Smart Miner: A New Framework for Mining Large Scale Web Usage Data [*]

Murat Ali Bayir[†]
Department of Computer Science and Engineering
University at Buffalo, SUNY
14260, Buffalo, NY, USA
mbayir@cse.buffalo.edu

Ismail Hakki Toroslu
Department of Computer Engineering, METU NCC,
Kalkanli, Guzelyurt, TRNC,
Mersin, Turkey
toroslu@ceng.metu.edu.tr

Ahmet Cosar
Intelligent Data Analysis Group
Department of Computer Engineering, METU
06531, Ankara, Turkey
cosar@ceng.metu.edu.tr

Guven Fidan
AGMLAB Information Technologies
CyberPark, Bilkent
06800, Ankara, Turkey
guven.fidan@agmlab.com

ABSTRACT

In this paper, we propose a novel framework called Smart-Miner for web usage mining problem which uses link information for producing accurate user sessions and frequent navigation patterns. Unlike the simple session concepts in the time and navigation based approaches, where sessions are sequences of web pages requested from the server or viewed in the browser, Smart Miner sessions are set of paths traversed in the web graph that corresponds to users' navigations among web pages. We have modeled session construction as a new graph problem and utilized a new algorithm, Smart-SRA, to solve this problem efficiently. For the pattern discovery phase, we have developed an efficient version of the Apriori-All technique which uses the structure of web graph to increase the performance. From the experiments that we have performed on both real and simulated data, we have observed that Smart-Miner produces at least 30% more accurate web usage patterns than other approaches including previous session construction methods. We have also studied the effect of having the referrer information in the web server logs to show that different versions of Smart-SRA produce similar results. Our another contribution is that we have implemented distributed version of the Smart Miner framework by employing Map/Reduce Paradigm. We conclude that we can efficiently process terabytes of web server logs belonging to multiple web sites by our scalable framework.

[*]This work is supported by The Scientific and Technical Research Council of Turkey with industrial project grant TEYDEB 7070405.
[†]corresponding author.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Web Usage Mining, Graph Mining, Map/Reduce, Parallel Data Mining, Web User Modeling

1. INTRODUCTION

As in classical data mining, the aim of web mining is to discover and retrieve useful and interesting patterns from large web dataset [7]. In the last fifteen years, the WWW become the largest information sources with its amazing growing rate. All of this huge data available in the World Wide Web can be mined mainly in three different dimensions, which are web content mining, web structure mining and web usage mining. This paper is related to the web usage mining which can be defined as the application of data mining techniques to web log data in order to discover user access patterns [9, 26]. Web usage mining has various applications such as link prediction, site reorganization and web personalization. The success of all of these applications is significantly related to the outcomes of web usage mining process which includes session construction and frequent navigation pattern discovery phases.

Producing accurate user sessions and navigation patterns is not an easy task since http protocol is stateless and connectionless. Also, in reactive session construction [8, 9, 23], where it is not possible to generate web log information to identify individual users (like cookies), all users behind a proxy server will have the same IP number and therefore, these users will be seen as a single client on the server side. These problems can be handled by proactive strategies [21, 24] by using cookies, which add client specific information into each page request by using dynamic server page codes. However, in order to employ proactive strategies, internal structure of web pages must be changed either by inserting JavaScript codes (called page tagging) or by using dynamic server pages codes to collect session data. Nowadays, many web site owners prefer page tagging method provided by a third party, which works by installing Java script codes into their web pages to collect access log data. This data is usually stored and processed on the servers of the third party.

However, for various reasons, such as security and changes in the internal structure of web site, some site owners may not want to use proactive approaches at all. Instead of that, these site owners prefer to process only their raw server logs, which contain access requests. We had received this kind of request from several customers of AGMLAB Information Technologies to process their huge web server logs for web usage mining and employ complex filters over navigation patterns for path analysis. Therefore, as part of this work we had focused on reactive approaches and proposed a new framework called Smart-Miner to meet this type of demands.

Smart-Miner is a novel framework including a new session construction technique and an efficient pattern discovery algorithm. Smart-Miner is also distributed framework which employs Map/Reduce Paradigm to process very large server logs. Our session construction algorithm, Smart-SRA is designed in such a way that it can even produce sessions by using the web topology and the server logs containing only the basic information, not even the referrer information (such as Common Log format[1]), from server log files. Furthermore, we have employed Smart-Miner on more comprehensive log formats like Combined Log Format which also contains referrer information and thus eliminates the need for the web topology.

Regardless of different web server logs, Smart-Miner employs two main steps to obtain frequent patterns from the log files. In the first phase, Smart-SRA algorithm is employed to generate user sessions containing set of paths traversed by users on the web graph. In the second phase, an efficient version of Apriori-All method is used to discover frequent navigation paths from user sessions. The reason for using Smart-SRA framework is to produce more correct sessions that captures more realistic user behavior as well as providing good coverage for navigated paths. In particular the followings are the contributions of this paper:

- We have implemented a full web usage mining framework, Smart Miner, as commercial software which is a sub module of our Web Analytics Service and works on a distributed architecture. Currently final polishing of Web Analytics Service is underway, and tests are performed on it before the final deployment. During the development of Smart-Miner framework, several novel ideas were also used, which are listed below.

- We propose a novel session construction method, called Smart-SRA, which models session construction process as a graph problem and produce maximal paths traversed on the web graph efficiently. Furthermore we have implemented two different version of Smart-SRA that can generate sessions by using only referrer information or by using only web topology.

- We introduce a strictly sequential version of Apriori-All algorithm which exploits the structure of web graph to improve the performance. In particular, our pattern discovery method generates candidate patterns only corresponding to the paths on the web graph.

- For simulating real user behavior, we have developed an agent simulator which can be used for comparing accuracies of different Web Usage Mining methods. Our agent simulator is based on the random surfer model of the page-rank algorithm. Our syntactic web topology also obeys power law distribution property of web graphs.

