A New Approach for Reactive Web Usage Data Processing

Murat Ali Bayir, Ismail H. Toroslu, Ahmet Cosar
Department of Computer Engineering,
Middle East Technical University,
Ankara, 06531 Turkey
{ali.bayir, toroslu, cosar }@ceng.metu.edu.tr

Abstract— **Web usage mining exploits data mining techniques to discover valuable information from navigation behavior of World Wide Web (WWW) users. The required information is captured by web servers and stored in web usage data logs. The first phase of web usage mining is the data processing phase. In the data processing phase, first, relevant information is filtered from the logs. After that, sessions are reconstructed by using heuristics that select and group requests belonging to the same user session. If we are processing requests after they are handled by the web server, this technique is called "reactive" while in "proactive" techniques the same (pre)processing occurs during the interactive browsing of the web site by the user. Reactive session reconstruction uses "time" and "navigation" oriented heuristics. We propose to combine these heuristics with "site topology" information in order to increase the accuracy of the reconstructed sessions. In this work, we have implemented an agent simulator, which models behavior of web users and generates web user navigation as well as the log data kept by the web server. By this way we know the actual user sessions and we can accurately evaluate and compare the performances of alternative session reconstruction heuristics (which will use only the web server log data). To the best of our knowledge, this paper is the first work that uses such an agent simulator, and therefore, is able to accurately evaluate different session reconstruction heuristics. By using the agent simulator, we attempt to show that our new heuristic discovers more accurate sessions than previous heuristics.**

Index Terms— **Web mining, web usage mining, session reconstruction, agent simulator and web topology.**

1. Introduction

As in classical data mining, in web mining [5] the aim is to discover and retrieve useful and interesting patterns from a large dataset. In web mining, this dataset is the huge web data. Web data contains different kinds of information, including, web documents data, web structure data, web log data, and user profiles data. All of these data can be mined mainly in three different dimensions, which are; Web content mining, Web structure mining, and Web usage mining.

In web content mining, web documents, such as text and multimedia, are used. Web content mining is similar to other text mining problems. Categorization or classifications of documents are typical applications. These systems are usually built on top of existing search engines and facilitate the web users' search for information.

In web structure mining, web structure data, such as HTML and XML tags are used. Web structure data describes the organization of the content on the web. Inter-page relationships are the most important data related to web structure. Also in web structure mining, the graph structure of the web is analyzed in order to discover the model underlying the link structure of the web. This information could be used to calculate organization and/or popularity of web pages and present the web users with intelligent guidance, or for directing an advertiser to web sites that are more likely to be visited by potential customers.

Web usage mining (WUM) [5, 13] can be defined as the application of data mining techniques to web log data in order to discover user access patterns. Web usage mining has various application areas such as web pre-fetching, link prediction, site reorganization and web personalization. Most important phases of WUM are the reconstruction of user sessions by using heuristics techniques, and discovering useful patterns from these sessions by using pattern discovery techniques like apriori or similar ones. WUM data is related to mainly users' navigation on the web. The most common action of the web user is navigation through web pages by using hyperlinks. A web page can be accepted as related to another web page if they are accessed in the same user session; also, similarity expectation increases if two pages are accessed in the same session of a user. However, since HTTP protocol is stateless and connectionless, discovering the user sessions from server logs is not an easy task. For reactive strategies, all users behind a proxy server will have the same IP number and will be seen as a single client machine and all of these users' log records will contain the same IP number in the web log data. Also, caching performed by the clients' browsers and proxy servers will make web log data even less reliable. These problems can be handled by proactive strategies by using cookies and/or java applets. However, cookies and applets could have been disabled by some clients for privacy and security considerations. In this case such solutions for proactive strategies would also become unusable. In previous works on reactive strategies, mainly sessions are reconstructed by using page access timestamps and navigation constraint heuristics [2].

The data source for web usage mining can vary with respect to the methods used. In proactive strategies [11, 9], the raw data is collected when client requests are being processed by web server,. Proactive strategies are more appropriate for dynamically created server pages. Also, in proactive strategies, association of an agent with a session is determined during the interaction of user with web site. However, in reactive strategies [12, 6, 7], the available raw data is mainly server logs containing information about client requests. In this work, we only consider reactive strategies because mining a huge collection of access data captured by web server can be more convenient after the interaction since it doesn't add extra load on the web server while it is busy serving client requests. Comparison of reactive approaches with proactive ones is not meaningful because of different input sets they use. Since web server logs are used for reactive processing, raw data has the same advantages and disadvantages for reactive heuristics.

When a user agent (Internet Explorer, Mozilla, Netscape, etc.) hit an URL in a web server's domain, the information related to that operation is recorded in that web server's access log file. An access log file contains its information in Common Logfile Format (CLF) [4]. In CLF, each client request for any URL corresponds to a record in access log file. Each CLF record is a tuple containing seven attributes that are given below:

- Client machine's IP address
- Access date and time
- Request method (GET or POST),
- URL of the page accessed
- Transfer protocol (HTTP 1.0, HTTP 1.1,)
- Success of return code
- Number of bytes transmitted

For the session reconstruction, IP address, request time, and URL are the only information needed from the user web access log in order to obtain users' navigation paths. Therefore, other attributes from the log data are ignored.

