METODY INŻYNIERII WIEDZY
METHODS OF KNOWLEDGE ENGINEERING

Bezpośrednia Eksploracja Informacji z Danych
Data Mining

Adrian Horzyk

horzyk@agh.edu.pl

Akademia Górniczo-Hutnicza w Krakowie

PROBLEM ANALIZY DANYCH

- ✓ **Większość repozytoriów zawiera ukryte informacje (mogące wzbogacić naszą wiedzę) w postaci regularności, korelacji, podobieństw, trendów, osobliwości, reguł..., lecz ich struktura i rozmiary uniemożliwiają ich „ręczną” analizę.**
- ✓ **W ostatnich latach powstało wiele ciekawych i efektywnych algorytmów, struktur i metod półautomatycznej lub automatycznej eksploracji danych (*data miningu*), zwanych też odkrywaniem wiedzy (*knowledge discovery*).**
- ✓ **Eksploracja danych (*data mining*) to skrót myślowy oznaczający eksplorację wiedzy z danych, a ściślej mówiąc wydobywanie informacji z danych, które tą wiedzę mogą ukształtować.**
- ✓ **Algorytmy te służą również do odkrywania związków (relacji) pomiędzy elementami lub grupami obiektów, które zapisywane są w postaci tzw. reguł asocjacyjnych.**

EKSPLORACJA WIEDZY Z DANYCH

- ✓ **Eksploracja danych** w zależności od:
  - rodzaju danych
  - sposobu przechowywania danych

  może być przeprowadzana za pośrednictwem:
  metod statystycznych, reguł, drzew lub diagramów decyzyjnych, badania zawierania, podzbiorów, bliskości lub podobieństw, wyszukiwania wzorców częstych lub rzadkich, systemów rozmytych,
  sieci neuronowych, metod skojarzeniowych itd.
- ✓ **Dane** mogą tworzyć charakterystyczne **wzorce** w postaci:
  - **Zbiorów** (np. encji, wektorów, macierzy),
  - **Sekwencji** (np. tekstów, ciągów instrukcji, sekwencji czasowych),
  - **Struktur złożonych** (np. podgrafów, obrazów, map, tekstur).

ZBIORY ELEMENTÓW I WSPARCIE

- ✓ Zbiór elementów (*itemset*) $I = \{i_1, i_2, ..., i_N\}$ – to zbiór wszystkich dostępnych elementów (obiektów, towarów), gdzie $N \geq 1$.
- ✓ Transakcja (*transaction*) T jest parą T = (id, X) składającą się z identyfikatora transakcji id oraz pewnego podzbioru towarów $X \subseteq I$, zakładając pewną skończoną ilość identyfikatorów transakcji $id \in Tid = \{id_1, id_2, ..., id_M\}$.
- ✓ Z punktu widzenia eksploracji danych interesuje nas częstość (*frequency*) wzorców (*patterns*) W, czyli powtarzalność różnych k-elementowych podzbiorów (k-zbiorów) w zbiorze transakcji nazywanym bazą transakcyjną $D = \{T_1, T_2, ..., T_M\}$.
- ✓ k-zbiór (*k-itemset*) to k-elementowy zbiór $X = \{x_1, ..., x_K\} \subseteq I$, który zwykle definiuje pewną transakcję $T_m = (id_m, X)$ lub wzorzec $W \subseteq I$, np.:
  W={kawa, cukier} $\subseteq$ X={kawa, cukier, jajka} $\subseteq I$,
  W={kawa, cukier} $\subseteq$ I={kawa, mleko, cukier, orzeszki, jajka, chleb, masło, miód}.
- ✓ Mówimy, że transakcja T = (id, X) pokrywa wzorzec (np. zbiór towarów) W, jeśli W $\subseteq$ X. Wzorzec W może być pokryty przez wiele transakcji. Zbiór transakcji pokrywających wzorzec W oznaczamy jako cover(W,D) = {T $\in$ D: T pokrywa W}.
- ✓ Mówimy, że wzorzec W jest częsty (*frequent*), jeśli jego pokrycie przez transakcje z rozważanej bazy transakcji D jest nie mniejsze niż ustalony próg σ.
- ✓ Wydobywanie częstych wzorców (mining frequent patterns) jest jednym podstawowych zadań eksploracji danych i niezbędnym krokiem w wydobywaniu reguł asocjacyjnych (Association Rule Mining) i analizie korelacji danych.

ZBIORY ELEMENTÓW I WSPARCIE

- ✓ **Wsparcie (*support*) s** – to częstotliwość wystąpień **wzorca W** (zbioru elementów X) w analizowanym zbiorze encji lub transakcji wyrażone w procentach. Wsparcie liczone jest jako stosunek ilości wystąpień **wzorca W** w rozważanym zbiorze transakcji D, wyrażonej jako |cover(W,D)|, w stosunku do ilości wszystkich rozważanych transakcji M = |D|:

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

- ✓ Korzystając z definicji wsparcia, mówimy, że **wzorzec W** jest **częsty (*frequent*)**, jeśli jego wsparcie (*support*) jest nie mniejsze niż ustalony próg σ (*min support*): $\sigma = s_{min}$ (*minimum support*).

PRZYKŁAD: Dla progu σ = 50% i zbioru transakcji określ, które elementy są częste (*frequent*)?

CZĘSTE > 50%

- ✓ Cukier (80%)
- ✓ Kawa (60%)
- ✓ Jajka (60%)
- ✓ Mleko (40%)
- ✓ Orzeszki (40%)
- ✓ Masło (40%)
- ✓ Chleb (20%)
- ✓ Miód (20%)

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

REGUŁY ASOCJACYJNE

- ✓ **Reguły asocjacyjne (*association rules*) elementów transakcji/wzorców: X → Y (s, c).**
- ✓ **Wsparcie (*support*) s dla reguł asocjacyjnych zdefiniowane jest przez prawdopodobieństwo, że określona transakcja zawiera zarówno X jak również Y, czyli X ∪ Y. Prawdopodobieństwo to jest liczone względem wszystkich możliwych transakcji, wyrażając prawdopodobieństwo zaistnienia takiej asocjacji, czyli wystąpienia takiej reguły asocjacyjnej.**
- ✓ **Zaufanie/Pewność/Wiarygodność (*confidence*) c – to prawdopodobieństwo warunkowe p(Y|X), że transakcja zawierająca X zawiera również Y.**
- ✓ **Eksploracja reguł asocjacyjnych polega na odnalezieniu wszystkich reguł X → Y o określonym minimalnym wsparciu s_{min} oraz o określonej minimalnej pewności c_{min} : np. $s \geq s_{min} = 40\%$ oraz $c \geq c_{min} = 50\%$. Wtedy taką regułę asocjacyjną nazywamy silną.**
- ✓ **Wielowymiarowe reguły asocjacyjne zawierają rozbudowane reguły tj.:**
  wiek (X, „18-24”) ∧ zawód (X, „student”) ⇒ kupuje (X, „cola”)
  wiek (X, „18-24”) ∧ kupuje (X, „pop-corn”) ⇒ kupuje (X, „cola”)

REGUŁY ASOCJACYJNE

Eksploracja reguł asocjacyjnych oznaczają wyszukiwanie takich reguł, które przewidują wystąpienie elementu na podstawie wystąpienia innych elementów w transakcji.

