Research on Algorithm of Frequent Subtree Mining

Xin Guo[1], ChangGuo Xiang[1]

[1] Key laboratory for ecotourism of Hunan province Jishou University

hunter2011@foxmail.com

Keywords: Data Mine; Frequent Subtree; Frequent Subgraph; Ordered Tree

Abstract. With the development of computer and information technology, frequent mine become important researching value, from subset mine to structured data mine, it includes frequent sub tree mining and frequent sub graph mining. In this paper, we introduces frequent sub tree mining and describes research status, and analyzes ordered tree mine algorithm and unordered tree mine algorithm. We achieve the algorithm and compare the efficiency of the tree algorithm.

Introduction

Frequent itemsets are many data mining tasks of the key issues is the core algorithm of association rule mining. It is widely used in mining association rules, sequence mining algorithms. Following frequent subtree mining is another important research direction after frequent item set mining, compared with collection and sequences, since the tree structure better able to express the relationship between things, so more and more frequent subtree mining causes people's attention, especially those with a very high value in the field of bioinformatics, Web structure, XML data and the like. Such as ribonucleic acid (RNA) in organisms can be expressed as the molecular structure of the tree structure [1], the researchers in order to get a new RNA-related information, it must be aware of the RNA structure with which to compare, with the same search topology, thereby obtaining the function information of the new RNA. Also users can visit a Web site set represented as a tree structure, access history of all users constitutes a tree database, then you can find the most frequent users of the site access mode for easy website design personnel adjustments Web site architecture, website structure optimization.

In other countries, early in 1998, K.Wang [2], who proposed the unordered tree rooted in centralized mining frequent tree algorithm derived formula, 2002 Zaki [1] and Asai [3], who proposed separately based technology to extend the rightmost ordered frequent subtree mining algorithm: TreeMiner and FREQT. 2003 Y.chi, Y.Yang and RR Muntz [4] proposed a disorderly tree mining algorithm: Free Tree Miner, proposes a disorderly trees standardized methods and standardized methods FreeTree, and in 2004 years and presents a hybrid algorithm: HybridTeeMiner [5], can tap Unordered tree and Free tree. In 2004 Y.chi, Y.Yang and RR Muntz proposed mining algorithm largest enclosed frequent subtrees: CMTreeMiner [6]. In China, 2003 Pan Jin et al proposed Chopper [7] algorithm and 2004 Zhu Yongtai et al proposed ESPM [8] algorithm, Zhao Chuanshen others on the basis of ESPM algorithm proposed FTPB in 2005 [9] algorithm , it is the use of sequence mining algorithms to improve efficiency of the algorithm.

Basic concept

Definitions 1. Rooted ordered labeled tree (Ordered Tree). A rooted ordered labeled tree T = (V, E) is a root node having a label and directed acyclic graph, where V represents the tree node set, E = {(x, y) | x, y V} represents the set of edges in the tree, with L = {l0, l1, ..., ln} represents a label set, node V and L there is a correspondence between l: V → L, li called node label. Ordered sub-tree means that for each node in the tree, all of its child nodes from left to right in the order form sibling relationships.

The definition of the disorder rooted labeled tree (Unordered Tree). A disorder rooted labeled tree T = (V, E) is a root node has a label and a directed acyclic graph, where V represents the tree node set, E = {(x, y) | x, y V} represents the set of edges in the tree, with L = {l0, l1, ..., ln} represents a label set, node V and L there is a correspondence between l: V $\rightarrow$ L, li called node label. Disorder refers to each node in the tree, all of its child nodes sequential relationship does not exist around.

Definition of unrooted unordered labeled tree (FreeTree). A tree without roots disorder label T = (V, E) is a root of having no clear reference directed acyclic graph, where V represents the tree node set, E = {(x, y) | x, y V} represents the set of edges in the tree, with L = {l0, l1, ..., ln} represents a label set, node V and L there is a correspondence between l: V $\rightarrow$ L, li called node label.

Definitions 4. The tree node relationship [8] Given a tree root is vo T, consider the path from the beginning of any vo: If u v in front, then u is the ancestor of v, v is u of sons, if u, v only between one edge, then u is the father of v, v is u child. This side is called the branch, denoted by b = (u, v). If some nodes have the same father, so these nodes are brothers.

Definition 5. Given the frequent subtrees ordered labeled tree database TDB and sub-tree T (introns tree, export sub-tree), support for the definition of T s (T) = | p (T) / N |, where p (T) is contained in T TDB tree trees number, N is the number of trees trees in TDB. When the support T s (T) $\geqslant$minsup, minsup$\in$ [0,1], called T as TDB in frequent subtree. Wherein, minsup support for the user-specified threshold.

