CWID: A20555359

CS422 Data Mining

Assignment 2

1. Recitation Exercises

1.1 Chapter 8

(1) Frequent Itemset Mining with Apriori and FP-Growth

Given:

tid	itemset
t1	ABCD
t2	ACDF
t3	ACDEG
t4	ABDF
t5	BCG
t6	DFG
t7	ABG
t8	CDFG

a) Apriori Algorithm (minsup = 3/8)
(because minsup = 3/8 = 0.375 × 8 = 3)

Step 1: Calculate support counts (number of transactions containing each item)

Item counts:
A: appears in t1, t2, t3, t4, t7 -> 5 times
B: appears in t1, t4, t5, t7 -> 4 times
C: appears in t1, t2, t3, t5, t8 -> 5 times
D: appears in t1, t2, t3, t4, t6, t8 -> 6 times
E: appears in t3 -> 1 time
F: appears in t2, t4, t6, t8 -> 4 times

G: appears in t3, t5, t6, t7, t8 -> 5 times

Minimum support count required = 3 (because minsup = 3/8 = 0.375)

Discard E because support = 1 < 3 (does not meet minsup threshold)

Step 2: Frequent 1-itemsets (support ≥ 3):
{A}, {B}, {C}, {D}, {F}, {G}

Step 3: Generate candidate 2-itemsets from frequent 1-itemsets, count support, and prune those below minsup:
Examples:
- {A, B} appears in t1, t4, t7 → support = 3 (keep)
- {A, C} appears in t1, t2, t3 → support = 3 (keep)
- {A, D} appears in t1, t2, t3, t4 → support = 4 (keep)
- {C, D} appears in t1, t2, t3, t8 → support = 4 (keep)
- {D, F} appears in t2, t4, t6, t8 → support = 4 (keep)
- {C, G} appears in t3, t5, t8 → support = 3 (keep)
- {G, D} appears in t3, t6, t8 → support = 3 (keep)

Step 4: Candidate 3-itemsets:
- {A, C, D} appears in t1, t2, t3 → support = 3 (keep)

Final frequent itemsets summary:

Length 1:
{D} (6/8)
{A} (5/8)
{C} (5/8)
{G} (5/8)

{B} (4/8)
{F} (4/8)

Length 2:
{A, D}, {C, D}, {D, F}, {A, B}, {A, C}, {C, G}, {G, D}

Length 3:
{A, C, D}

b) FP-Growth Algorithm (minsup = 2/8)
(minimum support count = 2)

Item counts (same as above):
A: 5 ≥ 2 yes
B: 4 ≥ 2 yes
C: 5 ≥ 2 yes
D: 6 ≥ 2 yes
E: 1 < 2 no (discard)
F: 4 ≥ 2 yes
G: 5 ≥ 2 yes

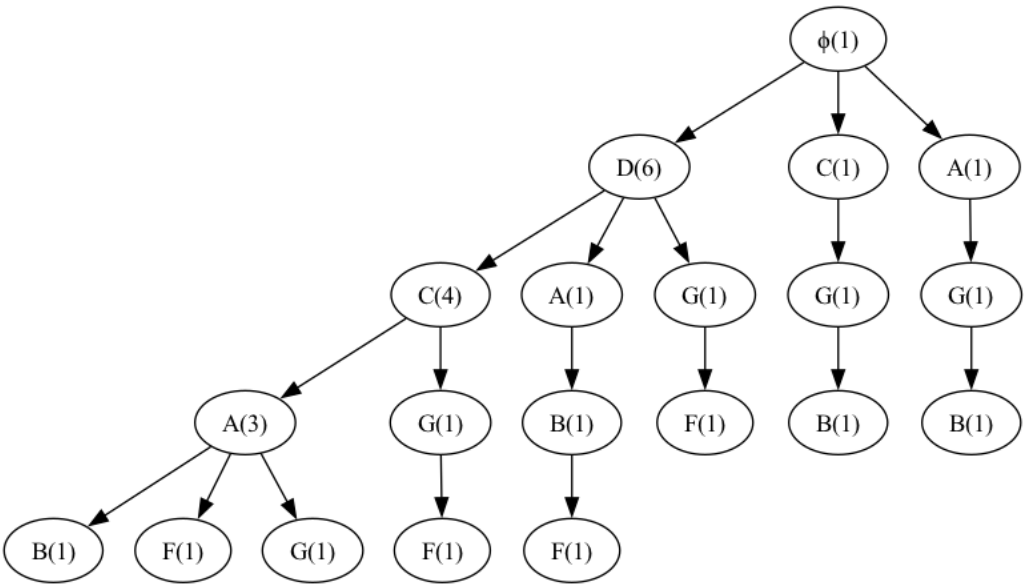
Step 1: Order frequent items by descending support (tie broken arbitrarily):
D(6), C(5), A(5), G(5), B(4), F(4)