Reguły asocjacyjne (*association rules*) stosowane są np. do:

- ✓ organizowania promocji i sprzedaży wiązanej (up-selling i cross-selling),
- ✓ konstruowania katalogów wysyłkowych,
- ✓ ustalenia sposobu rozmieszczenia towarów na półkach w marketach,
- ✓ określania najlepiej rotujących się towarów.

REGUŁY ASOCJACYJNE:

- ✓ Kawa → Cukier (60%, 100%)
- ✓ Cukier → Kawa (60%, 75%)
- ✓ Cukier → Jajka (40%, 50%)
- ✓ Jajka → Cukier (40%, 67%)

NIE SĄ NIMI dla s ≥ 50%, c ≥ 50%:

- ❖ Kawa → Jajka (20%, 33%)
- ❖ Jajka → Kawa (20%, 33%)

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

OKREŚLANIE REGUŁ ASOCJACYJNYCH

- ✓ **Wsparcie (*support*) s to ilość** transakcji zawierających zarówno X jak również Y, czyli $X \cup Y$.

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

- ✓ **Zaufanie/Pewność/Wiarygodność (*confidence*) c** – to prawdopodobieństwo warunkowe p(Y|X), że transakcja zawierająca X zawiera również Y.

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

Sposób obliczania wsparcia (*support*) s i pewności (*confidence*) c dla reguł asocjacyjnych określonych, dla których s ≥ 40%, c ≥ 50%:

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

Kawa → Cukier (s = 60% = 3 / 5, c = 100% = 3 / 3)
Cukier → Kawa (s = 60% = 3 / 5, c = 75% = 3 / 4)
Cukier → Jajka (s = 40% = 2 / 5, c = 50% = 2 / 4)
Jajka → Cukier (s = 40% = 2 / 5, c = 67% = 2 / 3)

OKREŚLANIE REGUŁ ASOCJACYJNYCH

- ✓ **Wsparcie (*support*) s to ilość** transakcji zawierających zarówno X jak również Y, czyli $X \cup Y$.

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

- ✓ **Zaufanie/Pewność/Wiarygodność (*confidence*) c** – to prawdopodobieństwo warunkowe p(Y|X), że transakcja zawierająca X zawiera również Y.

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

Sposób obliczania wsparcia (*support*) s i pewności (*confidence*) c dla reguł asocjacyjnych określonych, dla których s ≥ 40%, c ≥ 50%:

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

Jajka → Cukier (s = 40% = 2 / 5, c = 67% = 2 / 3)

Kawa → Cukier (s = 60% = 3 / 5, c = 100% = 3 / 3)

Cukier → Kawa (s = 60% = 3 / 5, c = 75% = 3 / 4)

Cukier → Jajka (s = 40% = 2 / 5, c = 50% = 2 / 4)

OKREŚLANIE REGUŁ ASOCJACYJNYCH

- ✓ **Wsparcie (*support*) s to ilość** transakcji zawierających zarówno X jak również Y, czyli $X \cup Y$.

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

- ✓ **Zaufanie/Pewność/Wiarygodność (*confidence*) c** – to prawdopodobieństwo warunkowe p(Y|X), że transakcja zawierająca X zawiera również Y.

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

Sposób obliczania wsparcia (*support*) s i pewności (*confidence*) c dla wielowymiarowej reguły asocjacyjnej:

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

Jajka $\wedge$ Cukier → Kawa (s = 20% = 1 / 5, c = 50% = 1 / 2)

REGUŁY ASOCJACYJNE

- ✓ **Reguły asocjacyjne** przypominają reguły decyzyjne, lecz decyzja (czyli prawa strona implikacji) nie jest z góry określona.
- ✓ **Reguły asocjacyjne** działają podobnie jak uczenie nienadzorowane (bez nauczyciela) dla problemów algorytmów grupowania (klasteryzacji). Taki algorytm nie ma z góry określonej prawidłowej odpowiedzi. Zamiast tego ma opisywać wewnętrzne zależności między atrybutami.
- ✓ **Reguły asocjacyjne** wzięły się z badań nad zagadnieniami **analizy koszykowej** (***Market Basket Analysis***) polegającej na odkrywaniu wzorców zachowania się klientów, czyli znajdywaniu grup produktów kupowanych razem oraz określania ich częstości:

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

- ✓ Do mierzenia obiektywizmu reguł asocjacyjnych wykorzystujemy dwa wskaźniki:
  - ✓ **Wsparcie (*support*)** – określające, ile procent spośród zbadanych transakcji występuje razem, np. w ilu transakcjach występuje kawa i cukier równocześnie.
  - ✓ **Zaufanie/Pewność/Wiarygodność (*confidence*)** – określa, ile procent transakcji zawiera wniosek (decyzję, czyli lewą stronę implikacji) przy założeniu, że spełniona jest lewa strona implikacji (transakcji), czyli: $c\ (X \Rightarrow Y) = s\ (X \cup Y)\ /\ s\ (X)$.
- ✓ Odpowiednie poziomy wymaganego wsparcia i wiarygodności określa użytkownik na podstawie potrzeb wynikających z danej dziedziny, wiedzy eksperta, zadania itp.
- ✓ Regułę asocjacyjną nazywamy **silną**, jeśli $s \geq s_{min}$ oraz $c \geq c_{min}$.

