Brief Introduction to Algorithmic Data Analysis in English

Esther Galbrun
Autumn 2021
Part I

Frequent Itemset Mining
Problem
Discover items that often co-occur in a dataset

Classical setting: *Shopping basket data*

- Each product of the supermarket is an item
- Record customer transactions as sets of items
- Identify products that are often bought together
  
  **Frequent itemset** \{butter, bread, ham, pickles\}

- Extract rules that capture typical buying behaviour
  
  **Association rules** \{bread, ham\} \Rightarrow \{butter, pickles\}

- Insights for marketing and shelf placement
Discover items that often co-occur in a dataset

**Shopping basket data**  Customer transactions
  Identify products often bought together

**Text mining**  Bag of word model
  Identify co-occurring terms and keywords

**More complex data types**  (spatio-)temporal data, graph data

**Other analysis tasks**  Building block for clustering, classification, outlier detection
A pizzeria offers to compose your pizza by freely choosing ingredients among *ham, jalapeno, mozzarella, olives and tuna*. To put together a menu, the pizzaiolo would like to know what are favorite combinations.
Data

The database $\mathcal{T}$ is a collection of sets, called transactions, from a universe $U$ of items

$$\mathcal{T} = \{T_1, T_2, \ldots, T_n\}, \text{ where } T_k \subseteq U, \forall k \in [1, n]$$

The total number of items is $m = |U|$

If we fix an order over $U$, each transaction can be represented as a binary vector of size $m$

Then, the database can be represented as a binary matrix with $n$ rows and $m$ columns

Each transaction has a unique identifier, its $tid$
The universe of items is the set of five ingredients
{ham, jalapeno, mozzarella, olives, tuna}
For short, $U = \{h, j, m, o, t\}$
Each pizza constitutes a transaction, represented by the corresponding set of ingredients
For instance, a ham and mozzarella pizza is represented as $T = \{h, m\}$, also simply denoted $hm$

Ordering the items alphabetically according to corresponding ingredient names, this pizza is represented by the binary vector $\langle 1, 0, 1, 0, 0 \rangle$, also simply written 10100
The database then records all pizzas sold

<table>
<thead>
<tr>
<th>tids</th>
<th>pizzas</th>
<th>sets</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>ham mozzarella olives</td>
<td>{h, m, o}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>2)</td>
<td>mozzarella</td>
<td>{m}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>3)</td>
<td>jalapeno mozzarella</td>
<td>{j, m}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>4)</td>
<td>ham jalapeno mozzarella olives</td>
<td>{h, j, m, o}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>5)</td>
<td>ham jalapeno mozzarella olives</td>
<td>{h, j, m, o}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>6)</td>
<td>ham</td>
<td>{h}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>7)</td>
<td>ham jalapeno mozzarella tuna</td>
<td>{h, j, m, t}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>8)</td>
<td>mozzarella</td>
<td>{m}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>9)</td>
<td>olives</td>
<td>{o}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>10)</td>
<td>ham jalapeno mozzarella olives tuna</td>
<td>{h, j, m, o, t}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>11)</td>
<td>ham mozzarella tuna</td>
<td>{h, m, t}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td>12)</td>
<td>ham mozzarella</td>
<td>{h, m}</td>
<td>■■■■■■■■■■</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
An itemset $I$ is a set of items, i.e. $I \subseteq U$

A $k$-itemset is an itemset that contains exactly $k$ items, i.e. such that $|I| = k$

The support set of an itemset $I$ in $\mathcal{T}$ is the set of transactions from $\mathcal{T}$ that contain $I$

$$\text{supp}_\mathcal{T}(I) = \{ T \in \mathcal{T}, I \subseteq T \}$$

We call $|\text{supp}_\mathcal{T}(I)|$ the absolute support of $I$ in $\mathcal{T}$

and $|\text{supp}_\mathcal{T}(I)| / |\mathcal{T}|$ its fractional support

We denote $\text{supp} \%_\mathcal{T}(I)$ the fractional support given as a percentage, i.e.

$$\text{supp} \%_\mathcal{T}(I) = 100 \cdot \frac{|\text{supp}_\mathcal{T}(I)|}{|\mathcal{T}|}$$
An itemset $I$ is a set of items, i.e. $I \subseteq U$

