KNOWLEDGE-BASED COMPUTATIONAL INTELLIGENCE AND DATA MINING AND BIOMEDICINE

Data Mining and Knowledge Exploration

Adrian Horzyk

horzyk@agh.edu.pl

AGH

AGH University of Science and Technology
Krakow, Poland

DATA ANALYSIS PROBLEM

- ✓ Most repositories contain hidden information (which may enrich our knowledge) in the form of regularity, correlations, similarities, trends, peculiarities, rules ..., but their structure and size prevent their "manual" analysis.
- ✓ In recent years, many interesting and effective algorithms, structures, and methods of semi-automatic or automatic data mining have been created, also known as knowledge discovery.
- ✓ Data mining is a mental abbreviation that means exploring knowledge from data, or more precisely, extracting information from data that can shape knowledge of individuals.
- ✓ These algorithms are also used to discover relationships between elements or groups of objects that are saved in the form of the so-called association rules.

KNOWLEDGE EXPLORATION FROM DATA

Data mining depending on:

- **the type of data**
- **the method of data storage**

can be carried out through:

statistical methods, rules, decision trees or diagrams, testing of concluding, subsets, closeness or similarities, searching for frequent or rare patterns, fuzzy systems, neural networks, associative methods etc.

The data can create characteristic patterns in the form of:

- **Sets (e.g. entities, vectors, matrices),**
- **Sequences (e.g. texts, instruction strings, time sequences),**
- **Complex structures (e.g. subgraphs, images, maps, textures).**

COLLECTIONS AND SUPPORT

- ✓ Set of elements (*itemset*) $I = \{i_1, i_2, ..., i_N\}$ – is a set of all available elements (objects, items), where $N \geq 1$.
- ✓ Transaction T is a pair of T = (id, X) consisting of transaction id and a certain subset of items $X \subseteq I$, assuming some finite number of transaction id $\in$ Tid = $\{id_1, id_2, ..., id_M\}$.
- ✓ From the point of view of data mining, we are interested in the frequency of patterns W, the repeatability of various k-element subsets (k-sets) in the transaction set called transaction base D = $\{T_1, T_2, ..., T_M\}$.
- ✓ *k-itemset* is a k-element set $X = \{x_1, ..., x_K\} \subseteq I$, which usually defines a certain transaction $T_m = (id_m, X)$ or the pattern $W \subseteq I$, e.g.
  W={coffee, sugar} $\subseteq$ X={coffee, sugar, eggs} $\subseteq I$,
  W={coffee, sugar} $\subseteq$ I={coffee, milk, sugar, nuts, eggs, bread, butter, honey}.
- ✓ We say that transaction T = (id, X) covers a pattern (e.g. a set of items) W, if $W \subseteq X$. Pattern W can be covered by many transactions. The set of transactions covering the pattern W is denoted as cover cover(W,D) = {T $\in$ D: T covers W}.
- ✓ We say that the pattern W is frequent if its coverage by transactions from the transaction base D under consideration is not less than the chosen threshold σ.
- ✓ Mining frequent patterns is one of the basic tasks of data mining and an indispensable step in the extraction of Association Rule Mining and data correlation analysis.

COLLECTIONS AND SUPPORT

- ✓ ***Support s*** – is the frequency of occurrences of the **pattern W** (set of elements X) in the analyzed set of entities or transactions expressed as a percentage. Support is calculated as the ratio of the number of occurrences of the **pattern W** in the considered set of transactions D, expressed as |cover(W,D)|, in relation to the number of all considered transactions M =|D|:

  $s = |cover(W,D)| \ / M$

- ✓ Using the definition of support, we say that the **pattern W** is frequent if its support is not less than the threshold:
  threshold σ (*min support*): $\sigma = s_{min}$ (*minimum support*).

EXAMPLE: For the σ = 50% threshold and transaction set, specify which elements are frequent?

FREQUENT > 50%

- ✓ Sugar (80%)
- ✓ Coffee (60%)
- ✓ Eggs (60%)
- ✓ Milk (40%)
- ✓ Nuts (40%)
- ✓ Butter (40%)
- ✓ Bread (20%)
- ✓ Honey (20%)

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

ASSOCIATION RULES

- ✓ **Association rules for transaction / pattern elements: X → Y (s, c).**
- ✓ **Support for association rules is defined by the probability that a specific transaction contains both X and Y, i.e. X ∪ Y. This probability is calculated relative to all possible transactions, expressing the probability of such association, i.e. the occurrence of such association rule.**
- ✓ ***Confidence* c – is the conditional probability p(Y|X) that the transaction containing X also contains Y.**
- ✓ **The exploration of association rules consists of finding all X → Y rules with a specific minimum support s_{min} and a certain minimum confidence c_{min} : e.g. $s \geq s_{min} = 40\%$ and $c \geq c_{min} = 50\%$. Then we call such an association rule strong.**
- ✓ **Multidimensional association rules contain extensive rules, i.e. age:**
  age (X, „18-24”) ∧ job (X, „student”) ⇒ buys (X, „cola”)
  age (X, „18-24”) ∧ buys (X, „pop-corn”) ⇒ buys (X, „cola”)

ASSOCIATION RULES

Associative rule exploration means searching for rules that predict the occurrence of an element based on the occurrence of other elements in the transaction.

Association rules are applied e.g. to:

- organizing promotions and tying (up-selling and cross-selling),
- constructing shipping catalogs,
- determining the manner of placing goods on shelves in supermarkets,
- determining the best rotating goods.

