Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 2, Number 10; April-June, 2015 pp. 874-877
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/jbaer.html

Mining Ordered Embedded and Induced Tree Patterns: An Overview

Geetika Munjal[1] and Swati Mittal[2]

[1]ITM University
[2]Student ITM University
E-mail: [1]geetika@itmindia.edu, [2]s21.mittal@gmail.com

Abstract—*Mining frequent subtrees from databases of labeled trees is a new research field that has many practical applications in areas such as computer networks, Web mining, bioinformatics, XML document mining, etc. This paper provides an overview of a wide range of algorithms for mining rooted ordered trees. The focus is on theoretical concepts of candidate generation and frequency counting of frequent subtree mining algorithms. It discusses the techniques involved in mining embedded and induced subtrees.*

Keywords: *Frequent subtree mining, embedded, induced, canonical representation*

1. INTRODUCTION

A tree can be called as semi-structured data structure. A tree is a hierarchical structure which comprises of a set of linked nodes. Data can be viewed as trees conceptually and abounds in domain like XML databases, bioinformatics and weblog analysis.

Earlier data mining was only limited to tabular data sets and data mining algorithms can only be used with tabular data but now the growth of semi-structured data has provided numerous opportunities.

There are a number of algorithms that are used to discover frequent subtrees from labelled trees in a database. Mostly, the algorithms are based on the techniques of Market-Basket analysis which is used for association rule mining. In this paper an overview regarding the tree mining algorithms have been presented.

2. TREE BASICS

Let T is a labelled ordered tree which is rooted and is said to be an acyclic directed connected graph represented by T={V,E $\leq$,L,Vo,$\sum$ } where V is the set of nodes and E is the set of edges.L is the labelling function defined by L:V$\rightarrow$ $\sum$ where $\sum$ is the alphabet[7]. L assigns labels to nodes in V from alphabet E. Root of the tree denoted by vroot. An edge (u,v) belongs to E where u is the parent and v is the child. Parent of u is said to be the ancestor of v and v is called the descendant of u. To represent the ordering among the siblings, a binary relation"$\leq$"$\subset$V2 is used. The size of a tree is equal to the number of nodes it has and denoted by |T|. Finally, node index is the preorder of a node according to the preorder traversal. The unrooted trees are free trees and are useful in analysis of moleculer evolution i.e phylogeny.

There are two main steps to find frequent patterns:

1. Candidate Generation
2. Frequency Counting
   Rightmost path extension method is widely used for candidate generation. In the second step, the frequencies of patterns are calculated using some algorithms.

3. TYPES OF SUBTREES

3.1.1 Embedded subtrees: Suppose we have a rooted tree T=(V,E,L) then T' is said to be an embedded tree of T where T' is represented by T'=(V',E',L') if and only if the following two conditions are fulfilled[4]:

1. Ancestral relationship is maintained 2. Left to Right order of siblings is same

Embedded subtrees of fig T1 subtreeabc is represented as 016,034,036,035

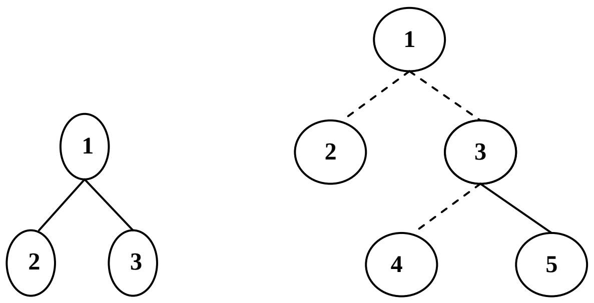


Fig. 1: a) Node 2 and 3 are siblings
b) Node 2 and 4 are embedded siblings

3.1.2 Induced subtree: In a tree T, vertex set is V and edge set is E, then T' with V' and E' as the vertex set and edge set respectively is said to be the induced subtree of T only when

1. Labeling of E' and V' is preserved in T

2. Left to right order of siblings in T' must be a subordering of vertices

By removing the leaf nodes repeatedly, an Induced subtree can be obtained.