Step 2: Reorder transactions by this order (remove E):

t1: D C A B
t2: D C A F
t3: D C A G

t4: D A B F
t5: C G B
t6: D G F
t7: A G B
t8: D C G F

Step 3: Build FP-tree



<u>FP-tree Construction:</u> Step-by-Step Explanation

Step 0: Preprocessing – Frequency Count and Header Table Creation

1. Count the frequency of each item in all transactions.
   Example:
   D: 6, C: 5, A: 5, B: 3, F: 3, G: 5

2. Filter out items with frequency less than the minimum support (min_support = 2).
   All items remain since all have support ≥ 2.

3. Create the header table, which stores for each frequent item:
   - Its support count
   - A pointer to the first node in the FP-tree containing that item (initially None)

Step 1: Insert Transaction ['D', 'C', 'A', 'B']
- Filter and sort items by frequency: ['D', 'C', 'A', 'B']
- Insert nodes:
  φ (root) → D(1) → C(1) → A(1) → B(1)
- Update header table node-links for D, C, A, B

Step 2: Insert Transaction ['D', 'C', 'A', 'F']
- Sorted: ['D', 'C', 'A', 'F']
- Shared prefix with Step 1: D → C → A (increment counts)
- Add new node: F(1) under A
- Path after insertion:
  φ → D(2) → C(2) → A(2) → F(1)
- Update header table node-link for F

Step 3: Insert Transaction ['D', 'C', 'A', 'G']
- Sorted: ['D', 'C', 'A', 'G']
- Shared prefix: D → C → A (increment counts)
- Add new node: G(1) under A
- Path after insertion:
  φ → D(3) → C(3) → A(3) → G(1)
- Update header table node-link for G

Step 4: Insert Transaction ['D', 'A', 'B', 'F']

- Sorted: ['D', 'A', 'B', 'F']
- Increment D count: D(4)
- Add new branch under D:
  A(1) → B(1) → F(1)
- Update header table node-links accordingly

Step 5: Insert Transaction ['C', 'G', 'B']
- Sorted: ['C', 'G', 'B']
- Add new branch from root:
  C(1) → G(1) → B(1)
- Update header table node-links for C, G, B

Step 6: Insert Transaction ['D', 'G', 'F']
- Sorted: ['D', 'G', 'F']
- Increment D count: D(5)
- Add new branch under D:
  G(1) → F(1)
- Update header table node-links for G, F

Step 7: Insert Transaction ['A', 'G', 'B']
- Sorted: ['A', 'G', 'B']
- Add new branch from root:
  A(1) → G(1) → B(1)
- Update header table node-links for A, G, B

Step 8: Insert Transaction ['D', 'C', 'G', 'F']

- Sorted: ['D', 'C', 'G', 'F']
- Increment counts: D(6), C(4)
- Add new branch under C:
  G(1) → F(1)
- Update header table node-links for G, F

Step 4: Mine FP-tree for frequent itemsets with support ≥ 2:

Frequent 1-itemsets:
{D} (6), {C} (5), {A} (5), {G} (5), {B} (4), {F} (4)