- For estimating the accuracy of alternative Web Usage Mining methods, we have proposed geometric accuracy model which considers both recall and precision. This model enables us to find the best method in terms of the number of sessions captured correctly and the number of sessions generated by session construction methods to capture the correct sessions

- Using both simulated and real data, we have showed that in terms of our accuracy measure the maximal frequent patterns discovered by Smart-Miner framework is at least 30% much better than the ones obtained by web usage mining techniques utilizing previous session construction methods.

- We have also implemented distributed version of Smart-Miner using Map-Reduce framework which enables us to process huge web server logs of multiple sites simultaneously. Our results show that we have a significant performance improvement in the distributed version of the Smart-Miner. In particular, we have showed that our run time performance increases linearly and our framework can be scaled-up easily to process any size of data.

This paper is organized as follows. The next section is dedicated to the previous session construction methods and our new method Smart-SRA. Section 3 discusses pattern discovery problem by introducing strictly sequential Apriori-All technique. After that, we introduce the agent simulator which was used to evaluate different session construction heuristics. The first subsection of the experimental results is dedicated to the description of the accuracy metric for comparing different web usage mining methods. In the next two subsections, the experimental results on simulated and real data are presented. The fourth subsection presents experimental results related to the impact of the referrer information on real case study. The fifth subsection discusses the map/reduce version of the Smart-Miner framework and scalability issues. Finally, we give our conclusions in section 6.

2. SESSION CONSTRUCTION

2.1 Previous Heuristics and Related Work

Before going into details of the previous work, it is better to inform the reader with the more general definition of web user session in the literature [8, 9, 12, 21, 23, 24].

Definition (Session): Session is a sequence of requests made by a single user with a unique IP address on a Web

[1]http://www.w3.org/daemon/user/config/logging.html

site during a specified period of time. Each request item in the session can be provided by either web server or cache systems from local client or proxies.

The most basic session definition comes with Time Oriented Heuristics[8, 12, 23] which are based on time limitations on total session time or page-stay time. They are divided into two categories with respect to the thresholds they use:

- In the first one, the duration of a session is limited with a predefined upper bound, which is usually accepted as 30 minutes according to [5]. In this type, a new page can be appended to the current session if the time difference with the first page doesn't violate total session duration time. Otherwise, a new session is assumed to start with the new page request.
- In the second time-oriented heuristic, the time spent on any page is limited with a threshold. This threshold value is accepted as 10 minutes according to [5]. If the timestamps of two consecutively accessed pages is greater than the threshold, the current session is terminated after the former page and a new session starts with the latter page.

Navigation-Oriented approach [8, 9] uses the web graph [11, 17] constructed by using hyperlinks among web pages. In this approach, it is not necessary to have a hyperlink between every two consecutive web page requests. Let $P =[P_1, P_2, \ldots, P_k, P_{k+1}, \ldots, P_n]$ be a session containing web pages with respect to their timestamp orders. In this session, for every page P_k, except the initial page P_1, there must be at least one page P_j in the session which is referring to P_k and has a smaller timestamp than P_k. If there are several pages referring to P_k with smaller timestamps, then, among these pages, the one with the largest timestamp is assumed to be the page visited before P_k is viewed. Therefore, during session construction, backward browser movements up to the closest page referring to P_k are appended to the session.

More formally, if $P =[P_1, P_2, \ldots, P_k, P_{k+1}, \ldots, P_n]$ are pages forming a session, then, $\forall k$ satisfying $1 < k \leq n$, $\exists j$ such that, $T(P_k) > T(P_j)$ and P_j refers to P_k. During the construction of a new session, if $P =[P_1, P_2, \ldots, P_k, P_{k+1}, \ldots, P_n]$ is the current session and P_{n+1} is the next page request, then, the page P_{n+1} can be appended to this session as follows:

- If the $Link(P_n, P_{n+1}) = true$ then the new session becomes: $P^* = [P_1, P_2, \ldots, P_k, P_{k+1}, \ldots, P_n]$.
- If the $Link(P_n, P_{n+1}) = false$ then
  - If $\exists P_k \in P$ satisfying following properties such that:
    - $Link(P_k, P_{n+1}) = true$, and
    - $(\forall P_x \in P$ and $T(P_x) > T(P_k)) \Rightarrow Link(P_x, P_{n+1}) = false$, that means If P_k is the nearest (with the largest timestamp smaller than the timestamp of P_{n+1}) page that refers to P_{n+1}, then the new session becomes $P^* =[P_1, P_2, \ldots, P_k, P_{k+1}, \ldots, P_n, P_{n-1}, P_{n-2}, \ldots, P_k, P_{n+1}]$
  - If there $\nexists P_k \in P$ satisfying the properties above, then P_{n+1} becomes the first page of a new session.

Although the session construction mentioned above are not new, most of the recent applications [3, 13, 14, 15, 16, 18, 22] use them during session generation. The problem with these three methods is that the application side suffer from low performance due to the incorrect set of user sessions generated. It is very important to produce high quality sessions in session construction since the quality of the user sessions effects the quality of the frequent patterns discovered in Web Usage Mining process. Our main goal here is to produce high quality sessions, so that it leads to more correct frequent patterns that will increase the performance in applications like recommendation systems, personalization, web user clustering and semantic web usage mining. In the next subsection, we explain our new session construction method in detail.

2.2 Smart-SRA

Smart-SRA stands for Smart Session construction Algorithm which is designed to overcome deficiencies of the time and the navigation oriented heuristics. As it is given in the more general session definition, the session concept is also a sequence of page requests in Smart-SRA. Moreover, the ultimate aim of web usage mining is to determine frequent user access paths. Thus, session construction from server logs is an intermediate step. In order to determine frequent user access paths, potential paths should be captured in the user sessions. Therefore, rather than constructing just user request sequences from server logs, Smart-SRA uses a novel approach to construct user session as a set of paths in the web graph where each path corresponds to users' navigations among web pages. That is, in Smart-SRA, server request log sequences are processed to reconstruct web user session not as a sequence of page requests, but, as a set of valid navigation paths. The definition of the Smart-SRA session is given below:

Definition (Smart-SRA Session): $S = \{S_1, S_2, \ldots, S_n\}$ is a session constructed by Smart-SRA having n paths, such that each $S_x = [P_1, \ldots, P_i, Pi + 1, \ldots, P_n] \in S$, satisfies the following conditions:

