Study on A Fast Algorithm for Mining Disorder Tree

Xin Guo [1]

[1] School of Software Outsourcing, Jishou University, Zhangjiajie, Hunan, 427000

hunter2011@foxmail.com

Keywords: Data Mining; Frequent Subtree; Disorderly Tree; Export Subtree; Intron Tree

Abstract. Unordered tree mining has important research value in the field of XML data, bioinformatics, Web structure. In this paper, we propose a disorderly tree mining algorithms-UTMiner (Unordered Trees Miner). Because the tree has a disordered mining property, so in order to avoid digging out the same sub-tree, this paper proposes an efficient method of standardization disorder tree unordered tree into subtrees standardization, reuse rapid proposed ordered tree mining algorithm to get all the standardization sub-tree.

Introduction

Since the tree has a wide range of application, compared to the sets and sequences, the tree structure can better express the relationship between things, so frequent subtree mining has drawn increasing attention. Frequent subtree mining can be roughly divided into disordered and ordered tree tree mining excavation, unordered tree [1] as a tree structure, due to its structure as compared with other trees have more general characteristics (all nodes disorder), and therefore have a high value in the field of bioinformatics, Web structure, XML data and the like.

Unordered Trees Miner

Standardization Policy. When standardization, depth-first coding algorithm to represent a tree, in the depth of the tree traversal process nodes hierarchically numbered, and is uniquely represented by a binary tree node, wherein the level number, the node label , FIG. 4, the depth-first tree is encoded as:. To facilitate the design and implementation of a recursive algorithm to achieve the standard operating thinking: If a tree is a standard tree, then it's all sub-tree is also a standard tree, and all the sub-tree root lexicographical order from left to right arrangement. During execution, the algorithm sub-tree T using the following strategy:

1) Only if the sub-tree root, no operation is performed.

2) If the sub-tree contains only a subtree, no operation is performed.

3) If the sub-tree containing multiple sub-trees, the first sub-tree for each of them to standardize the operation, then the sub-tree root for each of them are arranged in order according to the dictionary, if a two lexicographical same sub-tree root, and where no one child node, it is converted to right-sibling nodes.

Figure 1 shows the standardization process of the tree, where the tree for the final standardization subtree.

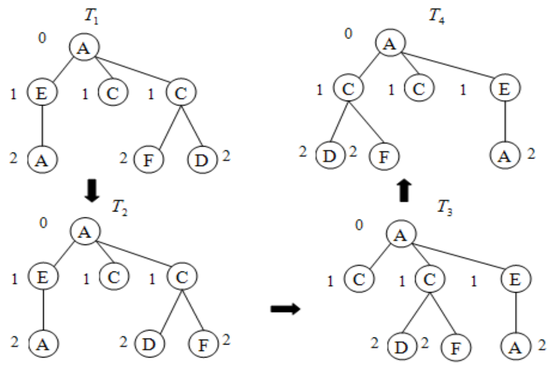


Fig. 1 Standardization

Data Structure. Any node of the tree, with a triple to uniquely represent a node, the node number which represents the parent node representing the node number, indicates the rightmost leaf node to node as the root of the subtree node number, if the node is the root of the tree, then, said the triples represented as nodes.

With a four-tuple to represent a sub-tree, it represents the sub-tree in the forest where the tree number (database). Indicates whether this subtree is exported subtree (1 for the exported subtree, 0 for non-export sub-tree), that intro tree prefix indicates the rightmost leaf node of this subtree node in the tree represents the original, if only one sub-tree node, then, says this quad tree representation for the child. In Figure 3, it is assumed in the database tree number is 0 sub-tree is available, he said sub-tree can be used to represent.

Miner Algorithm. Firstly, scan the database to get a sub-tree frequently, because all a sub-tree itself is a standard tree, there is no need to standardize the operation, and then all of a sub-tree frequently to construct a hash table, and into this hash table a linear table structure, then the value of the hash table is generated in accordance with the policy of candidate pairwise merge two sons get frequent tree, the same tree two sons do not need to be standardized operation, this time to build a new two subtrees containing all frequent Kazakhstan Greek table, and put it into a table structure in which the key is the value of the hash table tree node, the value stored hash table prefix and sub-tree to tree vector and other information, and frequently obtained three sub-tree, because it is unordered tree mining, so now contains some of the same tree structure algorithm to standardize operations, and remove the same tree structure, hash table consisting of all standard tree into a linear table, and so on will be All sub-tree no repeat of standardization.

UTMiner algorithm is described as follows:

Algorithm 1. unordered tree mining algorithm UTMiner

Input: database D, the minimum support minsup.

Output: All standard unordered trees.

Algorithm 1. unordered tree mining algorithm UTMiner

Input: database D, the minimum support minsup.

Output: All standard unordered trees.