ASSOCIATION RULES:

- ✓ Coffee → Sugar (60%, 100%)
- ✓ Sugar → Coffee (60%, 75%)
- ✓ Sugar → Eggs (40%, 50%)
- ✓ Eggs → Sugar (40%, 67%)

ARE NOT FOR: s ≥ 50%, c ≥ 50%:

- ❖ Coffee → Eggs (20%, 33%)
- ❖ Eggs → Coffee (20%, 33%)

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

DETERMINATION OF ASSOCIATION RULES

- ✓ **Support s** is the number of transactions containing both X and Y, i.e. $X \cup Y$.
- ✓ **Confidence c** – is the conditional probability p(Y|X) that the transaction containing X also contains Y.

$$s\ (A \Rightarrow B, D) = \frac{|\text{cover}(A \cup B\ ,\ D)|}{|D|}$$

The method of calculating the **support s** and the **confidence c** for the association rules specified for which **s ≥ 40%**, **c ≥ 50%**:

$$c\ (A \Rightarrow B, D) = \frac{s\ (A \cup B\ ,\ D)}{s\ (A\ ,\ D)}$$

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

Coffee → Sugar (s = 60% = 3 / 5, c = 100% = 3 / 3)

Sugar → Coffee (s = 60% = 3 / 5, c = 75% = 3 / 4)

Sugar → Eggs (s = 40% = 2 / 5, c = 50% = 2 / 4)

Eggs → Sugar (s = 40% = 2 / 5, c = 67% = 2 / 3)

DETERMINATION OF ASSOCIATION RULES

- ✓ **Support s** is the number of transactions containing both X and Y, i.e. $X \cup Y$.
- ✓ **Confidence c** – is the conditional probability p(Y|X) that the transaction containing X also contains Y.

$$s\ (A \Rightarrow B, D) = \frac{|cover(A \cup B\ ,\ D)|}{|D|}$$

The method of calculating the **support s** and the **confidence c** for the association rules specified for which **s ≥ 40%**, **c ≥ 50%**:

$$c\ (A \Rightarrow B, D) = \frac{s\ (A \cup B\ ,\ D)}{s\ (A\ ,\ D)}$$

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

Eggs → Sugar (s = 40% = 2 / 5, c = 67% = 2 / 3)

Coffee → Sugar (s = 60% = 3 / 5, c = 100% = 3 / 3)

Sugar → Coffee (s = 60% = 3 / 5, c = 75% = 3 / 4)

Sugar → Eggs (s = 40% = 2 / 5, c = 50% = 2 / 4)

DETERMINATION OF ASSOCIATION RULES

- ✓ **Support s** is the number of transactions containing both X and Y, i.e. $X \cup Y$.
- ✓ **Confidence c** – is the conditional probability p(Y|X) that the transaction containing X also contains Y.

$$s\ (A \Rightarrow B, D) = \frac{|cover(A \cup B\ ,\ D)|}{|D|}$$

How to calculate **support (s)** and **confidence (c)** for **a multidimensional association rule**:

$$c\ (A \Rightarrow B, D) = \frac{s\ (A \cup B\ ,\ D)}{s\ (A\ ,\ D)}$$

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

Eggs $\wedge$ Sugar → Coffee (s = 20% = 1 / 5, c = 50% = 1 / 2)

ASSOCIATION RULES

- ✓ **Association rules** resemble decision rules, but the decision (i.e. the right side of the implication) is not predetermined.
- ✓ **Association rules** work similarly to unsupervised training for problems of grouping algorithms (clustering). Such an algorithm has no predefined correct answer. Instead, it should describe the internal relationships between attributes.
- ✓ **Association rules** came from research on the issues of **Market Basket Analysis** consisting in discovering patterns of customer behavior, i.e. finding product groups bought together and determining their frequency:

$$\bigwedge_{i \in I} a_i = v_i \Rightarrow a_k = v_k$$

- ✓ We use two indicators to measure the objectivity of association rules:
  - ✓ **Support** - specifying how many percents of the transactions tested occur together, e.g. how many transactions occur in coffee and sugar simultaneously.
  - ✓ **Confidence** - determines how many percents of transactions the application contains (decision, i.e. the left side of the implication) assuming that the left side of the implication (transaction) is met, i.e.: $c\,(X \Rightarrow Y) = s\,(X \cup Y)\,/\,s\,(X)$.
- ✓ Appropriate levels of required support and reliability are determined by the user based on the needs arising from the given field, expert knowledge, tasks, etc.
- ✓ An association rule is called **strong** if $s \geq s_{min}$ and $c \geq c_{min}$.

CREATION OF ASSOCIATION RULES

- ✓ Creating association rules is usually preceded by the process of setting frequent patterns for which we create these rules, because usually (e.g. in trade) what is often repeated is important, e.g. important are often bought products that occur together in various transactions. This allows you to better locate these products on store shelves in order to increase their sales.
- ✓ Therefore, we are usually looking for strong association rules with properly defined minimum support and minimum confidence.
- ✓ Sometimes patterns that were first or rarely interesting may be interesting, e.g. in astronomy, physics (e.g. hadron collider), genetics, cognitive science, etc., then only patterns with low support or unique patterns may be sought.
- ✓ For setting association rules and searching for common patterns, the very popular Apriori algorithm is used, which also has numerous extensions to accelerate its operation.

SUBPATTERNS

- ✓ Large and long patterns contain (combinatorically speaking) a large number of sub-patterns:
  - ✓ subsets of items
  - ✓ subsequences of elements
  - ✓ subgraphs of elements
  - ✓ segments/areas of elements (e.g. for images, maps)

- ✓ Subpatterns allow you to find similarities and differences and create associative relationships between patterns.
- ✓ Due to the sometimes high combinatorial complexity necessary to make comparisons, it pays to first analyze and compare subpatterns with fewer constituent objects, and only then determine, for example, frequency, rarity, support, or certainty for patterns with more constituents.

CLOSED PATTERNS

- ✓ **Closed patterns X** are patterns that are **frequent** and there is no super-pattern Y ⊃ X, that has **the same support** as the X pattern.
- ✓ **The closed pattern** is, therefore, a lossy compression of all the frequent patterns contained in them, because information about their support is lost.

Example: Is the "coffee" pattern a closed one? It is a common pattern with ≥ 50%, support, but it is not a closed pattern, as there is the "coffee ∪ sugar" pattern that has the same support of 60% and contains it. However, the "coffee ∪ sugar" pattern is closed because there is no pattern with the same support that would cover it.

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

MAXIMUM PATTERNS

- ✓ **The maximum patterns (max-patterns) X are those that are frequent and there is no super-pattern Y ⊃ X.**
- ✓ **The max-patterns represent all frequent patterns that all elements contain.**
- ✓ **The max-pattern is, therefore, a lossy compression of all common patterns consisting of its elements.**

Example: Whether "sugar" is a max-pattern? coffee ∪ sugar ⊃ sugar

The "coffee ∪ sugar" pattern is not only closed but also maximum, as there is no common pattern that contains it.

