UNIVERSITY OF GRONINGEN
FACULTY OF SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTING SCIENCE

Information Systems

Market Basket Analysis - Association Rules

Group 07

AUTHORS
Frans Simanjuntak (S3038971)
Stefan Cretu(S3048438)

January 16 2018

Contents

1. Introduction

In this document is presented the solution for the Association Rules Analysis, part of the assignment 5 of Information Systems course. The purpose of the assignment is to implement one of association rules algorithms so-called apriori algorithm and obtain the rules. In this assignment, we analyse transactions of file groceries.txt that contains 9835 rows.

The rest of the document is structured as following: Chapter 2 describes the theoretical aspect, followed by the implementation of Apriori algorithm in Chapter 3. Finally, Chapter 4 answers all questions given in this assignment.

2. Theoretical Aspects

Association rule mining is a procedure which is meant to find frequent patterns, correlations, associations, or causal structures from data sets found in various kinds of databases such as relational databases, transactional databases, and other forms of data repositories [1]. Given a set of transactions, association rule mining aims to find the rules which enable us to predict the occurrence of a specific item based on the occurrences of the other items in the transaction.

The main applications of association rule mining:

- ***Basket data analysis*** is to analyze the association of purchased items in a single basket or single purchase as per the examples given above.
- ***Cross marketing*** is to work with other businesses that complement your own, not competitors. For example, vehicle dealerships and manufacturers have cross marketing campaigns with oil and gas companies for obvious reasons.
- ***Catalog design*** is the selection of items in a business catalog are often designed to complement each other so that buying one item will lead to buying of another. So these items are often complements or very related.

An association rule has two parts, an antecedent and a consequent. An antecedent(if) is an item found in the data. A consequent(then) is an item that is found in combination with the antecedent [2]. Association rules are created by analyzing data for frequent if/then patterns and using the criteria *support* and *confidence* to identify the most important relationships. *Support* is an indication of how frequently the items appear in the database. *Confidence* indicates the number of times the if/then statements have been found to be true.

The association rules mining problem can be formally stated as
"Given a set of transactions T, find all the rules having support ≥ minimum support and confidence ≥ minimum confidence, where minimum support and minimum confidence are the corresponding support and confidence threshold".

A common strategy adopted by many association rule mining algorithms is to decompose the problem into two major subtasks:

1. **Frequent Itemset Generation**, whose objective is to find all the itemsets that satisfy the minimum support threshold. These itemsets are called frequent itemsets.
2. **Rule Generation**, whose objective is to extract all the high-confidence rules from the frequent itemsets found in the previous step. There rules are called strong rules.

This strategy is also adopted by the most popular association rule mining algorithm so-called Apriori algorithm. The principle of Apriori algorithm is if an itemset is frequent, then all of

its subset must also be frequent. The implementation of this algorithm is explained in chapter 3.

3. Implementation

The implementation of the apriori algorithm, in the case of assignment 6, is related to an input text file, called *groceries.txt,* which contains the input data for the algorithm. Thus, firstly the file is parsed in order to obtain the itemset which is sent as input to the function that implements the apriori algorithm. Afterwards, the apriori algorithm is called to compute the frequent itemsets from the input itemset. The last step is to mine the association rules in the frequent itemset output at the previous step.

Therefore, our implementation model is structured in 3 steps, as follows: input data manipulation, computing the frequent itemsets (apriori algorithm), generate association rules. These steps are detailed below.

3.1 Input data manipulation

As mentioned above, the input data is stored in a text file that contains 9835 transactions. Each transaction is on a separate row and has at least one element that represent one product bought in the market. If the transaction contains multiple elements, those are separated by commas. Each transaction ends with a backslash.

Once the file is open, it is read line by line. Then each line is parsed, with the backslash not being taken into consideration. For each row is created a set containing its elements and each such a set is appended to a list of transactions. In the end, the list of transaction contains all the elements in the input file in an appropriate form to be further processed.

3.2 Computing frequent itemsets - Apriori algorithm

The apriori algorithm is called to compute the frequent items in the transaction list processed at the previous step. It is implemented in the function called *apriori_frequent_itemset,* which takes as input a list of candidates (Ck) and the value for the minimum support.

Then, in a loop, it is computed the set of items which have the minimum support (Lk) by calling the function *itemset_support,* which takes the same inputs, namely the candidates_list and the minimum support. The call of this function represents the pruning step.

