Finding frequent (closed) sets with tree structures
FP-tree, FP-growth, CLOSET

- FP-tree data structure
- FP-growth algorithm for finding all frequent sets
- CLOSET algorithm for finding frequent closed sets
- Literature for this part
Key properties

Problem: discovery of frequent sets

- a compressed representation of the database (FP-tree)
- no explicit generation of candidates
- recursive partitioning of search space
Key ideas

- scan database once, compute the frequencies of singletons
- scan the database for a second time and store it as a tree, also store counts in the tree
- while building the tree, prune and sort items by their frequency (try to minimize the tree size)
- determine frequent sets using the tree, without accessing the database again
Example relation

(here $a, b, \ldots \in R$ are items)

<table>
<thead>
<tr>
<th>Row</th>
<th>Ordered frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a, c, d, f, g, i, m, p$</td>
<td>$f, c, a, m, p$</td>
</tr>
<tr>
<td>$a, b, c, f, l, m, o$</td>
<td>$f, c, a, b, m$</td>
</tr>
<tr>
<td>$b, f, h, j, o$</td>
<td>$f, b$</td>
</tr>
<tr>
<td>$b, c, k, p, s$</td>
<td>$c, b, p$</td>
</tr>
<tr>
<td>$a, c, e, f, l, m, n, p$</td>
<td>$f, c, a, m, p$</td>
</tr>
</tbody>
</table>

Frequency threshold $= 3/5$. 

Constructing FP-tree

- first database scan: frequent sets and absolute frequencies are 
  \[ f : 4, c : 4, a : 3, b : 3, m : 3, p : 3 \]
- initialize the FP-tree (frequent pattern tree) \( T \):
  \[ T = \text{node labeled “null”} \]
- second database scan: for each row
  - read the row
  - remove infrequent items and sort the frequent ones in descending order by frequency
  - add the resulting string to \( T \), update counts as necessary
**FP-tree data structures**

- a tree, with the root labeled “null”, and with paths in the tree representing item prefixes
- links across the tree, linking all occurrences of the same item in the tree
- each node (except null) consists of
  - item name: item identifier
  - count: nr of rows reaching this node
  - node link: link to next node in the tree with the same item identifier
- frequent item header table: starting point for the cross links
String insertion procedure

- procedure insert_tree(string \([p|P]\), tree rooted at \(T\))
- \(p\) is the first item of the string and \(P\) is the remaining string
- the 2nd database scan: for each row \(t \in r\) call insert_tree\((t', T)\), where \(t'\) is the pruned and resorted contents of the row, and \(T\) is the root of the tree

1. if \(T\) has a child node \(N\) such that \(N.itemname = p\) then
2. \(N.count++\);
3. else
4. create a new node \(N\);
5. \(N.itemname := p\); \(N.count := 1\);
6. update nodelinks for \(p\) to include \(N\);
7. if \(P\) is non-empty
8. call insert_tree\((P, N)\);
Analysis

- Time complexity:
  - 2 scans over the database
  - tree building: $O(||r||)$ (total number of items)

- Space complexity:
  - $O(||r||)$
  - average complexity much better!? (pruning and sorting of items)
  - tree height bounded by the size of the maximal row
Finding frequent sets

FP-growth algorithm

- for all frequent items $A$, in increasing order of frequency (i.e., starting from the bottom of the header table and the tree):
  - traverse all occurrences of $A$ in the tree using the node links
  - at each node $N$ with $N.itemname = A$, determine the frequent sets in which $A$ occurs
  - do this by only looking at the path from root to $N$
    (all sets including nodes below $N$ have been generated already in earlier iterations)
Example

item \( p \)

- **two paths**
  - \( f : 4, c : 3, a : 3, m : 2, p : 2 \)
  - \( c : 1, b : 1, p : 1 \)

- i.e., \( fcam \) occurs twice with \( p \) and \( cb \) once; \( p \)'s frequency is \( 2+1=3 \)

- \( \Rightarrow p \)'s **conditional pattern base** (note: \( p \) removed, counts adjusted):
  - \( f : 2, c : 2, a : 2, m : 2 \)
  - \( c : 1, b : 1 \)

- frequent sets that contain \( p \) are determined by the conditional pattern base!
• \( \Rightarrow \) recursively build an FP-tree for the conditional base and find frequent sets there, then add \( p \) to them all

• the recursive call:

• conditional “data base” given as input
  - \( f : 2, c : 2, a : 2, m : 2 \)
  - \( c : 1, b : 1 \)

• 1st database scan: only \( c : 3 \) is frequent

• \( \Rightarrow cp \) is frequent, frequency \( 3/5 \)
item $m$

- two paths
  - $f : 4$, $c : 3$, $a : 3$, $m : 2$
  - $f : 4$, $c : 3$, $a : 3$, $b : 1$, $m : 1$

