Frequent Item Identification through Pattern Growth

Let:

- \( I = \{a_1; a_2; \ldots ; a_m\} \) be a set of items,
- a transaction database \( DB = \{T_1; T_2; \ldots ; T_n\} \) where \( T_i \) \((i \in [1::n])\) is a transaction which contains a set of items in \( I \).
- The support (or occurrence frequency) of a pattern \( A \), which is a set of items, is the number of transactions containing \( A \) in \( DB \). \( A \) is a frequent pattern if \( A \)'s support is no less than a predefined minimum support threshold.

Given a transaction database \( DB \) and a minimum support threshold, \( \text{minSup} \), the problem of finding the complete set of frequent patterns is called the **frequent pattern mining problem**, or frequent itemset generation.

We will be using the following DB as an example:

<table>
<thead>
<tr>
<th>TID</th>
<th>Items Bought</th>
<th>(Ordered) Frequent Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>( f, a, c, d, g, i, m, p )</td>
<td>( f, c, a, m, p )</td>
</tr>
<tr>
<td>200</td>
<td>( a, b, c, f, l, m, o )</td>
<td>( f, c, a, b, m )</td>
</tr>
<tr>
<td>300</td>
<td>( b, f, h, j, o )</td>
<td>( f, b )</td>
</tr>
<tr>
<td>400</td>
<td>( b, c, k, s, p )</td>
<td>( c, b, p )</td>
</tr>
<tr>
<td>500</td>
<td>( a, f, c, e, l, p, m, n )</td>
<td>( f, c, a, m, p )</td>
</tr>
</tbody>
</table>

Table 1: A transaction database as running example.

We begin by creating a Frequent Pattern Tree or FP-Tree, which has the following structure:

- one root labeled as "null"
- a set of item prefix subtrees as the children of the root where Each node in the item prefix subtree consists of four fields:
  - item-name: registers which item this node represents
  - item-count: registers the number of transactions represented by the portion of the path reaching this node
  - node-link: links to the next node in the FP-tree carrying the same item-name or null if there is none
  - parent-node: links to the parent node in this branch of the tree.
- a frequent-item header table where each entry in the frequent-item header table consists of two fields:
  - item-name
  - head of node-link: points to the rest node in the FP-tree carrying the item-name.

Given a DB and minim support \( \text{minSup} \) we generate the tree as follows:
• Scan the transaction database DB once.
• Collect the set of frequent items F and their supports.
• Sort F in support descending order as L, the list of frequent items.
• Create the root of an FP-tree, T, and label it as "null".
• For each transaction Trans in DB:
  ◦ Select and sort the frequent items in Trans according to the order of L.
  ◦ Let the sorted frequent item list in Trans be [p|P], where p is the first element and P is the remaining list
  ◦ Call insert_tree([p|P]; T):
    ▪ IF: T has a child N such that N.item-name = p.item-name
      • then increment N's count by 1;
    ▪ ELSE:
      • create a new node N
      • let its count be 1
      • its parent link be linked to T, and its node-link
      • be linked to the nodes with the same item-name via the node-link structure.
    ▪ IF: P is nonempty,  
      ◦ RECURSE: call insert_tree(P,N).

![FP Tree Example](image)

Figure 1: The FP tree in Example 1.
Next we mine for frequent itemsets by passing in the generated FP-Tree to the FP-growth algorithm and the empty list of Frequent ItemSets:

Procedure FP-growth (Tree, freqItemsets)
{
    IF: Tree contains a single path P
        For each combination of nodes c in the path P
            generate frequent itemset c U freqItemsets with support = minimum support of all nodes in c
    ELSE:
        For each node a in the header of Tree
            generate frequent itemset c = a U freqItemsets with support = a.support
            generate FP-Tree for c = TreeC
            IF: TreeC contains more than just the root node
                RECURSE: call FP-growth(TreeC, c)
}

<table>
<thead>
<tr>
<th>item</th>
<th>conditional pattern base</th>
<th>conditional FP-tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>{(f:2, c:2, a:2, m:2), (c:1, b:1)}</td>
<td>{(c:3)}</td>
</tr>
<tr>
<td>m</td>
<td>{(f:4, c:3, a:3, m:2), (f:4, c:3, a:3, b:1, m:1)}</td>
<td>{(f:3, c:3, a:3)}</td>
</tr>
<tr>
<td>b</td>
<td>{(f:4, c:3, a:3, b:1), (f:4, b:1), (c:1, b:1)}</td>
<td>{}</td>
</tr>
<tr>
<td>a</td>
<td>{(f:3, c:3)}</td>
<td>{(f:3, c:3)}</td>
</tr>
<tr>
<td>c</td>
<td>{(f:3)}</td>
<td>{(f:3)}</td>
</tr>
<tr>
<td>f</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

Table 2: Mining of all patterns by creating conditional (sub) pattern bases

References: