Chapter 7.29 Discovering Frequent Embedded Subtree Patterns from Large Databases of Unordered Labeled Trees

Yongqiao Xiao
SAS Inc., USA

Jenq-Foung Yao
Georgia College & State University, USA

Guizhen Yang
University at Buffalo, State University of New York,USA

ABSTRACT

Recent years have witnessed a surge of research interest in knowledge discovery from data domains with complex structures, such as trees and graphs. In this paper, we address the problem of mining maximal frequent embedded subtrees which is motivated by such important applications as mining "hot" spots of Web sites from Web usage logs and discovering significant "deep" structures from tree-like bioinformatic data. One major challenge arises due to the fact that embedded subtrees are no longer ordinary subtrees, but preserve only part of the ancestor-descendant relationships in the original trees. To solve the embedded subtree mining problem, in this article we propose a novel algorithm, called TreeGrow, which is optimized in two important respects. First, it obtains frequency counts of root-to-leaf paths through efficient compression of trees, thereby being able to quickly grow an embedded subtree pattern path by path instead of node by node. Second, candidate subtree generation is highly localized so as to avoid unnecessary computational overhead. Experimental results on benchmark synthetic data sets have shown that our algorithm can outperform unoptimized methods by up to 20 times.

INTRODUCTION

Tree mining has been applied to Web usage mining, mining semi-structured data and bioinformatics, and so forth. The focus of this article is on mining maximal frequent embedded subtrees from unordered labeled trees. As a motivating example, consider mining the Web logs at a particular Web site. Several types of traversal patterns — for example, sequential patterns and maximal traversal patterns — have been proposed to analyze the browsing behavior of the user (Lan, Bressan, & Ooi, 1999; Mannila, Toivonen, & Verkamo, 1995; Pei, Han, Mortazavi-Asl, & Zhu, 2000; Spiliopoulou 2000; Chen, Park, & Yu, 1998; Xiao & Dunham 2001). They are all one-dimensional patterns. One drawback of such one-dimensional traversal patterns for the Web logs is that the document structure of the Web site, which is essentially hierarchical (a tree) or a graph, is not well captured.

In this article, we uncover the maximal frequent embedded subtree structures from the access sessions. The access sessions are regarded as trees instead of sequences. The trees are assumed to be unordered (the order of the child nodes does not matter), because we think when analyzing the user's browsing from a Web page (e.g., home page) on a Web server, it would be more interesting to know which pages the user follows from the starting page, regardless of the order of the access. The frequent subtrees are embedded subtrees (ancestor/descendant relationship is kept in the subtree) and maximal (it is not a subtree of another frequent subtree). Such embedded subtrees are able to uncover patterns hidden within the trees, which might be missed by induced subtrees (only parent/child relationship is allowed) (Xiao, Yao, Li, & Dunham, 2003). The maximality of subtrees can reduce the number of meaningful patterns. Notice that we do not intend to imply that other types of frequent subtrees are not or less important.

Also in this article, we propose a novel algorithm, TreeGrow, to discover all such maximal frequent subtrees given some minimum support threshold. It uses compression of the trees to allow localized candidate subtree generation. An optimization method is further proposed to efficiently count the frequency of the candidate subtrees.

The rest of the article is organized as follows. In the next section the tree mining problem is formally defined, followed by a description of the TreeGrow algorithm. We then report the experimental results, related work is described, and we conclude the article with future work.

PROBLEM STATEMENT

A *tree* is defined as $T = <N,B,r,L>$, where N is the set of nodes, B is the set of branches (directed edges), $r \in N$ is the root of the tree, and L is the set of labels on the nodes. For each branch, where $<n_1, n_2> \in N$, we call n_1 the parent of n_2, and n_2 a child of n_1. On each node $n_i \in N$, there is a label $l_i \in L$. The labels in a tree could be unique, or duplicate labels are allowed for different nodes. In both cases, the label of a node is given by a function, $l : N \rightarrow L$. Without loss of generality, the labels are represented by positive integers.

Paths and Root Paths

A *path* is a sequence of connected nodes $<n_1, n_2, \ldots, n_k>$, where $<n_i, n_{i+1}> \in$ B ($1 \leq i < k$), and k is the number of nodes on the path. A node u is called an ancestor of another node v if there is a path from u to v in the tree. Correspondingly, v is called a descendant of u. A path starting from the root node is called a *root path*. Each root path in a tree can be uniquely identified by the last node on the path in the tree. In Figure 1, node n_3 represents the root path $<n_1, n_2, n_3>$, and the labels on the path are <1,2,3>.

v-path and v-node

A *v-path* (virtual path) is a sequence of nodes $<n_1, n_2, \ldots, n_k>$, where n_i is an ancestor of n_{i+1} ($1 \leq i < k$), that is, two adjacent nodes on a *v-path* are not necessarily directly connected. By this definition, all paths are *v-paths*. A *root v-path* is similarly a *v-path* staring at the root. Notice that a *root v-path* might not be uniquely represented by the last node on it, since it is likely to have two *root v-paths*