TWORZENIE REGUŁ ASOCJACYJNYCH

- ✓ Tworzenie reguł asocjacyjnych poprzedzone jest zwykle procesem wyznaczania częstych wzorców, dla których te reguły tworzymy, gdyż zwykle (np. w handlu) jest istotne to, co często się powtarza, np. istotne są często kupowane produkty, które występują razem w różnych transakcjach. Pozwala to na lepsze zlokalizowanie tych produktów na półkach sklepowych, w celu zwiększenia ich sprzedaży.
- ✓ Poszukujemy wobec tego zwykle silnych reguł asocjacyjnych o odpowiednio zdefiniowanym minimalnym wsparciu oraz minimalnej ufności (wiarygodności).
- ✓ Czasami ciekawe mogą być te wzorce, które wystąpiły po raz pierwszy lub występują rzadko, np.: w astronomii, fizyce (np. zderzaczu hadronów), genetyce, kognitywistyce itd., wtedy poszukiwane mogą być tylko wzorce o niewielkim wsparciu lub wzorce unikalne.
- ✓ Do wyznaczania reguł asocjacyjnych oraz poszukiwania wzorców częstych wykorzystywany jest bardzo popularny algorytm Apriori, który posiada również liczne rozszerzenia mające na celu przyspieszenie jego działania.

WZORCE SKŁADOWE

- ✓ Duże i długie wzorce zawierają (kombinatorycznie rzecz biorąc) sporą ilość wzorców składowych – subwzorców (*sub-patterns*):
  - ✓ podzbiorów elementów
  - ✓ subsekwencji elementów
  - ✓ podgrafów elementów
  - ✓ wycinków/obszarów elementów (np. dla obrazów, map)

- ✓ Subwzorce (*sub-patterns*) umożliwiają znajdywanie podobieństw i różnic oraz tworzenie asocjacyjnych związków pomiędzy wzorcami.
- ✓ Ze względu na czasami dużą kombinatoryczną złożoność niezbędnych do wykonania porównań, opłaca się najpierw analizować i porównywać subwzorce o mniejszej ilości obiektów składowych, a dopiero potem na ich podstawie określać, np. częstość, rzadkość, wsparcie czy pewność dla wzorców o większej ilości obiektów.

WZORCE ZAMKNIĘTE

- ✓ **Zamknięte wzorce (*closed patterns*) X to takie wzorce, które są częste (*frequent*) i nie istnieje żaden nadwzorzec (*super-pattern*) $Y \supset X$, który miałby takie same wsparcie (*support*) jak wzorzec X.**
- ✓ **Zamknięty wzorzec (*closed pattern*) jest więc kompresją stratną wszystkich zawartych w nich częstych (frequent) wzorców, gdyż tracona jest informacja o ich wsparciu (*support*).**

Przykład: Czy wzorzec „kawa" jest wzorcem zamkniętym (*closed*)?
Jest wzorcem częstym o wsparciu ≥ 50%, lecz nie jest wzorcem zamkniętym, gdyż istnieje nadwzorzec „kawa ∪ cukier", który ma takie samo wsparcie równe 60% i go zawiera.
Natomiast wzorzec „kawa ∪ cukier" jest zamknięty, gdyż nie istnieje żaden wzorzec o takim samym wsparciu, który by go obejmował.

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

WZORCE MAKSYMALNE

- ✓ Maksymalne wzorce (*max-patterns*) X to takie wzorce, które są częste (*frequent*) i nie istnieje żaden częsty nadwzorzec (*super-pattern*) $Y \supset X$.
- ✓ Maksymalne wzorce (*max-patterns*) reprezentują wszystkie częste wzorce (*frequent patterns*), których wszystkie elementy zawierają.
- ✓ Maksymalny wzorzec (*max-pattern*) jest więc kompresją stratną wszystkich częstych wzorców składających się z jego elementów.

Przykład: Wzorzec „kawa ∪ cukier” jest nie tylko zamknięty, lecz również maksymalny, gdyż nie istnieje żaden częsty wzorzec, który by go zawierał.

Wzorce zamknięte od maksymalnych różnią się tym, iż wzorce zamknięte mogą posiadać częste nadwzorce o mniejszym wsparciu, zaś wzorce maksymalne takich nadwzorców nie posiadają.

ID TRANSAKCJI	ELEMENTY TRANSAKCJI
1	kawa, mleko, cukier, orzeszki
2	kawa, cukier, jajka
3	kawa, chleb, cukier, masło
4	orzeszki, cukier, miód, jajka
5	masło, mleko, jajka

ALGORYTMY DATA MININGU

Najpopularniejsze algorytmy (*state-of-the-art algorithms for the search of frequent patterns*) wyszukiwania zbiorów wzorców częstych:

- ✓ **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, Wei-keng Liao), PD-Club etc.**
- ✓ **Mining Close Frequent Patterns (…)**
- ✓ **MaxPattern Eclat (…)**
- ✓ **Partition, MAFIA, LCM, kDCI i wiele innych…**

Wszystkie te algorytmy posiadają wersje sekwencyjne i równoległe.

Równoległe wersje algorytmów można zaimplementować np. w zorientowanym obiektowo języku programowania Charm++, który wykorzystuje paradygmat programowania równoległego oparty o asynchroniczną wymianę komunikatów, aczkolwiek wymiana synchroniczna również jest możliwa:

- ➢ **Aplikacje w Charm++ są kolekcjami kontenerów równoległych (*chares*), które mogą swobodnie migrować pomiędzy fizycznymi jednostkami (procesorami lub rdzeniami) przetwarzającymi je.**
- ➢ **Komunikacja pomiędzy kontenerami równoległymi odbywa się poprzez wysyłanie komunikatów,**
- ➢ **Po odebraniu komunikatu wywoływana jest metoda wejściowa (*entry method*) kontenera,**
- ➢ **Operacja przekazania wiadomości jest nie blokująca.**

ALGORYTMY APRIORI

Apriori iteracyjny stopniowo generuje i testuje częstość kandydatów dzieląc ich na zbiory częste o długości od 1 do k aż do momentu gdy znalezione zostaną wszystkie wzorce częste.

Apriori równoległe dzielimy na następujące grupy algorytmów:

- **Count Distribution** – algorytmy bazujące na dekompozycji danych, tzn. baza dzielona jest na statyczne partycje, w których równolegle zliczane jest wsparcie dla kandydatów na zbiory częste (niezależnie od siebie).
- **Data Distribution** – algorytmy dążące do efektywniejszego wykorzystania pamięci RAM poprzez podzielenie bazy na partycje i rozdzielenie listy wyszukiwanych kandydatów pomiędzy procesory lub rdzenie. Każdy kandydat jest wyszukiwany przez tylko jeden proces, więc procesy muszą wymieniać się partycjami w czasie każdej iteracji.
- **Candidate Distribution** – algorytmy rozdzielające listę wyszukiwanych kandydatów i replikujące transakcje z bazy danych w celu równoległego ich przetwarzania.

