Finding All Maximal Paths In Web User Sessions

Murat Ali Bayir*
Microsoft Bing Ads R&D Center
Bellevue, WA, USA, 98004

Ismail Hakki Toroslu
Middle East Technical University
Ankara, 06530, Turkey

ABSTRACT

This paper introduces a new method for the session construction problem, which is the first main step of the web usage mining process. The proposed method is capable of extracting all possible maximal navigation sequences of web users. Through experiments, it is shown that when our new technique is used, it outperforms previous approaches in web usage mining applications such as next-page prediction.

Keywords

Web Mining, Linked Data, Graph Theory

1. INTRODUCTION

The purpose of Web Usage Mining (WUM) [4] is to find interesting knowledge about navigation behaviors of web users. The first step of WUM includes the session construction from user logs which directly affects the quality of patterns discovered in WUM process. Previous approaches [5] for session reconstruction have two problems. They are either using time information without link data or add artificial backward movements (noise) to complete paths in web topology. These problems can be handled by using cookies and adding client specific information in server requests. 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.

Our previous method Smart-SRA [2] solved most of the problems mentioned above. However, it still can not capture particular user behaviors due to its greedy nature. Consider the following example on web topology given in Figure 1. A web user follows the path $[P_1, P_{13}, P_{49}]$ and goes back to P_{13}. Then, the same user visits P_{34} from links on P_{13}. In this case, there will be two maximal sessions sequences that represents user's paths in the graph $\{[P_1, P_{13}, P_{49}], [P_1, P_{13}, P_{34}]\}$.

*Corresponding author, email: mbayir@microsoft.com

WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2889351.

However, none of the previous heuristics [5] including Smart-SRA [2] are capable of extracting both.

To overcome the problems of Smart-SRA and previous approaches, we propose a new technique, called as Complete Session Reconstruction Algorithm (C-SRA). C-SRA is very powerful algorithm which produces complete set of maximal paths that can be obtained from given page request sequence and web topology. It produces sessions that better represent complex navigation behavior where user goes and backs between web pages and jumps to different nodes in the topology of given web site.

The next section describes the details of C-SRA. Next, the pattern discovery on top of C-SRA sessions is explained. Finally, we present our preliminary experimental results.

2. C-SRA ALGORITHM

C-SRA is a two phased method that produces sessions as a set of all possible maximal sequences. A Maximal sequence is a path in the graph which is not subsequence[1] of any other sequence generated from the same session.

In the first phase of C-SRA, user log sequences from server logs including (IP, URL, Time) tuples are partitioned into smaller candidate sessions by using time constraints. The second phase of C-SRA constructs all maximal navigation sequences from the candidate sessions generated at the first phase. The input and outputs of second phase of C-SRA is given below:

Input: A possibly cyclic directed graph $G = (V, E)$ such that $V = \{v_1, v_2, \ldots, v_n\}$ is vertex set and $E \subset V \times V$ is a set of edges, and an ordered sequence of vertices $S = [vs_1, vs_2, \ldots, vs_k]$ where each $vs_i \in V$ (without any repetition for our problem, since the second request of the same page is always provided by the browser cache for limited time interval).

Output: Set of maximal sequences O, where each $O_j = [vs_{j1}, vs_{j2}, \ldots, vs_{jm}] \in O$ is a maximal navigation sequence that corresponds to a path in G. That is, for every pair of consecutive vertices in a sequence O_j, such as vs_{jp} and $vs_{j(p+1)}$, there exists an edge $\langle vs_{j(p)}, vs_{j(p+1)} \rangle \in E$. In addition, in order to satisfy the maximality property, there is no sequence $O_q \in O$ such that O_j is a sub-string of O_q.

The main part of the C-SRA consumes the site graph and vertex sequence (G, S) and produces the output O mentioned above. The details of each phase are given below:

Phase 1 constructs candidate session set from user page request sequence by using time thresholds (applying both

[1]subsequence relation here is same as substring relation

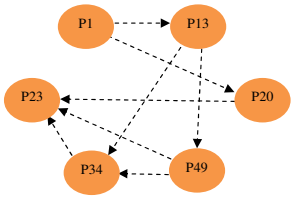

Figure 1: Topology

Page	P_1	P_{20}
Temp Sequences		([P_1]:1:F)
New Sequence	([P_1]:2:T)	([P_1, P_{20}]:1:T)
CMaximals		
Page	**P_{23}**	**P_{13}**
Temp Sequences	([P_1]:1:F) ([P_1, P_{20}]:0:F)	([P_1]:0:F)
New Sequence	([P_1, P_{20}, P_{23}]:0:T)	([P_1, P_{13}]:2:T)
CMaximals	([P_1, P_{20}, P_{23}]:0:T)	([P_1, P_{20}, P_{23}]:0:T)
Page	**P_{34}**	
Temp Sequences	([P_1, P_{13}]:1:F)	
New Sequence	([P_1, P_{13}, P_{34}]:1:T)	
CMaximals	([P_1, P_{20}, P_{23}]:0:T) ([P_1, P_{13}, P_{34}]:1:T)	

