Expert Systems with Applications 42 (2015) 94–100

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Research on a frequent maximal induced subtrees mining method based on the compression tree sequence

Jing Wang, Zhaojun Liu, Wei Li, Xiongfei Li *

College of Computer Science and Technology, Jilin University, China

ARTICLE INFO

Article history:
Available online 9 August 2014

Keywords:
Data mining
Frequent subtree
Induced subtree
Maximal subtree
Compression
CFMIS

ABSTRACT

Most complex data structures can be represented by a tree or graph structure, but tree structure mining is easier than graph structure mining. With the extensive application of semi-structured data, frequent tree pattern mining has become a hot topic. This paper proposes a compression tree sequence (*CTS*) to construct a compression tree model; and save the information of the original tree in the compression tree. As any subsequence of the *CTS* corresponds to a subtree of the original tree, it is efficient for mining subtrees. Furthermore, this paper proposes a frequent maximal induced subtrees mining method based on the compression tree sequence, CFMIS (compressed frequent maximal induced subtrees). The algorithm is primarily performed via four stages: firstly, the original data set is constructed as a compression tree model; then, a cut-edge reprocess is run for the edges in which the edge frequent is less than the threshold; next, the tree is compressed after the cut-edge based on the different frequent edge degrees; and, last, frequent subtree sets maximal processing is run such that, we can obtain the frequent maximal induced subtree set of the original data set. For each iteration, compression can reduce the size of the data set, thus, the traversal speed is faster than that of other algorithms. Experiments demonstrate that our algorithm can mine more frequent maximal induced subtrees in less time.

1. Introduction

Structures of data are becoming increasingly complicated with the fast development of the Internet and storage technology. Most data with a complicated structure can be represented by a tree or graph structure. With the extensive application of semi-structured data, the research priority of frequent pattern mining has expanded from frequent item set mining (Liu, Lin, & Han, 2011; Wang & Chen, 2011; Yang & Huang, 2010) to frequent subtree mining (Balcázar, Bifet, & Lozano, 2010; Li, Li, & Zhao, 2010) and frequent subgraph mining (Hou, Ong, Nee, et al., 2011; Jiang, Coenen, & Zito, 2013). The complexity of tree mining is lower than that of graph mining, and tree mining algorithms can be applied to graph mining instances that contains a small amount of rings, so it is of great significance to be able to mine data represented by a tree structure. Frequent subtree mining has become an important field of data mining research.

Frequent subtree mining is the process of mining a subtree set from a given data set that satisfies user attention (support or frequent degree). Frequent subtree mining has a high value in computer vision, text acquisition, Web log analysis, XML document analysis, XML association rule mining, XML query pattern mining territory, semi-structured data analysis, analysis of biometric information and structural analysis of compounds. For example, through application of the frequent subtree mining method to web logs, users' degree of interest can be known by deep analysis of the information represented in a tree structure, and it is convenient to optimize the structure of the network. In analyzing the XML document, the frequent subtree mining method can find a frequent data structure that is implicit and represented by a tree structure.

When mining frequent subtrees on a given tree data set, the number of frequent subtrees increases exponentially with the decrease of the minimum degree, and the frequent subtrees in the result data set contains redundant information, so simplifying the result set is necessary. Finding the closed subtrees and the largest subtrees are two common methods to simplify the result set. Closed subtree T can express information that all other subtrees closed by T can express, so subtrees closed by T can be deleted from the result set. Recently, there has been great interest in mining closed subtrees, and many efficient closed subtree mining algorithms have been proposed. Maximal subtree T' can express information that all other subtrees maximized by T' can express, so the number of subtrees in the result set can be minimized. This is significant for the growth of large-scale data.

* Corresponding author.

http://dx.doi.org/10.1016/j.eswa.2014.07.053
0957-4174/© 2014 Elsevier Ltd. All rights reserved.

In this paper, we propose an efficient method, CFMIS, based on the compression tree sequence that focuses on mining frequent maximal induced subtrees. The algorithm is primarily performed via four stages: firstly, the original data set is constructed as a compression tree model; then, a cut-edge reprocess is run for the edges in which the edge frequent is less than the threshold; next, the tree is compressed after the cut-edge based on the different frequent edge degrees; and, last, frequent subtree sets maximal processing is run such that, we can obtain the frequent maximal induced subtree set of the original data set. We have demonstrated experiment that the proposed algorithm in this paper can mine frequent maximal induced subtrees in a rapid and efficient way.

2. Related work

Methods for mining frequent subtrees are classified into mining methods based on generation–test strategy and mining methods based on pattern growth strategy. The main idea of mining method based on generation–test strategy is to produce the corresponding candidate subtree first, and then traverse the corresponding data set to test whether the candidate subtree is frequent. This strategy is mainly used to expand or merge subtrees to generate a new candidate subtree, and then to test whether the new candidate subtree is frequent or not by traversing the database. Mining methods based on pattern growth strategy iterate through the database to find the frequent tree extension points through repeated search, until mining out all hidden frequent subtrees.