Apriori asocjacyjne wykorzystują asocjacyjne grafowe struktury danych AGDS i mogą działać zarówno w wersji sekwencyjnej, jak i równoległej, a ze względu na fakt, iż nie muszą przeglądać całej kolekcji transakcji wielokrotnie, gdyż przechowują relacje pomiędzy transakcjami i elementami oraz częstość elementów jest określona w strukturze AGDS, działają najszybciej.

REGUŁA OCZYSZCZANIA APRIORI

Reguła oczyszczania Apriori mówi, iż każdy podzbiór zbioru częstego (*frequent itemset*) jest częsty (*frequent*).

Wnioski wynikające z tej reguły:

- ✓ **Każdy podzbiór zbioru częstego nie może być rzadki, ale musi być częsty.**
- ✓ **Może być częstszy, czyli jego wsparcie (*support*) może być większe.**
- ✓ **Jeśli jakikolwiek podzbiór zbioru S jest rzadki (*infrequent*), wtedy zbiór S jest również rzadki (*infrequent*).**

Powyższy wniosek umożliwia odfiltrowanie wszystkich nadwzorców (*super-patterns*), które zawierają rzadkie (*infrequent*) podzbiory (*itemsubsets*), w celu podniesienia efektywności przeszukiwania wzorców w trakcie ich eksploracji, gdyż wszystkie nadwzorce wzorców rzadkich są rzadkie, więc nie trzeba ich rozważać w trakcie dalszego przeszukiwania.

Reguła oczyszczania Apriori (*pruning principle*) mówi, iż jeśli istnieje jakikolwiek podzbiór (*itemsubset*), który jest rzadki (*infrequent*), wtedy jego dowolny zawierający go zbiór (*superset*) nie powinien być uwzględniany/generowany w procesie eksploracji.

ALGORYTM OCZYSZCZANIA APRIORI

Algorytm Apriori w C++ (szybszy)

Algorytm Apriori w Python (wolniejszy)

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

Opis algorytmu APRIORI:

1. **Krok oczyszczania (*prune step*):**
   Przeszukujemy wszystkie wzorce w celu ustalenia ilości każdego kandydata w k-elementowym podzbiorze C_k. Ilości te są porównywane z ustalonym minimalnym wsparciem (suport) s, w celu ustalenia, czy dany kandydat może zostać umieszczony w zbiorze L_k częstych wzorców.
   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. **Krok łączenia (*join step*):**
   Wzorce ze zbioru L_k są naturalnie łączone ze sobą w celu wygenerowania następnych k+1 elementowych kandydatów C_{k+1} (badane są ich kombinacje). Najważniejsze jest przeszukanie wszystkich wzorców w celu wyznaczenia ilości każdego podzbioru dla każdego k-elementowego kandydata C_k. W wyniku przeszukiwania należy określić wszystkie częste (powyżej pewnego progu minimalnego) podzbiory, jakie powstaną w wyniku takiego złączenia.
   L_k is natural 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.

ALGORYTM APRIORI

Algorytm APRIORI:

```
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;
```

ALGORYTM APRIORI

Algorytm APRIORI:

```
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;
}
```

Algorytm Apriori do znajdywania reguł asocjacyjnych:

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

EKWIWALENTNA TRANSFORMACJA KLAS

Ekwiwalentna Transformacja Klas ECLAT (*Equivalence Class Transformation*) to algorytm przeszukiwania w głąb (*DFS depth-first search*) wykorzystujący przecięcie zbiorów. Służy do eksploracji częstych wzorców poprzez badanie ich wertykalnego (kolumnowego) formatu, przyspieszając działanie Apriori, gdyż nie musi przeszukiwać bazy danych w celu określenia wsparcia dla elementów k+1 elementowych:

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

$t(E) = \{T_1, T_2, T_3\}$ ➔ diffset $(BE, E) = \{T_1\}$ – zbiór różnic

HORYZONTALNY FORMAT DANYCH	
TRANSAKCJE	ELEMENTY ZBIORU
1	A, C, D, E
2	A, B, E
3	B, C, E

TRANSFORMACJA

WERTYKALNY FORMAT DANYCH	
ELEMENT	LISTA TRANSAKCJI
A	1, 2
B	2, 3
C	1, 3
D	1
E	1, 2, 3

tablica asocjacji

Dzięki takiej transformacji wiemy, w których transakcjach występują poszczególne elementy, więc łatwiej można je analizować!

Dodatkowy materiał poszerzający opis tej transformacji:
http://research.ijcaonline.org/volume90/number8/pxc3894337.pdf

ALGORYTM ECLAT
Equivalent CLAss Transformation

Klasyczne generowanie reguł asocjacyjnych obejmuje dwa kroki:

1. **Kosztowne obliczeniowo generowanie zbiorów elementów częstych, czyli takich dla których wsparcie przekracza pewny przyjęty próg (czyli wsparcie minimalne).**
2. **Generowanie reguł asocjacyjnych.**

Wykorzystanie ECLAT przebiega następująco:

1. **Efektywne generowanie zbiorów elementów częstych dzięki zmianie formatu reprezentacji z horyzontalnego na wertykalny.**
2. **Generowanie reguł asocjacyjnych.**

HORYZONTALNY FORMAT DANYCH	
TRANSAKCJE	ELEMENTY ZBIORU
1	A, C, D, E
2	A, B, E
3	B, C, E

TRANSFORMACJA

WERTYKALNY FORMAT DANYCH	
ELEMENT	LISTA TRANSAKCJI
A	1, 2
B	2, 3
C	1, 3
D	1
E	1, 2, 3

tablica asocjacji

```
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)
}
```

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

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

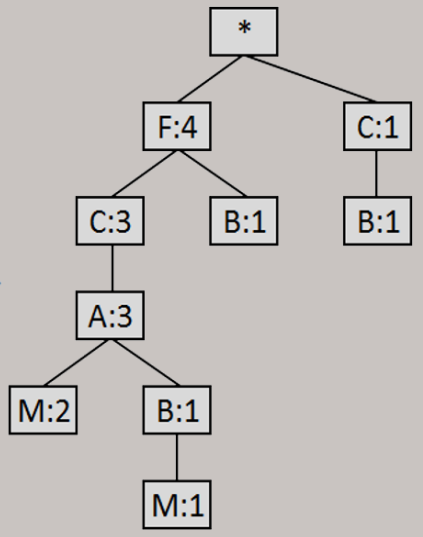
Transformacja wzorców do FP-drzewa

Algorytm FP-Growth dokonuje najpierw kompresję stratną reprezentacji wzorców w postaci drzewa, które następnie jest wykorzystywane do eksploracji:

1. Znajdź wszystkie 1-elementowe zbiory częste w bazie transakcji D.
2. Transformuj każdą transakcję $T_i \in D$ do skompresowanej (uproszczonej) postaci $\breve{T}_i$ polegającej na usunięciu z T_i wszystkich elementów, które nie są częste.
3. Posortuj elementy transakcji skompresowanych $\breve{T}_i$ według malejących wartości ich wsparcia tworząc listę elementów.
4. Transformuj posortowane transakcje $\breve{T}_1, \breve{T}_2, \ldots, \breve{T}_N$ do FP-drzewa.

