Chapter XV

Efficient Web Mining for Traversal Path Patterns

Zhixiang Chen
The University of Texas - Pan American, USA

Richard H. Fowler
The University of Texas - Pan American, USA

Ada Wai-Chee Fu
The Chinese University of Hong Kong, Hong Kong

Chunyue Wang
The University of Texas - Pan American, USA

ABSTRACT

A maximal forward reference of a Web user is a longest consecutive sequence of Web pages visited by the user in a session without revisiting some previously visited page in the sequence. Efficient mining of frequent traversal path patterns, that is, large reference sequences of maximal forward references, from very large Web logs is a fundamental problem in Web mining. This chapter aims at designing algorithms for this problem with the best possible efficiency. First, two optimal linear time algorithms are designed for finding maximal forward references from Web logs. Second, two algorithms for mining frequent traversal path patterns are devised with the help of a fast construction of shallow generalized suffix trees over a very large alphabet. These two algorithms have respectively provable linear and sublinear time complexity, and their performances are analyzed in comparison with the a priori-like algorithms and the Ukkonen algorithm. It is shown that these two new algorithms are substantially more efficient than the a priori-like algorithms and the Ukkonen algorithm.
INTRODUCTION

Because of its significant theoretical challenges and great application and commercial potentials, Web mining has recently attracted extensive attention (e.g., Buchner et al., 1998, 1999; Catledge & Pitkow, 1995; Chen et al., 1998; Cooley et al., 1997). Surveys of recent research in Web mining can be found in Kosala and Blockeel (2000) and Srivastava et al. (2000). One of the major concerns in Web mining is to discover user traversal (or navigation) path patterns that are hidden in vast Web logs. Such discovered knowledge can be used to predict where the Web users are going, that is, what they are seeking, so that it helps the construction and maintenance of real-time intelligent Web servers that are able to dynamically tailor their designs to satisfy users’ needs (Perkowitz & Etzioni, 1998). It has significant potential to reduce, through prefetching and caching, Web latencies that have been perceived by users year after year (Padmanabhan & Mogul, 1996; Pitkow & Pirolli, 1999). It can also help the administrative personnel to predict the trends of users’ needs so that they can adjust their products to attract more users (and customers) now and in the future (Buchner & Mulvenna, 1998).

Traversal path pattern mining is based upon the availability of traversal paths that must be obtained from raw Web logs. A maximal forward reference of a Web user, a longest consecutive sequence of Web pages visited by the user without revisiting some previously visited page in the sequence, is a typical traversal path pattern (Chen et al., 1998; Cooley et al., 1999). This chapter studies the problem of efficient mining of frequent traversal path patterns that are large sequences of maximal forward references. The previously known algorithms for the problem are the two a priori-like algorithms FullScan and SelectiveScan designed in Chen, Park and Yu (1998).

This chapter aims at designing algorithms for mining traversal path patterns with the best possible efficiency. The main contributions are summarized as follows. First, two algorithms, ISMFR (Interval Session Maximal Forward References) and GSMFR (Gap Session Maximal Forward References), are given for finding maximal forward references from raw Web logs. The first algorithm is designed for interval sessions of user accesses and the second for gap sessions. The two algorithms have linear, hence optimal, time complexity, and are substantially more efficient than the sorting based method devised in Chen, Park and Yu (1998) for finding maximal forward references. Second, the problem of mining frequent traversal path patterns from maximal forward references is investigated with the help of a fast construction of shallow generalized suffix trees over a very large alphabet (Chen et al., 2003a, 2003b). Precisely, a shallow generalized suffix tree is built for a set of maximal forward references obtained by the algorithm ISMFR (or GSMFR) such that each tree node contains the frequency of the substring represented by that node. Once such a tree is built, a simply traversal of the tree outputs frequent traversal path patterns, that is, frequent substrings, with respect to some given frequency threshold parameter. Two algorithms, SbSfxMiner (Sorting Based Suffix Tree Miner) and HbSfxMiner (Hashing Based Suffix Tree Miner), are designed to overcome some well-understood obstacles of suffix tree construction (Gusfield, 1997, pp. 116-119): the complexity of suffix tree construction is dependent on the size of the underlying alphabet and a suffix tree does not have nice locality properties to support memory paging. These two algorithms, SbSfxMiner and HbSfxMiner, have respectively provable sublinear and linear time complexity. Performances of these two algorithms are analyzed in comparison with the two a priori-like algorithms FullScan and SelectiveScan in Chen, Park and Yu.
Related Content

**PowKMeans: A Hybrid Approach for Gray Sheep Users Detection and Their Recommendations**
[www.irma-international.org/article/powkmeans/198359/](www.irma-international.org/article/powkmeans/198359/)

**Secure Online DNS Dynamic Updates: Architecture and Implementation**
[www.irma-international.org/article/secure-online-dns-dynamic-updates/2630/](www.irma-international.org/article/secure-online-dns-dynamic-updates/2630/)

**A Survey of Web Service Discovery Systems**
[www.irma-international.org/chapter/survey-web-service-discovery-systems/5037/](www.irma-international.org/chapter/survey-web-service-discovery-systems/5037/)

**Pen Testing for Web Applications**
[www.irma-international.org/article/pen-testing-web-applications/72989/](www.irma-international.org/article/pen-testing-web-applications/72989/)

**Productivity Evaluation of Self-Adaptive Software Model Driven Architecture**
[www.irma-international.org/chapter/productivity-evaluation-self-adaptive-software/72762/](www.irma-international.org/chapter/productivity-evaluation-self-adaptive-software/72762/)