Frequent subtree mining is given in the user minsup case, in the tree database TDB, the TDB in digging out all frequent subtrees.

Frequent Subtree Mining Algorithm

Depending on the type of the tree, frequent subtree mining algorithm can be divided into: a rooted ordered tree (Ordered Tree) mining algorithms disorder rooted tree (Unordered Tree) mining algorithms and unrooted tree disorder (FreeTree) *mining* algorithms.

Ordered Tree. Asiai of FREQT [3] algorithm and Zaki's TreeMiner [1] algorithm is rooted ordered tree mining representatives algorithm, both algorithms are based on the far right to extend the generated candidate sub tree, and then calculate the number of each candidate subtree support to give all frequent subtrees according minsup. Although both candidates are generated based on the rightmost expansion, but the former is carried all the rightmost frequently extend a node for each sub-tree generate candidate frequent K K + 1 sub-tree, which is carried out for each of the frequent subtree K generating a candidate merger between two K + 1 sub-tree, which generates a candidate space smaller than the former, it is possible to improve the operational efficiency of the algorithm in a certain program.

This algorithm can have an ordered tree roots dig out all frequent intron trees, and offers two counts each candidate subtree supports several methods: the weight and non-weight count count. Algorithm, a new data structure (scope-list) to represent the various sub-tree, the data structure can be shown in detail the position of each sub-tree appears in the database and the number of times the message appears to facilitate the implementation of mining algorithms. A candidate generation algorithm strategy, that generate pairwise merger candidate subtrees according to all frequent subtrees, to facilitate the candidate generation algorithm a data structure: equivalence class (Equivalence Classes).

TreeMiner algorithm in the tree is a tree of the string representation (Definition 1), this data structure is simple, easy to implement and can uniquely represent a tree, can improve the design efficiency of the algorithm. Tree node using a tuple SX = (lx, μx) uniquely, where lx represents X number of nodes, μx node number represents the rightmost leaf node to node X as the root of the subtree , node ID number for each node at the depth priority access.

In algorithm with a new data structure (scope-list) to represent the information of each sub-tree, this data structure are some triples (t, m, s) formed, where t represents the sub-tree in the forest where the trees (database) number, m represents primer prefix tree, s represents the rightmost leaf node of this subtree SX in the original tree.

For all of the same sub-tree prefix (after removing the rightmost sub-tree leaf node remaining node string representation) consisting of a sub-tree forest called equivalence classes (Equivalence Classes). In the equivalence class, for each sub-tree leaves rightmost edge connector available a tuple (X, i) to indicate, where X represents the rightmost node label leaf node, i indicates the connection node No. rightmost path sequence, as in all sub-equivalence class tree, only the rightmost leaf node is different, so the equivalence class can be used to uniquely represent this tuple, call these binary group element.

Candidate generation algorithm strategy is based on the rightmost leaf node expansion strategy by the equivalence class (Equivalence Classes) in all sub-tree pairwise connection, according to the following theorems to generate candidate subtrees:

Theorem 1 [1]. Equivalence class extension.

For a prefix P k-1 sub-tree into an equivalent class prefix, in which the presence of any two Element: (x, i) and (y, j). Define a join operation , denoted by (x, i) (y, j), the following rules on the two Element: case 1 (i = j), a) If P is not empty, the (y, j) rightmost leaf node and (y, ni) added to the equivalence class subtree (x, i) represented, can form two trees candidate subtrees, where ni (x, i) represented by the subtree The depth-first number. subtree b) If P is not empty, the (y, j + 1) added to (x, i) represented go. Case 2 (i> j), the (y, j) added to the (x, i) represented by the sub-tree go. Case 3 (i> j), no candidate subtree.