Reactive strategies are mostly applied on static web pages. Because the content of dynamic web pages changes in time, it is difficult to predict the relationship between web pages and obtain meaningful navigation path patterns. Therefore we restrict our work to static web pages.

As mentioned above, previously designed reactive strategies [2] for session reconstruction use two types of heuristics. In time-oriented heuristics [12, 6], session data is reconstructed by analyzing the session duration time or the time between consecutive web page requests (page stay time). In navigation-oriented approach, session reconstruction is performed by analyzing the hyperlinks among the pages user requested [6, 7]. This heuristic must estimate (speculate) browser movements by providing path completion. In this work, we propose a novel approach, which combines time-oriented and navigation oriented approaches in order to obtain more accurate sessions and do this more efficiently. As in navigation-oriented heuristic, our technique also uses the web site topology and includes path completion with a different method. It is a reactive strategy designed for discovering user session patterns on static web pages. Because we assume a static web, the target web site to be mined can be easily modeled as a static web graph [1, 8, 10]. The adjacency matrix of this graph represents the relationships among the web pages. We compare our heuristics with all three previously studied reactive strategies. We don't perform any comparison with proactive strategies as they would definitely use more information (e.g. cookies) instead of using only the web log data, and any comparison would be unfair to reactive strategies.

This paper is organized as follows. The next section summarizes previously proposed reactive strategies, namely *time* and *navigation* oriented heuristics. Section 3 introduces our heuristic technique for session reconstruction. Section 4 introduces the agent simulator that we have developed in order to evaluate and compare different session reconstruction heuristics. Section 5 compares the performance of our new heuristic, with respect to previous heuristics. Finally, we give conclusions.

2. Previous Heuristics for Session Reconstruction

2.1 Time-oriented heuristics

Time oriented heuristics [12, 6] are based on time limitations on total session time or page-stay time. There are two types of time-oriented heuristics. In the first one, the duration of a discovered session cannot be greater than a predefined upper bound, δ. The upper bound δ is usually accepted as 30 minutes according to [3]. Any page requested with timestamp t_i can be appended to the current session under consideration if the time difference between the requested page's timestamp and the timestamp of the first page t_0 of that session is smaller than δ ($t_i - t_0 \leq \delta$). The first page with a timestamp greater than $t_0 + \delta$ becomes the first page of the next session. In other words, if [WP_1, WP_2, …., WP_N] are web pages forming a session (in increasing order of access time), then access_time (WP_N) – access_time (WP_1) ≤ δ.

Table 1: An example web page request sequence.

Page	P_1	P_{20}	P_{13}	P_{49}	P_{34}	P_{23}
Timestamp	0	6	15	29	32	47

By using a δ value of 30 minutes, we obtain two sessions from the web page request sequence given in Table 1; the first session is [P_1, P_{20}, P_{13}, P_{49}] and the second session is [P_{34}, P_{23}].

In the second time-oriented heuristic, the time spent on any page is limited with a threshold of ρ. This threshold value is accepted as 10 minutes according to [3]. If t_j is the timestamp of the most recently accessed page P_J, and t_{J-1} is the timestamp of page P_{J-1} accessed immediately before page P_J, then, $t_J - t_{J-1} \leq \rho$ must be satisfied. Otherwise this new request becomes the first page of the new session. In other words, if [WP_1, WP_2, …, WP_K, WP_{K+1}, … ,WP_N] are pages forming a session, then, $1 \leq K < N$ access_time(WP_{K+1}) – access_time(WP_K) ≤ ρ.

By using ρ as 10 min, we obtain three sessions from the web page request stream given in Table 1; these sessions are [P_1, P_{20}, P_{13}], [P_{49} , P_{34}], and [P_{23}].

In time-oriented approaches, it is very challenging to mine session data correctly, since they do not consider the web page *connectivity*. In real life, most of the web users request a web page from another one having a hyperlink to it. Also a web page referring to another page can be accepted as related. Thus, it is better to group these pages in the same session. On the other hand, it is better to put two pages into two different sessions if the first one accessed does not have any links to the next one even though the second one is accessed immediately after the first one. Most probably these pages will be unrelated to each other.

2.2 Navigation-oriented heuristic

Navigation-oriented approach [6, 7] uses web topology, based on graph models [1, 8, 10] constructed using the hyperlinks among web pages, in order to discover sessions. However, in a session, it is not necessary to have a hyperlink between every two consecutive web pages. For every page WP_K (except the initial page WP_1) in a session there must be at least one page WP_J with a hyperlink to WP_K in the same session, having a smaller timestamp than WP_K. In other words, if [WP_1, WP_2, …, WP_J, …, WP_K, … , WP_N] are pages forming a session, then, ∀K ∃J such that, access_time(WP_K) > access_time(WP_J) and there exist a hyperlink from P_J to P_K. If there are several pages having hyperlinks to WP_K with smaller timestamps, then, among these pages, the one with the largest timestamp, WP_{Jmax}, is assumed to be used for accessing the page WP_K. Therefore, during the session reconstruction, backward browser movements until page WP_{Jmax} with a hyperlink to WP_K are appended to that session.