Frequent 2-itemsets (selected from Jupyter output):
{C, D} (4), {A, D} (4), {D, F} (4), {G, C} (3), {A, C} (3), {A, B} (3), {G, D} (3), {A, G} (2), {B, C} (2), {B, D} (2), {G, B} (2), {A, F} (2), {C, F} (2), {G, F} (2)

Frequent 3-itemsets (selected):
{A, C, D} (3), {G, C, D} (2), {A, B, D} (2), {A, D, F} (2), {C, D, F} (2), {D, F, G} (2)

(2) Proof: sup(Xab) = sup(Xa) - |d(Xab)|

Step 1:
Let Xa = {x1, x2, ..., x(k-1), xa}
Let Xb = {x1, x2, ..., x(k-1), xb}
So, they share the common (k-1)-itemset:
X = {x1, x2, ..., x(k-1)}

Step 2:
Let Xab = Xa ∪ Xb = {x1, ..., x(k-1), xa, xb}

Step 3:
Define support:
sup(Xa) = number of transactions containing Xa = |T(Xa)|
sup(Xab) = number of transactions containing Xab = |T(Xab)|

Step 4:
Since Xab is the union of Xa and Xb,
Then T(Xab) = T(Xa) ∩ T(Xb)

Step 5:
Define the diffset:
d(Xab) = T(Xa) – T(Xab)

Step 6:
So, the size of the diffset:
|d(Xab)| = |T(Xa)| - |T(Xab)| = sup(Xa) – sup(Xab)

Step 7:
Rewriting:
sup(Xab) = sup(Xa) - |d(Xab)|

Conclusion:
This proves the relationship using the diffset method.

(3) From the dataset in Table 8.4, show all association rules that can be generated from the itemset ABE.

Dataset:

- tid: t1
  itemset: [A, C, D]
- tid: t2
  itemset: [B, C, E]
- tid: t3
  itemset: [A, B, C, E]
- tid: t4
  itemset: [B, D, E]
- tid: t5
  itemset: [A, B, C, E]
- tid: t6
  itemset: [A, B, C, D]

Step 1: Identify all non-empty proper subsets of the itemset ABE.

Subsets:

- Size_1: [A, B, E]
- Size_2: [AB, AE, BE]

Step 2: List all possible association rules from the itemset ABE by pairing subsets with their complements.

Rules:

- A → BE
- B → AE
- E → AB
- AB → E
- AE → B
- BE → A

Step 3: Calculate support counts for the itemset ABE and all its subsets using the given dataset.

Support_counts:

- ABE: 2 # transactions t3, t5
- A: 4 # transactions t1, t3, t5, t6
- B: 5 # transactions t2, t3, t4, t5, t6
- E: 4 # transactions t2, t3, t4, t5
- AB: 3 # transactions t3, t5, t6
- AE: 2 # transactions t3, t5
- BE: 4 # transactions t2, t3, t4, t5

Step 4: Compute the confidence for each association rule using the formula:

confidence(X → Y) = support(X ∪ Y) / support(X)

Confidence_values:

- A → BE: 2 / 4 = 0.50
- B → AE: 2 / 5 = 0.40
- E → AB: 2 / 4 = 0.50
- AB → E: 2 / 3 ≈ 0.67
- AE → B: 2 / 2 = 1.00
- BE → A: 2 / 4 = 0.50

Step 5: Summarize all valid association rules derived from itemset ABE with their confidence values.

Final_rules:

- Rule: "A → BE"
  Confidence: 0.50
- Rule: "B → AE"
  Confidence: 0.40
- Rule: "E → AB"
  Confidence: 0.50

- Rule: “AB → E”
  Confidence: 0.67
- Rule: “AE → B”
  Confidence: 1.00
- Rule: “BE → A”
  Confidence: 0.50

1.2 Chapter 9

(1) True or False.

(a) Maximal frequent itemsets are sufficient to determine all frequent itemsets with their supports.

