SS-ZG548: ADVANCED DATA MINING

14 Mining with Tree Data

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

Oct 27, 2021 ONLINE (WILP @ BITS-Pilani July-Dec 2021)

http://ktiwari.in/adm

Complex data forms[1]

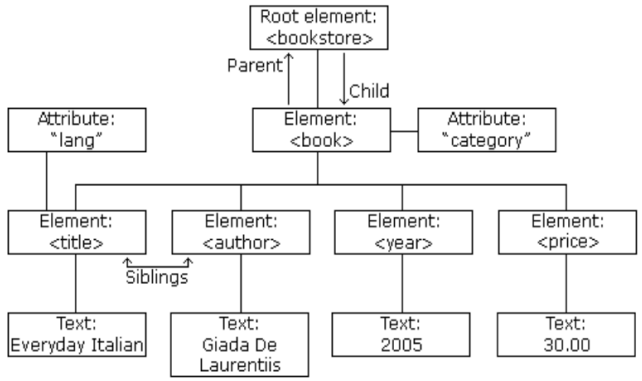
Data can be categorized as 1) Transactional, 2) Relational, 3) Sequential, 4) Semi-structured, or 5) Unstructured

XML: Extensible Markup Language

Data has a hierarchical relationship between nodes. NO loops

```
<?xml version="1.0"?>
  <bookstore>
          <book category ="COOKING">
                  <title lang="en">Everyday Italian>
                  <author>Giada De Laurentiis</author>
                  <year>2005</year>
                  <price>30.00</price>
                  </title>
          </book >

          <book category ="CHILDREN">
                  <title lang="en">Harry Potter>
                  <author>J K. Rowling</author>
                  <year>2005</year>
                  <price>29.99</price>
                  </title>
          </book >
 </bookstore>
```



- Mine frequent patterns from XML (such as frequent trees)
- Can be modeled as a tree

[1] Book: Mining of Data with Complex Structures, By Fedja Hadzic, Henry Tan, Tharam S. Dillon

Frequent Subtree Mining Problem

Given a tree database T_{db} and a minimum support threshold σ, find all subtrees that occur at least σ times in T_{db}

Support

1. Transaction-Based Support: It is the number of transactions in T_{db} that contain at least one occurrence of subtree t

$$\sum_{k_i \in T_{db}} d(t, k_i)$$

where d is an indicator variable for presence of subtree t in k_i

2. Occurrence-Based Support: Takes the repetition of items in a transaction and counts the subtree occurrences in the database as a whole

$$\sum_{k_i \in T_{db}} g(t, k_i)$$

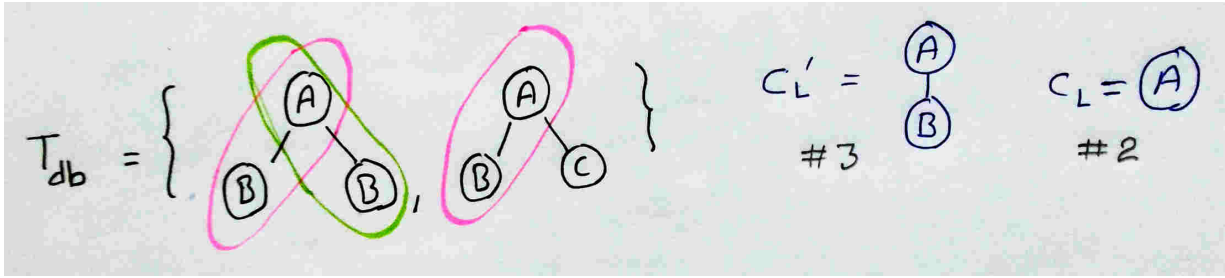
where g denotes the total number of occurrences of subtree t in k_i

3. Hybrid-Match Support

AR Mining

Can we use apriory like algorithms (generate&test based approach)?

