What Is Association Rule Mining?

- Association rule mining
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items in transaction databases
  - Understand customer buying habits by finding associations and correlations between the different items that customers place in their "shopping basket"

- Applications
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, web log analysis, fraud detection (supervisor->examiner)

Rule form

- Antecedent $\rightarrow$ Consequent [support, confidence]

(support and confidence are user defined measures of interestingness)

Examples

- buys(x, "computer") $\rightarrow$ buys(x, "financial management software") [0.5%, 60%]
- age(x, "30..39") ^ income(x, "42..48K") $\rightarrow$ buys(x, "car") [1%, 75%]
How can Association Rules be used?

Stories – Beer and Diapers

• Diapers and Beer. Most famous example of market basket analysis for the last few years. If you buy diapers, you tend to buy beer.
• T. Bischok headed Terradata’s Industry Consulting group.
• K. Heath ran self join in SQL (1990), trying to find two itemsets that have baby items, which are particularly profitable.
• Found this pattern in their data of 50 stores/90 day period.
• Unlikely to be significant, but it’s a nice example that explains associations well.

Probably mom was calling dad at work to buy diapers on way home and he decided to buy a six-pack as well.

The retailer could move diapers and beers to separate places and position high-profit items of interest to young fathers along the path.

How can Association Rules be used?

Let the rule discovered be

{Bagels,...} → {Potato Chips}

• Potato chips as consequent ⇒ Can be used to determine what should be done to boost its sales
• Bagels in the antecedent ⇒ Can be used to see which products would be affected if the store discontinues selling bagels
• Bagels in antecedent and Potato chips in the consequent ⇒ Can be used to see what products should be sold with Bagels to promote sale of Potato Chips

Basic Concepts

• Given:
  (1) database of transactions,
  (2) each transaction is a list of items purchased by a customer in a visit

• Find:
  all rules that correlate the presence of one set of items (itemset) with that of another set of items

  E.g., 35% of people who buys salmon also buys cheese

Application

• "Baskets" = documents; "items" = words in those documents.
  • Lets us find words that appear together unusually frequently, i.e., linked concepts.

• "Baskets" = sentences, "items" = documents containing those sentences.
  • Items that appear together too often could represent plagiarism.
Applications

- “Baskets” = Web pages; “items” = linked pages.
- Pairs of pages with many common references may be about the same topic.
- “Baskets” = Web pages $p$; “items” = pages that link to $p$.
- Pages with many of the same links may be mirrors or about the same topic.

Rule basic Measures

$A \Rightarrow B \ [s, c]$

**Support:** denotes the frequency of the rule within transactions. A high value means that the rule involves a great part of database.

$$\text{support}(A \Rightarrow B \ [s, c]) = p(A \cup B)$$

**Confidence:** denotes the percentage of transactions containing $A$ which also contain $B$. It is an estimation of conditioned probability.

$$\text{confidence}(A \Rightarrow B \ [s, c]) = \frac{p(B|A)}{sup(A)} = \frac{\text{sup}(A,B)}{\text{sup}(A)}.$$  

Example

**Itemset:**

- $A,B$ or $B,E,F$

**Support of an itemset:**

$\text{Sup}(A,B)=1$ $\text{Sup}(A,C)=2$

**Frequent pattern:**

- Given min. sup=2, $\{A,C\}$ is a frequent pattern

For minimum support $= 50\%$ and minimum confidence $= 50\%$, we have the following rules:

- $A \Rightarrow C$ with 50% support and 66% confidence
- $C \Rightarrow A$ with 50% support and 100% confidence

Mining Association Rules

- What is Association rule mining
- **Apriori Algorithm**
- Measures of rule interestingness
- Advanced Techniques
Boolean association rules

Each transaction is converted to a Boolean vector

An Example

For rule \( A \Rightarrow C \):

\[
\text{support} = \text{support}\left(\{A, C\}\right) = 50\%
\]

\[
\text{confidence} = \frac{\text{support}\left(\{A, C\}\right)}{\text{support}(A)} = 66.6\%
\]

Finding Association Rules

- Approach:
  - 1) Find all itemsets that have high support
    - These are known as frequent itemsets
  - 2) Generate association rules from frequent itemsets

Mining Frequent Itemsets (the Key Step)