TreeMiner algorithm scans the database frequently get a sub-tree and frequent two subtrees, and get an equivalence class, then in this equivalence class of Element generate pairwise merger candidate subtrees and calculate the number of candidate sub-tree support and recursively This procedure generates all the equivalence classes until no equivalence class generation, then you can get the database of all frequent subtrees, pseudo code algorithm is described as follows:

```
Algorithm 1[10] Procedure TreeMiner
Input: Ordered labeled tree databaseD，Minimum support minsup
Output: All frequent subtrees
1:   TREEMINER(D,minusp) :
2:   F1={frequent 1-subtrees};  F2={classes[P]1 of frequent 2-subtrees};
3:   For all [P]1 ∈E do Enumerate-frequent-subtrees([P]1);
4:   Enumerate-frequent-subtrees([P]):
5:   For each element (x,i) ∈[P] do
6:       For each element (y,j) ∈[P] do
7:           R={(x,i) ⊕(y,j)};
8:           L(R)={ L(X,i) ⊕L(Y,j)};
9:           If for any R∈R,R is frequent then
10:              [N]=[N]∪{R};
11:      Enumerate-frequent-subtrees([N]);
```

Unordered Tree and Free Tree. Y.chi, Y.Yang and RR Muntzr of HybridTreeMiner mining algorithm is rooted and unrooted tree disorderly disorderly representative tree algorithm. In order to facilitate the process of unordered tree matching subtree mining operations, disorderly tree mining algorithm differs from the ordered tree mining algorithm, the algorithm first need to standardize conversion disorder tree, will be converted into disorderly ordered tree after tree mining operation.

Y.chi et al proposed a string encoded using breadth-first way to represent a tree, with two special characters "$" and "#" to represent the hierarchical structure of the tree, and if "#" is greater than "$" and greater than lexicographic tree node label and define the width of the smallest priority string encoded as standard encoding (BFCS: breadth-first canonical string), corresponding to the tree as a standard tree. From the bottom to the top through a process to build a standard tree, the first node to the bottom of the order according to the dictionary to standardize small to large, does not meet the requirements of the exchange of two nodes, then one layer of nodes normalized until the top node (the root), then the tree has become a standard tree structure.

From Definition 3 shows, FreeTree no established root, but can lead to a series of procedures to construct a root FreeTree will FreeTree into a disorderly tree and standardize operations, and then in a standard tree mining operations, so it can be FreeTree excavation into Unordered Tree of excavation work.

First of all the leaf nodes and edges connecting FreeTree removed, repeat this process until FreeTree only one node or two nodes, the situation for the next first last only one node, He said FreeTree centered tree [4], in the case of the second last only two nodes, called FreeTree double heart tree [4]. For all FreeTree only the existence of these two cases.

HybridTreeMiner algorithm can solve FreeTree and Unordered Tree excavation work, this algorithm is similar with TreeMiner algorithms are generating two expansion strategy based on two candidates, the candidate subtree count is similar, except that the algorithm needs to resolve without HybridTreeMiner standardization order tree.

Test Analysis

This paper analyzes tree miner algorithm time complexity, the algorithm in the calculation frequently a sub-tree, you need to scan the entire database, for any of a tree, then the time complexity is O (n), n is the number of nodes in the tree, Similarly, in the calculation of the frequent two subtrees, for any of a tree time complexity is O (n2), n is a tree of nodes. K for seeking frequent subtree, if an equivalence class [P] there are n Element, because in seeking a candidate subtrees need to intersect pairwise Element ((x, i) (y, j)), and At the same time you want to operate cross scope-list (L (X, i) L (Y, j)), and therefore seek equivalence under this category to frequent K subtrees time complexity is O (an2), where a is the L (X, i) L (Y, j) the time required.

In this paper, a C ++ implementation of the program and TreeMiner random tree generation algorithm, and TreeMiner algorithm under different parameters were a lot of experiments, experiments show that the algorithm is efficient and feasible higher. Experimental environment for Pentium (R) 4 cpu 2.80GHz, memory is 512MB, operate as Windows XP, experimental platform is vc6.0. In this paper, the random tree generation program to generate tree database, this random tree generation program can dynamically set various parameters (tree database size, height of the tree, the tree node fan-out, etc.) to control the random tree complexity. First, in the same circumstances generated parameters (tree height of 7, fan-out of 6) in size from 10,000 to 50,000 in five tree database, by default the same support threshold value of 1% of the cases, the implementation of mining program, the experimental results Fig 1:

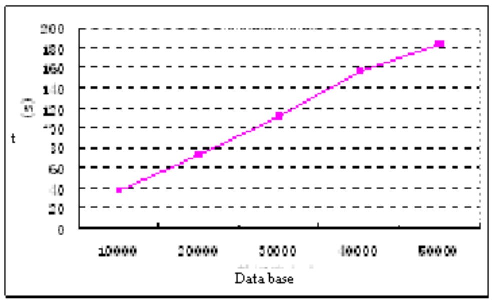

Fig. 1 Run time with the size of the database changes

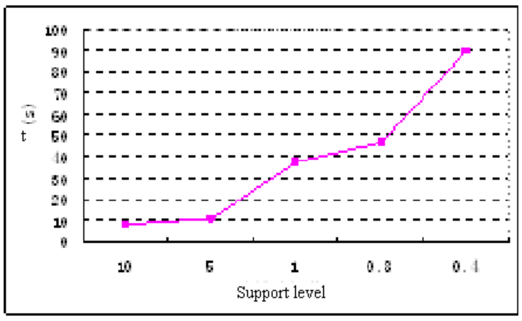

Fig. 2 Uptime with support threshold conversion

Then different support threshold experiment, random tree generator parameters using tree size is 10000, tree height of 7, 6 fanout, in support threshold varies from 10% to 0.4% The results shown in Figure 10:

Tree algorithm also a factor, the paper in the case of different tree height and fan-out by experiment, this experiment is divided into two small experiments, each of the small experiment in six tasks, each task allocation and the number of frequent subtree experiments produced the following table :(parameter setting is expressed as (tree height, fan-out))

Table 1 Task Allocation Table

Task	Change of height	Num of Freq tree	Change of tree fan	Num of Freq tree
1	(5,6)	3341	(7,3)	493
2	(6,6)	4552	(7,4)	3111
3	(7,6)	11069	(7,5)	5963
4	(8,6)	44596	(7,6)	12005
5	(9,6)	115429	(7,7)	40912
6	(10,6)	454132	(7,8)	132193

Conclusion

This article discusses the basic concepts and common algorithms of frequent subtree mining, based on the characteristics of the tree, this article divides frequent subtree mining algorithm into three categories: Ordered Tree mining algorithm, unordered tree mining algorithm and free tree mining algorithms. This article focuses on the ordered tTree mining algorithm for mining algorithm unordered tree is briefly introduced, and the tree mining algorithm lot of tests to verify the efficiency and effectiveness of the algorithm in different states. Frequent subtree mining is an important research in the field of data mining. The current frequent subtree mining algorithms always keep in finding frequent subtree and are less frequent on the largest sub-tree and closed frequent subtrees study, so the maximal frequent subtrees and closed frequent subtree mining research is an important research direction. For the efficiency of the algorithm, we can consider parallel processing to the algorithm in order to improve the efficiency of the algorithm.

Acknowledgements

Project: Open Fund FY2014 Key Laboratory of Eco-tourism in Hunan

References

[1] M.J.Zaki. Efficiently Mining Frequent Trees In A Forest: Algorithms And Applications.In Ieee Transaction On Knowledge And Data Engineering, Special Issue On Mining Biological Data.Vol.17, No.8, Pp 1021~1035, 2005.

[2] Wang K. And Liu H., Discovering Typical Structures Of Documents: A Road Map Approach, Proceedings Of Acm Sigir The 1998 International Conference On Research And Development In Information Retrieval,1998.

[3] T. Asai, K. Abe, Et Al. Efficient Substructure Discovery Form Large Semi-Structured Data. The 2th Siam Int' 1 Conf. Data Mining (Sdm 2002), Arlington, Va, Usa, 2002.

[4] Y.Chi, Y.Yang, And R.R. Muntz. Indexing And Mining Free Trees. Proc.Thind Ieee Int'1 Conf.Data Mining, 2003.

[5] Y.Chi, Y.Yang and R.R. Muntz. Hybridtreeminer: an Efficient Algorithm For Mining Frequent Rooted Trees and Free Trees Using Canonical Forms. Proc.16th Int'1 Conf. Scientific And Statistical Database Management,2004.

[6] Y. Chi, Y. Yang, Y. Xia, and R.R. Muntz. Cmtreeminer: Mining Both Closed and Maximal Frequent Subtrees.in The Eighth Pacific Asia Conference on Knowledge Discovery and Data Mining (Pakdd'04), May, 2004.

[7] Panjin, Etc. Chopper: An Efficient Algorithm for Mining Frequent Ordered Labeled Tree Structure. 20th Annual National Database (Ndbc2003), Changsha, 2003.

[8] Zhu Yongtai, And So On. Espm- Frequent Subtree Mining Algorithm. Computer Research and Development, 2004 (10): 1720–1726.

[9] Zhao Chuanshen. Mining Algorithm Based on Projection Branch Fast Frequent Subtree. Computer Research and Development, 2006 (3): 456 – 462.

[10] M.J.Zaki. Efficiently Mining Frequent Trees In A Forest. Pro.Eighth Acm Sigkdd Int'1 Conf. Knowledge Discovery and Data Mining ,July 2002.