This statement is False. The fundamental issue lies in the distinction between determining membership and determining exact support values. While maximal frequent itemsets can tell us which itemsets are frequent due to the antimonotone property (any subset of a frequent itemset must also be frequent), they cannot provide the specific support counts for these subsets. For instance, if we know that itemset {A,B,C} is maximal frequent with 30% support, we can conclude that all its subsets like {A,B}, {A,C}, and {A} are frequent, but we have no way to determine whether {A,B} has 30%, 45%, or any other support value greater than the minimum threshold. The maximal itemset only preserves its own support information, not the support distribution of its constituent subsets.

(b) An itemset and its closure share the same set of transactions.

This statement is **True**. The closure of an itemset is defined as the maximal set of items that co-occur in exactly the same transactions as the original itemset. By this very definition, both the original itemset and its closure must appear in identical transaction sets. This occurs because the closure operation finds the intersection of all items across the transactions where the original itemset appears. If the closure appeared in additional

transactions, those transactions would necessarily contain the original itemset as well. Conversely, if the closure appeared in fewer transactions, the original itemset couldn't possibly appear in the excluded transactions. This identical transaction coverage directly implies that both itemsets have exactly the same support value.

© The set of all maximal frequent sets is a subset of the set of all closed frequent itemsets.

This statement is **True**. The relationship stems from the inherent properties of maximal itemsets. Consider any maximal frequent itemset M – by definition, it cannot be extended with additional items while maintaining the minimum support threshold. Now, if M were not closed, this would mean that its closure contains additional items while appearing in the same transactions (and thus having the same support). However, this creates a contradiction because the closure would be both frequent (same support as M) and a proper superset of M, which violates the maximality of M. Therefore, every maximal frequent itemset must also be closed, establishing that the set of maximal frequent itemsets is indeed a subset of closed frequent itemsets.

(d) The set of all maximal frequent sets is the set of longest possible frequent itemsets.

This statement is **True**. The concept of "longest possible" in the context of frequent itemsets refers to itemsets that cannot be extended further while maintaining the frequency requirement. This is precisely what defines maximal frequent itemsets – they represent the boundary between frequent and infrequent patterns in the itemset lattice. Any attempt to add items to a maximal frequent itemset would either result in an infrequent itemset or an itemset that doesn't exist in the database with sufficient support. These maximal itemsets effectively capture the most comprehensive frequent patterns possible in the dataset,

representing the maximum extent to which items can be combined while still meeting the support threshold. Therefore, the set of maximal frequent itemsets exactly corresponds to the set of longest possible frequent itemsets.

1.3 Chapter 12

(1) Show that if X and Y are independent, then the conviction of the rule (X → Y) = 1.

Step 1: Define conviction
Formula: conv(X → Y) = P(X) * P(¬Y) / P(X ∧ ¬Y)

Step 2: Apply the definition of independence
- If X and Y are independent:
  - P(X ∧ Y) = P(X) * P(Y)
  - P(X ∧ ¬Y) = P(X) * P(¬Y)

Step 3: Substitute independence into conviction formula
- numerator: P(X) * P(¬Y)
- denominator: P(X) * P(¬Y)

conv(X → Y) = [P(X) * P(¬Y)] / [P(X) * P(¬Y)] = 1
result: conv(X → Y) = 1

Therefore, if X and Y are independent, then conviction equals 1.
This confirms that conviction detects dependency between X and Y.

(2) Show that if X and Y are independent, then oddsratio (X → Y) = 1.

Step 1: Define odds for Y given X

Formula: odds(Y|X) = P(X ∧ Y) / P(X ∧ ¬Y)

This compares how likely Y is when X occurs versus when it does not occur.

Numerator is the probability of both X and Y.

Denominator is the probability of X and not Y.

Step 2: Define odds for Y given ¬X

Formula: odds(Y|¬X) = P(¬X ∧ Y) / P(¬X ∧ ¬Y)

Similar to step 1, but for when X does not occur.

This captures how likely Y is when X is absent.

Step 3: Define odds ratio

Formula: oddsratio(X → Y) = odds(Y|X) / odds(Y|¬X)

= [P(X ∧ Y) * P(¬X ∧ ¬Y)] / [P(X ∧ ¬Y) * P(¬X ∧ Y)]

This is the ratio of two odds values. The formula uses all four cells in the contingency table.