- Find the frequent itemsets: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - Generate length \((k+1)\) candidate itemsets from length \(k\) frequent itemsets, and
    - Test the candidates against DB to determine which are in fact frequent
Apriori principle

Any subset of a frequent itemset must be frequent

- A transaction containing \{beer, diaper, nuts\} also contains \{beer, diaper\}
- \{beer, diaper, nuts\} is frequent $\rightarrow$ \{beer, diaper\} must also be frequent

Apriori principle

- No superset of any infrequent itemset should be generated or tested
  - Many item combinations can be pruned

Itemset Lattice

Apriori principle for pruning candidates

If an itemset is infrequent, then all of its supersets must also be infrequent

Found to be Infrequent

Pruned supersets
How to Generate Candidates?

- Step 1: self-joining $L_{k-1}$

  insert into $C_k$
  
  select $p.item_1, p.item_k, ..., q.item_{k-1}$
  from $L_{k-1} p, L_{k-1} q$
  
  where $p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$

How to Generate Candidates?

- Step 2: pruning

  for all itemsets $c$ in $C_k$
do
  
  for all $(k-1)$-subsets $s$ of $c$ do

  if ($s$ is not in $L_{k-1}$) then delete $c$ from $C_k$

Example of Generating Candidates

- $L_3 = \{abc, abd, acd, ace, bcd\}$

  **Self-joining:** $L_3 * L_3$

  - $abcd$ from $abc$ and $abd$
  - $acde$ from $acd$ and $ace$

  **Pruning** (before counting its support):

  - $acde$ is removed because ade is not in $L_3$

- $C_F = \{abcd\}$
The Apriori Algorithm

- $C_k$: Candidate itemset of size $k$  
- $L_k$: frequent itemset of size $k$

**Join Step:** $C_k$ is generated by joining $L_{k-1}$ with itself

**Prune Step:** Any (k-1)-itemset that is not frequent cannot be a subset of a frequent $k$-itemset

**Algorithm:**

$L_1 = \{\text{frequent items}\}$

for $(k = 1; L_k \neq \emptyset; k++)$

1. $C_{k+1}$ = candidates generated from $L_k$
2. for each transaction $t$ in database do
   - increment the count of all candidates in $C_{k+1}$ that are contained in $t$
3. $L_{k+1}$ = candidates in $C_{k+1}$ with min_support

end

return $L = \bigcup_k L_k$

---

Counting Support of Candidates

- Why counting supports of candidates a problem?
  - The total number of candidates can be very large
  - One transaction may contain many candidates

**Method:**

- Candidate itemsets are stored in a hash-tree
- Match candidates to transactions supporting it

---

Counting Support of Candidate Itemsets

- Frequent Itemsets are found by counting candidates
- Naive way:
  - Search for each candidate in each transaction
  - Expensive!!

<table>
<thead>
<tr>
<th>Candidates</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
</tbody>
</table>

Naïve approach requires $O(NM)$ comparisons

Reduce the number of comparisons (NM) by using hash tables to store the candidate itemsets

---

Reducing Number of Comparisons

- Store the candidate itemsets in a hash structure
  - Hash each transaction to its appropriate buckets
  - Match transaction only to candidates stored in the hashed bucket(s)

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transactions</th>
<th>Hash Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
</tr>
<tr>
<td>3</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>M</td>
</tr>
<tr>
<td>5</td>
<td>N</td>
</tr>
</tbody>
</table>

Buckets
Generating AR from frequent itemsets

- Confidence \((A \Rightarrow B) = P(B|A) = \frac{\text{support_count}(\{A,B\})}{\text{support_count}(\{A\})}\)
- For every frequent itemset \(x\), generate all non-empty subsets of \(x\)
- For every non-empty subset \(s\) of \(x\), output the rule "\(s \Rightarrow (x-s)\)" if \(\frac{\text{support_count}(\{x\})}{\text{support_count}(\{s\})} \geq \text{min_conf}\)

From Frequent Itemsets to Association Rules

- Q: Given frequent set \(\{A,B,E\}\), what are possible association rules?
  - \(A \Rightarrow B, E\)
  - \(A, B \Rightarrow E\)
  - \(A, E \Rightarrow B\)
  - \(B \Rightarrow A, E\)
  - \(B, E \Rightarrow A\)
  - \(E \Rightarrow A, B\)
  - \(\_ \Rightarrow A, B, E\) (empty rule), or true \(\Rightarrow A, B, E\)