Closed and maximal patterns differ in that closed patterns may have frequent over-the-head with less support, while maximum patterns do not have such overlays.

TRANSACTION ID	TRANSACTION ELEMENTS
1	coffee, milk, sugar, nuts
2	coffee, sugar, eggs
3	coffee, bread, sugar, butter
4	nuts, sugar, honey, eggs
5	butter, milk, eggs

ALGORITHMS FOR DATA MINING

The most-popular state-of-the-art algorithms for the search of frequent patterns are:

- ✓ **Apriori (Rakesh Agrawal, John C. Shafer)**
- ✓ **Eclat (Mohammed J. Zaki, Mitsunori Ogihara, Srinivasan Parthasarathy, Wei Li)**
- ✓ **FP-growth (Jiawei Han, Jian Pei, Yiwen Yin)**
- ✓ **D-Club (Jianwei Li, Alok Choudhary, Nan Jiang, Weikeng Liao), PD-Club etc.**
- ✓ **Mining Close Frequent Patterns (…)**
- ✓ **MaxPattern Eclat (…)**
- ✓ **Partition, MAFIA, LCM, kDCI i many more…**

All of these algorithms have sequential and parallel versions.

Parallel versions of the algorithms can be implemented, e.g. in the object-oriented programming language Charm ++, which uses the parallel programming paradigm based on asynchronous message exchange, although the synchronous exchange is also possible:

- ➢ **Charm ++ applications are collections of parallel containers (chares) that can freely migrate between physical units (processors or cores) processing them.**
- ➢ **Communication between parallel containers is done by sending messages,**
- ➢ **When the message is received, the container entry method is called,**
- ➢ **The operation of forwarding the message is not blocking.**

APRIORI ALGORITHMS

Iterative Apriori gradually generates and tests the frequency of candidates by dividing them into frequent sets of lengths from 1 to k until all frequent patterns are found.

We divide **parallel Apriori** into the following groups of algorithms:

- **Count Distribution** – algorithms based on data decomposition, i.e. the database is divided into static partitions, in which support for candidates for frequent collections is counted (independently of each other).
- **Data Distribution** – algorithms seeking to use RAM more efficiently by partitioning the database and partitioning the list of searched candidates between processors or cores. Each candidate is searched for by only one process, so processes must exchange partitions during each iteration.
- **Candidate Distribution** – algorithms separating the list of searched candidates and replicating transactions from the database for parallel processing.

Associative Apriori uses associative graphs of AGDS data structures and can operate in both sequential and parallel versions, and due to the fact that they do not have to browse the entire transaction collection repeatedly, because they store the relationship between transactions and elements, and the frequency of elements is defined in the AGDS structure, work fastest.

APRIORI PRUNING RULE

The Apriori pruning rule says that every subset of the frequent itemset is frequent.

Conclusions resulting from this rule:

- ✓ **Each subset of a frequent set may not be rare but must be frequent.**
- ✓ **It may be more frequent, i.e. its support may be greater.**
- ✓ **If any subset of the S set is infrequent, then the S set is also infrequent.**

The above conclusion allows the filtering of all super-patterns that contain *infrequent itemsubsets* in order to increase the efficiency of searching patterns during their exploration, because *all infrequent superpatterns of the infrequent patterns are infrequent*, so they do not need to be considered in the further searching.

The Apriori pruning principle says that if there is any itemsubset that is infrequent, then any of its supersets should not be included/generated in the exploration process.

APRIORI ALGORITHM

Description of the APRIORI algorithm:

1. **Prune step:**
   It scans the entire database to find the count of each candidate in C_k where C_k represents candidate k-itemset.
   The count of each itemset in C_k is compared with a predefined minimum support count to find whether that itemset can be placed in frequent k-itemset L_k.
2. **Join step:**
   L_k is naturally joined with itself to generate the next candidate k+1-itemset C_{k+1}. The major step here is the prune step which requires scanning the entire database for finding the count of each itemset in every candidate k-itemset. If the database is huge then it requires more time to find all the frequent itemsets in the database.

Apriori Algorithm in C++ (faster)

Apriori Algorithm in Python (slower)

Presentation: https://www.youtube.com/watch?v=WGlMlS_Yydk

APRIORI ALGORITHM

```
L1 = {1-element frequent pattern sets};
for (k=2; Lk-1.IsNotEmpty(); k++) do
{
        Ck = apriori_gen(Lk-1);
        foreach t∈D do  // t - transaction, D -transaction set
        {
            Ct = subset(Ck, t);
            foreach c in Ct do      // c - candidate set
                c.count++;    // count the number of occurrences
        }
        Lk = {c∈Ck | c.count >= minsup}
}
all_frequent_pattern_set = ∪Lk;
```

APRIORI ALGORITHM

```
function apriori_gen(Ck)
{
    insert into Ck
    select p.item1, p.item2, ..., p.itemk-1, q.itemk-1
    from Lk-1 p, Lk-1 q
    where p.item1 = q.item1, ..., p.itemk-2 = q.itemk-2, p.itemk-1 <
    q.itemk-1;
    forall itemsets c in Ck do
        forall (k-1)-subsets s of c do
            if ( s not in Lk-1 ) then
                delete c from Ck;
}
```

Apriori algorithm for finding association rules:

http://www.borgelt.net/apriori.html

http://www.borgelt.net/docs/apriori.pdf

Equivalence Class Transformation

Equivalence Class Transformation **is a depth-first search (DFS) algorithm that uses an intersection of sets. It is used to explore common patterns by examining their vertical (column) format, accelerating the operation of the Apriori, as it does not need to search the database to determine support for k + 1 elements:**

$t(B) = \{T_2, T_3\}$; $t(C) = \{T_1, T_3\}$ ➔ $t(BC) = \{T_3\}$

$t(E) = \{T_1, T_2, T_3\}$ ➔ diffset $(BE, E) = \{T_1\}$ – set of differences

HORIZONTAL DATA FORMAT	
TRANSACTIONS	ELEMENTS
1	A, C, D, E
2	A, B, E
3	B, C, E

TRANSFORMATION

WERTYKALNY FORMAT DANYCH	
ELEMENT	TRANSACTION LIST
A	1, 2
B	2, 3
C	1, 3
D	1
E	1, 2, 3

association table

Thanks to this transformation, we know which transactions each element is in, so it's easier to analyze!

