CS5228 – Tutorial 4

Association Rule Mining

Association rule mining is a method with the goal to identify frequently occurring patterns, correlations, or associations from transactional or similar datasets. The most popular application context is in Market Basket Analysis where Association Rule Mining is used to identify supermarket products that are frequently bought together (but we also saw another example in the lecture)

1. **Basic definitions.** Association Rule Mining aims to find "interesting" rules, interesting co-occurrences of items. To quantify "interesting", a series of different metrics exist. In the lecture, we mainly covered *support*, *confidence*, and *lift*. For the following tasks, let's assume the following basic dataset comprising 8 transactions.

tid	transactions
1	B, C, D, E, F
2	E, C
3	D, B, D, A
4	G, E, H, C
5	H, A, G, D, B
6	B, E, G
7	B, A, D
8	A, D, B, C

   (a) **Calculate the following values:**

   - $support(\{A\})$, $support(\{B\})$ $support(\{A, B\})$
   - $support(\{A\} \rightarrow \{B\})$, $support(\{B\} \rightarrow \{A\})$
   - $confidence(\{A\} \rightarrow \{B\})$, $confidence(\{B\} \rightarrow \{A\})$
   - $lift(\{A\} \rightarrow \{B\})$, $lift(\{B\} \rightarrow \{A\})$

   Solution:

   - $support(\{A\}) = 1/2$, $support(\{B\}) = 3/4$ $support(\{A, B\}) = 1/2$
   - $support(\{A\} \rightarrow \{B\}) = 1/2$, $support(\{B\} \rightarrow \{A\}) = 1/2$
   - $confidence(\{A\} \rightarrow \{B\}) = 1.0$, $confidence(\{B\} \rightarrow \{A\}) = 2/3$
   - $lift(\{A\} \rightarrow \{B\}) = 4/3$, $lift(\{B\} \rightarrow \{A\}) = 4/3$

(b) **Connections between metrics.** In a) we calculated the *support*, *confidence*, and *lift* for different itemsets and association rules. However, we do not need to calculate all values individually. Based on the definitions of the metrics, which calculations can we skip?

> **Solution:**
>
> - $support(X \to Y) = support(X \cup Y)$
> - $support(X \to Y) = support(Y \to X)$
> - $lift(X \to Y) = lift(Y \to X)$ based on its definition

(c) **"Usefulness" of different metrics.** What makes *support* and *confidence* more useful compared to other metrics such as *lift*, *conviction*, *collective strength*, *leverage*?

> **Solution:** Both *support* and *confidence* have the anti-monotone property which can be exploited for the Apriori Algorithm for finding association rules

(d) **"Importance" of different metrics.** What makes an association rule interesting? A high *support*, high *confidence*, high *lift*, high *conviction*, etc.?

> **Solution:** No metric is intrinsically the most indicative one for describing how interesting an association rule is. All metrics look at different aspects that make a rule interesting and which aspect might be most important typically depends on the exact application context.

2. **Finding the relevant rules.** In the lecture, we were mainly interested in association rules with a high support and a high confidence. This was also convenient since large(r) values for *minsup* and *minconf* typically speed up the execution of the Apriori algorithm.

   In the following, let's assume a transaction dataset for medical data analysis similar to what we saw on the lecture slides. In more details, each transaction contains as items a set of symptoms, together with one item indicating a positive or negative COVID-19 test result. For example, our dataset might look like this

tid	transactions
1	cough, fatigue, COVID-19-negative
2	anosmia, cough, fatigue, COVID-19-positive
3	anosmia, fatigue, headache, heart palpitations, COVID-19-positive
4	cough, fatigue, headache, COVID-19-negative
5	headache, stomach pain, COVID-19-negative
6	cough, heart palpitations, COVID-19-negative
7	anosmia, headache, stomach pain, COVID-19-positive
...	...

Our task is now to find rules that indicate which set of symptoms are most likely associated with a positive COVID-19 test. For example, we want to find rules like {anosmia, fatigue}→{COVID-19-positive}.

(a) **Choice of *minsup* and *minconf*.** How might the setup and the task above affect which values for *minsup* and *minconf* are meaningful? Hint: Assume that a large majority of the COVID-19 test results in our dataset are negative.

> **Solution:** If we assume that most COVID-19 test results have been negative, then any rules where {COVID-19-positive} is on the right-hand side won't be very frequent (compared to rules where {COVID-19-negative} is on the right-hand side). As such, we cannot set *minsup* too high, or we won't find any relevant rule.

(b) **Tweaking the dataset.** Can we simplify this task by only considering those transactions that contain COVID-19-positive, and remove all transactions that contain COVID-19-negative?

> **Solution:** No, we should not do this as that might yield misleading results. For example, we might get a rule with a high support, say, {cough}→{COVID-19-positive}. However, if a cough is so common even for negative results, we would overestimate its importance as an indicator for a positive test result. Also note that any rules with {COVID-19-positive} on the right-hand side would have a confidence of 1.0.

3. **Complexity Analysis.** The most naive approach for mining association rules would be to generate all possible rules and check if their support and confidence exceeds the specified thresholds *minsup* and $minconf$. In the lecture, you have learned that, given d unique items in a dataset of transactions, there are $3^d - 2^{d+1} + 1$ possible rules.

   Proof that d unique items result in $3^d - 2^{d+1} + 1$ possible rules! (Hint: Write out all possible rules for $d = 2, 3, 4, ...$ items; you should quickly spot the pattern that will allow you to validate the formula).

> **Solution:**
>
> - Each item has 3 possibilities to appear in a rule: on the left side of the rule, on the right side of the rule, or not at all. That reflects the 3^d possibilities.
> - However, these 3^d rules include invalid ones where the left and/or right side of the rule is empty. Of the 3^d rules, there are 2^d where the left side is empty, and 2^d where the right side is empty. We have to subtract these invalid combinations, and $2^d + 2^d = 2^{d+1}$.
> - Note that we now have subtracted the rule $\{\} \rightarrow \{\}$ twice. So we need '+1' to correct for this.