TRANSAKCJE	ELEMENTY TRANSAKCJI	UPORZĄDKOWANE CZĘSTE ELEMENTY
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

ELEMENTY minsup = 3	CZĘSTOTLIWOŚĆ
F	4
C	4
A	3
B	3
M	3



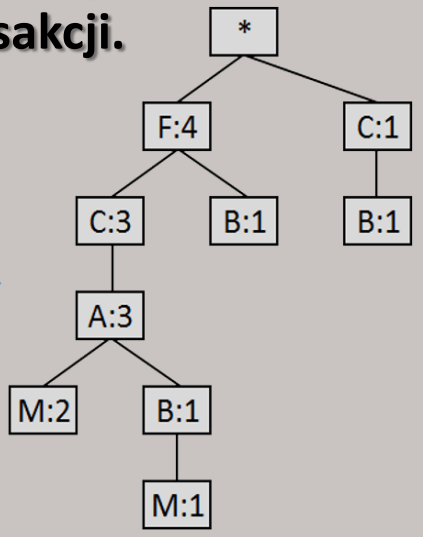
Tworzenie FP-drzewa

Budowa FP-drzewa:

- Korzeń grafu posiada etykietę "null", a pozostałe wierzchołki grafu, zarówno wierzchołki wewnętrzne jak i liście, reprezentują 1-elementowe zbiory częste wraz z informacją o ich liczności (wsparciu) w zbiorze transakcji D.
- Kolejne skompresowane (uproszczone) transakcje $\breve{T}_i$ z posortowanymi elementami dodawaj po kolei do FP-drzewa poruszając się od korzenia przez istniejące węzły, jeśli dany element jest już reprezentowany, inkrementując ich liczniki, albo dodaj nowy element do drzewa, jeśli nie było jeszcze dla danego poprzednika reprezentowany oraz nadaj mu licznik 1.
- W taki sposób tworzone jest drzewo prefiksowe dla transakcji $\breve{T}_i$ na bazie agregacji wspólnych prefiksów dla elementów transakcji.

TRANSAKCJE	ELEMENTY TRANSAKCJI	UPORZĄDKOWANE CZĘSTE ELEMENTY
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

ELEMENTY minsup = 3	CZĘSTOTLIWOŚĆ
F	4
C	4
A	3
B	3
M	3

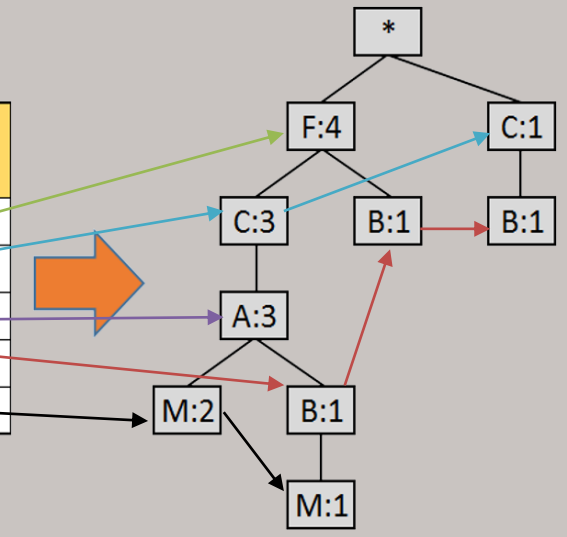


Odczytywanie wsparcia z FP-drzewa

- Informację o wsparciu wzorca reprezentowanego przez prefiks określamy na podstawie liczności ostatniego elementu prefiksu (czyli jego wsparcia), np. FCA ma wsparcie 3, gdyż A ma wsparcie 3.
- W przypadku elementów występujących wielokrotnie w drzewie, np. B lub C, przydatna jest tablica nagłówkowa, przechowująca listy wskaźników do poszczególnych wystąpień danego elementu w drzewie, co przyspiesza przeszukiwanie FP-drzewa (jak zaprezentowano strzałkami poniżej).

TRANSAKCJE	ELEMENTY TRANSAKCJI	UPORZĄDKOWANE CZĘSTE ELEMENTY
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

ELEMENTY minsup = 3	CZĘSTOTLIWOŚĆ
F	4
C	4
A	3
B	3
M	3



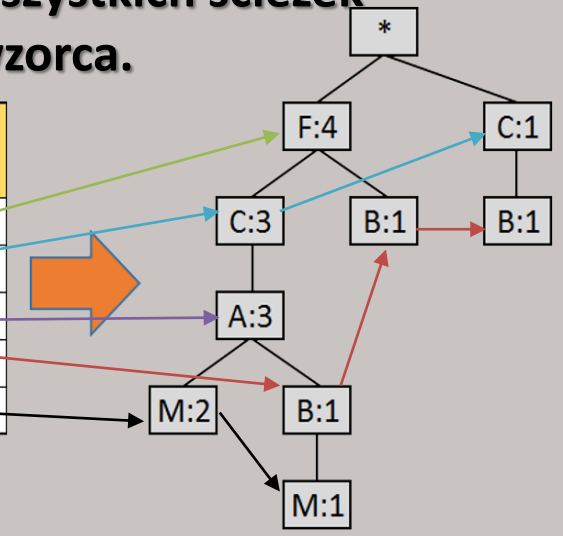
Eksploracja FP-drzewa

Proces eksploracji FP-drzewa bazuje na obserwacji, iż dla każdego 1-elementowego zbioru częstego α wszystkie częste nadzbiory zbioru α są reprezentowane w FP-drzewie poprzez ścieżki zawierające wierzchołek (wierzchołki) α:

1. **Dla każdego 1-elementowego zbioru częstego α znajdujemy w FP-drzewie wszystkie ścieżki, których końcowym wierzchołkiem jest wierzchołek reprezentujący zbiór α. Taką ścieżki nazywamy ścieżkami prefiksowymi wzorca α.**
2. **Z każdą prefiksową ścieżką wzorca α związany jest licznik częstości ścieżki, którego wartość równa jest wartości licznika transakcji wierzchołka końcowego ścieżki reprezentującego zbiór α. Zbiór wszystkich ścieżek prefiksowych wzorca tworzy tzw. warunkową bazę wzorca.**