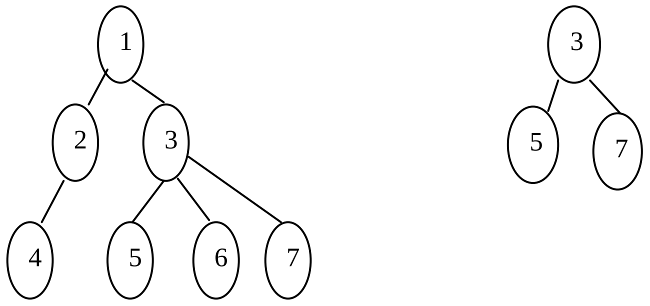


Fig. 2: a). Tree T b) Induced subtree of T

3.1.3Canonical representations of trees [8]

Depth-first codification: A tree can be represented by a string which is the sequence of the labels in a depth-first order. A special symbol is used. The depth-first codification for the example tree in Fig. 3 would ABDF↑G↑E↑C

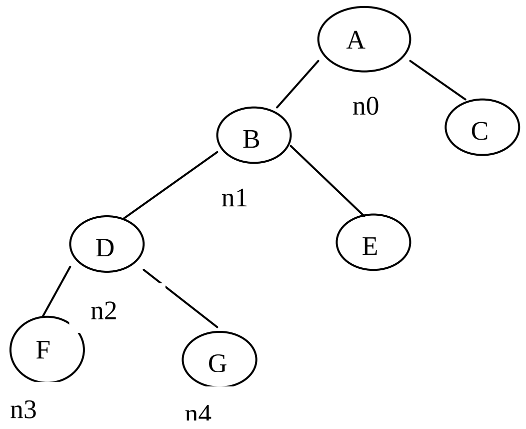


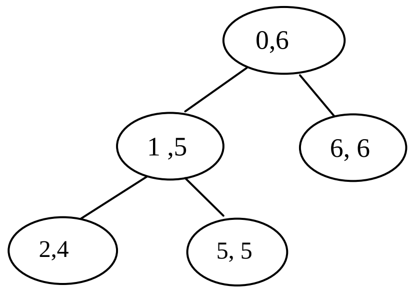
Fig. 3. Tree T1

Breadth-first codification: In this scheme, the string is generated according to breadth first order where levels are considered one by one. A special symbol $ is used which separates one sibling family from aother. The breath-firstcodification for the figure1 is A$BC$DE$FG

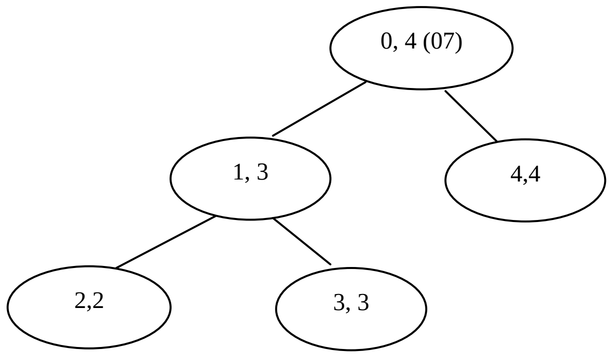
Depth-sequence-based coficlation: Here, depth first traversal is done, in addition it also stores the depth of every node in the tree. Therefore, the string has pair (d,l) where d is the depth and l is the node label. The depth sequence for the tree used in Figure1 is (0, A) (1, B) (2,D) (3,F) (3,G) (2,E) (1,C).

4. ALGORITHMS FOR MINING ORDERED TREES

4.1 Treeminer Treeminer is an algorithm for mining embedded ordered frequent trees introduced by Zaki[1]. In this algorithm, scopelists are used represented by the triplet (t,m,s) where t is the tree id, m is the match label and s is the scope. Scopelist is used for fast support counting and defines join operation for frequency counting. Fig 2 shows the scopelist representation of tree in figure1, scope is combination of preorder traversal number of the node, and preorder traversal number for right most nodes in subtree rooted at that node.