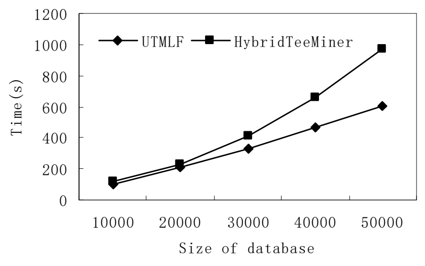
```
UTMiner (D, minusp):
F1 = {frequent 1-subtrees};
Hash ←F1 and Line [1]     Hash;
Node = 1;
While (Line [Node]! = Null) do
     Node = Node + 1;
     new_hash = null;
     For all [T] in Hash do
          For each element (x, i)  ∈  [T] do
          For each element (y, j)  ∈  [T] do
          R = {(x, i)      (y, j)}; // The first step in generating a candidate subtree merge
          If Node <     then // is less than the pruning threshold, or not to prune operation
               If all subtrees of R if frequent then
               Delete R; continue; // pruning operations, and jump out of the loop,
               No longer perform the second step merge operation
               L (R) = {L (X, i)     L (Y, j)}; // Step subtree merge vectors, seeking support
               If support (R)> minusp then
               Standard (R);
               New_hash    R;
               End IF
End for all;

     If new_hash! = Null then
          Line [Node]     new_hash;
```

```
    End while
  End UTMiner
```

Experiment and Analysis of Algorithms

We use the program to generate a random tree generator 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, and assuming experiment the pruning threshold is 6. We first case under the same parameters (tree height of 7, fan-out of 6), generating in size from 10,000 to 50,000 in five tree database, the same support in the default threshold is 1% of the mining program execution, results As shown in Figure 2.

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 3.



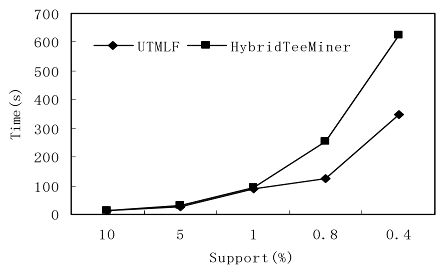


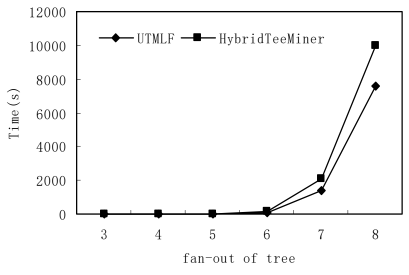
Fig. 2 Run time vs size of database Fig. 3 Run time Vs threshold of support

Tree algorithm is also a factor, we conducted experiments under different tree height and fan-out, and 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 generated by experiments shown in Table 1. :(parameter setting is expressed as (tree height, fan-out))

Table 1 The numbers of frequent subtrees

Task	Height of tree	subtrees	Fan-out of tree	subtrees
1	(4,6)	2110	(7,3)	1124
2	(5,6)	3562	(7,4)	3325
3	(6,6)	8775	(7,5)	6236
4	(7,6)	19453	(7,6)	25234
5	(8,6)	52367	(7,7)	106584
6	(9,6)	184325	(7,8)	446976

Corresponding to the running time of each task in Fig. 4, Fig. 5.



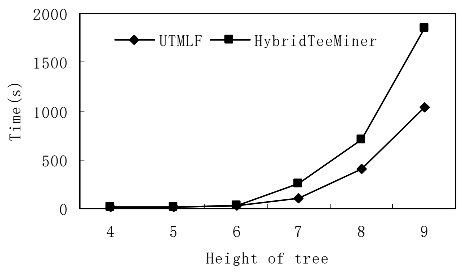


Fig. 4 Run time vs fan-out of tree Fig. 5 Run time vs high of tree

Conclusion

Algorithm adopt rightmost leaf node expansion candidate generation strategy, but what is different with other similar algorithm is this algorithm uses a vector-based sub-tree and hash table structure combine to build a multi-layered data structure to store all frequent subtrees because the sub-tree vector contains subtrees position and support information in the database, generation strategy by the candidate as defined herein, it is possible to obtain candidate subtree can be obtained at the same time their support, rather than scanning the database again, so the algorithm only needs to scan a database, improve operational efficiency.

Acknowledgements

Project: A Project Supported by Scientific Research Fund of Hunan Provincial Education Department

Project number:14B143

References

[1] K. Wang. and H. Liu. Discovering Typical Structures of Documents: A Road Map Approach, Proceedings of ACM SIGIR The International Conference on Research and Development In Information Retrieval, 1998:146-154.

[2] M.J. Zaki. Efficiently Mining Frequent Trees in A Forest. Proc .of Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, 2002:71-80.

[3] T. Asai, K. Abe, Et Al. Efficient Substructure Discovery form Large Semi-Structured Data. Proceedings of The Second SIAM International Conference on Data Mining, Arlington, USA, 2002:158-174.

[4] Y.Chi, Y.Yang,And R.R. Muntz. Indexing and Mining Free Trees. Proc.of Thind IEEE Int'1 Conf. on Data Mining, 2003: 509 - 512.

[5] Y.Chi, Y.Yang,And R.R. Muntz. Hybridtreeminer: An Efficient Algorihtm for Mining Frequent Rooted Trees and Free Trees Using Canonical forms. Proc.16th Int'1 Conf. Scientific And Statistical Database Management, 2004: 11-20.

[6] Pan Jin .Chopper: An Efficient Algorithm for Mining Frequent Ordered Labeled Tree Structure of the 20th Session of The National Database Annual Meeting (NDBC2003), Changsha, 2003.

[7] Zhu Yongtai, Wang Chen, Hong Mingshen. ESPM- Frequent Subtree Mining Algorithm. Computer Research and Development, 2004 (10): 1720-1726.

[8] C.S. Zhao, Z.H. Sun, J. Zhang. Mining Algorithm Based On Projection Branch Fast Frequent Subtree. Computer Research And Development, 2006 (3): 456-462.

[9] 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, 2005: 1021-1035.