- Timestamp Ordering Rule:
  - $\forall i : 1 \leq i < n, T(P_i) < T(Pi + 1)$
  - $\forall i : 1 \leq i < n, T(P_{i+1}) - T(P_i) < \sigma$(page stay time)
  - $T(P_n) - T(P_1) < \delta$(session duration time)
- Topology Rule:
  - $\forall i : Link(P_i, P_{i+1}) = true$
- Maximality Rule:
  - $\forall S_x \in S \nexists S_y$ such that $S_y \in S$ and $S_x \subset S_y$

The timestamp ordering condition simply represents the standard user session definition, so that each path in the session constructed by Smart-SRA satisfies the requirements of a session. The topology condition is introduced to force each user navigation path to correspond to a path in the web site graph. Even though a user may perform actions like going to a previously visited page by using the back button of the browser or follow a link from a previously visited and still retained page, an ultimate goal we are interested in frequent user access paths, especially access sequence from web server during navigation on web graph. In addition, Smart-SRA only produces maximal paths in the session set generated. The motivation behind maximal paths are the coverage of possible paths they provided. We proved with maximality property that all possible paths that can be generated from user access sequence must be subset of at least one maximal path.

As it is clearly seen from the above rules, Smart-SRA uses three important properties which must be satisfied by all paths of a session generated by Smart-SRA. It is important that Smart-SRA eliminates the need for inserting the backward browser movements of navigation-oriented heuristic and it preserves the timestamp order of web pages. The two main phases of Smart-SRA are explained below:

- In the first phase, the access data stream of web users are partitioned into shorter page request sequences called candidate sessions, by using session duration time and page-stay time rules. The page request sequences obtained in this manner correspond to sessions obtained by using both of the time-oriented heuristics mentioned above.
- In the second phase, candidate sessions are divided into maximal sub-sessions such that for each consecutive page pair in the sequence there exists a link from previous one to latter one. At the same time, page stay time rule for consecutive pages is also satisfied. Since the total session duration time is checked in the first phase, there is no need to re-check it. However, page stay time criteria must be checked again since the consecutive page pairs cannot match in the sub-sessions with candidate session obtained after first phase. The second phase also adds topology rule and eliminates the need for inserting backward browser movements. This is achieved by repeating the following steps until all pages in a candidate session obtained after the first phase have been processed:

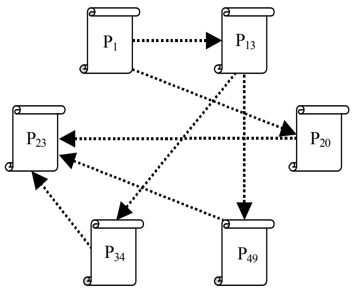


Figure 1: An example web site topology graph

The pseudocode for the second phase of Smart-SRA is given in Algorithm 1. In phase-2 of Smart-SRA, only maximal sequences are kept through the iterations. Also, only required sessions are propagated, and thus, there is no redundant session construction. Consider the Table 1, that shows a sample web page requests sequence of a web user obtained after the first phase of the algorithm, and Figure 1 representing the web topology graph. The application of the inner loop (while loop) of the second phase of the session construction algorithm is given in Table 2. For this example, when the algorithm is executed, it discovers the three maximal sessions showed in 4th iteration of NewSessionSet satisfying three conditions described in 2.2.

Algorithm 1 Second Phase of Smart-SRA

```
1: ForEach CandSession in CandidateSessionSet
2:   $NewSessionSet := \{\}$
3:   while $CandSession \neq []$
4:     $TSessionSet := \{\}$
5:     $TPageSet := \{\}$
6:     ForEach $Page_i$ in CandSession
7:       $StartPageFlag := TRUE$
8:       ForEach $Page_j$ in CandSession with $j > i$
9:         If $(Link[Page_i, Page_j] = true)$ and
           $(TimeDiff(Page_j, Page_i) \leq \sigma)$ Then
10:          $StartPageFlag := FALSE$
11:      End For
12:      If StartPageFlag = TRUE Then
13:        $TPageSet := TPageSet \cup \{Page_i\}$
14:    // Remove the selected pages from the current seq.
15:    $CandSession := CandSession - TPageSet$
16:    If $NewSessionSet = \{\}$ Then
17:      ForEach $Page_i$ in $TPageSet$
18:        $TSessionSet := TSessionSet \cup \{[Page_i]\}$
19:    Else
20:      ForEach $Page_i$ in $TPageSet$
21:        ForEach $Session_j$ in $NewSessionSet$
22:          If $(Link[Last(Session_j), Page_i] = true)$ and
           $(TimeDiff(Last(Session_j), Page_i) \leq \sigma)$ Then
23:            $TSession := Session_j$
24:            $TSession.mark := UNEXTENDED$
25:            $TSession := TSession \bullet Page_i$ // Append
26:            $TSessionSet := TSessionSet \cup \{TSession\}$
27:            $Session_j.mark := EXTENDED$
28:          End If
29:        End For
30:      End For
31:    End If
32:    ForEach $Session_j$ in $NewSessionSet$
33:      If $Session_j.mark \neq EXTENDED$ Then
34:        $TSessionSet := TSessionSet \cup \{Session_j\}$
35:      End If
36:    End For
37:    $NewSessionSet := TSessionSet$
38:  End While
39: End For
```

Table 1: Example Web Page Request Sequence

Page	P_1	P_{20}	P_{13}	P_{49}	P_{34}	P_{23}
TimeStamp	0	6	9	12	14	15

2.3 Extension to Referrer Case

We have also extended Smart-SRA to work with other log formats which has referrer information. In order to use this information, Smart-SRA keeps the id of referrer with the current page id in the candidate session. The extension to referrer case is very easy since we only need to change the topology check operation in the 9th and 22nd lines of the Algorithm 1. Instead of checking link existence $if(Link[Page_j,\ Page_i] = true)$, referrer check $if(Page_i.$