Additional material that broadens the description of this transformation:
http://research.ijcaonline.org/volume90/number8/pxc3894337.pdf

ECLAT ALGORITM
Equivalent Class Transformation

The classic generation of association rules involves two steps:

1. **Computationally expensive generation of sets of frequent elements, i.e. those for which support exceeds a certain accepted threshold (i.e. minimum support).**
2. **Generating association rules.**

The use of ECLAT is as follows:

1. **Efficient generation of frequent element sets by changing the representation format from horizontal to vertical.**
2. **Generating association rules.**

HORIZONTAL DATA FORMAT	
TRANSACTIONS	ELEMENTS
1	A, C, D, E
2	A, B, E
3	B, C, E

TRANSFORMATION

WERTYKALNY FORMAT DANYCH	
ELEMENT	TRANSACTION LIST
A	1, 2
B	2, 3
C	1, 3
D	1
E	1, 2, 3

association table

```
ECLAT (S_{k-1})
{
        forall itemsets I_a, I_b ∈ S_{k-1}, where a < b do
        {
                C = I_a ∩ I_b
                if (C.support ≥ minsup) add C to L_k
        }
        partition L_k into prefix-based (k-1)-lenght prefix classes S_k
        foreach class S_k in L_k do ECLAT (S_k)
}
```

Supplementary materials: http://ijctjournal.org/Volume2/Issue3/IJCT-V2I3P17.pdf , https://www.youtube.com/watch?v=oBiq8cMkTCU

Eclat algorithm: http://www.borgelt.net/eclat.html

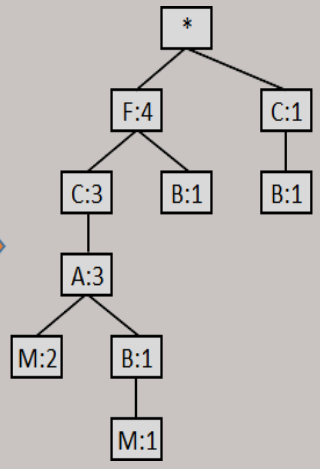
Transformation of patterns into FP-trees

The FP-Growth algorithm first performs lossy compression of the representation of patterns in the form of a tree, which is then used for exploration:

1. Find all 1-element frequent sets in transaction database D.
2. Transform each transaction $T_i \in D$ into a compressed (simplified) form $\breve{T}_i$ consisting in removing from the T_i elements that are not frequent.
3. Sort the compressed transaction elements $\breve{T}_i$ by decreasing the values of their support by creating a list of elements.
4. Transform the sorted transactions $\breve{T}_1, \breve{T}_2, \ldots, \breve{T}_N$ into an FP-tree.

TRANSACTIONS	TRANSACTION ELEMENTS	ORDERED FREQUENT ELEMENTS
1	A,C,D,G,F,I,M,P	F,C,A,M
2	A,B,C,F,L,M,R	F,C,A,B,M
3	B,F,H,J,R,W	F,B
4	B,C,K,S,P	C,B
5	A,C,E,F,L,M,N	F,C,A,M

ELEMENTS minsup = 3	FREQUENCY
F	4
C	4
A	3
B	3
M	3



ELEMENT	CONDITIONAL PATTERN BASES
A	FC:3
C	F:3
B	F:1, C:1, FCA:1
M	FCA:2, FCAB:1

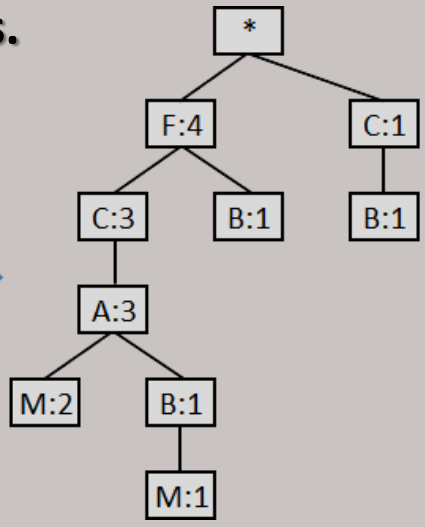
Creating FP-tree

FP-tree structure:

- The root of the graph has the "null" label, and the remaining graph vertices, both internal vertices, and leaves, represent 1-element frequent sets together with information about their number (support) in the transaction set D.
- Subsequent compressed (simplified) transactions $\breve{T}_i$ with sorted elements add in turn to the FP-tree moving from the root through existing nodes, if the given element is already represented, incrementing their counters, or add a new element to the tree, if it was not represented for a given predecessor and give it counter 1.
- In this way, a prefix tree is created for transaction $\breve{T}_i$ transactions based on an aggregation of common prefixes for transaction elements.

TRANSACTIONS	TRANSACTION ELEMENTS	ORDERED FREQUENT ELEMENTS
1	A,C,D,G,F,I,M,P	F,C,A,M
2	A,B,C,F,L,M,R	F,C,A,B,M
3	B,F,H,J,R,W	F,B
4	B,C,K,S,P	C,B
5	A,C,E,F,L,M,N	F,C,A,M