Rule Generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - \(L = \{A,B,C,D\}\):
    \[
    c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)
    \]
    - Confidence is non-increasing as number of items in rule consequent increases

Generating Rules: example

<table>
<thead>
<tr>
<th>Trans-ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACD</td>
</tr>
<tr>
<td>2</td>
<td>BCE</td>
</tr>
<tr>
<td>3</td>
<td>ABCE</td>
</tr>
<tr>
<td>4</td>
<td>BE</td>
</tr>
<tr>
<td>5</td>
<td>ABCE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Conf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{BC} =&gt; {E}</td>
<td>100%</td>
</tr>
<tr>
<td>{BE} =&gt; {C}</td>
<td>75%</td>
</tr>
<tr>
<td>{CE} =&gt; {B}</td>
<td>100%</td>
</tr>
<tr>
<td>{B} =&gt; {CE}</td>
<td>75%</td>
</tr>
<tr>
<td>{C} =&gt; {BE}</td>
<td>75%</td>
</tr>
<tr>
<td>{E} =&gt; {BC}</td>
<td>75%</td>
</tr>
</tbody>
</table>

Min_support: 60%
Min_confidence: 75%
Exercício

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk, Chips, Mustard</td>
</tr>
<tr>
<td>2</td>
<td>Beer, Diaper, Bread, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Beer, Coke, Diaper, Milk</td>
</tr>
<tr>
<td>4</td>
<td>Beer, Bread, Diaper, Milk, Chips</td>
</tr>
<tr>
<td>5</td>
<td>Coke, Bread, Diaper, Milk</td>
</tr>
<tr>
<td>6</td>
<td>Beer, Bread, Diaper, Milk, Mustard</td>
</tr>
<tr>
<td>7</td>
<td>Coke, Bread, Diaper, Milk</td>
</tr>
</tbody>
</table>

Converte os dados para o formato booleano e para um suporte de 40%, aplique o algoritmo Apriori.

<table>
<thead>
<tr>
<th>Bread</th>
<th>Milk</th>
<th>Chips</th>
<th>Mustard</th>
<th>Beer</th>
<th>Diaper</th>
<th>Eggs</th>
<th>Coke</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

0.4*7 = 2.8

<table>
<thead>
<tr>
<th>C1</th>
<th>L1</th>
<th>C2</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread 6</td>
<td>Milk 6</td>
<td>Bread 5</td>
<td>Milk 5</td>
</tr>
<tr>
<td>Milk 5</td>
<td>Bread 4</td>
<td>Milk 3</td>
<td>Beer 3</td>
</tr>
<tr>
<td>Chips 2</td>
<td>Mustard 2</td>
<td>Beer 4</td>
<td>Milk 3</td>
</tr>
<tr>
<td>Mustard 2</td>
<td>Diaper 6</td>
<td>Coke 3</td>
<td>Dairy 4</td>
</tr>
<tr>
<td>Beer 4</td>
<td>Diaper 6</td>
<td>Milk, Beer 3</td>
<td>Coke 3</td>
</tr>
<tr>
<td>Diaper 6</td>
<td>Coke 3</td>
<td>Milk, Coke 3</td>
<td>Beer, Diaper 4</td>
</tr>
<tr>
<td>Coke 3</td>
<td>Milk, Coke 3</td>
<td>Beer, Diaper 4</td>
<td>Coke 3</td>
</tr>
</tbody>
</table>

C3

<table>
<thead>
<tr>
<th>Bread, Milk, Beer 2</th>
<th>Bread, Milk, Diaper 4</th>
<th>Bread, Beer, Diaper 3</th>
<th>Milk, Beer, Diaper 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread, Milk, Diaper 4</td>
<td>Bread, Beer, Diaper 3</td>
<td>Milk, Beer, Diaper 3</td>
<td>Milk, Diaper, Coke 3</td>
</tr>
<tr>
<td>Milk, Beer, Coke 3</td>
<td>Milk, Diaper, Coke 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8 + \( C^3_2 + C^3_3 \) = 92 >> 24

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates

- Improving Apriori: general ideas
  - Reduce number of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Improving Apriori’s Efficiency

- Problem with Apriori: every pass goes over whole data.
- AprioriTID: Generates candidates as apriori but DB is used for counting support only on the first pass.
  - Needs much more memory than Apriori
  - Builds a storage set \( C^k \) that stores in memory the frequent sets per transaction
- AprioriHybrid: Use Apriori in initial passes; Estimate the size of \( C^k \). Switch to AprioriTid when \( C^k \) is expected to fit in memory
  - The switch takes time, but it is still better in most cases
Improving Apriori's Efficiency

- Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans.