Table 2: Evaluation of Example Session by Smart-SRA

Iteration	1	2
CandidateSession	$[P_1, P_{20}, P_{13}, P_{49}, P_{34}, P_{23}]$	$[P_{20}, P_{13}, P_{49}, P_{34}, P_{23}]$
TempPageSet	$\{P_1\}$	$\{P_{20}, P_{13}\}$
NewSessionSet	$[P_1]$	$[P_1, P_{20}]$ $[P_1, P_{13}]$
Iteration	**3**	**4**
CandidateSession	$[P_{49}, P_{34}, P_{23}]$	$[P_{23}]$
TempPageSet	$\{P_{49}, P_{34}\}$	$\{P_{23}\}$
NewSessionSet	$[P_1, P_{13}, P_{34}]$ $[P_1, P_{13}, P_{49}]$ $[P_1, P_{20}]$	$[P_1, P_{13}, P_{34}, P_{23}]$ $[P_1, P_{13}, P_{49}, P_{23}]$ $[P_1, P_{20}, P_{23}]$

$referrer = Page_j$) is used in this new version of Smart-SRA. Obviously referrer case produce less number of sessions than original Smart-SRA since each page has exactly one referrer. However, for original Smart-SRA, we have considered restricted topology containing only pages within the candidate session instead of using the whole topology. This property enables original version of Smart-SRA to produce small number of sessions as the referrer based version also. In the experimental results section, we will show that there is no significant difference in terms of the number of generated sessions for both versions.

3. DISCOVERING PATTERNS

Sequential pattern discovery is the next phase of the Web Usage Mining. In this phase, frequent access patterns are determined from reconstructed sessions. There are several algorithms in the literature for the sequential pattern mining, such as GSP [25], SPADE [28] and PrefixSpan [20] etc. Although it is not the most recent or the most efficient one, we have used a modified version of the AprioriAll [1] technique. AprioriAll is very suitable for our problem since we can make it very efficient by pruning most of the candidate sequences generated at each iteration step of the algorithm. This pruning can be done because of the topological constraint requirement mentioned above, that is for every subsequent pair of pages in a sequence the former one must have a hyperlink to the latter one. We will call this new version of AprioriAll as Sequential Apriori Algorithm. In particular, we prune majority of the possible sequences in the search space during candidate sequence generation. Therefore, our focus is on the quality of discovered web usage patterns rather than performance of sequential pattern mining methods. Another different constraint in our domain is that a string matching constraint should be satisfied between two sequences in order to have support relation. To illustrate; the sequence $< 1, 2, 3 >$ does not support $< 1, 3 >$ although 3 comes after 1 in both of them. However, sequence $< 1, 3, 2 >$ supports $< 1, 3 >$. A session S supports a pattern P if and only if P is a subsequence of S not violating string matching constraint. We call all the sessions supporting a pattern as its support set.

Sequential AprioriAll Algorithm (Algorithm 2): In the beginning, each page with sufficient support forms a length-1 supported pattern. Then, in the main step, for each k value greater than 1 and up to the maximum reconstructed session length, supported patterns (patterns satisfying the support condition) with length k+1 are constructed by using the supported patterns with length k and length 1 as follows:

- If the last page of the length-k pattern has a link to the page of the length-1 pattern, then by appending that page length-k+1 candidate pattern is generated.
- If the support of the length-k+1 pattern is greater than the required support, it becomes a supported pattern. In addition, the new length-k+1 pattern becomes maximal, and the extended length-k pattern and the appended length-1 pattern become non-maximal.
- If the length-k pattern obtained from the new length-(k+1) pattern by dropping its first element was marked as maximal in the previous iteration, it also becomes non-maximal.
- At some k value, if no new supported pattern is constructed, the iteration halts.

Notice that in the sequential apriori algorithm, the patterns with length-k are joined with the patterns with length-1 by considering the topology rule. This step significantly eliminates many unnecessary candidate patterns before even calculating their supports, and thus increases the performance drastically. In addition, since the definition of the

Algorithm 2 Sequential Apriori

```
1: Input: Minimum support frequency: $\delta$, Reconstructed Sessions: S
2: Topology information as matrix: Link, The Set of all Web Pages: P
3: Output: Set of maximal frequent patterns: Max
4: Procedure sequentialApriori ($\delta$, S, Link, P)
5:   $L_1 := \{\}$ // Set of frequent length-1 patterns
6:   For i:=1 to $|P|$ do
7:       $L_1 := L_1 \cup [P_i]$ | If Support($[P_i]$,S) $> \delta$
8:   For $k = 1$ to $N - 1$ do
9:       If $L_k =$ then
10:        Halt
11:      Else
12:        $L_{k+1} := \{\}$
13:        For Each $I_i \in L_k$
14:          For Each $P_j \in P$
15:            If Link[Last($I_i$), $P_j$] = true then
16:              $T := I_i \bullet P_j$ // Append $P_j$ to $I_i$
17:              If $Support(T, S) > \delta$ then
18:                $T.maximal := TRUE$
19:                $I_i.maximal := FALSE$ // since extended
20:                $V := [T_2, T_3, \ldots, T_{|T|}]$ // drop first element
21:                if $V \in L_k$ then
22:                  $V.maximal := FALSE$
23:                  $L_{k+1} := L_{k+1} \cup$ {T}
24:          End For
25:        End For
26:      End If
27:  End For
28:  $Max := \{\}$
29:  For $k := 1$ to $N - 1$ do
30:      $Max := Max \cup$ {S | S $\in L_k$ and S.maximal = true }
31:  End For
32: End Procedure
```

support automatically controls the timestamp ordering rule with the sub-session check, all discovered patterns will satisfy both the topology and the timestamp rules, which are very important in web usage mining.

$$Support(I, S) = \frac{|\{S_i | \forall i \, I \, is \, substring \, of \, S_i\}|}{|S|} \tag{1}$$

An auxiliary function Support(I, S) determines whether a given pattern (I) has sufficient support from the given set of reconstructed user sessions (S). Support of a pattern I is defined as a ratio between the numbers of constructed sessions supporting the pattern I, the number of all sessions. Clearly, the support for each candidate pattern at step-n can be calculated only by one scan through the transaction database by keeping candidate sessions in hashmap.