A $k$-itemset is an itemset that contains exactly $k$ items, i.e. such that $|I| = k$

The support set of an itemset $I$ in $\mathcal{T}$ is the set of transactions from $\mathcal{T}$ that contain $I$

$$\text{supp}_\mathcal{T}(I) = \{T \in \mathcal{T}, I \subseteq T\}$$

We call $|\text{supp}_\mathcal{T}(I)|$ the absolute support of $I$ in $\mathcal{T}$ and $|\text{supp}_\mathcal{T}(I)| / |\mathcal{T}|$ its fractional support

There are variations in the use of support terminology

The database is often left out from the notation, as it is clear from the context
Pizzeria example

<table>
<thead>
<tr>
<th>tid</th>
<th>set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{h, m, o}</td>
</tr>
<tr>
<td>2</td>
<td>{m}</td>
</tr>
<tr>
<td>3</td>
<td>{j, m}</td>
</tr>
<tr>
<td>4</td>
<td>{h, j, m, o}</td>
</tr>
<tr>
<td>5</td>
<td>{h, j, m, o}</td>
</tr>
<tr>
<td>6</td>
<td>{h}</td>
</tr>
<tr>
<td>7</td>
<td>{h, j, m, \textbf{t}}</td>
</tr>
<tr>
<td>8</td>
<td>{m}</td>
</tr>
<tr>
<td>9</td>
<td>{o}</td>
</tr>
<tr>
<td>10</td>
<td>{h, j, m, o, \textbf{t}}</td>
</tr>
<tr>
<td>11</td>
<td>{h, m, \textbf{t}}</td>
</tr>
<tr>
<td>12</td>
<td>{h, m}</td>
</tr>
</tbody>
</table>

In this database, for itemset \( I = \{ \textbf{t} \} \)

\[
\text{supp}(I) = \{7, 10, 11\} \\
|\text{supp}(I)| = 3 \\
\text{supp \%}(I) = 25
\]
<table>
<thead>
<tr>
<th>tid</th>
<th>set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>{h, m, o}</td>
</tr>
<tr>
<td>2)</td>
<td>{m}</td>
</tr>
<tr>
<td>3)</td>
<td>{j, m}</td>
</tr>
<tr>
<td>4)</td>
<td>{h, j, m, o}</td>
</tr>
<tr>
<td>5)</td>
<td>{h, j, m, o}</td>
</tr>
<tr>
<td>6)</td>
<td>{h}</td>
</tr>
<tr>
<td>7)</td>
<td>{h, j, m, t}</td>
</tr>
<tr>
<td>8)</td>
<td>{m}</td>
</tr>
<tr>
<td>9)</td>
<td>{o}</td>
</tr>
<tr>
<td>10)</td>
<td>{h, j, m, o, t}</td>
</tr>
<tr>
<td>11)</td>
<td>{h, m, t}</td>
</tr>
<tr>
<td>12)</td>
<td>{h, m}</td>
</tr>
</tbody>
</table>

In this database, for itemset \( I = \{h, m\} \)

\[
\text{supp}(I) = \{1, 4, 5, 7, 10, 11, 12\}
\]

\[|\text{supp}(I)| = 7\]

\[\text{supp} \% (I) = 58.33\]

Frequent Itemset Mining

Given a set of transactions $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$, where each transaction $T_i$ is a subset of items from $U$, and a minimum support threshold $\sigma$, determine all itemsets $I$ that occur as a subset of at least $\sigma$ transactions in $\mathcal{T}$. 
## Pizzeria example