During the construction of a new session, if [WP_1, WP_2, …, WP_N] is the current session and WP_{N+1} is a new page, then, the page WP_{N+1} can be added to this session as follows:

- If WP_N has a hyperlink to WP_{N+1}, new session becomes [WP_1, WP_2, …, WP_K, WP_{K+1}, … , WP_N, WP_{N+1}] .
- If WP_N does not have a hyperlink to WP_{N+1}, and WP_{Kmax} is the nearest (with the largest timestamp smaller than the timestamp of WP_{N+1}) page having a hyperlink to WP_{N+1}, then, the new session becomes [WP_1, WP_2, …, WP_K, WP_{K+1}, … , WP_N, **WP_{N-1}, WP_{N-2}, ..., WP_{Kmax}**, WP_{N+1}]. The subsequence represented in bold has been added to represent the backward browser movements which are normally served from the browser's local cache.

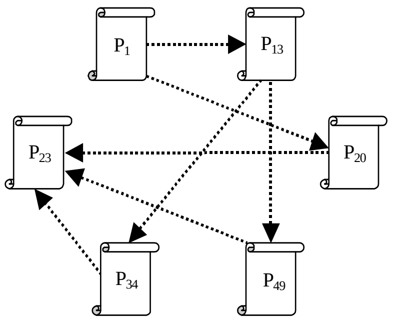

Figure 1. An example web topology graph.

Consider again the Table 1 that shows a sample web page requests sequence of an agent in increasing timestamp order, and Figure 1 representing the web topology graph between the requested pages, where each page is a node and each directed edge represents the hyperlink from the referring page to the target page. The process used for evaluation of the final session is given in Table 2. Backward browser movements are given in bold.

Table 2: Evaluation of the example session for Table 1 by using navigation oriented heuristic.

Curent Session	Condition	New Page
[ ]	–	P_1
[P_1]	Link[P_1, P_{20}] = 1	P_{20}
[P_1, P_{20}]	Link[P_{20}, P_{13}] = 0 Link[P_1, P_{13}] = 1	P_{13}
[P_1, P_{20}, **P_1,** P_{13}]	Link[P_{13}, P_{49}] = 1	P_{49}
[P_1, P_{20}, **P_1,** P_{13}, P_{49}]	Link[P_{49}, P_{34}] = 0 Link[P_{13}, P_{34}] = 1	P_{34}
[P_1, P_{20}, **P_1,** P_{13}, P_{49}, **P_{13},** P_{34}]	Link[P_{34}, P_{23}] =1	P_{23}
[P_1, P_{20}, **P_1,** P_{13}, P_{49}, **P_{13},** P_{34}, P_{23}]	–	-

In navigation-oriented approach artificially inserting backward browser movements is a major problem, since although the rest of the session always corresponds to forward movements in web topology graph, backward movements represent movements in reverse direction of the edges, and it is difficult to interpret patterns obtained in this manner. Another problem is the length of sessions. Sessions tend to become much longer due to insertion of backward movements, and if a navigation-oriented heuristic is used without any time limitation, it is possible to obtain very long sessions. Lastly, discovering useful patterns from these resulting longer patterns becomes more difficult and inefficient.

3. A New Reactive Strategy for Session Reconstruction

In our method we propose solutions to the deficiencies of time and navigation oriented heuristics by combining both of these heuristics and by using the site topology in a way that eliminates the need for inserting the backward browser movements of navigation-oriented heuristic, which will mean increased efficiency with shorter sequences.

We propose a method, called as *Smart Session Reconstruction Algorithm (Smart-SRA)* with two phases. In the first phase, shorter request sequences are constructed by using overall session duration time and page-stay time criteria. A sub-session constructed by using these two criteria corresponds to a session formed according to the time-oriented heuristic using both limitations. In the second phase, sessions are partitioned into maximal sub-sessions satisfying both the time and the topology rules. That means each session [P_1, … P_i, P_{i+1}, … P_n] satisfies the following two conditions:

- *Timestamp Ordering Rule*: The request time of the first page in each consecutive page pair must be smaller than the request time of the second page (Timestamp(P_i) < Timestamp(P_{i+1})), guaranteeing that the web page sequence is in the increasing order of web page request timestamps, and the access time difference between two consecutive pages is less than a predefined limit(page-stay time).
- *Topology Rule*: Between each consecutive page pair in a session there must be a hyperlink from the first page to the second page (i.e., $\forall i \leq n$, there is at least one hyperlink from P_i to P_{i+1}).

Notice that, the overall session duration time limit is already guaranteed after performing the first phase. The second phase adds referrer constraints, while still ensuring the satisfaction of the second time constraint and eliminating the need for inserting backward browser moves. The second phase of the following algorithm extracts these sequences.