ELEMENTS minsup = 3	FREQUENCY
F	4
C	4
A	3
B	3
M	3

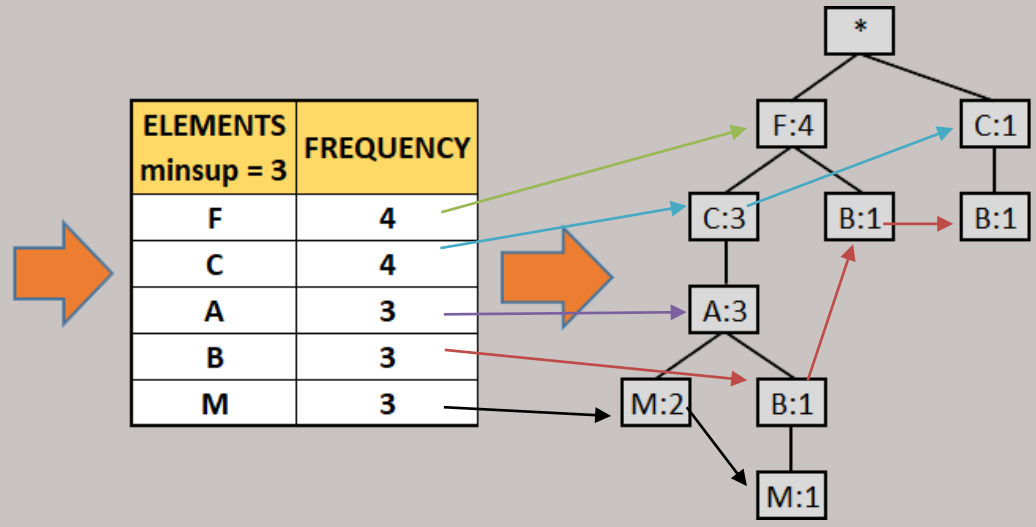


Reading support from FP-tree

- Information about the support of the pattern represented by the prefix is determined on the basis of the size of the last element of the prefix (i.e. its support), e.g. FCA has support 3, because A has support 3.
- For elements that appear repeatedly in a tree, e.g. B or C, a header array is useful, storing lists of indicators for individual instances of a given element in the tree, which speeds up searching the FP-tree (as shown by the arrows below).

TRANSACTIONS	TRANSACTION ELEMENTS	ORDERED FREQUENT ELEMENTS
1	A,C,D,G,F,I,M,P	F,C,A,M
2	A,B,C,F,L,M,R	F,C,A,B,M
3	B,F,H,J,R,W	F,B
4	B,C,K,S,P	C,B
5	A,C,E,F,L,M,N	F,C,A,M

ELEMENTS minsup = 3	FREQUENCY
F	4
C	4
A	3
B	3
M	3



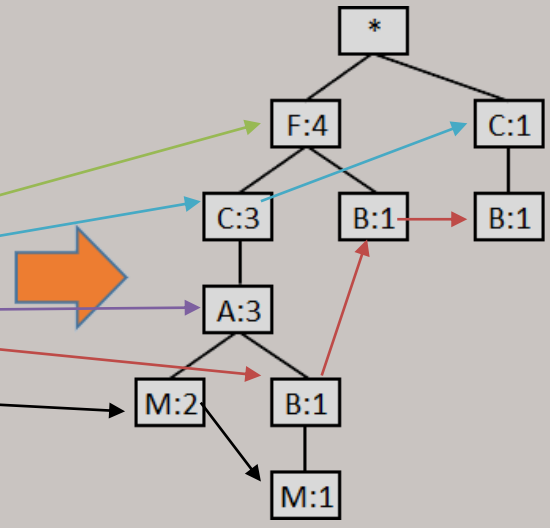
FP-tree Exploration

The FP-tree exploration process is based on the observation that for each 1-element frequent set α, all common supersets of the set α are represented in the FP-tree through paths containing the α vertex or vertices:

1. For each 1-element frequent set α, we find in the FP-tree all paths whose final vertex is the vertex representing the set α. We call this path the prefix paths of the α pattern.
2. Each prefix path of the α pattern is associated with a path frequency counter whose value is equal to the value of the transaction counter of the end vertex of the path representing the set α. The set of all pattern prefix paths forms the so-called conditional base of the pattern.

TRANSACTIONS	TRANSACTION ELEMENTS	ORDERED FREQUENT ELEMENTS
1	A,C,D,G,F,I,M,P	F,C,A,M
2	A,B,C,F,L,M,R	F,C,A,B,M
3	B,F,H,J,R,W	F,B
4	B,C,K,S,P	C,B
5	A,C,E,F,L,M,N	F,C,A,M

ELEMENTS minsup = 3	FREQUENCY
F	4
C	4
A	3
B	3
M	3



FP-tree Exploration

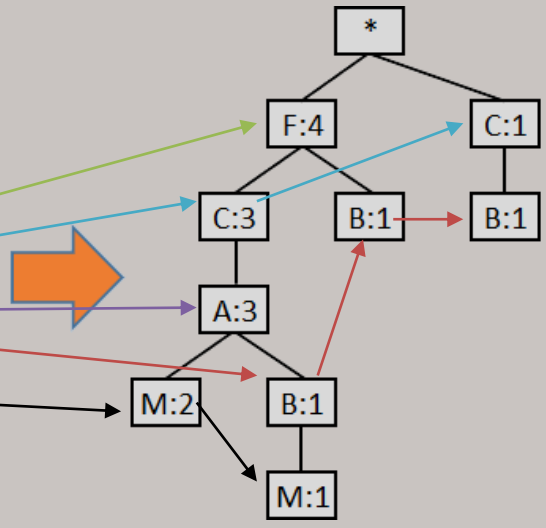
3. The conditional pattern base is used to construct the so-called conditional FP-tree of the α pattern, designated Tree-α.
4. Then, the conditional FP-tree is recursively explored to find all frequent sets containing the α set.

The FP-Growth algorithm finds all common sets.

The initial parameters of the FP-Growth procedure at the time the procedure is initiated are as follows: Tree = FP-tree and α = null.

TRANSACTIONS	TRANSACTION ELEMENTS	ORDERED FREQUENT ELEMENTS
1	A,C,D,G,F,I,M,P	F,C,A,M
2	A,B,C,F,L,M,R	F,C,A,B,M
3	B,F,H,J,R,W	F,B
4	B,C,K,S,P	C,B
5	A,C,E,F,L,M,N	F,C,A,M

ELEMENTS minsup = 3	FREQUENCY
F	4
C	4
A	3
B	3
M	3



Pseudocode of the FP-growth Algorithm

The FP-Growth algorithm finds all common sets.