(a)



(b)

Fig. 4: a) Tree T2: node scope representation of tree T1 b) Tree T3

String representation T2 0123-14-1-15-1-16-1

String representation T3 012-13-1-14-1

Embedded trees are more informative then induced trees as they discover tree patterns which are hidden deep within larger trees.

Table 1: Vertical scopelist for nodes of T2 and T3

Node (a)	Node(b)	Node(c)	Node(d)
1[0,4] 2[0,6]	1[1,3] 1[2,2] 2[1,5] 2[3,3]	1[4,4] 1[3,3] 2[4,4] 2[5,5] 2[6,6]	2[2,4]

Join operation in Treeminer Join of ScopeLists of nodes is based on interval algebra, Let $s_x = [l_x,u_x]$ be a scope for node x, and $s_y = [l_y,u_y]$ a scope for y.We say that s_x contains s_y,denoted$s_x \supset s_y$, iff $l_x \leq l_y$ and $u_x \geq u_y$.So to perform join for node a and node be in tree1 scope 1[0,4] and 1[1,3] $l_x \leq l_y$ and $u_x \geq u_y$hence S_x contains $S_y$so the new list obtained after join operation is shown in table 2 node d is removed as it does not cover minimum support criteria with minimum support 2.

Table 2: Scopelist after join

Node (a---b)	Node(a---c)	Node(b---c)
1,0[1,3]	1,0[4,4]	1,1[4,4]
1,0[2,2]	1,0[3,3]	1,1[3,3]
2,0[1,5]	2,0[4,4]	2,2[4,4]
2,0[3,3]	2,0[5,5]	2,2[5,5]
	2,0[6,6]	

Table3: Scopelist join for Frequent subtree ab-1c-1

1,01[4,4]
1,02[4,4]
1,02[3,3]
2,01[4,4]
2,01[6,6]
2,03[4,4]
2,03[5,5]
2,03[6,6]

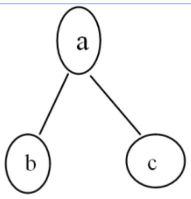


Fig. 5: Frequent subtree abc of T2 and T3

Oinduced[2] This algorithm is used for finding frequent ordered induced patterns using breadth first method for candidate generation. Indexing scheme is used to extend every candidate by frequent trees. It also uses two new encoding schemes which are m coding and cm coding for frequency counting and per tree support and accordingly and develop an efficient approach for frequency. Occurence match support also known as tree support is also performed by these encoding schemes. Tree support can be defined as the number of occurrences of a subtree in a database. For example in Fig. 6 occurence of embedded subtree is 4. Input for the OInduced algorithm is an integer value given by the user and database of rooted ordered trees in string format. Value defined by the user is known as minsup value and is selected either per tree or occurrence match. It is responsible for reducing running time and is affected by the size of minsup value.

Candidate Generation in Oinduced:It uses rightmost extension technique which uses breadth first method to generate candidates. Rightmost path extension method is complete and non-redundant for generating induced and embedded trees.

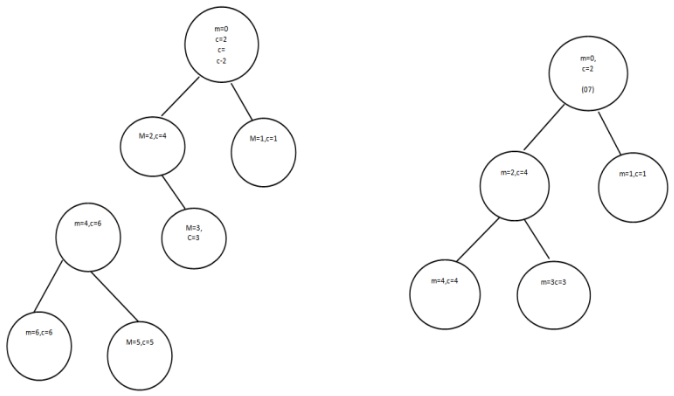


Fig. 6: a) C-M coding for Tree T2 b) C-M coding for T3

Freqt[5] : It is based on rightmost expansion i.e. extending a tree by adding a node to the right most path or to the rightmost sibling.