- Sampling: mining on a subset of given data.
  - The sample should fit in memory.
  - Use lower support threshold to reduce the probability of missing some itemsets.
  - The rest of the DB is used to determine the actual itemset count.

Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB (2 DB scans).

- (support in a partition is lowered to be proportional to the number of elements in the partition)

Phase I

- Trans. in D
- Divide D into n Non-overlapping partitions
- Find frequent itemsets local to each partition (parallel alg.)

Phase II

- Combine results to form a global set of candidate itemset
- Find global frequent itemsets among candidates
- Freq. itemsets in D

Dynamic itemset counting: partitions the DB into several blocks each marked by a start point.

- At each start point, DIC estimates the support of all itemsets that are currently counted and adds new itemsets to the set of candidate itemsets if all its subsets are estimated to be frequent.
- If DIC adds all frequent itemsets to the set of candidate itemsets during the first scan, it will have counted each itemset's exact support at some point during the second scan;
- thus DIC can complete in two scans.

Dynamic itemset counting: partitions the DB into several blocks each marked by a start point.

- At each start point, DIC estimates the support of all itemsets that are currently counted and adds new itemsets to the set of candidate itemsets if all its subsets are estimated to be frequent.
- If DIC adds all frequent itemsets to the set of candidate itemsets during the first scan, it will have counted each itemset's exact support at some point during the second scan;
- thus DIC can complete in two scans.
Comment

- Traditional methods such as database queries:
  - support hypothesis verification about a relationship such as the co-occurrence of diapers & beer.

- Data Mining methods automatically discover significant associations rules from data.
  - Find whatever patterns exist in the database, without the user having to specify in advance what to look for (data driven).
  - Therefore allow finding unexpected correlations

Mining Association Rules

- What is Association rule mining
- Apriori Algorithm
- **Measures of rule interestingness**
- Advanced Techniques

Interestingness Measurements

- Are all of the strong association rules discovered interesting enough to present to the user?
- How can we measure the interestingness of a rule?

- Subjective measures
  - A rule (pattern) is interesting if
    - it is *unexpected* (surprising to the user); and/or
    - *actionable* (the user can do something with it)
    - (only the user can judge the interestingness of a rule)

Objective measures of rule interest

- Support
- Confidence or strength
- Lift or Interest or Correlation
- Conviction
- Leverage or Piatetsky-Shapiro
- Coverage
Criticism to Support and Confidence

- Example 1: (Aggarwal & Yu, PODS98)
  - Among 5000 students
    - 3000 play basketball
    - 3750 eat cereal
    - 2000 both play basketball and eat cereal
  - 2000 basketball not basketball sum(row)
  - cereal 2000 1750 3750 75%
  - not cereal 1000 250 1250 25%
  - sum(col.) 3000 2000 5000

  play basketball ⇒ eat cereal [40%, 66.7%]
  misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
  play basketball ⇒ not eat cereal [20%, 33.3%]
  is more accurate, although with lower support and confidence

Lift of a Rule

- Lift (Correlation, Interest)
  \[
  \text{Lift}(A \rightarrow B) = \frac{\text{sup}(A,B)}{\text{sup}(A) \cdot \text{sup}(B)} = \frac{P(B | A)}{P(B)}
  \]
  - A and B negatively correlated, if the value is less than 1:
    otherwise A and B positively correlated
  - Example 1 (cont)
    - play basketball ⇒ eat cereal [40%, 66.7%]
      \[
      \text{LIFT} = \frac{2000}{5000 \times 5000} = 0.89
      \]
    - play basketball ⇒ not eat cereal [20%, 33.3%]
      \[
      \text{LIFT} = \frac{1000}{5000 \times 5000} = 1.33
      \]

Problems With Lift

- Rules that hold 100% of the time may not have the highest possible lift. For example, if 5% of people are Vietnam veterans and 90% of the people are more than 5 years old, we get a lift of 0.05/(0.05*0.9)=1.11 which is only slightly above 1 for the rule
  - Vietnam veterans ⇒ more than 5 years old.