Enumerate all distinct pizzas

<table>
<thead>
<tr>
<th></th>
<th>count</th>
<th>tids</th>
<th></th>
<th>count</th>
<th>tids</th>
<th></th>
<th>count</th>
<th>tids</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmo</td>
<td>65</td>
<td>{1...}</td>
<td>hj</td>
<td>16</td>
<td>{...}</td>
<td>hjm</td>
<td>42</td>
<td>{...}</td>
</tr>
<tr>
<td>m</td>
<td>74</td>
<td>{2,8...}</td>
<td>hjo</td>
<td>8</td>
<td>{...}</td>
<td>hjot</td>
<td>0</td>
<td>{}</td>
</tr>
<tr>
<td>jm</td>
<td>28</td>
<td>{3...}</td>
<td>hjt</td>
<td>4</td>
<td>{...}</td>
<td>hmo</td>
<td>7</td>
<td>{...}</td>
</tr>
<tr>
<td>hjmo</td>
<td>47</td>
<td>{4,5...}</td>
<td>hjm</td>
<td>42</td>
<td>{...}</td>
<td>hjo</td>
<td>8</td>
<td>{...}</td>
</tr>
<tr>
<td>h</td>
<td>74</td>
<td>{6...}</td>
<td>hjo</td>
<td>8</td>
<td>{...}</td>
<td>hjot</td>
<td>0</td>
<td>{}</td>
</tr>
<tr>
<td>hjmt</td>
<td>30</td>
<td>{7...}</td>
<td>hjot</td>
<td>0</td>
<td>{}</td>
<td>hmo</td>
<td>7</td>
<td>{...}</td>
</tr>
<tr>
<td>o</td>
<td>178</td>
<td>{9...}</td>
<td>ho</td>
<td>43</td>
<td>{...}</td>
<td>hjm</td>
<td>42</td>
<td>{...}</td>
</tr>
<tr>
<td>hjmot</td>
<td>93</td>
<td>{10...}</td>
<td>hot</td>
<td>28</td>
<td>{...}</td>
<td>hmo</td>
<td>7</td>
<td>{...}</td>
</tr>
<tr>
<td>hmt</td>
<td>49</td>
<td>{11...}</td>
<td>ht</td>
<td>7</td>
<td>{...}</td>
<td>hjo</td>
<td>8</td>
<td>{...}</td>
</tr>
<tr>
<td>hm</td>
<td>96</td>
<td>{12...}</td>
<td>j</td>
<td>29</td>
<td>{...}</td>
<td>hjot</td>
<td>0</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>t</td>
<td>8</td>
<td>{...}</td>
<td>hjm</td>
<td>42</td>
<td>{...}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>hmt</td>
<td>49</td>
<td>{...}</td>
</tr>
</tbody>
</table>
Pizzeria example: Enumerating all distinct pizzas
Pizzeria example: Aggregating supports

\[
\begin{array}{cccccccc}
\emptyset & h & j & m & o & t \\
0 & 74 & 29 & 74 & 178 & 8 \\
\{6\ldots\} & \{\ldots\} & \{2,8\ldots\} & \{9\ldots\} & \{\ldots\} \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
\{hj\} & \{hm\} & \{ho\} & \{ht\} & \{jm\} & \{jo\} & \{jt\} & \{mo\} & \{mt\} & \{ot\} \\
16 & 96 & 43 & 7 & 28 & 16 & 14 & 20 & 13 & 17 \\
\{\ldots\} & \{12\ldots\} & \{\ldots\} & \{3\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
\{hjm\} & \{hjo\} & \{hjt\} & \{hmo\} & \{hmt\} & \{hot\} & \{jmo\} & \{jmt\} & \{jot\} & \{mot\} \\
42 & 8 & 4 & 65 & 49 & 28 & 7 & 3 & 3 & 27 \\
\{\ldots\} & \{\ldots\} & \{\ldots\} & \{1\ldots\} & \{11\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
\{hjmo\} & \{hjmt\} & \{hjot\} & \{hmot\} & \{jmot\} \\
140 & \{4,5,10\ldots\} & \{4,5\ldots\} & \{\ldots\} & \{\ldots\} \\
\{4,5\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} \\
\end{array}
\]

\[
\begin{array}{cccccccc}
\{hjmot\} \\
93 & \{10\ldots\} \\
\end{array}
\]
Pizzeria example: Aggregating supports

```
∅

h
j
m
o

hj
hm
ho
ht

jm

hjm

hjmo

hjmt

hjmot

259 {3, 4, 5, 7, 10 ...}

{6 ...} {...} {2, 8 ...} {9 ...} {...}

16 96 43 7 16 14 20 13 17

{...} {12 ...} {...} {...} {...} {...} {...} {...} {...}

42 8 4 65 49 28 7 3 3

{...} {...} {1 ...} {11 ...} {...} {...} {...} {...}

47 30 0 108

{4, 5 ...} {7 ...} {...} {...}

93

{10 ...}
```
Support properties