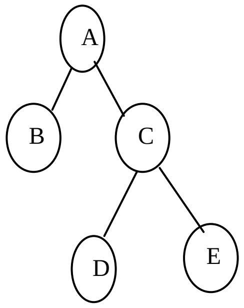


Fig. 7: Tree T4

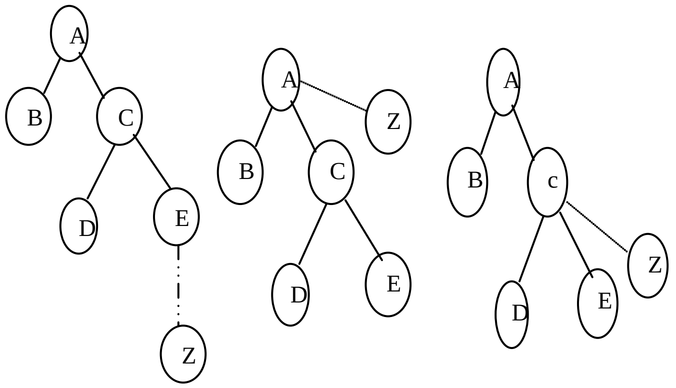


Fig. 8: Possible Right most expansion of tree T4

In order to extract regular substructures from a collection of web pages, the FreqT algorithm is very useful. For example, information can be extracted from web or a query can be modified in semi-structured database language.

5. RESULT

Average running time of treeminer algorithm is directly proportional to the size of frequent trees. As the size of the tree increases, treeminer average running time increases. In FreqT, size of pattern trees has no effect on the evaluation time per occurrences.

In terms of running time, OInduced is better than FreqT and scales linearly with respect to the size of input trees. In OInduced algorithm, M and CM coding are very helpful as they compute Candidate frequency quickly.

6. CONCLUSION

The algorithms studied so far are based on candidate generation approach which uses Apriori algorithm. Different techniques followed by different mining algorithms for rooted trees are discussed in the paper with examples. But these algorithms cannot be applied to trees where nodes are partially ordered which are used extensively while handling XML documents. .The application of FP-Growth-like algorithms to tree mining is an open research problem.

REFERENCES

[1] Mohammed J. Zaki “Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications” IEEE Transactions on Knowledge and Data Engineering, 17(8):1021-1035. Aug 2005

[2] MostafaHaghirChehreghani,MortezaHaghirChehreghani, Caro Lucas, and MasoudRahgozar “OInduced: An Efficient Algorithm for Mining Induced Patterns from Rooted Ordered Trees” IEEE Transactions on systems, man and cybernetics part A: systems and hymans Vol. 41, no. 5,September 2011

[3] R.Agrawal and R. Srikant. “Mining Sequential Patterns” Proceedings of the 20th VLDB Santiago, Chile, 1994, PP 487-499

[4] Aída Jiménez, Fernando Berzal, and Juan Carlos Cubero “Mining Different Kinds of Trees: A Tree Mining Overview”CEDIsept 2007 PP 343-352.

[5] Tatsuya Asai, Kenji Abe,ShinjiKawasoe,HirokiArimura, Hiroshi sakamoto,andSetsuoArikawa “Efficient Substructure Discovery from Large Semi-structured Data”SIAM PP 153-174.

[6] J. Han, J. Pei, T. Yin, Mining frequent patterns without candidate generation,In Proc. SIGMOD 2000, ACM, 2000, pp. 1–11.

[7] Aho, A. V., Hopcroft, J. E., Ullman, J. D., Data Structures and Algorithms, Addison-Wesley, 1983.

[8] Tatsuya Asai1, Hiroki Arimura1, Takeaki Uno2, and Shin-ichi Nakano3,”Discovering Frequent Substructures in Large Unordered Trees”, *Conference on discovery science*, 2003, pp.47-61