TRANSAKCJE	ELEMENTY TRANSAKCJI	UPORZĄDKOWANE CZĘSTE ELEMENTY
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

ELEMENTY minsup = 3	CZĘSTOTLIWOŚĆ
F	4
C	4
A	3
B	3
M	3



Eksploracja FP-drzewa

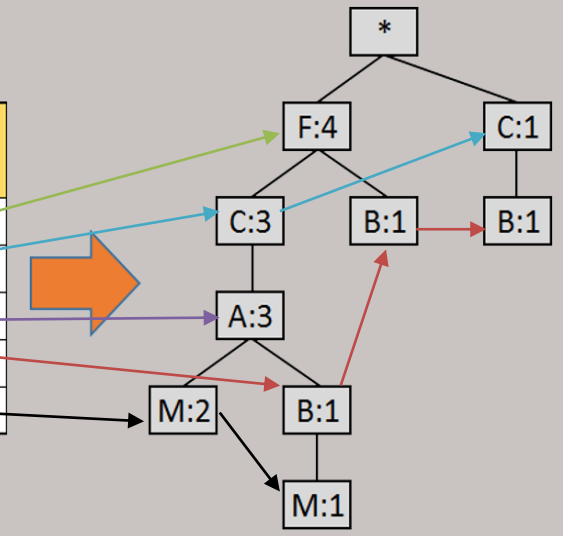
3. Warunkowa baza wzorca służy do konstrukcji tzw. warunkowego FP-drzewa wzorca α, oznaczanego Tree-α.
4. Następnie, warunkowe FP-drzewo jest rekurencyjnie eksplorowane w celu znalezienia wszystkich zbiorów częstych zawierających zbiór α.

Algorytm FP-Growth znajduje wszystkie zbiory częste.

Parametry początkowe procedury FP-Growth, w momencie inicjacji procedury, są następujące: Tree = FP-drzewo oraz α = null.

TRANSAKCJE	ELEMENTY TRANSAKCJI	UPORZĄDKOWANE CZĘSTE ELEMENTY
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

ELEMENTY minsup = 3	CZĘSTOTLIWOŚĆ
F	4
C	4
A	3
B	3
M	3



Algorytm FP-growth w pseudokodzie

Algorytm FP-Growth znajduje wszystkie zbiory częste.