The initial parameters of the FP-Growth procedure at the time the procedure is initiated are as follows: Tree = FP-tree and α = null.

```
procedure FP-Growth (Tree, α)
    if Tree zawiera pojedynczą ścieżkę P
    then for each kombinacji β wierzchołków ścieżki P
    do
        generuj zbiór β U α o wsparciu równym minimalnemu wsparciu
        elementów należących do β
    end
    else for each α-i należącego do tablicy nagłówków elementów Tree
    do
        generuj zbiór β = α-i U α o
        wsparciu = wsparcie(α-i);
        utwórz warunkową bazę wzorca β;
        utwórz warunkowe FP-drzewo wzorca β - Tree-β;
    if Tree-β ≠ Ø then FP-Growth (Tree-β, β);
end;
```

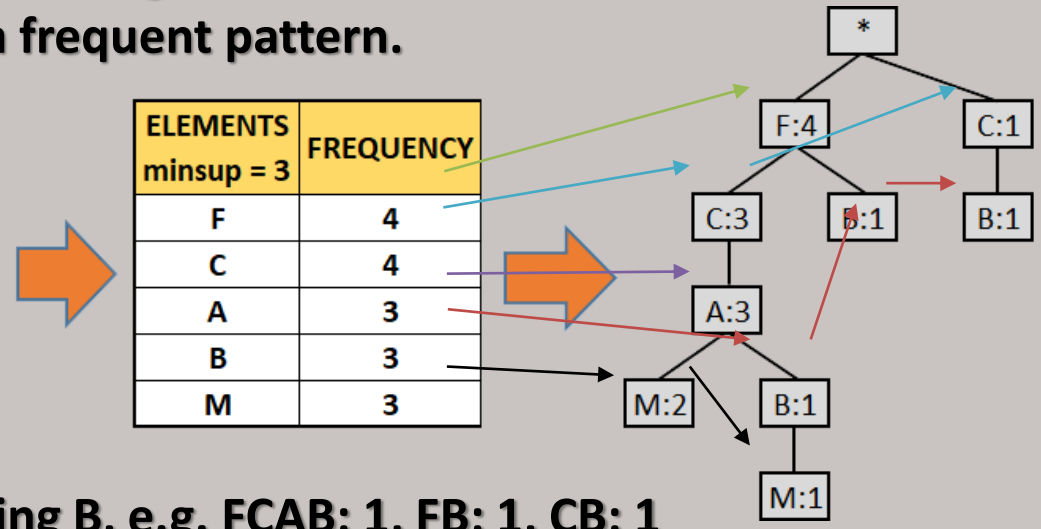
FP-growth Algorithm for Exploration

Growing of frequent FP-Growth patterns (Frequent Pattern Growth):

1. Find individual one-element frequent patterns and divide the database by them.
2. Recursively enlarge frequent patterns for each part of a divided database, the so-called conditional database.
3. A tree structure FP-tree (frequent pattern tree) will be created.
4. Recursively construct and explore FP-trees until the resulting FP-tree is empty or contains only one path that generates all combinations of its sub-paths, each of which is a frequent pattern.

TRANSACTIONS	TRANSACTION ELEMENTS	ORDERED FREQUENT ELEMENTS
1	A,C,D,G,F,I,M,P	F,C,A,M
2	A,B,C,F,L,M,R	F,C,A,B,M
3	B,F,H,J,R,W	F,B
4	B,C,K,S,P	C,B
5	A,C,E,F,L,M,N	F,C,A,M

ELEMENTS minsup = 3	FREQUENCY
F	4
C	4
A	3
B	3
M	3

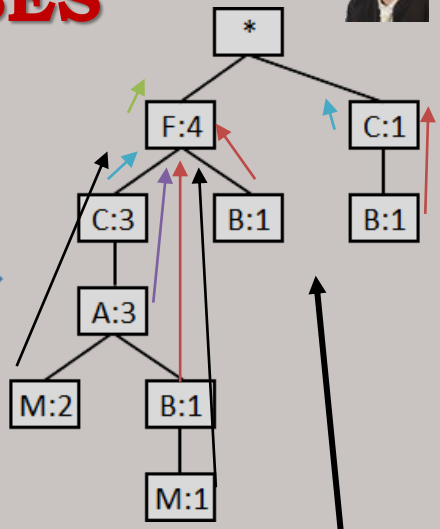


5. Exploration of patterns containing B, e.g. FCAB: 1, FB: 1, CB: 1

CONDITIONAL PATTERN-BASES

TRANSACTIONS	TRANSACTION ELEMENTS	ORDERED FREQUENT ELEMENTS
1	A,C,D,G,F,I,M,P	F,C,A,M
2	A,B,C,F,L,M,R	F,C,A,B,M
3	B,F,H,J,R,W	F,B
4	B,C,K,S,P	C,B
5	A,C,E,F,L,M,N	F,C,A,M

ELEMENTS minsup = 3	FREQUENCY
F	4
C	4
A	3
B	3
M	3



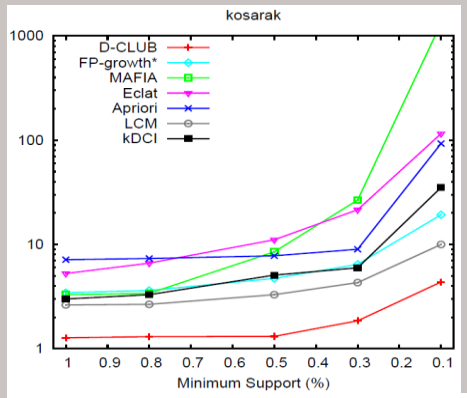
6. Exploration of standards not containing B: FCA:3, FCAM:2, C:1. creates a conditional base of patterns relative to element B.
7. Conditional pattern-bases are determined relative to the elements on the paths in the FP-tree that contain them, including all their prefixes:
8. Exploring the M-conditional pattern-base:
   FCAM: 2 has the prefix FCA:2, and FCABM:1 has the prefix FCAB: 1, so for these prefixes we take the frequent part: FCA:2, FCAB:1 → FCA:3
9. Exploring the B- conditional pattern-base:
   F:1 , C:1, FCA:1 → {}
10. Exploring the AM- conditional pattern-base:
    we take into account separately the prefixes of patterns containing A and M, i.e. for FCA:3 we have FC:3, and for those containing FCAM:2 and FCABM:1 we have FCA:2 and FCAB:1, so we are finally considering the prefixes: FC:3, FCA:2, FCAB:1 → FC:3

ELEMENT	CONDITIONAL PATTERN BASES
A	FC:3
C	F:3
B	F:1, C:1, FCA:1
M	FCA:2, FCAB:1

EXPLORATION OF D-CLUB PATTERNS

D-CLUB algorithm:

- ✓ is used for quick search of common patterns
- ✓ it gradually groups the database into a condensed associative bitmap using a differentiation technique to remove dense patterns and then extract the remaining small bitmaps by fast bit operations on such aggregates.
- ✓ uses bitmaps organized in rectangular, two-dimensional matrices that are improved in regions that require further calculations.



is much faster than Apriori and faster for many other common pattern search methods, e.g. Eclat, FP-tree.