Session Reconstruction Algorithm (Smart-SRA)
Input: Page request sequence of a user, given in timestamp order, and named as *UserRequestSequence*. The web topology graph including only the nodes appearing in *UserRequestSequence*, and represented with the adjacency matrix, called *Link*.
Output: The set of reconstructed sessions.
Phase 1: Construct candidate sessions from *UserRequestSequence* by using both of the time-oriented heuristics. Whenever time difference between two page accesses exceeds the *page-stay* threshold (10 minutes), *UserRequestSequence* is broken between these two pages into two separate session candidates. Also, if the difference between the current page and the first page in session candidate exceeds the *session duration* threshold (30 minutes), the current session candidate is terminated, and a new session candidate is started. That means for each candidate session constructed, both the total duration of the whole session and the time spent on each page in a session will be within the given thresholds. The set of sub-sessions constructed in this step is called as *CandSessionSet*.
Phase 2: Construct sessions from each candidate session in *CandSessionSet*, by using the web topology. First, the web pages without any referrers are determined in the candidate session (Step I). Second, these pages are removed from *CandSessionSet* (Step II) and then, they are appended to the previously constructed sessions, if there is a hyperlink from the last page of a session to new web pages (Step III). Iterations terminate when all pages in the candidate session have been processed. This algorithm is given in Figure 2.

```
Procedure: Session Reconstruction (Phase 2)
ForEach CandSession in CandSessionSet
  NewSessionSet := { }
  While CandSession ≠ { }
    TSessionSet := { }
    // Step I
    TPageSet := { }
    ForEach Page_i in CandSession
      StartPageFlag := TRUE
      ForEach Page_j  Where j>i in CandSession
        If (Link[Page_j, Page_i]=1) AND
          (TimeDiff(Page_j, Page_i) ≤ 10) Then
          StartPageFlag := FALSE
        If StartPageFlag = TRUE Then
          TPageSet := TPageSet U {Page_i}
    // Step II
    CandSession:= CandSession – TPageSet
    If NewSessionSet = { } Then  // Step III-a
      ForEach Page_i in TPageSet
        TSessionSet := TSessionSet U {[Page_i]}
    Else // Step III-b
      ForEach Page_i in TPageSet
        ForEach Session_j in NewSessionSet
          // If the last element of current session has a link to
          // current page and satisfies time requirements
          If  (Link[LastElement(Session_j), Page_i] = 1)
          AND (TimeDiff(Page_j, Page_i) ≤ 10) Then
            TSession := Session_j
            TSession.mark := UNEXTENDED
            TSession := TSession ● Page_i  // Append
            TSessionSet := TSessionSet U {TSession}
            Session_j.mark := EXTENDED
          EndIf
        EndFor
      EndFor
    EndIf
    ForEach Session_j in NewSessionSet
      If Session_j.mark ≠ EXTENDED Then
        TSessionSet := TSessionSet U { Session_j}
     NewSessionSet := TSessionSet
  EndWhile
EndFor
```

Figure 2. Phase 2 of the session reconstruction algorithm.

Notice that only maximal sequences are kept through the iterations and thus, there is no redundant session construction. Moreover, if the web topology graph contains vertices corresponding to web pages that do not appear in the candidate session being processed, these vertices and their incident edges must be removed from the graph prior to the execution.

Table 3: 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

Consider Table 3 that shows a sample web page requests sequence of an agent obtained by using the first phase of the above algorithm, and the web topology is as given in Figure 1. The application of the inner loop (while loop) of the second phase of the session reconstruction algorithm is given in Table 4. For this example, Smart-SRA discovers the following maximal sessions satisfying both timestamp ordering and topology conditions:

1. [P_1, P_{13}, P_{34}, P_{23}]
2. [P_1, P_{13}, P_{49}, P_{23}]
3. [P_1, P_{20}, P_{23}]

4. Agent Simulator

It is not possible to use real user navigation data for evaluating and comparing different web user session reconstruction heuristics since all of the actual user requests cannot be captured by processing server side access logs. Especially the sessions containing access requests served from a client's and/or proxy's local cache cannot be accurately determined. Therefore, we have developed an agent simulator that generates web agent requests by simulating an actual web user.

Our agent simulator first randomly generates a typical web page topology and then simulates a user agent that accesses this domain from its client site and navigates in this domain like a real user. In this way, we will have full knowledge about the sessions beforehand, and later we can use a heuristic to process user access log data to discover the sessions. Then, we 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 the full navigation history at the client side, it can determine navigation requests that are served by the web server.

Agent simulator will produce an access log file at server side containing requests provided by web server. The sessions discovered by the heuristics are compared with the original complete session file. For example consider an agent with complete page sequences of [P_1, P_{13}, P_{34}] and [P_1, P_{20}] generated by the agent simulator, which are the real sessions. However, in the produced web server log this sequence can appear as [P_1, P_{13}, P_{34}, P_{20}] because browser provides the movement from P_{34} to P_1 through P_{13} using its local cache, meaning these last two movement will not be sent to the web server. We execute heuristics on the server side log data and produce candidate session sequences. These candidate sequences are compared with real session sequences in order to determine the accuracy of evaluated heuristics.

Table 4: Evaluation of example session.

Iteration	**1**	**2**	**3**	**4**
Candidate Session	[P_1, P_{20}, P_{13}, P_{49}, P_{34}, P_{23}]	[P_{20}, P_{13}, P_{49}, P_{34}, P_{23}]	[P_{49}, P_{34}, P_{23}]	[P_{23}]
New Session Set (before)		[P_1]	[P_1,P_{20}] [P_1,P_{13}]	[P_1,P_{13},P_{34}] [P_1, P_{13}, P_{49}] [P_1, P_{20}]
Temp Page Set	{P_1}	{P_{20}, P_{13}}	{P_{49}, P_{34}}	{P_{23}}
Temp Session Set	[P_1]	[P_1,P_{20}] [P_1,P_{13}]	[P_1,P_{13},P_{34}] [P_1, P_{13}, P_{49}]	[P_1, P_{13}, P_{34}, P_{23}] [P_1, P_{13}, P_{49}, P_{23}] [P_1, P_{20}, P_{23}]
New Session Set (after)	[P_1]	[P_1,P_{20}] [P_1,P_{13}]	[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}]
Explanation	P_1 is the start page.	Both P_{20} and P_{13} are reachable from P_1	Both P_{49} and P_{34} are reachable from P_{13}, but not from P_{20}	P_{23} is reachable from P_{34}, P_{49} and P_{20}