Figure 2: Execution Table

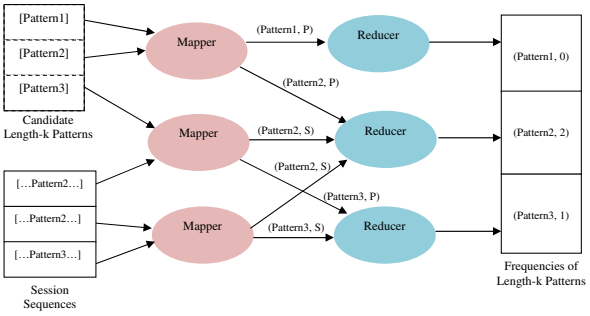

Figure 3: Frequency Calculation

total duration time constraint and limiting the time spent on a page in a session).

Phase 2 constructs all maximal navigation sequences from the candidate sessions generated at the first phase. The following features are used in this phase:

- **Maximality:** During the execution of phase two, each new sequence which is either constructed by adding a page to an existing sequence or constructed from a single page, is maximal at the beginning. A sequence becomes non-maximal if a new navigation sequence is constructed from it by adding a page to its tail.
- **Degree:** The degree of a sequence indicates how many new sequences can be constructed from it by adding new pages to its tail. Thus, the degree of a sequence is equal to the out-degree of it's last page when constructed. With the extension by appending a new page, the degree of the current navigation sequence is decreased and it is marked as non-maximal.

In second phase of C-SRA, each page in the candidate session is processed from left to right to determine if that page can expand existing navigation sequences or it can initiate a new sequence. At any particular step, if the degree of any maximal sequence decreases to zero, it is automatically added to final sequence set since there is no page remained in the candidate session to expand current sequence. After completing the processing of each page of a candidate session, remaining maximal sequences with out-degrees greater than zero are also added to the final sequence set.

Figure 2 illustrates the execution of the Phase 2 of C-SRA for the candidate session $[P_1, P_{20}, P_{23}, P_{13}, P_{34}]$ for example topology (Figure 1). In the table, each column represents the processing single page of the candidate session. The tuples in cells represent temporary variables as (sequence: degree: maximality flag). When page P_1 is processed, a new sequence containing only page P_1 is created. Since it is new sequence, it is maximal (T represent the maximality is true), and its out-degree is the out-degree of the page P_1, which is 2. When the second page, P_{20}, is processed, the sequence $[P_1]$ will be extended and a new sequence $[P_1, P_{20}]$ is generated. The new Sequence $[P_1, P_{20}]$ becomes be maximal, whereas $[P_1]$ is marked as non-maximal (degree is decreased one) and kept in the temporary sequence set. Although its out-degree is decreased, it can still be extended by using unused links. The rest of the execution is given in Figure 2.

3. PATTERN DISCOVERY

In this step, frequent navigation patterns are discovered from web user sessions [1] by using distributed version of sequential Apriori method [3] on Hadoop platform. In each step, we create candidate $length^{(k+1)}$ patterns by joining frequent $length^k$ and frequent $length^1$ locally. Then, in map/reduce job, we calculate the frequency of each candidate patterns and mark them as frequent or not (Figure 3).

4. EXPERIMENTAL RESULTS

For experimental purpose, we have implemented complete web usage mining pipeline (C-SRA + Pattern Discovery) to extract frequent patterns from the server logs of one of the largest mobile operators in Turkey. The test data includes both static and dynamic content with more than 1M page view requests daily (total data size is around 2.4M). The web site offers various sets of features ranging from purchasing different services and sending SMS online.

We compared C-SRA on page prediction application with previous heuristics [2, 4, 5]. We measured the success of next-page prediction on Bayesian Model inferred from frequent patterns. Our experiments show that C-SRA outperforms other techniques in terms of prediction success by generating better frequent patterns (Table 1).

Table 1: Comparision of Heuristics

	C-SRA	S-SRA	Navigation Oriented	Time Oriented
Success	90.40%	75.23%	65.12%	61.07%

5. REFERENCES

[1] M. A. Bayir, I. H. Toroslu, and A. Cosar. Performance comparison of pattern discovery methods on web log data. In *AICCSA 2006*, pages 445–451, 2006.

[2] M. A. Bayir, I. H. Toroslu, M. Demirbas, and A. Cosar. Discovering better navigation sequences for the session construction problem. *Data Knowl. Eng.*, 73:58–72, 2012.

[3] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[4] B. Liu, B. Mobasher, and O. Nasraoui. Web usage mining. In *Web Data Mining*, Data-Centric Systems and Applications, pages 527–603. Springer Berlin Heidelberg, 2011.

[5] B. Mobasher. Data mining for web personalization. In *The Adaptive Web*, pages 90–135, 2007.