**Monotonicity of support**
The support of every subset $J$ of $I$ is at least equal to that of the support of itemset $I$

$$\forall J \subseteq I, \quad \text{supp}(I) \subseteq \text{supp}(J)$$

and hence

$$|\text{supp}(I)| \leq |\text{supp}(J)|$$

**Downward closure property**
Every subset of a frequent itemset is also frequent
Pizzeria example: Lattice of ingredient combinations
Pizzeria example: Frequent ingredient combinations

For minimum support

\[ \sigma = 116 = 0.10 \cdot |\mathcal{T}| \]
Pizzeria example: Frequent ingredient combinations

for minimum support \( \sigma = 289 = 0.25 \cdot |\mathcal{T}| \)
Maximal and closed itemsets

A frequent itemset $I$ is **maximal** at a given minimum support level $\sigma$, if it is frequent, and no superset of it is frequent.

An itemset $I$ is **closed**, if none of its supersets have exactly the same support count as $I$.

**Condensed representations**

Knowledge of maximal frequent itemsets allows to reconstruct the set of frequent itemsets, but not their supports. Knowledge of closed frequent itemsets allows to also recompute the supports.
Pizzeria example: Frequent and maximal itemsets

For minimum support \( \sigma = 116 = 0.10 \cdot |T| \)
Pizzeria example: Frequent and maximal itemsets

For minimum support $\sigma = 289 = 0.25 \cdot |T|$
Algorithms
Algorithms for mining frequent itemsets

Support counting is expensive

Explore the space of itemsets by increasing lengths, i.e. level-wise enumeration

Avoid generating itemsets twice by using a canonical order

Exploit the downward closure property to prune itemsets
Level-wise enumeration: *Apriori* algorithm

\[
k \leftarrow 1 \\
\mathcal{F}_k \leftarrow \{\text{all frequent singleton itemsets}\} \\
\textbf{while } \mathcal{F}_k \neq \emptyset \textbf{ do} \\
\quad \text{Generate } \mathcal{C}_{k+1} \text{ by extending itemsets from } \mathcal{F}_k \\
\quad \text{Prune itemsets that violate downward closure} \\
\quad \mathcal{F}_{k+1} \leftarrow \{S \in \mathcal{C}_{k+1}, \text{supp}_D(S) \geq \theta\} \\
\quad k \leftarrow k + 1 \\
\textbf{return } \bigcup_i \mathcal{F}_i
Level-wise enumeration: *Apriori* algorithm

Candidate generation

\[
k \leftarrow 1 \\
F_k \leftarrow \{\text{all frequent singleton itemsets}\} \\
\textbf{while } F_k \neq \emptyset \textbf{ do} \\
\quad \text{Generate } C_{k+1} \text{ by extending itemsets from } F_k \\
\quad \text{Prune itemsets that violate downward closure} \\
\quad F_{k+1} \leftarrow \{S \in C_{k+1}, \text{supp}_D(S) \geq \theta\} \\
\quad k \leftarrow k + 1 \\
\textbf{return } \bigcup_i F_i
\]
Level-wise enumeration: Apriori algorithm

Candidate pruning

\[ k \leftarrow 1 \]
\[ \mathcal{F}_k \leftarrow \{ \text{all frequent singleton itemsets} \} \]
\[ \text{while } \mathcal{F}_k \neq \emptyset \text{ do} \]
\[ \quad \text{Generate } \mathcal{C}_{k+1} \text{ by extending itemsets from } \mathcal{F}_k \]
\[ \quad \text{Prune itemsets that violate downward closure} \]
\[ \quad \mathcal{F}_{k+1} \leftarrow \{ \mathcal{S} \in \mathcal{C}_{k+1}, \text{supp}_D(\mathcal{S}) \geq \theta \} \]
\[ \quad k \leftarrow k + 1 \]
\[ \text{return } \bigcup_i \mathcal{F}_i \]
Level-wise enumeration: \textit{Apriori} algorithm