An important feature of our agent simulator is its ability to represent dynamic behaviors of a web agent. It simulates four basic behaviors of a web user. These behaviors can be used to construct more complex navigation sequences in a single session. These four basic behaviors constructing complex navigations are given below:

1. A Web user can start a new session with any one of the possible entry pages of a web site: Most of the time a web user can enter a web site from external domains via links or users can directly type in the web page address in their browser. Regardless of the entrance type, in each web site there are starting pages such as “index.html”. These pages can be the starting page of many web agents with a high probability.

For static pages, of course all pages can be typed in address bar and accessed directly. However, not all of the pages are likely to take the first hit from the web users with very high probability. So, most of the pages cannot be accepted as a session starting page. While agent simulator creates site topology, it also determines the starting pages for the topology. When a user starts a session, the fist page of session is randomly selected from the set of these starting pages. Also, during the navigation, web user can request a new starting page, which cannot be accessible from previous pages. User may type the web page address in the address bar. In this case this new page becomes the first page of a new session. We are going to use the example web topology given in Figure 3 to illustrate this type of behavior. In this example, gray pages represent starting pages of the domain. Since P_1 and P_{49} are the starting pages of this topology, the only possible real session list of any agent are in the form of [P_1, …] or [P_{49}, …]. While a web user is navigating after starting at some start page, s/he can jump to another start page, which is not accessible from any other previously visited page. In this case the current session terminates and a new session starts. For example, for the navigation of user illustrated in Figure 3, if the current session is [P_1, P_{20}] and user requests P_{49} and P_{23} consecutively, immediately agent simulator creates a new session [P_{49}, P_{23}] starting with P_{49}, and ends the session [P_1, P_{20}].

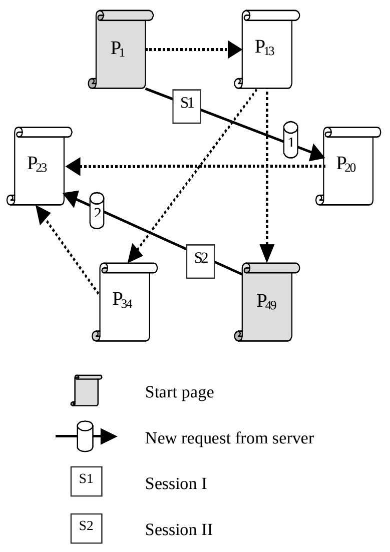

Figure 3. An example navigation of behavior type 1.

2. A Web user can select a new page having a link from the most recently accessed page: This is the most typical behavior of a web user. When a user is browsing a page, most probably s/he selects one of the links on that page to go to the next one. In order to generate web user navigation, agent simulator first finds pages having links from the current page. Then, one of them is randomly chosen and appended to the end of the current session. This behavior is illustrated in Figure 4. If [P_1, P_{13}] is the current session and the user is browsing page P_{13}, since P_{13} has links to P_{34} and P_{49}, the next page can be one of these two pages. After one of them is selected and a page is appended to the current session, the user navigation sequence becomes [P_1, P_{13}, P_{34}] or [P_1, P_{13}, P_{49}]. In Figure 4 web user select P_{34} and session becomes [P_1, P_{13}, P_{34}].

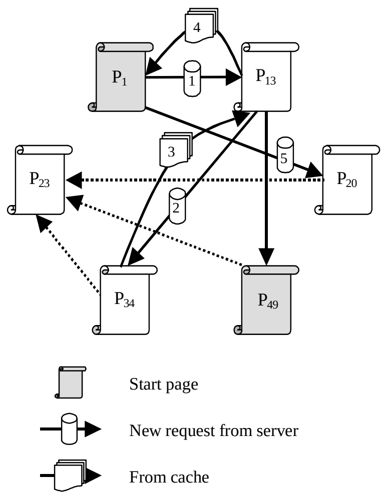

Figure 4. An example navigation of behavior type 2.

3. A Web user can select as the next page a page having a link from any one of the previously browsed pages (i.e., pages accessed before the most recently accessed one): This behavior is provided by a web browser. Agent simulator generates browser movements on the client site. However, it also eliminates these movements on the server site while generating log data. By using the web browser, web user can use "back" and "forward" buttons or a link on the current page in order to navigate back to a previously browsed page from the local cache, which has been previously obtained from the web server. A number of movements towards target page can be provided from the browser cache. Then, user can request a new page through web server from target page. In this case, agent simulator selects one of the previously accessed pages that have a link to one of the new pages not accessed before. For example, if the current session is [P_1, P_{13}, P_{34}], user can return to page P_1 then navigate to P_{20}. In this case the user's requests are: [P_1, P_{13}, P_{34}, $\mathbf{P_{13}}$, $\mathbf{P_1}$, P_{20}], bold movements are provided by browser. Agent simulator eliminates such browser movements and, it adds a new session starting from previous page having link to the next page. New real session sequences become [P_1, P_{13}, P_{34}] and [P_1, P_{20}]. Notice that agent simulator generated sessions will guarantee that P_i refers to P_{i+1}. This type of behavior is illustrated in Figure 5.