This function firstly creates a list of tuples (item, item_support), by searching for each item in the transaction list and counting its apparitions, then computing the support by dividing the number of apparitions to the length of the transaction list. Thereafter, the list of tuples is converted to a dictionary containing all tuples whose support is greater than or equal to the minimum support. This dictionary represents the Lk.

However, in the above-mentioned loop is checked the iteration number, as from the second one onwards it is applied the self joining step, as each Lk computed in the previous iteration joins itself in order to generate the list of candidates for the current iteration, Ck+1. For the first step it is not necessary as the list of candidates is the entire transaction list sent as input.

The self joining logic is performed by calling the function *selfjoin* which takes as inputs the itemset Lk-1 and the number of current iteration, that is *k*. It searches for each item in the input set, in a double loop, and joins the items as long as the resulting set matches the length corresponding to step *k*, that is *k*. The result of this logic is a set representing the set of candidates for step *k*.

After each iteration, the Lk is appended to a dictionary containing the frequent itemsets. The loop ends when an Lk cannot be computed anymore (it is empty) and the current values in the dictionary being the set of frequent items.

3.3 Generate association rules

The association rules computation is done in function *generate_association_rules*, which takes as inputs the values for minimum support and for minimum confidence, and outputs a list of rules that have minimum confidence, as well as the frequent itemset.

Here, the transaction list is further processed as it is needed to compute the itemset, that contains all elements in the transaction list, each one being included once. Afterwards, this itemset represents the input to the apriori algorithm, and acts as the list of candidates for the first iteration. Thereafter, the frequent itemset is computed by calling the logic described in section 3.2.

From this point, the mining of association rules starts by iterating through the frequent itemset. Then, for each set in the itemset, which has at least 2 elements, association rules are generated. This happens the following way: each element in the given set is considered as first object, let's call it A, and forms the antecedent set. The difference between the initial set and the set A is the second object, that forms the consequent set B. If the set B contains at least one element, then it is computed the confidence of each rule from A to each element in the set B, including B itself, as support(A|element of B)/support(A).

If the result value is greater than or equal than the input minimum confidence, the rule is appended to the list of association rules. This list contains triples (object 1, object 2, confidence).

4. Association Rules

In this chapter are presented the results for the questions listed in the assignment tasks.

4.1 Histogram of items

Figure 1 shows the histogram of Groceries.

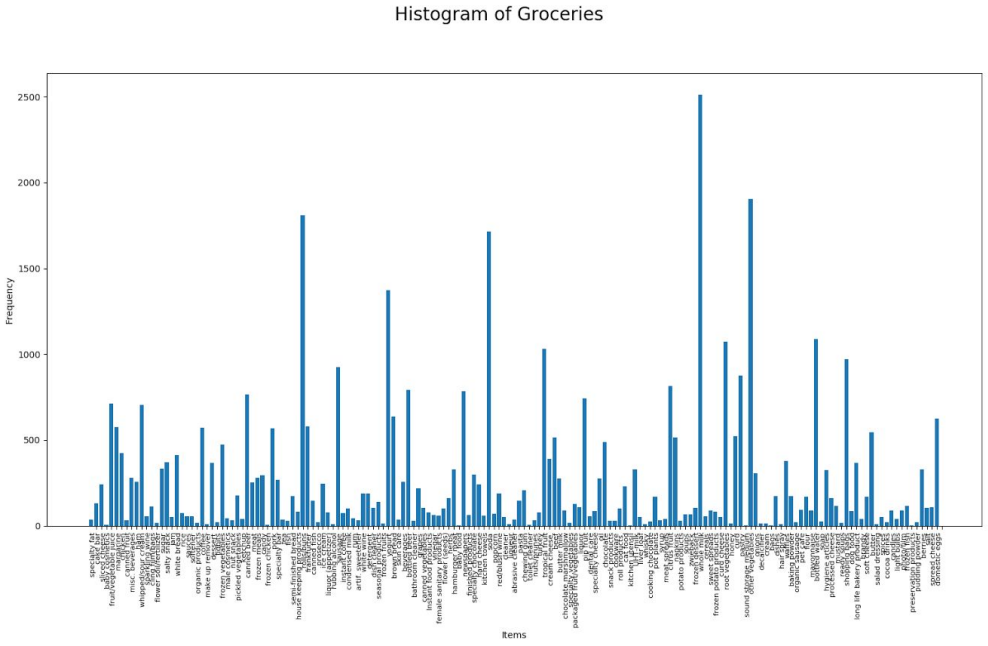

Figure 1. Histogram of Groceries

* Note: Please refer to file histogram.png for better view

It can be seen from figure 1 that the item *whole milk* is the most bought item which appears 2513 times in the whole transactions. It is followed by other vegetables, rolls/buns, and soda in which they appear in 1903,1809, 1715 transactions respectively. Conversely, the least bought items are baby food and sound storage medium which only appear in one transaction only. On average, each items is present in 256 transaction with standard deviation 377. A larger standard deviation indicates the items are more spread out over a wider range of values.

