CS-Mine: An Efficient WAP-Tree Mining for Web Access Patterns

Baoyao Zhou, Siu Cheung Hui, and ACM Fong

School of Computer Engineering, Nanyang Technological University, Singapore
zhouby@pmail.ntu.edu.sg, {asschui, ascmfong}@ntu.edu.sg

Abstract. Much research has been done on discovering interesting and frequent user access patterns from web logs. Recently, a novel data structure, known as Web Access Pattern Tree (or WAP-tree), was developed. The associated WAP-mine algorithm is obviously faster than traditional sequential pattern mining techniques. However, WAP-mine requires re-constructing large numbers of intermediate conditional WAP-trees during mining, which is also very costly. In this paper, we propose an efficient WAP-tree mining algorithm, known as CS-mine (Conditional Sequence mining algorithm), which is based directly on the initial conditional sequence base of each frequent event and eliminates the need for re-constructing intermediate conditional WAP-trees. This can improve significantly on efficiency comparing with WAP-mine, especially when the support threshold becomes smaller and the size of database gets larger.

1 Introduction

Web usage mining [1] discovers interesting and frequent user access patterns from web usage data that can be stored in web server logs, proxy logs or browser logs. Essentially, a web access pattern [2] is a sequential pattern in a large set of pieces of web logs, which is pursued frequently by users. Most of the previous studies for web access patterns mining have adopted sequential pattern mining techniques [3], such as AprioriAll [3] and GSP [5]. However, these Apriori-based algorithms encounter the same problem that requires expensive multiple scans of database in order to determine which candidates are actually frequent. Recently, Pei *et al.* [2] proposed a compressed data structure known as Web Access Pattern Tree (or WAP-tree), which facilitates the development of algorithms for mining web access patterns efficiently from web logs. The associated WAP-mine algorithm [2] avoids the problem of generating explosive numbers of candidates. Experimental results have shown that WAP-mine is obviously faster than traditional sequential pattern mining techniques. However, the conditional search strategy in WAP-mine requires re-constructing large numbers of intermediate conditional WAP-trees during mining, which is also very costly.

In this paper, we propose a new web access pattern mining algorithm based on WAP-tree structure, known as CS-mine (Conditional Sequence mining algorithm). To improve efficiency, CS-mine eliminates the need for re-constructing intermediate conditional WAP-trees. The rest of this paper is organized as follows. In Section 2, we introduce the related work on the WAP-tree structure and WAP-tree based mining algorithms. The proposed CS-mine algorithm is presented in Section 3. Section 4 shows the experimental results. Finally, the conclusion is given in Section 5.

2 WAP-Tree and Mining Web Access Patterns

Generally, web logs can be regarded as a collection of sequences of access events from one user or session in timestamp ascending order. Preprocessing tasks [4] can be applied to the original log files to obtain web access sequences after data cleaning, user identification, session identification, etc., for mining purposes. In this section, we review the WAP-tree structure and the related WAP-tree based mining algorithms.

2.1 WAP-Tree Structure and Construction

Let E be a set of access events, which represents web resources accessed by users, i.e. web pages, URLs. A web access sequence $S = e_1e_2\ldots e_n$ ($e_i \in E$ for 1 i n) is a sequence of access events, and $|S| = n$ is called the *length* of S. Note that it is not necessary that e_i e_j for i j in S, that is repeat of items is allowed. For example, suppose we have a set of web access sequences with the set of access events $E = \{a, b, c, d, e, f\}$. A simple web access sequence database is shown in Table 1.

Table 1. A database of web access sequences

User ID	Web Access Sequence	Frequent Sub-sequence
100	*cadba*	*caba*
200	*cacbeae*	*cacba*
300	*ceafbab*	*cabab*
400	*cfcabfa*	*ccaba*

A web access sequence $S' = e_1'e_2'\ldots e_m'$ is called a *sub-sequence* of $S = e_1e_2\ldots e_n$, denoted as $S' \subseteq S$, if there exists some i, 1 $i_1 < i_2 < \ldots < i_m$ n, such that $e_j' = e_{i_j}$ for 1 j m. In $S = e_1e_2\ldots e_k\ e_{k+1}\ldots e_n$, $S_{prefix} = e_1e_2\ldots e_k$ is called a *prefix sequence* of S, or a *prefix sequence* of e_{k+1} in S, and $S_{suffix} = e_{k+1}e_{k+2}\ldots e_n$ is called a *suffix sequence* of S or a *suffix sequence* of e_k in S. A web access sequence can be denoted as $S = S_{prefix} + S_{suffix}$. For example, S = *cadba* = *c*+*adba* = *ca*+*dba* = … = *cadb*+*a*. Let S_1 and S_2 be two prefix sequences of e_i in S, and S_1 is also the prefix sequence of e_i in S_2. Then S_1 is called the *sub-prefix sequence* of S_2 and S_2 is the *super-prefix sequence* of S_1. The prefix sequence of e_i in S without any super-prefix sequence is called the *long prefix sequence* of e_i in S. For example, if S = *cadba*, then $S_1 = c$ is the sub-prefix sequence of S_2 = *cadb* and S_2 is the super-prefix sequence of S_1. S_2 is also the long prefix sequence of a in S. $WAS_{DB} = \{S_1, S_2, \ldots, S_m\}$ is a given a web access sequences database, and $|WAS_{DB}| = m$. The *support* of S in WAS_{DB} is defined in equation (1).

$$\sup(S) = \frac{|\{S_i \mid S \subseteq S_i, S_i \in WAS_{DB}\}|}{|WAS_{DB}|} \quad (1)$$