4. A Web user can terminate the session: This is a typical behavior. Users can close a browser window, or a time out event could force the session to be terminated invalidating browser links, or user can switch to a new site. In Figure 6 a user terminates his/her session in P_{23}.

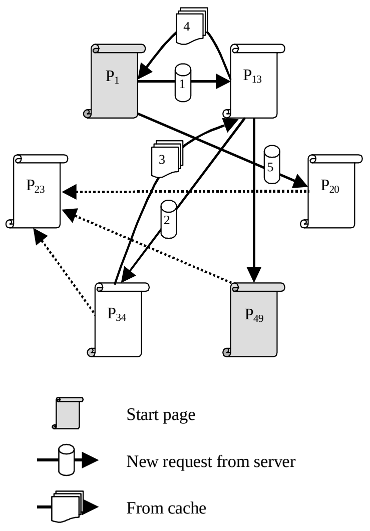

Figure 5. An example navigation of behavior type 3.

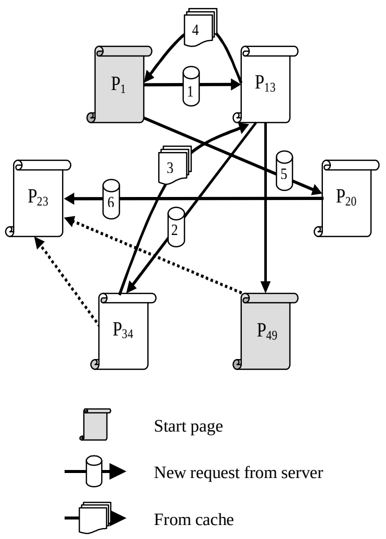

Figure 6. An example navigation of behavior type 4.

Agent simulator also uses time considerations while simulating the behaviors described above. In the second and the third behaviors, the time difference between two consecutive page requests is smaller than 10 minutes. Also, in these behaviors, time differences of access time of next page and current page obeys normal distribution. In addition, the median value is taken as 2.12 minutes (from [2]), and the standard deviation is taken as 0.5 minutes. The generated time differences set for each type of these behaviors constitute a normal distribution.

Four primitive basic behaviors given above are implemented in our agent simulator. Also, the following probability parameters are used for simulating navigation behavior of web user.

- **Session Termination Probability (STP):** It is the probability of terminating the current session at any page. The probability of terminating a session should increase as the number of requests of a session increases. The probability of terminating a session until n^{th} request is determined as $(1 - (1\text{-STP})^n)$. STP is given between (0, 1). For example if STP=0.5 the probability of terminating a session at the 2^{nd} request is 0.75.

- **Link from Previous pages Probability (LPP):** LPP is the probability of referring next page from previous pages except the most recently accessed one. This parameter is used to allow the generation of backward movements from browser. LPP can be given between [0,1).

- **New Initial page Probability (NIP):** NIP represents the probability of selecting one of the starting pages of a web site during the navigation. New Initial page Probability (NIP) is provided by the user to control this probability.

Simulating a sequence of web page requests of an agent is done by the simple algorithm given in Figure 7.

Procedure: Agent Simulator

```
EndSession := FALSE
NewPage := SelectInitialPage() // Select an initial page
PageSequence := { }
AllSeqOfAgent := { }
For Each Request While (EndSession = FALSE) do
   CurrentPage :=NewPage
   PageSequence := PageSequence ● CurrentPage // Append
   If (STP> random()) Then
       EndSession := TRUE
   Else If (NIP> random()) Then
       // Select a new, un-accessed initial page
       NewPage := SelectInitialPage()
       AllSeqOfAgent := AllSeqOfAgent U PageSequence
       PageSequence := { }
   Else If  (LPP> random()) Then
       // One of the previous pages reachable from current one
       CurrentPage := SelectPreviousPage(CurrentPage)
       NewPage := SelectPage(CurrentPage)
   Else  // Reachable from current page
       NewPage := SelectPage(CurrentPage)
EndWhile
```

Figure 7. Agent Simulator.

5. Performance Evaluation

We claim that sessions generated by our heuristic Smart-SRA are more accurate than the sessions constructed by the previous heuristics. For this purpose, we first define a method to calculate the accuracy of a reconstructed session, and then compare the accuracies of Smart-SRA with previous heuristics.

5.1 Accuracy Metric

An accurate session must satisfy both the timestamp and the topology rules explained in previous sections. The sessions generated by agent simulator satisfy both of these rules. Comparisons of session reconstruction heuristics are performed with respect to 3 parameters, namely STP, LPP and, NIP. The accuracy of a heuristic is defined as the ratio of correctly reconstructed sessions over the number of real sessions generated by the agent simulator.