- $p$ can now be ignored, sets containing it were found already

- $m$'s conditional pattern base:
  - $f : 2$, $c : 2$, $a : 2$
  - $f : 1$, $c : 1$, $a : 1$, $b : 1$

- $m$'s conditional FP-tree: just one path $f : 3$, $c : 3$, $a : 3$

- recursively find frequent patterns in the conditional FP-tree, first for $a$, then for $c$ and $f$
Example, conditional FP-tree of $m$

Global FP-tree

Conditional pattern base of "m"

Header table

<table>
<thead>
<tr>
<th>item</th>
<th>head of node-links</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

Conditional FP-tree of "m"
Example, conditional FP-trees of *ma* and *mac*

Conditional pattern base of "am": (f:3, c:3)  
Conditional pattern base of "cam": (f:3)

Conditional FP-tree of "am"  
Conditional FP-tree of "cam"

Observation: if the tree consists of just one path, one can simply generate all combinations of items
**FP-growth algorithm**

**Algorithm** FP-growth((conditional) FP-tree $T$, condition $X \subseteq R$)
First call: FP-growth(root, $\emptyset$)

1. if tree $T$ consists of a single path then
2. for all combinations $Y$ of items in the path
3. output set $X \cup Y$ and the minimum count
   of nodes in $Y$;
4. else for each item $A$ in the header table of $T$
5. output $Z := X \cup \{A\}$ and the count $A$.count;
6. construct $Z$'s conditional FP-tree $T_Z$;
7. if $T_Z \neq \emptyset$ then call FP-growth($T_Z$, $Z$);
Experimental results

![Graph showing runtime vs. support threshold for D1 and D2 FP-growth algorithms and D1 Apriori algorithm.](image)
CLOSET: closed sets with FP-tree

- use conditional pattern bases to locate closed sets
- **Lemma** Consider the conditional database of some set $X$ and the (possibly empty) set $Y$ of items that appear in every row of the conditional database. $X \cup Y$ is closed if no closed set $Z$ has been yet found such that $X \cup Y \subseteq Z$ and $Z$’s count is identical to $Y$’s count in the conditional database.
CLOSET optimizations to FP-growth

- the set $Y$ of items that appear in every row of the conditional database — if not empty — is a prefix of the only path from the root of the conditional FP-tree
  \[ \Rightarrow \text{handle these directly, not recursively} \]
- in more general, if there exists a single prefix path from the root, possibly several closed sets can be extracted directly
- if $X \subset Y$, the counts are equal, and $Y$ is closed, then there are no closed sets that contain $X$ but not $Y$
  \[ \Rightarrow \text{such sets } X \text{ can be pruned} \]
Single path optimization

\[ \text{root} \]
\[ i_{1:n-1} \]
\[ i_{1:n_1} \]
\[ i_{(k-1):n_2} \]
\[ i_{k:n_{1+l_1}} \]

- frequent closed itemset
  \[ i_1 \ldots i_{k:1:n_{1+l_1}} \]

\[ i_{(k:1):n_2} \]
\[ i_{k:1:n_l} \]

- frequent closed itemset
  \[ i_1 \ldots i_{k:1:n_l} \]

\[ i_{(k:1):n_2} \]
\[ i_{k:1:n_l} \]

- frequent closed itemset
  \[ i_1 \ldots i_{k:1:n_l} \]

\[ \ldots \]
Experimental results

- Again: number of frequent closed set vs. frequent sets

<table>
<thead>
<tr>
<th>Support</th>
<th>#F.C.I</th>
<th>#F.I</th>
<th>#F.I/#F.C.I</th>
</tr>
</thead>
<tbody>
<tr>
<td>64179 (95%)</td>
<td>812</td>
<td>2,205</td>
<td>2.72</td>
</tr>
<tr>
<td>60801 (90%)</td>
<td>3,486</td>
<td>27,127</td>
<td>7.78</td>
</tr>
<tr>
<td>54046 (80%)</td>
<td>15,107</td>
<td>533,975</td>
<td>35.35</td>
</tr>
<tr>
<td>47290 (70%)</td>
<td>35,875</td>
<td>4,129,839</td>
<td>115.12</td>
</tr>
</tbody>
</table>

Table 2: The number of frequent closed itemsets and frequent itemsets in dataset *Connect-4*. (F.C.I for frequent closed itemsets and F.I for frequent itemsets.)
- Performance comparison with other algorithms for closed frequent sets
Literature

- **FP-tree, FP-growth:**
  Jiawei Han, Jian Pei, Yiwen Yin: Mining Frequent Patterns without Candidate Generation. 2000 ACM SIGMOD Intl. Conference on Management of Data.

- **CLOSET:**
  Jian Pei, Jiawei Han, Runying Mao: CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2000.