- And, lift is symmetric:
  - not eat cereal ⇒ play basketball [20%, 80%]
    \[
    \text{LIFT} = \frac{1000}{5000 \times 5000} = 1.33
    \]
**Conviction of a Rule**

Note that $A \rightarrow B$ can be rewritten as $\neg(A, \neg B)$

$$Conv(A \rightarrow B) = \frac{sup(A) \cdot sup(\overline{B})}{sup(A, \overline{B})} = \frac{P(A) \cdot P(\overline{B})}{P(A, \overline{B})} = \frac{P(A)(1 - P(B))}{P(A) - P(A, B)}$$

- Conviction is a measure of the implication and has value 1 if items are unrelated.

- **play basketball $\Rightarrow$ eat cereal** [40%, 66.7%]  
  - eat cereal $\Rightarrow$ play basketball $\ conv: 0.85$

- **play basketball $\Rightarrow$ not eat cereal** [20%, 33.3%]  
  - not eat cereal $\Rightarrow$ play basketball $\ conv: 1.43$

**Leverage of a Rule**

- **Leverage or Piatetsky-Shapiro**
  
  $$PS(A \rightarrow B) = sup(A, B) - sup(A) \cdot sup(B)$$

- **PS (or Leverage):**
  - is the proportion of additional elements covered by both the premise and consequence above the expected if independent.

**Coverage of a Rule**

$$\text{coverage}(A \rightarrow B) = sup(A)$$

**Association Rules Visualization**

The coloured column indicates the association rule $B \rightarrow C$. Different icon colours are used to show different metadata values of the association rule.
The Ball graph Explained

- A ball graph consists of a set of nodes and arrows. All the nodes are yellow, green or blue. The blue nodes are active nodes representing the items in the rule in which the user is interested. The yellow nodes are passive representing items related to the active nodes in some way. The green nodes merely assist in visualizing two or more items in either the head or the body of the rule.

- A circular node represents a frequent (large) data item. The volume of the ball represents the support of the item. Only those items which occur sufficiently frequently are shown.

- An arrow between two nodes represents the rule implication between the two items. An arrow will be drawn only when the support of a rule is no less than the minimum support.
Association Rules Visualization

Mining Association Rules

- What is Association rule mining
- Apriori Algorithm
- FP-tree Algorithm
- Additional Measures of rule interestingness
- Advanced Techniques

Multiple-Level Association Rules

- Items often form hierarchy.
- Flexible support settings: Items at the lower level are expected to have lower support.

- Fresh ⇒ Bakery [20%, 60%]
- Dairy ⇒ Bread [6%, 50%]
- Fruit ⇒ Bread [1%, 50%] is not valid

Multi-Dimensional Association Rules

- Single-dimensional rules:
  - \( \text{buys}(X, "milk") \Rightarrow \text{buys}(X, "bread") \)

- Multi-dimensional rules: \( \geq 2 \) dimensions or predicates
  - Inter-dimension association rules (no repeated predicates)
    - \( \text{age}(X, "19-25") \land \text{occupation}(X, "student") \Rightarrow \text{buys}(X, "coke") \)
  - Hybrid-dimension association rules (repeated predicates)
    - \( \text{age}(X, "19-25") \land \text{buys}(X, "popcorn") \Rightarrow \text{buys}(X, "coke") \)
Quantitative Association Rules

\[ \text{age}(X, "30-34") \land \text{income}(X, "24K - 48K") \Rightarrow \text{buys}(X, "high resolution TV") \]

Mining Sequential Patterns

10% of customers bought "Foundation" and "Ringworld" in one transaction, followed by "Ringworld Engineers" in another transaction.

Sequential Pattern Mining

- **Given**
  - A database of customer transactions ordered by increasing transaction time
  - Each transaction is a set of items
  - A sequence is an ordered list of itemsets

- **Example:**
  - 10% of customers bought "Foundation" and "Ringworld" in one transaction, followed by "Ringworld Engineers" in another transaction.
  - 10% is called the support of the pattern (a transaction may contain more books than those in the pattern)

- **Problem**
  - Find all sequential patterns supported by more than a user-specified percentage of data sequences

Sequential Patterns Rules

- \[ \langle (F, R) \rangle (RE) \Rightarrow (FE) \] with 3% support and 40% confidence.