In order to evaluate session reconstruction heuristics, first, our agent simulator produces simulated sessions and a corresponding web log file containing client requests for web pages. Then, we use each of the four heuristics discussed previously, to process this log file and generate candidate sessions. After that, the accuracies of the heuristics are calculated. These four heuristics are:

- Time oriented heuristic (total time ≤ 30 min) (heur1)
- Time oriented heuristic (page stay ≤ 10 min) (heur2)
- Navigation oriented heuristic (heur3)
- Smart-SRA heuristic (heur4)

We assume that a session H, reconstructed by a heuristic, captures a real session R, if R occurs as a subsequence of H (represented as $R \sqsubset H$). For example, if $R = [P_1, P_3, P_5]$ and $H = [P_9, P_1, P_3, P_5, P_8]$, then, $R \sqsubset H$ since P_1, P_3 and P_5 are elements of H and they preserve their exact order. On the other hand, if $H = [P_1, \mathbf{P_9}, P_3, P_5, P_8]$, then, $R \not\sqsubset H$, because P_9 interrupts R in H. Searching real sessions in candidate sessions produced by heuristics can be done by using a simple algorithm adopted from ordinary string searching algorithm.

5.2 Experimental Results

For comparisons of different heuristics random web sites and web agent navigations are generated by using the parameters in Table 5. The number of web pages in a web site and the average number of out degrees of the pages (number of links from one page to other pages in the same site) are taken from [14]. Varying values of the three parameters defined in the previous section, namely STP, LPP, and NIP, are used for testing the performances of the heuristics. In our experiments, we first fix two of these parameters and then experimentally evaluate performance results by varying the third parameter.

In the first experiment LPP and NIP are fixed with the values in Table 5, and STP is varying from 1% to 20%. Figure 8 depicts the real accuracy values of 4 heuristics. As it is seen from the figure, Smart-SRA outperforms the other 3 heuristics with a large difference, and the difference is very stable. Smart-SRA is almost 50% better than the other heuristics as STP (session termination probability) increases.

Table 5: Parameters used for generating user sessions and web topology.

Number of web pages (nodes) in topology	300
Average number of outdegree	15
Average number of page stay time	2,2 min
Deviation for page stay time	0,5 min
Number of agents	10000
Fixed Session Termination Probability (STP)	5% (0.05)
Fixed Link From Previous Page probability (LPP)	30% (0.3)
Fixed New Initial Page probability (NIP)	30% (0.3)

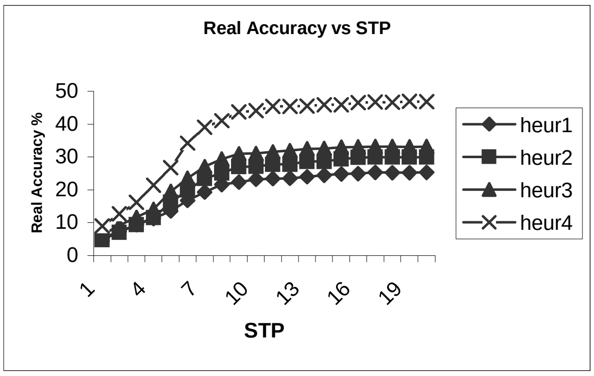

Figure 8: Reconstructed session accuracy comparison with increasing STP.

Increasing STP leads to sessions with fewer pages. In small sessions the effect of LPP and NIP is also small. For example, in a session with length 2, fixed LPP and NIP values are applied only for the second page. If the navigation is affected by LPP and NIP, then, the session becomes more complex. If there is no return back to an already visited page and there is no new initial page, then, the session becomes simple and it can be captured easily.

In the second experiment, LPP is varied from 0% to 90% and the other two parameters are fixed with the values in Table 5. The results of this experiment are given in Figure 9. As it is seen from the figure, as LPP increases the accuracies of reconstructed sessions decrease. Increasing LPP leads to more complex sessions. Path completion will be needed for discovering more accurate sessions. Although large LPP values are not very realistic, still we have presented the performance of all 4 heuristics for LPP values up to 90%. For the large LPP values, Smart-SRA captures nearly 25% of real sessions, whereas the other heuristics can determine only 6% to 7% of them. Moreover, as in the previous experiment, Smart-SRA performs at least 40% better than the best of the other heuristics for all LPP values.

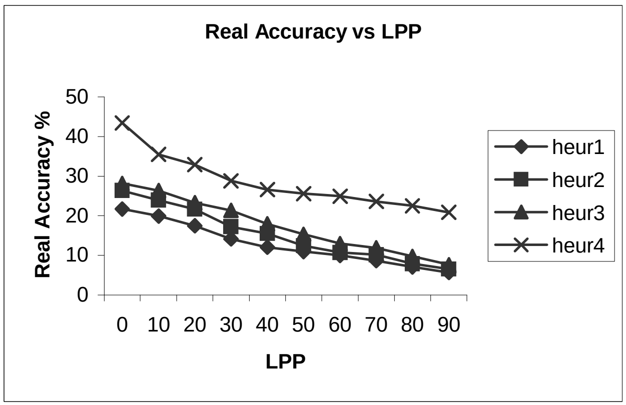

Figure 9: Reconstructed session accuracy comparison with increasing LPP.