4. AGENT SIMULATOR

Our agent simulator first randomly generates a typical web page topology which preserves power law property of web graphs as it is stated for graph generators [6]. After that, it simulates a user agent that accesses this domain from its client site and navigates (randomly) in this domain like a random surfer in pagerank [19] by using the parameters described below. In this way, we will have full knowledge about the sessions beforehand, and later we can use any heuristic to process user access log data to discover the sessions. Then, we can evaluate how successful that heuristic was in reconstructing the known sessions. While generating a session, our agent simulator eliminates web user navigations provided via a client's local cache. Since the simulator knows both the referrer of each page (i.e., from which page a hyperlink is used to access to this page) and the full navigation history at the client side, it can determine navigation requests that are served by the web server, and those served from the client/proxy cache. Each heuristic is executed using the server side log file as input, and produces its own sessions. After all of the sessions are discovered by the heuristics, sequential Apriori-All technique is applied on these sessions to determine frequent navigation paths.

The following parameters are used for simulating navigation behaviors of a web user:

Session Termination Probability (STP): STP is the probability of terminating session.

Link from Previous pages Probability (LPP): LPP is the probability of referring next page from one of the previously accessed pages except the most recently accessed one. This parameter is used to allow the generation of backward movements from browser.

Link from Current page Probability (LPC): LPC is the probability of referring next page from the most recently visited page.

New Initial page Probability (NIP): NIP represents the probability of selecting one of the starting pages of a web site during the navigation, thus starting a new page request sequence in the same domain. This behavior also represents browser actions to view new page without using links on the web pages like using bookmarks or writing new address to bar.

Our user model is similar to the random surfer model of the page rank algorithm [19]. In the original page rank algorithm, the web user terminates surfing with closing his/her browser or writing a new page URL to the address bar of browser with a damping factor, d (probability value between 0 and 1). Also, with probability $1-d$ the user continues navigation by randomly clicking links on the current page. Our agent simulator generates web user requests such that with probability $(1-d)$ a web user can select a link from current page (LCP) or press the back button (LPP). If this condition happens, the web user selects the next page from the links of the previous pages or from the links of the current page according to LPP and LPC values. Also with the probability d, web user can terminate the session (STP) or jump to a new start page (NIP) by typing its URL into the address bar of the browser. Corresponding actions are simulated with probabilities STP and NIP.

5. EXPERIMENTAL RESULTS

The first subsection is dedicated to the introduction of the accuracy metric for comparing different web usage mining methods. In the next two subsections, we compare the accuracies of the maximal frequent patterns obtained from sessions of Smart-SRA, navigation oriented and time oriented heuristics. In the fourth subsection, we have compared the referrer based version and the original version of Smart-SRA on web server logs of ceng.metu.edu.tr domain (Department of Computer Engineering at Middle East Technical University) which has more than 1500 unique sessions and nearly 5K web page requests every day. In the last subsection, we discuss distributed processing of web usage data on map/reduce framework.

5.1 Accuracy Metric

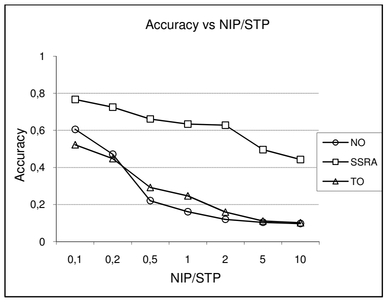
In the experiments mentioned in section 5.2 and 5.3, we have compared the most popular 3 heuristics with the Smart-SRA with respect to their accuracy performance. As it is mentioned in previous sections, sessions are processed by the sequential apriori algorithm after they are constructed by one of these four heuristics. Sequential apriori technique is also applied to original sessions generated by the agent simulator. Since, we know the frequent maximal patterns of the agent simulator (MP_A), which correspond to the correct frequent patterns, we can determine the accuracies of different heuristics (A_H stands for the accuracy of a heuristic H) by using the maximal frequent patterns generated by these heuristics (MP_H) as follows:

$$REC_H = \frac{|MP_A \cap MP_H|}{|MP_A|} \quad PRE_H = \frac{|MP_A \cap MP_H|}{|MP_H|} \tag{2}$$

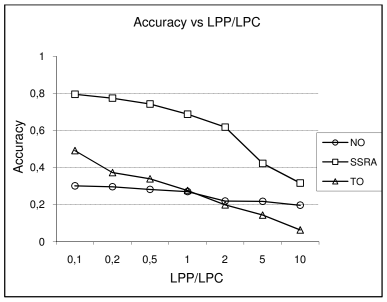
$$A_H = \sqrt{(REC_H * PRE_H)} \tag{3}$$

Here REC_H is the recall for heuristic H namely the ratio of the number of correct patterns captured by heuristic H over the number of all correct maximal patterns. PRE_H is the precision for heuristic H namely the ratio of the number of correct patterns captured by heuristics H over the number of all maximal patterns generated by heuristic H. We have defined accuracy A_H of heuristic H as a geometric mean of recall and precision for that heuristic.

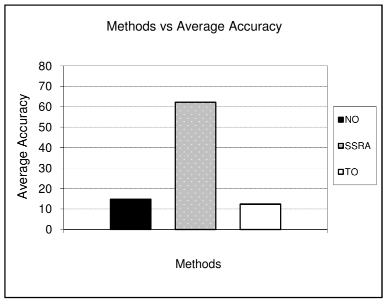
In order to balance the trade off between recall and precision, it is possible to define arithmetic mean or any linear function of these two variables as our accuracy metric. Since arithmetic mean is defined as $(REC_H + PRE_H)/2$, it may still produce incorrectly high accuracies whenever one of the REC_H or PRE_H is very large and the other one



(a) Accuracy vs NIP/STP



(b) Accuracy vs LPP/LPC



(c) Methods vs Avg. Accuracy

Figure 2: Experiments on Simulated Data

is very small. We are seeking for accuracy to be high if and only if both of them are high. The accuracy should be small even if one of these values is very large and the other one is very small. In general, due to the problems mentioned in arithmetic mean, any weighted linear functions using $d * PRE_H + (1 - d) * REC_H (d \in [0, 1])$ also fail. The geometric mean function [2, 4] solves all problems mentioned above. For obtaining high accuracy both precision and recall must be high. Therefore, we decided to use geometric mean as our accuracy metric thought the rest of this paper.