- **Confidence:** 40% of occurrences of \[ \langle (F, R) \rangle (RE) \] are followed by (FE).
  - (FE - foundation and empire)

- **Problem Decomposition:**
  - Find all sequential patterns with minimum support.
  - Use the sequential patterns to generate rules.

Applications

- **Applications of sequential pattern mining**
  - Attached mailing, e.g., customized mailings for a book club.
  - Customer satisfaction / retention
  - Web log analysis
  - Medical research, e.g., sequences of symptoms

Applications
The AprioriAll Algorithm [AgSr95]

- **A 3-phase algorithm**
  - **Phase 1:** finds all frequent itemsets with minimum support, that is, with a minimum number of customers containing it
  - **Phase 2:** transforms the DB s.t. each transaction is replaced by the set of all frequent itemsets contained in the transaction
  - **Phase 3:** finds all sequential patterns

### The AprioriAll Algorithm (cont.)

- **Phase 2:** transforms the DB s.t. each transaction is replaced by the set of all frequent itemsets contained in the transaction

<table>
<thead>
<tr>
<th>Customer Id</th>
<th>Original Customer Sequence</th>
<th>Transformed Customer Sequence</th>
<th>After Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{300} (900)</td>
<td>{((300))} (900)</td>
<td>((1) (9))</td>
</tr>
<tr>
<td>2</td>
<td>{400} 60 70 900</td>
<td>{((400) (60) (70) (900))}</td>
<td>(1) {2, 3, 4}</td>
</tr>
<tr>
<td>3</td>
<td>{30 60 70}</td>
<td>{((30) (60) (70))}</td>
<td>(1) {3}</td>
</tr>
<tr>
<td>4</td>
<td>{30} 60 70 900</td>
<td>{((30) (60) (70) (900))}</td>
<td>(1) {3, 4, 5}</td>
</tr>
<tr>
<td>5</td>
<td>{900}</td>
<td>{((900))}</td>
<td>(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Large Itemsets</th>
<th>Mapped to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(40)</td>
<td>1</td>
</tr>
<tr>
<td>(60)</td>
<td>2</td>
</tr>
<tr>
<td>(70)</td>
<td>3</td>
</tr>
<tr>
<td>(900)</td>
<td>4</td>
</tr>
<tr>
<td>(90)</td>
<td>5</td>
</tr>
</tbody>
</table>

### Application Difficulties

- Wal-Mart knows that customers who buy Barbie dolls (it sells one every 20 seconds) have a 60% likelihood of buying one of three types of candy bars. What does Wal-Mart do with information like that?
- 'I don’t have a clue,' says Wal-Mart’s chief of merchandising, Lee Scott.

See - KDnuggets 98:01 for many ideas
www.kdnuggets.com/news/98/n01.html

### Some Suggestions

- By increasing the price of Barbie doll and giving the type of candy bar free, Wal-Mart can reinforce the buying habits of that particular types of buyer
- Highest margin candy to be placed near dolls.
- Special promotions for Barbie dolls with candy at a slightly higher margin.
- Take a poorly selling product X and incorporate an offer on this which is based on buying Barbie and Candy. If the customer is likely to buy these two products anyway then why not try to increase sales on X?
- Probably they can not only bundle candy of type A with Barbie dolls, but can also introduce new candy of Type N in this bundle while offering discount on whole bundle. As bundle is going to sell because of Barbie dolls & candy of type A, candy of type N can get free ride to customers houses. And with the fact that you like something, if you see it often, Candy of type N can become popular.
References

- Jiawei Han and Micheline Kamber, “Data Mining: Concepts and Techniques”, 2000
- Vipin Kumar and Mahesh Joshi, “Tutorial on High Performance Data Mining”, 1999
  (http://www.cs.tau.ac.il/~fiat/dmsem03/Fast%20Algorithms%20for%20Mining%20Association%20Rules.ppt)
- Alípio Jorge, “selecção de regras: medidas de interesse e meta queries”,

Thank you !!! Thank you !!!!