4.2 How many rules satisfy the following input parameter values? support = 0.001, confidence = 0.8?

There are **409 rules** rules that satisfy the minimum support 0.001 and minimum confidence 0.8.

4.3 Sort the association rules by confidence in descending order and list the top 30 rules?

In order to obtain the top 30 rules, we execute this code.

```
association_rules, freq_itemset = generate_association_rules(min_support,
min_confidence)

counter = 0;

for obj1, obj2, confidence in sorted(association_rules, key=lambda (obj1, obj2,
confidence): confidence, reverse=True):

print('[R] {} => {} : {}'.format(tuple(obj1), tuple(obj2), round(confidence, 4)))

If (counter == 30):
    break;

counter = counter+1
```

The top 30 rules are shown below

No	Rules	Confidence (%)
1	('rice', 'sugar') => ('whole milk',)	100
2	('butter', 'pip fruit', 'hygiene articles') => ('whole milk',)	100
3	('citrus fruit', 'rolls/buns', 'pastry', 'whipped/sour cream') => ('whole milk',)	100
4	('cream cheese', 'domestic eggs', 'sugar') => ('whole milk',)	100
5	('pip fruit', 'brown bread', 'whipped/sour cream') => ('other vegetables',)	100
6	('whole milk', 'newspapers', 'rolls/buns', 'soda') => ('other vegetables',)	100
7	('whole milk', 'pip fruit', 'ham', 'tropical fruit') => ('other vegetables',)	100
8	('tropical fruit', 'yogurt', 'oil', 'root vegetables') => ('whole milk',)	100
9	('yogurt', 'other vegetables', 'oil', 'root vegetables') => ('whole milk',)	100
10	('flour', 'whipped/sour cream', 'root vegetables') => ('whole milk',)	100
11	('pip fruit', 'hygiene articles', 'root vegetables') => ('whole milk',)	100
12	('butter', 'tropical fruit', 'fruit/vegetable juice', 'whipped/sour cream') => ('other vegetables',)	100
13	('pip fruit', 'other vegetables', 'bottled water', 'root vegetables') => ('whole milk',)	100
14	('butter', 'rice', 'root vegetables') => ('whole milk',)	100
15	('cream cheese', 'napkins', 'domestic eggs') => ('whole milk',)	100

16	('butter', 'other vegetables', 'domestic eggs', 'whipped/sour cream') => ('whole milk',)	100
17	('whole milk', 'tropical fruit', 'yogurt', 'grapes') => ('other vegetables',)	100
18	('citrus fruit', 'tropical fruit', 'root vegetables', 'whipped/sour cream') => ('other vegetables',)	100
19	('butter', 'pork', 'other vegetables', 'whipped/sour cream') => ('whole milk',)	100
20	('butter', 'white bread', 'other vegetables', 'root vegetables') => ('whole milk',)	100
21	('pip fruit', 'tropical fruit', 'yogurt', 'ham') => ('other vegetables',)	100
22	('tropical fruit', 'sausage', 'rolls/buns', 'root vegetables') => ('whole milk',)	100
23	('citrus fruit', 'soft cheese', 'root vegetables') => ('other vegetables',)	100
24	('curd', 'domestic eggs', 'sugar') => ('whole milk',)	100
25	('hygiene articles', 'canned fish') => ('whole milk',)	100
26	('hygiene articles', 'whipped/sour cream', 'root vegetables') => ('whole milk',)	100
27	('tropical fruit', 'yogurt', 'other vegetables', 'oil', 'root vegetables') => ('whole milk',)	100
28	('butter', 'soft cheese', 'domestic eggs') => ('whole milk',)	100
29	('cream cheese', 'other vegetables', 'sugar') => ('whole milk',)	93.75
30	('tropical fruit', 'yogurt', 'sausage', 'root vegetables') => ('whole milk',)	93.75

4.4 Compare and contrast the following measures of interestingness: support, confidence, lift, conviction and leverage?

- **Support**

Support is an indication of how frequently the itemset appears in the dataset. Given an itemset A, the support of A is the percentage of transactions that contain A. For example, if 85% of all transactions contain itemset {milk}, then the support of {milk} is 0.85. Similarly, if 70% of all transactions contain itemset {bread,butter}, then the support of {bread,butter} is 0.7.