5.2 Accuracy Comparison on Simulated Data

In the first set of experiments, we have used agent simulator to produce syntactic data. The agent simulator first creates a random web topology with the parameters given in Table 3. There are works in the literature related with these parameters, and the average number of web pages in a web site is reported as 441 [27]. Also, the average number of out degrees of any web page is determined as 7.2 in [17]. Therefore, we decided to choose the ranges of parameters as in Table 3.

Table 3: Web Site and User Parameters

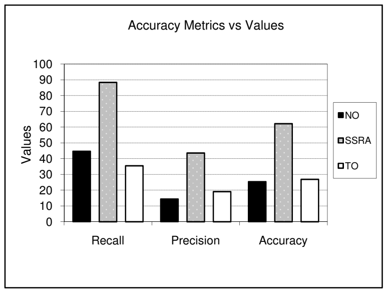
Parameter	Range
Number of web pages (nodes) in topology	[10, 1000]
Number of Users	[1000, 10000]

After the initializing steps of simulation mentioned above, we have studied the accuracy of alternative web usage mining processes (which contains the session construction phase followed by application of the sequential version of Apriori-All algorithm for frequent patter discovery). In this work, we compare the accuracies of the three web usage mining processes employing Smart-SRA, Navigation Oriented Heuristic and Time Oriented Heuristic using total session duration time threshold. We didn't plot the results for Time Oriented Heuristics using page stay time threshold since its accuracy is very close to Time Oriented Heuristics using total session duration time.

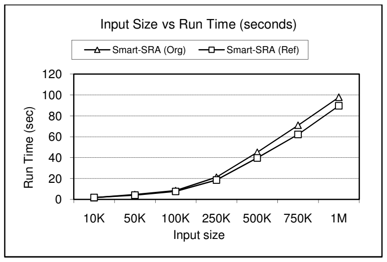
The main parameters used in our experiments are two probabilistic behavior metrics, namely the ratios (LPP/LPC) and (NIP/STP). We used 7 different values of these two parameters, namely the values $0.1, 0.2, 0.5, 1.0, 2.0$, 5.0, *and* 10.0 That is, 0.1 for (LPP/LPC) means, the probability of Link from Current Page behavior is ten times as the probability of Link From Previous Page behavior. Similarly, 10 for (LPP/LPC) means, the probability of LPP behavior is ten times as the probability of LPC behavior. Thus, with these 7 values we cover different distributions of LPP/LPC ratios. Since we have the same values for NIP/STP ratio also, in total we generate 49 different combinations of all of these 4 parameters. For each one of these 49 different cases, our agent simulator is executed 10 times with randomly chosen number of users, out degrees of web pages, and the number of web page parameters from the ranges in Table 3. Also for each test run of the agent simulator, we have executed pattern discovery phase with the 5 different support values from 0.001 to 0.010 (which are $0.001, 0.0025, 0.005$, 0.0075, *and* 0.01). Thus, for each of the 49 different combinations of $(NIP/STP$, $LPP/LPC)$, we have 50 different test runs (10 random different runs of agent simulator times 5 runs for each support values).

After that we take the average of 50 runs for each of the $(NIP/STP, LPP/LPC)$ combination. As a result, we obtain single accuracy value for each of the 49 combinations which can be represented as a $7x7$ matrix having varying LPP/LPC values on rows and varying NIP/STP values on columns. In order to provide complete visualization of all possible parameters, we plot the accuracy results of our experiments on syntactic data in 3 different views. In Figure 2(a), we plot the average accuracy of the 7 rows which corresponds to average accuracies with respect to varying NIP/STP values. Figure 2(b) contains the reverse result of the former one, since in this one we take the average of each column and plot the accuracies with respect to varying LPP/LPC values. The last figure (Figure 2(c)) shows the average of all 49 values for each session construction method in terms of accuracy, recall and precision values.

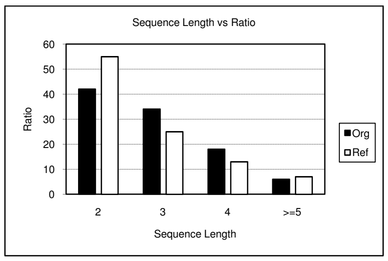
On the average, the maximal patterns obtained from Smart-SRAs sessions are at least 30% more accurate than patterns obtained from sessions of Time and Navigation Oriented Heuristics. Another observation is that increasing NIP/STP and LPP/LPC decreases the accuracy of all heuristics. This is due to the fact that LPP and NIP cause more complex behaviors in the user sessions than STP and LPC. Remember that LPP is the probability referring next page from one of the previously accessed pages except the most recently accessed one. High LPP means that web users tend to choose the next page by using links from previously accessed web pages. Clearly this behavior is more difficult to predict than choosing next page from the most recently



(a) Accuracy on Real Data



(b) Input Size vs RunTime



(c) Sequence Length vs Ratio

Figure 3: Experiments on Real Data

accessed page since in the latter case, the order is the same in the web server logs. The same argument also applies to NIP, since increasing NIP leads to web users to trigger new navigation sequence by writing new URL into the address bar or using external links from different sources such as search engines or bookmarks. However, increasing STP leads to web users to generate simple sessions that are terminated early with less number of pages. This also decreases the probability of having NIP and LPP conditions which enables heuristics to capture sessions easily.

5.3 Accuracy Comparision on Real Data

We have also evaluated the accuracy performance of both of the heuristics on real data captured from AGMLAB's company web site (www.agmlab.com). We tracked the actions of web users between March 2008 and June 2008. In this period of time, the number of visitors is 3801 which is calculated by using a 30 min session time-out. The web site for our case study contains 10 web pages and the link graph is densely connected.

For tracking client side actions of web users, the internal structure of web pages are changed by inserting an action tracking program into them. This program captures all actions of web users by using browsers on the client-side. These actions are stored as a text in a local cookie. When the user performs any page view action via browser, the information in local cookies is sent to server. In the server side, the cookie information is recorded to a log file by requested dynamic server page. By this way, we capture all of the information about the user on both the client and the server sides.