Database

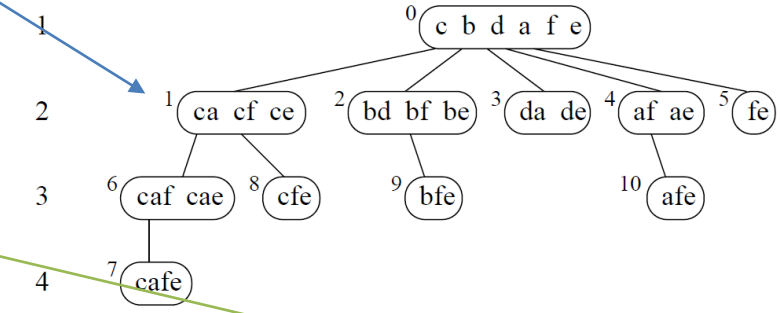
TID	Items
100	f d b e
200	f e b
300	a d b
400	a e f c
500	a d e
600	a c f e

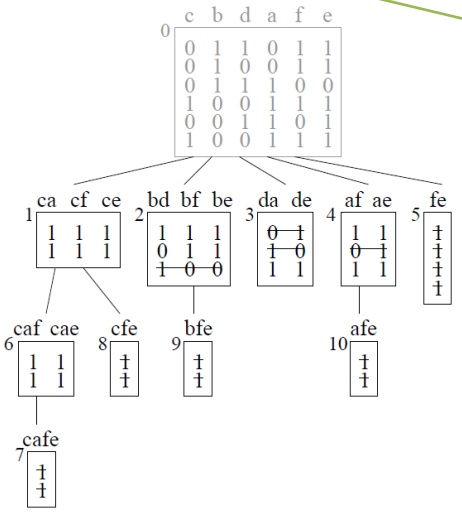
Frequent Itemsets (minsup = 33%)

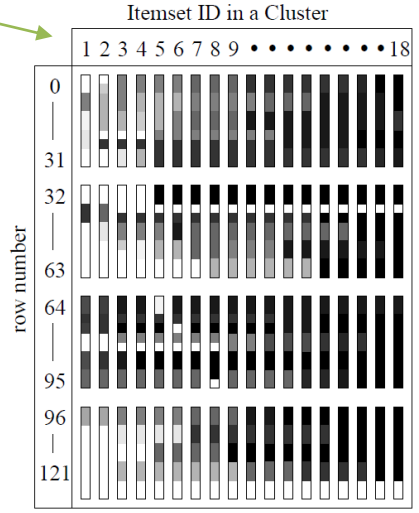
k	Frequent Itemsets : support
1	c:33% b:50% d:50% a:67% f:67% e:83%
2	ca:33% cf:33% ce:33% bd:33% bf:33% be:33% da:33% de:33% af:33% ae:50% fe:67%
3	caf:33% cae:33% cfe:33% bfe:33% afe:33%
4	cafe:33%

level/depth *(k)*

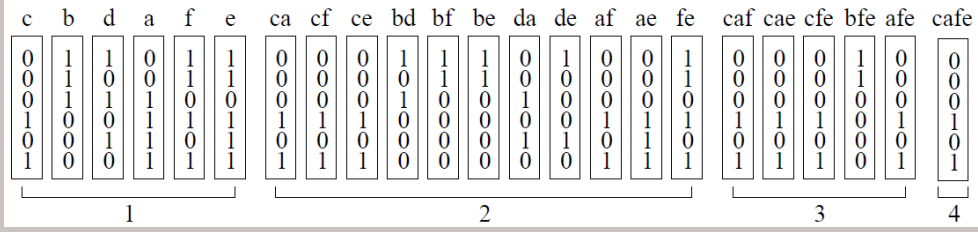
Cluster Tree of All FI's







Frequent Itemsets



EXPLORATION OF SEQUENTIAL PATTERNS

Sequential patterns consist of **sets of items**, also known as **events**, e.g.:

<EF(AB)(ABC)D(CF)G>

The elements of the sets making up the sequences are not ordered, i.e. their order does not matter: e.g. (ABC) = (CBA) = (ACB) - we write them in parentheses.

For the following sequence base and minimum support threshold (minsup = 3), we can find a *sequential pattern* <(AB)CA>

TRANSACTIONS	SEQUENTIAL PATTERNS
1	<D(ABC)(BC)A(DF)>
2	<(AE)C(BC)(AE)>
3	<(CF)(AB)(DC)CBA>
4	<EH(AF)CBC>
5	<C(AB)DF(CA)DA>

TRANSACTIONS	SEQUENTIAL PATTERNS
1	<D(ABC)(BC)A(DF)>
2	<(AE)C(BC)(AE)>
3	<(CF)(AB)(DC)CBA>
4	<EH(AF)CBC>
5	<C(AB)DF(CA)DA>

Sequential patterns are used in many applications in software engineering, analysis and comparison of DNA chains, proteins, time sequences and changes over time (e.g. on the stock exchange, exchange rates), medical treatment procedures, weather analysis and forecasting, analysis, individual adaptation of offers and optimization of promotional and advertising campaigns etc.

EXPLORATION OF SEQUENTIAL PATTERN APRIORI

Apriori-based sequential pattern mining is based on determining the frequency of occurrences (support) of one, then two, three, four ... element sequences: <A>, <B>, <C>, <D>, <E>, <F>, <H>

For which the minimum frequency or support (minsup) is above a certain threshold, e.g. ≥ 5.

We gradually generate candidates with a length of k + 1 on the basis of previously generated candidates with a length of k, and we always take into account only those candidates whose support is above a certain threshold. We proceed as long as there are longer candidates meeting this criterion (APRIORI).

Apriori allows us to examine only a limited number of candidates for whom the support is sufficiently large (above a large threshold), and not all substrings (e.g. for a small threshold, too many combinations should be considered and the algorithm would be computationally too complex).

The exploration of patterns generated and purified based on the Apriori rule is called **the Generalized Sequential Pattern (GSP) algorithm for Mining and Pruning**.