Support counting

\begin{align*}
k & \leftarrow 1 \\
\mathcal{F}_k & \leftarrow \{\text{all frequent singleton itemsets}\} \\
\text{while } \mathcal{F}_k \neq \emptyset \text{ do} \\
& \quad \text{Generate } \mathcal{C}_{k+1} \text{ by extending itemsets from } \mathcal{F}_k \\
& \quad \text{Prune itemsets that violate downward closure} \\
& \quad \mathcal{F}_{k+1} \leftarrow \{S \in \mathcal{C}_{k+1}, \text{supp}_D(S) \geq \theta\} \\
& \quad k \leftarrow k + 1 \\
\text{return } \bigcup_i \mathcal{F}_i
\end{align*}
Pizzeria example: Apriori algorithm

Enumerate singleton itemsets

\[
\begin{array}{c}
\emptyset \\
1156 \\
h \\
j \\
m \\
o \\
t \\
\end{array}
\]

for \( \sigma = 289 \)
Pizzeria example: Apriori algorithm

Count supports

1156
∅

710 349 711 669 413
h j m o t

for $\sigma = 289$
Pizzeria example: $Apriori$ algorithm

Frequent singleton itemsets

\[
\begin{align*}
\emptyset & : 1156 \\
\{h\} & : 710 \\
\{j\} & : 349 \\
\{m\} & : 711 \\
\{o\} & : 669 \\
\{t\} & : 413 \\
\end{align*}
\]

for $\sigma = 289$
Pizzeria example: *Apriori* algorithm

Generate candidates itemsets of length 2

for $\sigma = 289$
Pizzeria example: \textit{Apriori} algorithm

Count supports

\[
\begin{array}{c}
\text{for } \sigma = 289
\end{array}
\]
Pizzeria example: *Apriori* algorithm

Frequent itemsets of length up to 2

![Itemset Diagram](image)

for \( \sigma = 289 \)
Pizzeria example: *Apriori* algorithm

Generate candidates itemsets of length 3

for $\sigma = 289$
Pizzeria example: *Apriori* algorithm

Prune candidates

for $\sigma = 289$
Pizzeria example: *Apriori* algorithm

Count supports

for $\sigma = 289$
Pizzeria example: Apriori algorithm

Frequent itemsets of length up to 3

for $\sigma = 289$
Pizzeria example: \textit{Apriori} algorithm

Generate candidates itemsets of length 4

\[
\begin{array}{c}
\{\emptyset\} \\
\{h\} \\
\{j\} \\
\{m\} \\
\{o\} \\
\{t\} \\
\{hj\} \\
\{hm\} \\
\{ho\} \\
\{ht\} \\
\{hmj\} \\
\{hmo\} \\
\{hmt\} \\
\{hot\} \\
\{mot\}
\end{array}
\]

\text{for } \sigma = 289
Pizzeria example: *Apriori* algorithm

Prune candidates

for $\sigma = 289$
Pizzeria example: Apriori algorithm

Frequent itemsets of length up to 4

for $\sigma = 289$
Pizzeria example: Enumeration tree

Items ordered alphabetically, prefix growth

for $\sigma = 289$
Pizzeria example: Enumeration tree

Items ordered by decreasing frequency, prefix growth

for $\sigma = 289$
Items ordered by increasing frequency, prefix growth

for $\sigma = 289$
Algorithms for mining frequent itemsets

Support counting is expensive

According to the monotonicity of support

\[ \forall J \subseteq I, \quad \text{supp}(I) \subseteq \text{supp}(J) \]

Make support counting more efficient

- Prune irrelevant transactions
- Reuse support counting from previous steps

Recursively project the database down the enumeration tree
Vertical apriori algorithm

\[ k \leftarrow 1 \]
\[ \mathcal{F}_k \leftarrow \{ \text{all frequent singleton itemsets} \} \]
Generate \textit{tid} list for each frequent singleton itemsets

\textbf{while} \ \mathcal{F}_k \neq \emptyset \ \textbf{do}

\begin{itemize}
  \item Generate \( C_{k+1} \) by joining pairs of itemsets from \( \mathcal{F}_k \)
  \item Prune itemsets that violate downward closure
  \item Generate \textit{tid} list for each candidate by intersecting \textit{tid} lists of associated pair of \( k \)-itemsets
\end{itemize}

\[ \mathcal{F}_{k+1} \leftarrow \{ S \in C_{k+1}, \supp_D(S) \geq \theta \} \]
\[ k \leftarrow k + 1 \]

