998.

[16] M. Spiliopoulou, C. Pohle, and L. C. Faulstich. Improving the effectiveness of a web site with web usage mining. In B. Masand and M. Spiliopoulou, editors, *Advances in Web Usage Mining and User Profiling: Proceedings of the WEBKDD'99 Workshop, LNAI 1836*, pages 139–159. Springer Verlag, July 2000.

[17] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *5th Intl. Conf. Extending Database Technology*, Mar. 1996.

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

[19] S. Turner. Analog. http://www.analog.cx, 2000.

[20] M. J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42(1/2):31–60, Jan/Feb 2001. Special issue on Unsupervised Learning (D. Fisher, editor.).

[21] M. J. Zaki. Parallel sequence mining on shared-memory machines. *Journal of Parallel and Distributed Computing,*, 61(3):401–426, March 2001. Special issue on High Performance Data Mining (V. Kumar, S. Ranka and V. Singh, editors.).