Parametry początkowe procedury FP-Growth, w momencie inicjacji procedury, są następujące: Tree = FP-drzewo oraz 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;
```

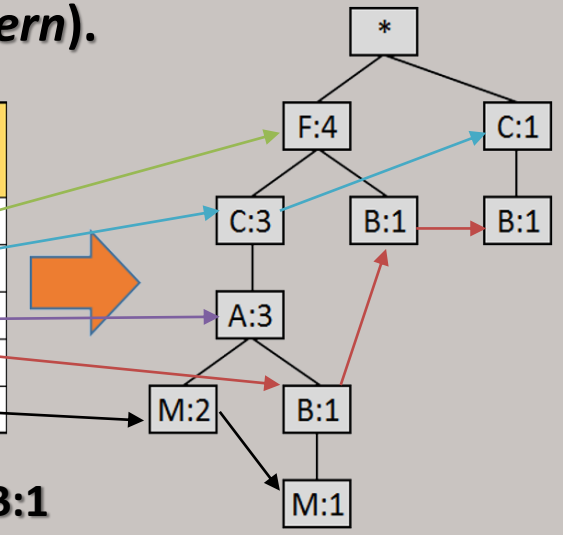
ALGORYTM FP-GROWTH W EKSPLORACJI

Rośnięcie częstych wzorców FP-Growth (*Frequent Pattern Growth*):

1. Znajdź pojedyncze częste jedno-elementowe wzorce i podziel bazę danych względem nich.
2. Rekurencyjnie powiększaj częste wzorce dla każdej części podzielonej bazy danych, tzw. warunkowej bazy danych (*conditional database*).
3. Powstanie struktura drzewiasta FP-tree (*frequent pattern tree*).
4. Rekurencyjnie konstruuj i eksploruj drzewa FP-trees, dopóki wynikowe drzewo FP-tree jest puste lub zawiera tylko jedną ścieżkę, która generuje wszystkie kombinacje swoich podścieżek (*sub-paths*), z których każda jest częstym wzorcem (*frequent pattern*).

TRANSAKCJE	ELEMENTY TRANSAKCJI	UPORZĄDKOWANE CZĘSTE ELEMENTY
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

ELEMENTY minsup = 3	CZĘSTOTLIWOŚĆ
F	4
C	4
A	3
B	3
M	3

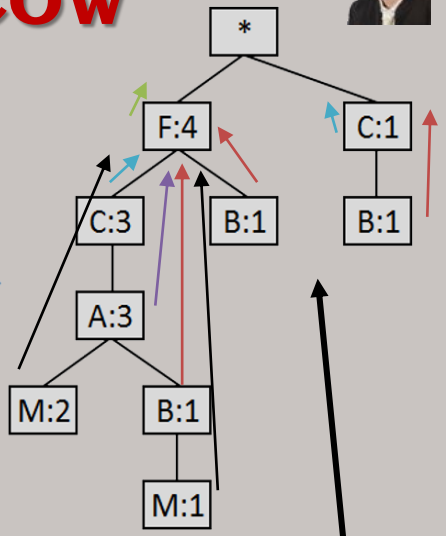


5. Eksploracja wzorców zawierających B, np.: FCAB:1, FB:1, CB:1

WARUNKOWE BAZY WZORCÓW

TRANSAKCJE	ELEMENTY TRANSAKCJI	UPORZĄDKOWANE CZĘSTE ELEMENTY
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

ELEMENTY minsup = 3	CZĘSTOTLIWOŚĆ
F	4
C	4
A	3
B	3
M	3



6. Eksploracja wzorców nie zawierających B: FCA:3, FCAM:2, C:1. tworzy warunkową bazę wzorców względem elementu B.
7. Warunkowe bazy wzorców wyznaczone są względem elementów na ścieżkach w drzewie FP-tree, które je zawierają, uwzględniając wszystkie ich prefiksy:
8. Eksplorując M-warunkową bazę wzorców: FCAM:2 ma prefiks FCA:2, zaś FCABM:1 ma prefiks FCAB:1, więc dla tych prefiksów bierzemy część wspólną: FCA:2, FCAB:1 → FCA:3
9. Eksplorując B-warunkową bazę wzorców: F:1 , C:1, FCA:1 → {}
10. Eksplorując AM-warunkową bazę wzorców (*AM-conditional pattern-base*): bierzemy pod uwagę osobno prefiksy wzorców zawierających A oraz M: czyli dla FCA:3 mamy FC:3, a dla tych zawierających FCAM:2 oraz FCABM:1 mamy FCA:2 oraz FCAB:1, więc finalnie rozważamy prefiksy: FC:3, FCA:2, FCAB:1 → FC:3

ELEMENT	WARUNKOWE BAZY WZORCÓW
A	FC:3
C	F:3
B	F:1, C:1, FCA:1
M	FCA:2, FCAB:1

EKSPLORACJA WZORCÓW D-CLUB

D-CLUB algorytm:

- ✓ służy do szybkiego wyszukiwania wzorców częstych
- ✓ stopniowo grupuje bazę danych do postaci skondensowanej asocjacyjnej bitmapy stosując technikę różnicowania w celu usunięcia gęstych wzorców, a następnie wyodrębnieniu pozostałych małych bitmap przez szybkie operacje bitowe na takich agregatach.
- ✓ stosuje mapy bitowe zorganizowane w prostokątne, dwuwymiarowe macierze, które są poprawiane w regionach, które wymagają dalszych obliczeń.
- ✓ jest dużo szybszy od Apriori oraz szybszy wielu innych metod wyszukiwania wzorców częstych, np. Eclat, FP-tree.

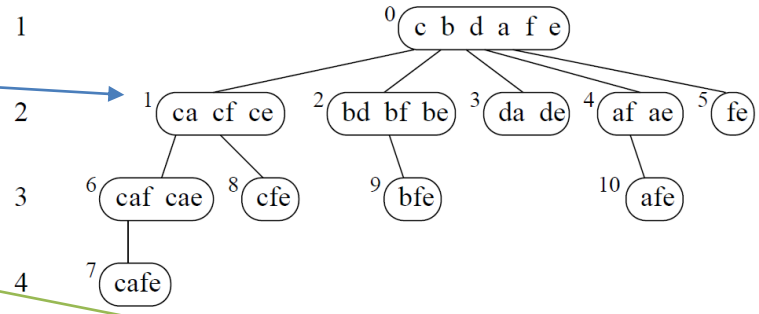
Database

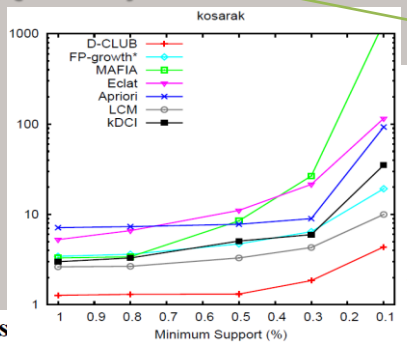
TID	Items
100	f d b e
200	f e b
300	a d b
400	a e f c
500	a d e
600	a c f e

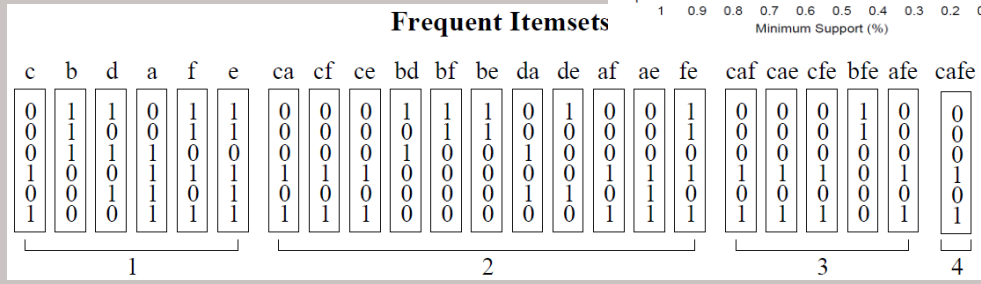
Frequent Itemsets (minsup = 33%)

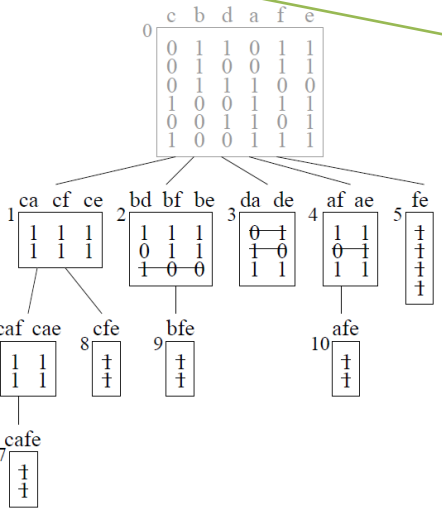
k	Frequent Itemsets : support
1	c:33% b:50% d:50% a:67% f:67% e:83%
2	ca:33% cf:33% ce:33% bd:33% bf:33% be:33% da:33% de:33% af:33% ae:50% fe:67%
3	caf:33% cae:33% cfe:33% bfe:33% afe:33%
4	cafe:33%

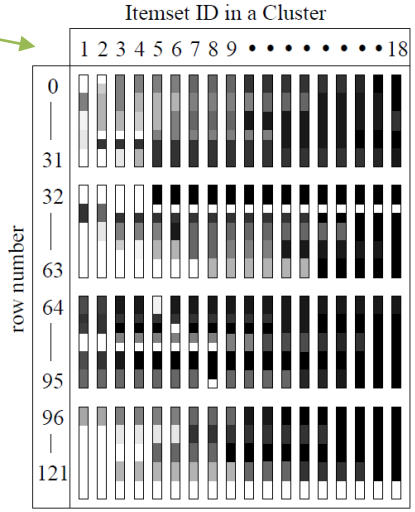
Cluster Tree of All FI's











EKSPLORACJA WZORCÓW SEKWENCYJNYCH

Wzorce sekwencyjne (*sequential patterns*) składają się z sekwencji zbiorów elementów (*sets of items*), zwanych też zdarzeniami (*events*), np.:

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

Elementy zbiorów tworzących sekwencje nie są porządkowane, tzn. ich kolejność nie ma znaczenia: np. (ABC) = (CBA) = (ACB) – zapisujemy je w nawiasach.

Dla poniższej bazy sekwencji i minimalnego progu wsparcia minsup = 3 otrzymamy sekwencyjny wzorzec (*sequential pattern*) <(AB)CA>

TRANSAKCJE	WZORCE SEKWENCYJNE
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>

Wzorce sekwencyjne mają liczne zastosowania, np. w: inżynierii oprogramowania, analizy i porównywania łańcuchów DNA, protein, sekwencji czasowych i zmian w czasie (np. na giełdzie kursów walut, akcji), procedur leczniczych w medycynie, analizie i przewidywaniu pogody, analizy, indywidualnego dostosowania ofert i optymalizacji akcji promocyjnych oraz reklamowych…

EKSPLORACJA APRIORI WZORCÓW SEKWENCYJNYCH

Eksploracja Apriori wzorców sekwencyjnych (apriori-based sequential pattern mining) polega na określeniu częstotliwości wystąpień (wsparcia/*support*) sekwencji jedno, następnie dwu, trzy, cztero… elementowych: <A>, <B>, <C>, <D>, <E>, <F>, <H>

Dla których minimalna częstotliwość czyli wsparcie (*minsup*) jest powyżej pewnego ustalonego progu, np. ≥ 5.

Stopniowo generujemy kandydatów o długości k+1 na podstawie wcześniej wygenerowanych kandydatów o długości k, przy czym zawsze bierzemy pod uwagę tylko tych kandydatów, których wsparcie jest powyżej pewnego ustalonego progu. Postępujemy tak dopóki istnieją dłuższi kandydaci spełniający to kryterium (APRIORI).