TRANSACTIONS	SEQUENTIAL PATTERNS
1	<D(ABC)(BC)A(DF)>
2	<(AE)C(BC)(AE)>
3	<(CF)(AB)(DC)CBA>
4	<EH(AF)CBC>
5	<C(AB)DF(CA)DA>

CANDIDATE	SUPPORT
<A>	10
<B>	7
<C>	11
<D>	5
<E>	3
<F>	4
<H>	1

CANDIDATES	<A>	<B>	<C>	<D>
<A>	<AA>	<AB>	<AC>	<AD>
<B>	<BA>	<BB>	<BC>	<BD>
<C>	<CA>	<CB>	<CC>	<CD>
<D>	<DA>	<DB>	<DC>	<DD>

CANDIDATES	<A>	<B>	<C>	<D>
<A>		<(AB)>	<(AC)>	<(AD)>
<B>			<(BC)>	<(BD)>
<C>				<(CD)>
<D>				

EXPLORATION OF TEXTS AND N-GRAMS & ALGORITHM OF AN ENTRY

Text exploration by searching for **n-grams**, or n-elemental word phrases where elements are words. You can also consider exploring phrases that contain other words (word sequences with gaps).

We often build N-grams based on (N-1) -grams, starting with bi-grams.

Exploration of texts by constructing phrases from often-repeated close words by combining, merging and ordering them (used in indexing and search engines).

We evaluate and organize the text phrases for the explored topic by:

- **Popularity** - i.e. the frequency of occurrence in relation to other phrases, e.g. "pattern mining" relative to "text pattern mining" or "sequential pattern mining".
- **Discriminativeness** - only frequent phrases in a given document in relation to other documents in which they are rare (infrequent).
- **Concordance** - a phrase consisting of words often occurring together in relation to others, which only occasionally occur together, e.g. "machine learning" in relation to "robust learning".
- **Completeness** - "vector machine" in relation to "support vector machine", if the latter occurs more often than the first in a different context with a different prefix.

These criteria allow you to compare phrases of different lengths (e.g. KERT algorithm).

EXPLORATION OF THE PHRASES AND THEMATIC MODELING USING TOPMine ALGORITHM

KERT (***Keyphrase Extraction and Ranking by Topics, Marina Danilevsky, Chi Wang, Nihit Desai, Jingyi Guo, Jiawei Han***) **first modeled the subject and then explored the phrases in the text.**

ToPMine first constructs phrases and then explores the subject of the text:

1. **First, we search for frequent sequence patterns consisting of neighboring elements (i.e. frequent candidate phrases) and count the number of their occurrences.**
2. **We combine frequent, one-element neighboring words (patterns) into phrases, determining their frequency of occurrence in the text.**
3. **Phrases create elements that often occur together.**
4. **We search for candidate phrases based on the frequency of their words in the text relative to the frequency of the entire candidate phrase.**

METHODS OF KNOWLEDGE EXPLORATION OF DATA

There are many other knowledge-based data mining methods.

This means that the data is not searched in a direct way, but a certain model of their representation is created, e.g. in:

- **neural systems,**
- **fuzzy systems,**
- **cognitive systems,**
- **association systems,**

which allow conclusions to be drawn based on some form of aggregated and internally represented data.

The conclusions obtained in this way may not only be a reproduction of the facts and rules collected, but also a generalization or summary thereof.

Important for achieving such functionality of the system will be:

- **way of data representation in the selected system,**
- **the possibility of aggregation and joint representation of the same and similar data,**
- **built-in inference and generalization mechanisms.**

REPRESENTATION OF DATA

The way data is represented significantly affects:

- Storing relationships between data
- Speed of access to data and their relations
- Knowledge exploration possibilities based on this data

During exploration, we usually look for:

- Frequent groups of data - patterns - determined on the basis of their similarity

We care about:

- Speed of access to data and their relations
- Possibilities of quick knowledge exploration based on this data

Knowledge of data - above all, information about:

- Relationships (relationships)
- Similarity and differences
- Classes, groups, and clusters
- …

BIBLIOGRAPHY AND COMPLEMENTARY LITERATURE

1. R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases, ACM SIGMOND Conf. Management of Data, 1993.
2. J. Han, M. Kamber. Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000.
3. G. Piatetsky-Shapiro, W. J. Frawley, Knowledge Discovery in Databases, AAAI, MIT Press, 1991.
4. G. S. Linoff, M. A. Berry, Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management, 3rd Edition, 2011.
5. U.S. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, AAAI, MIT Press, 1996.
6. Jianwei Li, Ying Liu, Wei-keng Liao, Alok Choudhary, Parallel Data Mining Algorithms for Association Rules and Clustering, CRC Press, LLC, 2006.
7. Jianwei Li, Alok Choudhary, Nan Jiang, Wei-keng Liao, Mining Frequent Patterns by Differential Refinement of Clustered Bitmaps (D-Club), SIAM SDM, 2006, https://epubs.siam.org/doi/pdf/10.1137/1.9781611972764.26
8. Mohammed J. Zaki, Parallel and distributed association mining: A survey, IEEE Concurrency, 7(4):14–25, 1999.
9. Mohammed J. Zaki, Mitsunori Ogihara, Srinivasan Parthasarathy, and Wei Li, Parallel data mining for association rules on shared-memory multi-processors, In Proc. of the ACM/IEEE Conf. on Supercomputing, November 1996.
10. FP-trees: http://wazniak.mimuw.edu.pl/images/c/c3/ED-4.2-m03-1.0.pdf
11. FP-Growth: http://www.borgelt.net/papers/fpgrowth.pdf
12. Implementations of various methods from data mining: http://www.borgelt.net/software.html
13. Daniel T. Larose, Odkrywanie wiedzy z danych, Wprowadzenie do eksploracji danych, PWN, 2006.
14. Stanisław Osowski, Metody i narzędzia eksploracji danych, BTC, Legionowo 2013.
15. Praca zbiorowa pod redakcją Andrzeja Karbowskiego i Ewy Niewiadomskiej-Szynkiewicz, Obliczenia równoległe i rozproszone, Oficyna Wydawnicza Politechniki Warszawskie, 2001.
16. Reguły asocjacyjne i algorytm Apriori: http://edu.pjwstk.edu.pl/wyklady/adn/scb/wyklad12/w12.htm

DATA MINING

EXPLORES INFORMATION FROM DATA AND EXPANDS KNOWLEDGE!