- NO for Occurrence-Based Support as anti monotone property[2] violates



- YES for Transaction-Based Support

Efficient Representation of XML tree

- Text format. No
- DFS string encoding: Traverse the tree in pre-order. Good for parent-child or ancestor-descendent or vertical relationships
- BFS string encoding: Traverse the tree in breadth-first order. Each sibling family is separated by the level coding

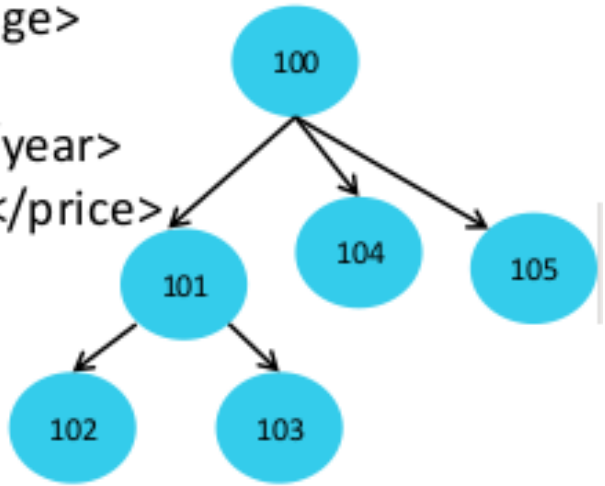
[2] Counter: In a tree database if $\exists$ candidate subtrees C_L & C_L' where $C_L \subseteq C_L'$ $s.t.$ C_L' is frequent but C_L is infrequent, We say that C_L' is pseudo frequent candidate subtree

Efficient Representation

'author[Giada De Laurentiis]'

String	Index
book[category="COOKING"]	100
author	101
name["Giada De Laurentiis"]	102
age["40"]	103
year["2005"]	104
price["30.00"]	105

```
<book category="COOKING">
  <author>
    <Name>Giada De Laurentiis</Name>
    <age>40</age>
  </author>
  <year>2005</year>
  <price>30.00</price>
</book>
```



Enumeration Methods

- Using join: ab + ac = abc
- TreeMiner by Zaki 2005
- Tree Model Guided (TMG)

TreeMiner: Candidate Generation [3]

- Uses join approach: frequent k-subtrees are used to generate candidate (k+1)-subtrees
- Two k-subtrees X,Y are in the same prefix equivalence class iff they share a common prefix up to the (k-1)th node
- An element of a equivalence class with label x and attached to position i^{th} is denoted as (x, i)
- An equivalence class with prefix P with elements $(x, i), (y, j), (z, k)$ is denoted as $[P] : (x, i), (y, j), (z, k)$
- Class Extension
  - Let P be a prefix class with encoding P, and let (x, i) and (y, j) denote any two elements in the class
  - Let $[P_x]$ denote the class representing extensions of element (x, i)

[3] TreeMiner: An Efficient Algorithm for Mining Embedded Ordered Frequent Trees. By Zaki, Mohammed J, At Advanced Methods for Knowledge Discovery from Complex Data, pp 123–151, Springer, 2005

TreeMiner: Candidate Generation

- Define a join operator $\otimes$ on the two elements, denoted $(x, i) \otimes (y, j)$ as follows