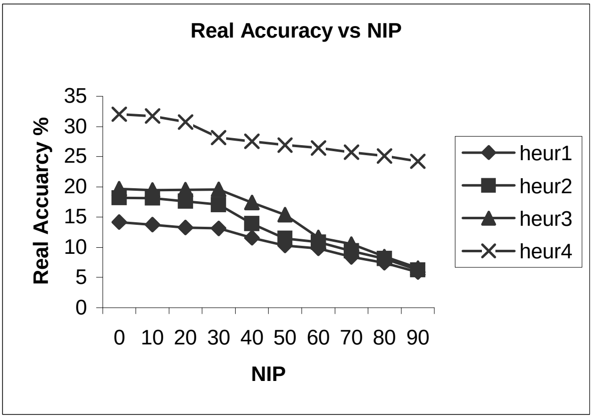

Figure 10: Reconstructed session accuracy comparison with increasing NIP.

In the third experiment, as in other experiments, two parameters are fixed with the values in Table 5 and, NIP is varied from 0% to 90%. Performances of all four heuristics are given in Figure 10. The results of this experiment are also very similar to the second one.

Increasing NIP causes more complex sessions, therefore, the accuracy decreases for all heuristics. However, large NIP values are not very realistic. The success of Smart-SRA is much higher (almost twice as good as the best of the other heuristics) for all NIP values.

6. Conclusions and Future work

This paper introduces a new session reconstruction heuristic which is based on user web page requests logs. Our heuristic, Smart-SRA, has been experimentally shown to be better than previously developed reactive, time and navigation oriented heuristics. Also, we do not allow page sequences with any unrelated (without any hyperlinks from the preceding page(s) to the next page) consecutive requests to be in the same session. Navigation oriented heuristics will insert artificial browser (back) requests into a session in order to guarantee that consecutive requests will have connectivity between each other. Since we don't insert such artificial page requests, our session sequences are much shorter and therefore, easier to process than those generated by navigation oriented heuristics. We also extend navigation oriented heuristics by using two time oriented heuristics. Another advantage of Smart-SRA is that it guarantees that all sessions generated will be maximal sequences and do not subsume any other session.

We have also implemented a novel agent simulator for generating simulated user sessions. Our agent simulator generates complete sessions satisfying both connectivity and timestamp rules. We have compared the sessions reconstructed by Smart-SRA and previous heuristics against the simulated sessions generated by the agent simulator. We have also defined a method to calculate the accuracy of the reconstructed sessions as a sequence – subsequence relationship. As it is seen from the experimental results, even when the behavior of simulated agent is very complex, Smart-SRA performs much better than previous heuristics with respect to real accuracy. In addition, it is experimentally shown that bigger NIP and LPP values leads to complex navigation behaviors, and thus, intelligent path completion and separation is needed for more accurate session reconstruction. Our experiments show that larger values of NIP and LPP decrease the accuracy of all session reconstruction heuristics while Smart-SRA is still much more successful than other heuristics for larger NIP and LPP values.

References:

[1] Andrei, B., Kumar, R., Farzin, M, *“Graph Structure in the Web,”* The Ninth International World Wide Web Conference, Amsterdam, May 2000.

[2] Berendt, B., Mobasher, B., Spiliopoulou, M., and Nakagawa, M. *“A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis,”* INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications Vol. 15, No. 2, 2003.

[3] Catledge, L.,Pitkow, J., “*Characterizing browsing behaviors on the world wide web,*” in Computer Networks and ISDN Systems, 27(6), 1995.

[4] Common Logfile Format, http://www.w3.org/Daemon/User/Config/Logging.html

[5] Cooley, R., Mobasher, B. and Srivastava, J.. *“Web mining: Information and pattern discovery on the world wide web.”* In proceedings of the 9th IEEE International Conference on Tools with Artifical Intelligence (ICTAI' 97), Newposrt Beach, CA.

[6] Cooley, R., Mobasher, B., and Srivastava, J., *“Data Preparation for Mining World Wide Web Browsing Patterns,”* Knowledge and Information Systems, vol. 1, no. 1, 1999.

[7] Cooley, R., Tan, P. and Srivastava, J., *“Discovery of interesting usage patterns from Web data,”* Advances in Web Usage Analysis and User Profiling. LNAI 1836, Springer, Berlin, Germany, pp.163-182, 2000.

[8] Cooper, C, Frieze, A., *“A general model of Web graphs,”* European Symposium on Algorithms: Algorithms-ESA 2001, pp. 500-511, 2001.

[9] Fu, Y. and Shih, M., “*A Framework for Personal Web Usage Mining,”* International Conference on Internet Computing, Las Vegas, NV, pp. 595-600, 2002.

[10] Kumar, R., Prabhagar, R., Sridhar, R., “*The Web As a Graph,”* Proc. of 19th ACM SIGMOD-SIGACT- SIGART Symp, PODS 2000.

[11] Shahabi, C., Kashani, F., “*Efficient and Anonymous Web-Usage Mining for Web Personalization,”* INFORMS Journal of Computing 15(2), pp. 123-147, 2003.

[12] Spiliopoulou, M., Faulstich, L.C. *"WUM: A tool for Web Utilization analysis,”* Proc.of EDBT workshop WebDB'98, LNCS 1590, Springer, Berlin, Germany, pp.184-203, 1998.

[13] Srivastava, J., Cooley, R., Desphande, M. and Tan, P. *“Web Usage Mining, Discovery and Applications of usage patterns from web data,”* SIGKDD Explorations 1(2):12-23, 2000.

[14] http://www.sims.berkeley.edu/research/ projects/how-much-info/internet/ rawdata.html