Step 4: Assume independence of X and Y

If X and Y are independent, then:

- P(X ∧ Y) = P(X) * P(Y)
- P(X ∧ ¬Y) = P(X) * P(¬Y)
- P(¬X ∧ Y) = P(¬X) * P(Y)
- P(¬X ∧ ¬Y) = P(¬X) * P(¬Y)

Step 6: Substitute using independence

numerator: P(X) * P(Y) * P(¬X) * P(¬Y)

denominator: P(X) * P(¬Y) * P(¬X) * P(Y)

formula:

oddsratio = [P(X) * P(Y) * P(¬X) * P(¬Y)] / [P(X) * P(¬Y) * P(¬X) * P(Y)]

Under statistical independence of X and Y, the odds ratio equals 1.

This indicates no association between X and Y in either direction.

(3) Show that for a frequent itemset X, the value of the relative lift statistic defined in Example 12.20 lies in the range
[1− |D| / minsup, 1]

Given Information:

- **Frequent itemset: sup(X, D)minsup**
- **Relative lift formula: rlift(X, D, Di) = 1 – sup(X, Di)/sup(X, D)**
- **Dataset size: |D|= number of transactions**
- **Randomized dataset: Di is a randomized dataset**

Step 1: Establish bounds for sup(X, Di)

Since Di is a randomized dataset, the support of itemset X in Di can range from minimum value of 0 when itemset X never occurs in randomized data to maximum value of |D| when itemset X occurs in every transaction.

Bounds:

- **minimum:sup(X, Di) = 0**
- **maximum: sup(X, Di) = |D|**

Step 2: Find the upper bound of relative lift

When sup(X, Di) = 0, which is the minimum possible support in randomized data, we get rlift(X, D, Di) = 1 – 0/sup(X, D) = 1 – 0 = 1.

Result: upper bound = 1

Step 3: Find the lower bound of relative lift

When sup(X, Di) = |D|, which is the maximum possible support in randomized data, we get rlift(X, D, Di) = 1 - |D|/sup(X, D). Since X is

frequent, we know sup(X, D) ≥ minsup. To minimize the relative lift, we want to maximize |D|/sup(X, D). This is maximized when sup(X, D) is at its minimum value, which is minsup. Therefore rlift(X, D, Di) = 1 - |D|/minsup.

Result: lower_bound = 1 - |D|/minsup

Step 4: Verification of the range

The relative lift statistic ranges from lower bound of 1 - |D|/minsup when randomized support is maximized to upper bound of 1 when randomized support is minimized. Range: [1 - |D|/minsup, 1]

Interpretation:

high_value: When **rlift ≈ 1**, the itemset rarely occurs in randomized data, suggesting it is a meaningful pattern.

Low_value: When **rlift ≈ 1 - |D|/minsup**, the itemset occurs very frequently in randomized data, possibly in every transaction, suggesting it might just be due to high marginal frequencies.

Zero_value: **When rlift ≈ 0**, the itemset has similar support in both original and randomized datasets.

Conclusion:

For a frequent itemset X, the relative lift statistic lies in the range [1 - |D|/minsup, 1].

Citations:

1. https://www.ceom.ou.edu/media/docs/upload/Pang-Ning_Tan_Michael_Steinbach_Vipin_Kumar_-_Introduction_to_Data_Mining-Pe_NRDK4fi.pdf

2. https://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf

3. **Zaki, M. J., & Meira Jr, W. (2020). Data Mining and Machine Learning: Fundamental Concepts and Algorithms, 2nd Edition.**

4. **Assistance was taken from a large language model (ChatGPT by OpenAI) to clarify concepts and provide explanations related to frequent itemset mining, association rule mining, and various related metrics. The model helped in organizing and summarizing key topics including algorithms (Apriori, FP-Growth), properties of frequent itemsets (maximal, closed), and association rule evaluation metrics (conviction, odds ratio, relative lift).**