The information in our cookies represents all actions of users on the client side. Also, in the sequence extracted from the cookie, the first instance of each requested page stands for access log information in the server side. To illustrate; if the cookie contains sequential page view as [**1, 2, 6, 3**, 2, 1, **5**, 3, 2] it is easily seen that bold ones are the first instances of requested web pages and these are provided by the web server and recorded to the access log file. So, the log file contains the sequence [1, 2, 6, 3, 5]. The access logs files generated from cookies in this manner is used as an input to each heuristic.

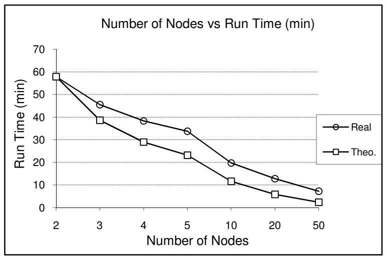
We have also extracted the real session sequences from the cookie information. For the example mentioned above, the second visit of page 2 ([1, 2, 6, 3, **2**, 1, 5, 3, 2]) is provided via browser cache, so our first sequence becomes [1, 2, 6, 3]. The first instance of page 5 comes after initial sequence [1, 2, 6, 3] and it is the page highlighted as bold ([1, 2, 6, 3, 2, 1, **5**, 3, 2]). Since the referrer of page 5 is page 1 in the cookies, the second sequence becomes [1, 5]. The browser provides page 1 from the local cache. Since page 1 is the first page of the sequence, there is no referrer of page 1 when it is requested from the server. Notice that the agent simulator produces sessions having forward references which are used for requesting web pages from the server. The user clicks a link on page 1 and requests page 5 from the server. However, this is not same for [2, 1] shown as bold, ([1, 2, 6, 3, **2, 1**, 5, 3, 2]). Finally, our real session contains two sequences {[1, 2, 6, 3], [1, 5]}. After all real sessions are extracted; maximal patterns are discovered by using Sequential Apriori method. After that, we evaluate the accuracy for each heuristic by using the methods described in the previous section.

For the evaluation of maximal patterns, we have used five different support values in the range [0.01, 0.1] with the values 0.01, 0.025, 0.05, 0.075, and 0.1. Then, we have calculated the average accuracy for these five different support values. The average accuracies for each heuristic on real data are given in Figure 3(a). As it is seen from the figure, Smart-SRA is much better than previous heuristics. While Smart-SRA has accuracy near 60%, the navigation oriented heuristic has accuracy near 25% and time oriented heuristic has accuracy near 27%. From the experimental results on our real data we can also observe several drawbacks of previous heuristics, which were also reflected in these results. First, time-oriented heuristics don't consider link information. In most of the cases, a web user navigates between pages by using hyperlinks in the pages. When the link information is not used, the sessions become an unrelated set of accessed web pages without navigation paths. In navigation-oriented approach, backward browser movements becomes a major problem, since the rest of the session always corresponds to forward movements in web topology graph. Also, Navigation oriented heuristics only consider the nearest page for navigating a new page which current one does not refer.

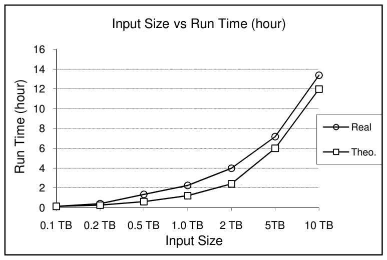
5.4 Experimental Results on Large Web Site

In this section, we provide experimental results over web server logs of ceng.metu.edu.tr for analyzing effect of the referrer information, seqeunce length and accuracy. Our test domain is visited by more than 1500 unique visitors requesting about 5K web pages daily. We have used six month server logs (which contains nearly 1M web page requests) for our experiments and focused on determining the effects of the referrer information on the performance of two versions of Smart-SRA, namely the one that uses the referrer infor-

(a) AGMLAB's Computer Cluster



(b) Performance on 100GB Data



(c) Performance with 50 nodes

Figure 4: Experiments on Large Scale Web Usage Data

mation and the one that does not. For comparison, we have measured elapsed time for generating user sessions and the number of sequences generated by two session construction algorithms. In these experiments, we ignored the candidate sessions having only one web page since there is no need to use any method to extract maximal sessions from them. We have also ignored the output sequences having single page since these sequences do not corresponds to paths in the web topology.

In the first experiment, we have determined the number of navigation sequences and run time for generating these sequences for varying log sizes. Starting from the first 10K web page requests of access logs, we have compared the two versions of Smart-SRA on 7 different input sizes up to 1M web page requests. The run time comparision of these two method is given in Figure 3(b). Actually the number of sequences generated graph is exactly same as run time graph. Thefore, we only give run time graph. As it is seen from these figures, referrer based version of smart-SRA is slightly better than the original version in terms of the execution time since it produces less sessions. However, it should be noticed that there is not a significant difference between the two versions of Smart-SRA since the original Smart-SRA uses the restricted web topology which prevents the explosion in the number of total sessions generated. As it is easily observed from the sequence length distribution graph (Figure 3(c)), the length distribution of sequences generated by these two methods are very similar.

In the last experiment, we have measured the accuracy of the navigation patterns obtained from the sessions generated by the original version of Smart-SRA and the referrer based version of Smart-SRA. Here, we assume that the navigation patterns obtained from referred based version as the correct set. Under this assumption, the accuracy of the original Smart-SRA is determined as 69% percent, which is also 35% better than the time and the navigation oriented heuristics.

5.5 Distributed Processing of Web Usage Data

We have implemented a distributed version of Smart-Miner framework using Map/Reduce [10] programming model which runs on AGMLAB's computer cluster having more than 100 computers (Figure 4(a)). With the help of Map/Reduce model, currently, we are processing web server logs of multiple web sites which are uploaded to our system daily. Map/Reduce [10] is a programming paradigm which expresses a large distributed computation as a sequence of distributed operations called map and reduce tasks on data sets of key/value pairs. In order to employ Map/Reduce model, during the computation, both input and internal data sources should be represented as (key, value) pairs.