Apriori pozwala badać tylko ograniczoną ilość kandydatów, dla których wsparcie jest odpowiednio duże (powyżej dużego progu), a nie wszystkie podciągi (np. dla małego progu należałoby rozważyć zbyt dużą ilość kombinacji i algorytm byłby obliczeniowo niewykonalny).

TRANSAKCJE	WZORCE SEKWENCYJNE
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>

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

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

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

Eksploracja wzorców wygenerowanych i oczyszczonych na podstawie reguły Apriori nazywana jest algorytmem *Generalized Sequential Pattern (GSP) algorithm for Mining and Pruning*.

EKSPLORACJA TEKSTÓW I FRAZ
N-GRAMY & ALGORYTM KERT

Eksplorację tekstów poprzez wyszukiwanie n-gramów, czyli n-elementowych fraz słownych, gdzie elementami są słowa. Można rozważać również eksplorację fraz, wewnątrz których występują inne słowa (sekwencje słów z odstępami (*gaps*)).

N-gramy konstruujemy często w oparciu o (N-1)-gramy, rozpoczynając od bi-gramów.

Eksploracja tekstów przez konstruowanie fraz z często powtarzających się bliskich słów poprzez ich łączenie, scalanie i porządkowanie (wykorzystywane w silnikach indeksujących i wyszukiwawczych – *indexing and search enginess*).

Frazy tekstowe dla eksplorowanego tematu oceniamy i porządkujemy według ich:

- **Popularności (*popularity*) – czyli częstości występowania w stosunku do innych fraz, np. „*pattern mining*" względem „*text pattern mining*" lub „*sequential pattern mining*"**
- **Dyskryminatywności (*discriminativeness*) – tylko częste (*frequent*) frazy w danym dokumencie w stosunku do innych dokumentów, w których są one rzadkie (*unfrequent*)**
- **Zgodności (*concordance*) – fraza składająca się ze słów często występujących razem w stosunku do innych, które tylko okazjonalnie występują razem, np. „*machine learning*" w stosunku do „*robust learning*"**
- **Kompletności (*completness*) – „*vector machine*" w stosunku „*support vector machine*", jeśli to drugie występuje częściej niż to pierwsze w innym kontekście z innym prefiksem**

Te kryteria pozwalają porównywać frazy o różnej długości (np. algorytm KERT).

EKSPLORACJA FRAZ I MODELOWANIE TEMATÓW ALGORYTM ToPMine

KERT (***Keyphrase Extraction and Ranking by Topics, Marina Danilevsky, Chi Wang, Nihit Desai, Jingyi Guo, Jiawei Han***) **najpierw modelował temat, a następnie eksplorował frazy w tekście.**

ToPMine najpierw konstruuje frazy, a następnie eksploruje temat tekstu:

1. **Najpierw wyszukujemy częste wzorce sekwencyjne składające się z sąsiadujących elementów (czyli częste frazy kandydujące) i liczymy ilości ich wystąpień.**
2. **Łączymy częste (*frequent*) jednoelementowe sąsiadujące słowa (wzorce) we frazy wyznaczając ich częstotliwość występowania w tekście.**
3. **Frazy tworzą elementy, które często występują razem.**
4. **Wyszukujemy frazy kandydujące na podstawie częstości występowania składających się na nie słów w tekście w stosunku do częstości występowania całej frazy kandydującej.**

METODY EKSPLORACJI DANYCH OPARTE NA WIEDZY

Istnieje wiele innych metod eksploracji danych opartych na wiedzy.

Oznacza to, iż dane nie są przeszukiwane w sposób bezpośrednich, lecz tworzony jest pewien model ich reprezentacji, np. w:

- **systemach neuronowych,**
- **systemach rozmytych,**
- **systemach kognitywistycznych,**
- **systemach asocjacyjnych,**

które pozwalają na wyciąganie wniosków na podstawie pewnej formy zagregowanych i reprezentowanych wewnętrznie danych.

Uzyskane w taki sposób wnioski mogą być nie tylko odtworzenie zebranych faktów i reguł, lecz również ich uogólnieniem lub podsumowaniem.

Istotne znaczenie dla uzyskania takiej funkcjonalności systemu wnioskującego odgrywać będzie:

- **sposób reprezentacji danych w wybranym systemie,**
- **możliwość agregacji i wspólnej reprezentacji takich samych i podobnych danych,**
- **wbudowane mechanizmy wnioskowania i generalizacji.**

REPREZENTACJA DANYCH

Sposób reprezentacji danych w istotny sposób wpływa na:

- **Przechowywanie relacji pomiędzy danymi**
- **Szybkość dostępu do danych i ich relacji**
- **Możliwości eksploracji wiedzy na podstawie tych danych.**

W trakcie eksploracji najczęściej poszukujemy:

- **Częstych (*frequent*) grup danych – wzorców (*patterns*) – określonych na podstawie ich podobieństwa**

Zależy nam na:

- **Szybkości dostępu do danych i ich relacji**
- **Możliwości szybkiej eksploracji wiedzy na podstawie tych danych.**

Wiedza o danych – to przede wszystkim informacje o ich:

- **Związkach (relacjach)**
- **Podobieństwie i różnicach**
- **Klasach, grupach i grupowaniu**

BIBLIOGRAFIA
I LITERATURA UZUPEŁNIAJĄCA

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

DATA MINING

WYDOBYWA INFORMACJE Z DANYCH I POSZERZA WIEDZĘ!