\textbf{return} \( \bigcup_i \mathcal{F}_i \)
Vertical apriori algorithm

Vertical database representation

\[ k \leftarrow 1 \]
\[ \mathcal{F}_k \leftarrow \{ \text{all frequent singleton itemsets} \} \]
Generate \textit{tid} list for each frequent singleton itemsets

\textbf{while} \ \mathcal{F}_k \neq \emptyset \ \textbf{do}

Generate \( \mathcal{C}_{k+1} \) by joining pairs of itemsets from \( \mathcal{F}_k \)
Prune itemsets that violate downward closure
Generate \textit{tid} list for each candidate by intersecting \textit{tid} lists of associated pair of \( k \)-itemsets

\[ \mathcal{F}_{k+1} \leftarrow \{ S \in \mathcal{C}_{k+1}, \text{supp}_D(S) \geq \theta \} \]
\[ k \leftarrow k + 1 \]

\textbf{return} \( \bigcup_i \mathcal{F}_i \)
Space–time trade-off

Tid lists

- Allow to compute supports faster
- Require memory space for storage

Use dedicated data structures that support efficient counting
The **FP-tree** is a compact representation of the database

- Extract conditional projected database for a given suffix
- Update counts efficiently

**FP-growth** is a recursive suffix-based pattern growth algorithm.
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 1
Transaction mho
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 3
Transaction mj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 4
Transaction mhoj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step 5
Transaction mhoj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 6
Transaction h
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 7
Transaction mhtj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step#  8
Transaction  m
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 10
Transaction mhotj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 11
Transaction mht
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 12
Transaction mh
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 100
Transaction mho
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 101
Transaction mot
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 1155
Transaction ho

UEF/School of Computing BADA:FIM
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 1156
Transaction tj
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix $m$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth for $\sigma = 289$
Recursive pattern growth

for $\sigma = 289$

Suffix h
Recursive pattern growth

for $\sigma = 289$

Suffix mh
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix: o
Recursive pattern growth

for $\sigma = 289$

Suffix $mo$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix ho
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix mho
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix t
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix $mt$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix $ht$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

For $\sigma = 289$

Suffix $j$
Items ordered by decreasing frequency, suffix growth
Association rules
Association rules

Frequent itemsets can be used to generate association rules

Classical setting: Shopping basket data

- Identify products that are often bought together
  Frequent itemset \{butter, bread, ham, pickles\}
- Extract rules that capture typical buying behaviour
  Association rules \{bread, ham\} \Rightarrow \{butter, pickles\}
- Insights for marketing and shelf placement
Frequent itemsets can be used to generate association rules.

Consider two itemsets $X$ and $Y$ such that

$$X \subset U, \quad \emptyset \neq Y \subseteq U, \quad X \cap Y = \emptyset$$

The confidence of the association rule $X \Rightarrow Y$ is the *conditional probability* that a transaction contains $X \cup Y$ given that it contains $X$

$$\text{conf}(X \Rightarrow Y) = \frac{|\text{supp}(X \cup Y)|}{|\text{supp}(X)|}$$

$X$ and $Y$ are called the *antecedent* and *consequent* of the rule, respectively.
$X \Rightarrow Y$ is an association rule at minimum support $\sigma$ and minimum confidence $\gamma$ if

$$\text{supp}(X \cup Y) \geq \sigma \quad \text{and} \quad \text{conf}(X \Rightarrow Y) \geq \gamma$$
1. Mine all the frequent itemsets for minimum support $\sigma$.
2. Split the frequent itemsets into association rules of minimum confidence $\gamma$.

Monotonicity of confidence:
Let $X_a$, $X_b$ and $I$ be itemsets such that $X_a \subseteq X_b \subseteq I$, then

$$\text{conf}(X_b \Rightarrow I \setminus X_b) \geq \text{conf}(X_a \Rightarrow I \setminus X_a)$$