A frequent itemset has items that appear together often enough. The term “often enough” is formally defined with minimum support criterion. If the minimum support is set to 0.4, any itemset can be considered a frequent itemset if at least 40% of the transactions contain this itemset. In other words, the support of a frequent itemset should be greater than or equal to the minimum support.

- **Confidence**

Confidence is defined as the measure of certainty or trustworthiness associated with each discovered rule. Mathematically, confidence is the percent of transactions that contain both X and Y out of all the transactions that contain X (see equation 1).

$$Confidence(X \text{ -> } Y = \frac{Support(X \wedge Y)}{Support(X)}) \qquad \text{(Equation 1)}$$

For example, if {bread, fish, milk} has a support of 0.15 and {bread,fish} also has a support of 0.15, the confidence of rule {bread,fish} -> {milk} is 1, which means 100% of the time a customer buys bread and fish, milk is bought as well. The rule is therefore correct for 100 % of the transactions containing bread and fish. A relationship may be thought as interesting when the algorithm identifies the relationship with a measure of confidence greater than or equal to predefined threshold. This predefined threshold is called the minimum confidence. A higher confidence indicates that the rule (X -->Y) is more interesting or more trustworthy, based on the sample dataset.

Even though confidence can identify the interesting rules from all the candidate rules, it comes with a problem. Given rule in the form of X-->Y, confidence considers only the antecedents (X) and the occurrence of X and Y. It does not take the consequent (Y) into consideration. Therefore, confidence can no tell if a rule contains true implication of the relationship or if the rule is purely coincidental.

- **Lift**

Lift measures how many times more often X and Y occur together than expected if they are statistically independent of each other. This approach is designed to address the issue related to the implication of the rule using confidence. Lift is a measure of how X and Y are really related rather than coincidentally happening together.

$$Lift(X \text{ -> } Y = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)}) \qquad \text{(Equation 2)}$$

Lift is 1 if X and Y are statistically independent of each other. In contrast, a lift of X-->Y greater than 1 indicates that there is some usefulness to the rule. A larger value of the lift suggests a greater strength of the associations between X and Y.

Assuming 2.000 transactions with {milk,fish} appearing in 600 of them , {milk} appearing in 1000, and {fish} appearing in 800, then Lift (milk → fish) = 0.3/(0.5*0.4) = 1.5. If {bread} appears in 800 transactions and {milk,bread} appears in 800, then Lift(milk → bread) = 0.4/(0.5*0.4) = 2. Therefore, it can be concluded that milk and bread have a stronger association than milk and fish.

- **Leverage**

Leverage is similar to Lift, but instead of using a ratio, leverage uses the difference. Leverage measures the difference in the probability of X and Y appearing together in the dataset compared to what would be expected if X and Y were statistically independent of each other. Leverage is formulated as

$$Leverage(X \rightarrow Y) = Support(X \wedge Y) - Support(X) * Support(Y) \quad \text{(Equation 3)}$$

Leverage is 0 when X and Y are statistically independent of each other. If X and Y have some kind of relationship, the leverage would be greater than zero. A larger leverage value indicates a stronger relationship between X and Y. For example, the Leverage (milk → Fish) = 0.3 - (0.5*0.4) = 0.1 and Leverage (milk → bread) = 0.4 - (0.5*0.4) = 0.2. It indicates that milk and bread have stronger association than milk and fish.

- **Conviction**

Conviction is another interest metric that also measures the independence of X and Y but goes a little bit further. Conviction was developed as an alternative to confidence which was found to not capture direction of associations adequately. It compares the probability that X appears without Y if they were dependent on the actual frequency of the appearance of X without Y.

In that respect it is similar to lift , however, in contrast to lift it is a directed measure since it also uses the information of the absence of the consequent. An interesting fact is that conviction is monotone in confidence and lift. Contrarily to lift, conviction is sensitive to rule direction (Lift (X → Y) = Lift(Y → X)). Equation 4 shows the formula of conviction.

$$Conviction(X \rightarrow Y) = \frac{1 - Support(Y)}{1 - Confidence(X \rightarrow Y)}) \quad \text{(Equation 4)}$$

Where $\overline{Y}$ is the event that Y does not appear in a transaction. If the value of conviction is 1, it indicates that the items are unrelated.

References

[1] https://www.techopedia.com/definition/30306/association-rule-mining accessed on 16-01-2018
[2] http://searchbusinessanalytics.techtarget.com/definition/association-rules-in-data-mining accessed on 16-01-2018