Due to the space limitations we will skip the details of the Map/Reduce version of our algorithms. However, the idea is quite simple for both session construction and pattern discovery phases. We have a single Map/Reduce job for session construction, the map phase of which hashes the IP address and domain name (since we process logs of multiple web sites) as key from single access logs and delivers all access logs belonging to same IP and domain name pair to the same reducers. In this way, we accumulate all access records of single IP belonging to same domain on single reducer machine. After that, we run the Algorithm 1 on single reducer machine. The pattern discovery phase is trickier. Unlike session construction case, we have implemented two Map/Reduce jobs for pattern discovery. The first job creates candidate sequences and the latter one calculates frequency of candidate sequences as well as eliminating both infrequent and non maximal patterns. Each job is repeated for each step-n of the Algorithm 2 until all maximal patterns are discovered. The offline framework continues to run in this manner until it generates all of the maximal patterns. The most important point here is that, we calculate the frequency of each candidate pattern in a distributed manner by partitioning the session database into small blocks which are processed by different map/reduce nodes. Each map/reduce node counts the local frequency of candidate pattern over its repository and local results are merged for finding global frequency of each candidate patterns. We measured the run time performance of our cluster (up to 50 nodes) by generating frequent navigation sequences of 1 month data (about 100 GB) from only most popular web sites analyzed by AGMLAB Information Technologies. As it is seen from Figure 4(b), our framework is capable of processing 100 GB data in 5 minutes with 50 nodes. In the second experiment, we have processed 6 months usage data of several web sites belongs to different customers of AGMLAB. In this experiment (Figure 4(c)), we have used 50 nodes and varied the the total input size from 100GB to 10 TB. The results show that our framework is capable of processing 10TB usage data in one day (14 hours) with 50 nodes. In fact, our scalable architecture shows similar curve like in the theoretical limit (run time proportional to the input size or number of nodes) which implies that we can process any size of data by increasing number of nodes in the cluster.

6. CONCLUSION AND FUTURE WORK

In this paper we introduced the new web usage mining framework, Smart-Miner, that we have developed as a sub module of commercial service. The framework contains several novel contributions to different web usage mining problems, such as session construction, frequent user pattern discovery and large scale web usage data processing. We have proved the quality and benefits of our new framework with several experiments by using syntactic data to large scale real web user data. We have also showed that our framework produce better result than previous methods and it is shown that our framework can process any size of web usage data with its scalable architecture.

As it is mentioned above, Smart-Miner framework is part of our Web Analytics Service. As a future work, we are planning to improve Web Analytics Service by intelligent applications that works on frequent navigation patterns generated by Smart-Miner. Specifically, we are planning to design a decision support system which will execute user defined filters over frequent access patterns for advertisement or fraud pattern analysis.

7. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.

[2] M. A. Bayir, D. Guney, and T. Can. Integration of topological measures for eliminating non-specific interactions in protein interaction networks. *Discrete Applied Mathematics*, doi:10.1016/j.dam.2008.06.034, 2008.

[3] J. Borges and M. Levene. Generating dynamic higher-order markov models in web usage mining. In *PKDD*, pages 34–45, 2005.

[4] S. Brohée and J. van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7:488, 2006.

[5] L. D. Catledge and J. E. Pitkow. Characterizing browsing strategies in the world-wide web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, 1995.

[6] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), 2006.

[7] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *ICTAI*, pages 558–567, 1997.

[8] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowl. Inf. Syst.*, 1(1):5–32, 1999.

[9] R. Cooley, P.-N. Tan, and J. Srivastava. Discovery of interesting usage patterns from web data. In *WEBKDD*, pages 163–182, 1999.

[10] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[11] D. Donato, L. Laura, S. Leonardi, and S. Millozzi. The web as a graph: How far we are. *ACM Trans. Internet Techn.*, 7(1), 2007.

[12] E. Frias-Martinez and V. Karamcheti. A customizable behavior model for temporal prediction of web user sequences. In *WEBKDD*, pages 66–85, 2002.

[13] A. A. Ghorbani and X. Xu. A fuzzy markov model approach for predicting user navigation. In *Web Intelligence*, pages 307–311, 2007.

[14] D. Godoy and A. Amandi. Learning browsing patterns for context-aware recommendation. In *IFIP AI*, pages 61–70, 2006.

[15] J. Jung. Semantic preprocessing of web request streams for web usage mining. *J. UCS*, 11(8):1383–1396, 2005.

[16] N. Khasawneh and C.-C. Chan. Active user-based and ontology-based web log data preprocessing for web usage mining. In *Web Intelligence*, pages 325–328, 2006.

[17] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. The web as a graph. In *PODS*, pages 1–10, 2000.

[18] O. Nasraoui, M. Soliman, E. Saka, A. Badia, and R. Germain. A web usage mining framework for mining evolving user profiles in dynamic web sites. *IEEE Trans. Knowl. Data Eng.*, 20(2):202–215, 2008.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. tech. rep. computer systems laboratory. Technical report, Stanford University, Stanford, CA., 1998.

[20] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng.*, 16(11):1424–1440, 2004.

[21] C. Shahabi and F. B. Kashani. Efficient and anonymous web-usage mining for web personalization. *INFORMS Journal on Computing*, 15(2):123–147, 2003.

[22] A. D. Silva, Y. Lechevallier, F. de A. T. de Carvalho, and B. Trousse. Mining web usage data for discovering navigation clusters. In *ISCC*, pages 910–915, 2006.

[23] M. Spiliopoulou and L. Faulstich. Wum - a tool for www ulitization analysis. In *WebDB*, pages 184–103, 1998.

[24] M. Spiliopoulou, B. Mobasher, B. Berendt, and M. Nakagawa. A framework for the evaluation of session reconstruction heuristics in web-usage analysis. *INFORMS Journal on Computing*, 15(2):171–190, 2003.

[25] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 3–17, 1996.

[26] 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, 2000.

[27] R. B. T. O'Neill, F. Lavoie. Trends in the evolution of the public web. *D-Lib Magazine*, 9 Number:4, 2003.

[28] M. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42:31–60, 2001.