EvoMiner (Deepak, Fernández-Baca, Tirthapura, et al., 2011) is an Apriori-like level-wise method, which uses a novel phylogeny-specific constant-time candidate generation scheme, a fingerprinting based technique for downward closure, and a lowest-common-ancestor-based support counting step. Bui, Hadzic, and Tagarelli et al. introduce an associative classification method (Bui, Hadzic, Tagarelli, et al., 2014) based on a structure preserving flat representation of trees in which subtrees are constrained by the position in the original trees, leading to a drastic reduction in the number of rules generated, especially with data that has great structural variation among tree instances. Nguyen, Doi, and Yamamoto propose a new top-down method (Nguyen, Doi, & Yamamoto, 2012) for mining unordered closed tree patterns from a database of trees such that every mined pattern must contain a common piece of information in the form of a tree specified by the user. Lee and Lee introduce a new type of problem called the frequent common family subtree mining problem (Lee & Lee, 2013) for a collection of leaf-labeled trees in their paper and present some characteristics for the problem. It proposes an algorithm to find frequent common families in trees. Nguyen and Yamamoto propose a novel and efficient incremental mining algorithm (Nguyen & Yamamoto, 2010) for closed frequent labeled ordered trees. They adopt a divide-and-conquer strategy and apply different mining techniques in different parts of the mining process. The algorithm requires no additional scan of the entire database. PTG (Li & Yang, 2011) (partition tree growth) is put forward based on the partition principle. In the PTG algorithm, the database is divided into several partitions, the TG (tree growth) algorithm creates the local frequent subtrees of every partition, and then creates the global frequent subtrees according to the global support value for filtering. Deng, Lv proposed Nodeset (Deng & Lv, 2014), a novel structure where a node is encoded only by pre-order or post-order code to solve the memory-consumption problem. Xiao and Yao proposed the classic PathJoin (Xiao & Yao, 2003) algorithm based on the Apriori algorithm to effectively implement mining of maximal frequent subtrees. The algorithm uses a compact data structure called FST-Forest, which compresses the trees and retains the original tree structure. PathJoin generates candidate subtrees by joining the frequent paths in FST-Forest.

MFPTM (Wu & Li, 2011) constructs an MP[1] tree based on fusion compression and the FP[2] tree principle to mine maximal frequent subtrees. MFPTM is an advanced algorithm as it solves the problem of frequent pattern mining based on the Apriori algorithm which generates a large quantity of candidate patterns and improves the efficiency of mining frequent subtrees. MFPTM outperforms the classic algorithm PathJoin. The proposed algorithm, CFMIS, and the state-of-the-art MFPTM both focus on frequent maximal subtrees, not just frequent subtrees. Furthermore, the two algorithms both retain subtrees that only contain frequent nodes by compression, although the compression methods are different. Therefore, the two algorithms are compared by experiments on both synthetic and real datasets.

3. CFMIS algorithm

In this section, we provide the definitions for some general and specific concepts that will be used in the remainder of the paper. We also give the details of our algorithm.

3.1. Prepared knowledge

A tree is generally defined as an acyclic connected graph, and they can be classified according to their structural characteristics. If sibling nodes of tree T are ordered, the tree T is called an ordered tree; otherwise, it is known as an unordered tree. If the nodes in the tree contain labels, the tree T is called a label tree, otherwise, it is known as a non-label tree. If the sibling nodes of the same parent node have no repeats, the tree T is called an attribute tree, otherwise, it is known as a non-attribute tree. An unordered label attribute tree is denoted as *ULAT*,

Definition 1 (*ULAT*). *An* unordered tag attribute tree is an acyclic connected graph, which is denoted as $ULAT = (V, E, \Sigma, L, r)$, where V is the node set; E is the edge set in which $(x, y) \in E$ represents that node x is the parent of node y; Σ is the label set in which elements can be compared and sorted; L is the mapping from the node set to label set, $L : V \rightarrow \Sigma$, and sibling nodes of the same parent node without the same label; and r is the root node.

The CFMIS algorithm addresses unordered label attribute tree sets, and the 'tree' mentioned below is *ULAT*, assuming that there is no repeat label in a same tree.

Definition 2 (*Induced subtree*). A tree $T' = (V', E', \Sigma', L', r')$ is an induced tree of $T = (V, E, \Sigma, L, r)$, denoted as $T' \subset T$, if and only if $V' \subset V$; $E' \subset E$; $\Sigma' \subset \Sigma$; $L' \subset L$.

Fig. 1 shows an induced tree of a source tree.

Reserving parent–child relationships between nodes in the source tree and the absence of affection among sibling nodes are features of induced trees. The CFMIS algorithm addresses the induced trees of an original data set.

Definition 3 (*Frequent subtree*). Let the tree structure data set be $D = \{T_1, T_2, \ldots, T_n\}$. ε is the minimum frequency threshold, $T' \subset T_i$, where $i \in [1, n]$, $T_i \in D$. $Occ(T, T')$ represents whether T' occurs in T, if T' occurs in T, then $Occ(T, T') = 1$, and else $Occ(T, T') = 0$. The frequency of T' is denoted as $Frq(T')$, and $Frq(T') = \sum_{i=1}^{n} Occ(T, T')$. T' is a frequent tree if and only if $Frq(T') \geqslant \varepsilon$.

Definition 4 (*Maximal subtree*). Let the tree structure data set be

[1] MP (maximal path) tree is proposed in reference Wu and Li (2011), each path from the root node to a leaf node in an MP tree is frequent.

[2] FP (frequent pattern) tree, constructs different branches of an FP tree by traversing each item in the transaction dataset.