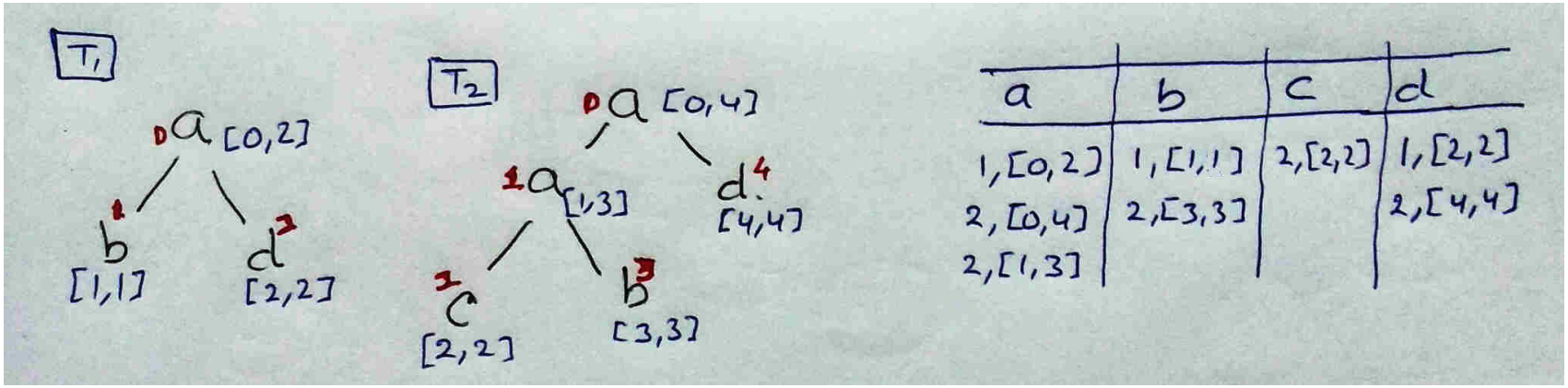
Case-1 $(i = j)$:
a) If $P \neq \phi$, add (y, j) and $(y, j + 1)$ to class $[P_x]$
b) If $P = \phi$, add $(y, j + 1)$ to class $[P_x]$

Case-2 $(i > j)$: add (y, j) to class $[P_x]$

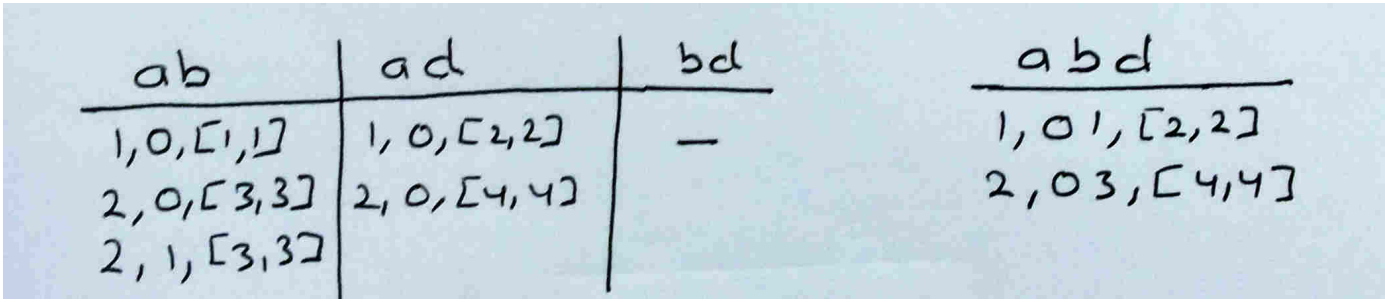
Case-3 $(i < j)$: no new candidate is possible in this case

- Enumerates all possible (k+1) subtrees with the common prefix P using k^{th}
- Counting of Frequency can be done using 1) Hashing approaches, 2) Store only hyperlinks of subtrees in a *tree database* instead of creating a local copy for each generated subtree, 3) Subtree encoding is used to uniquely identify subtrees
- DFS is used to find scope of node $n_l : [l, r]$ and scope-list that is a triple (t, m, s). t: tree ID, m: match label, s: scope

TreeMiner: Candidate Generation



- Two Type of Scope-list joins ($s_x = [l_x, u_x]$, $s_y = [l_y, u_y]$)
  1. In-scope Join: Used for ancestor-descendent or parent-child node extension. $s_x \supset s_y$ iff $l_x \leq l_y$ and $u_y \leq u_x$
  2. Out-scope Join: Used for sibling extension. $s_x < s_y$ iff $l_y \leq u_x$
- Add the triple $(t_y, m_y \cup l_x, s_y)$



- Perform all possible joins. Frequency count of a subtree is equal to the size of its scope-list

Thank You!

Thank you very much for your attention!

Queries ?