Homework 2: Discovery of Frequent Itemsets and Association Rules

Group 11:

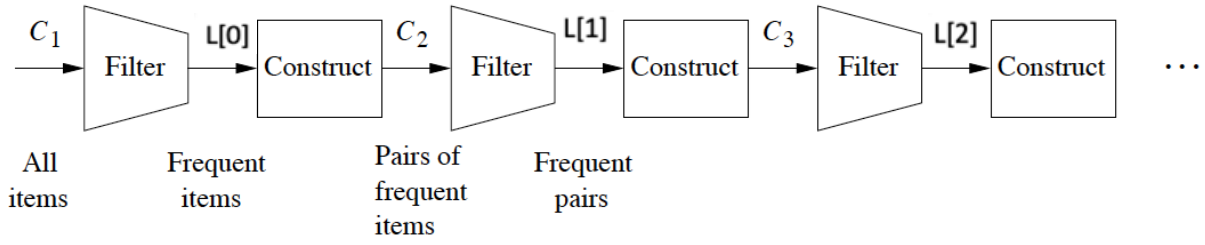
Boyu Xue (boyux@kth.se), Kristupas Bajarunas(kristupasbajarunas@gmail.com)

Introduction

We have a sale transaction dataset (T10I4D100K.dat) which includes 100,000 transactions. The objective of this report is to discover itemsets which are bought together more frequently and to find the association rules between the frequent itemsets by implementing A-priori algorithm in Python. The first part is a filter process that can filter and remove the non-frequent itemsets with a frequency lower than 1,000 (support <1,000), and finally return the truly frequent k-itemsets. The second part is generating association rules and finding the rules which have high confidence (> 0.5).

Part 1. Finding the frequent k-itemsets

The working flowchart of this part is shown in the following picture:



At first, we import the file of “T10I4D100K.dat” and return all the data. Then we use "frozenset" function to create the keys in the dictionary of candidate frequent singletons (1-itemset) named “C1” in the form of “key1 :value1, key2 :value2, …”. In C1, the values of a key is the number of the transactions which include this key, e.g. if key “25” appeared in the first and third transactions, we will have “ frozenset({25}) :{0,2}” .

Secondly, we created a “filter_remove” function which removes the non-frequent items with a frequency lower than the support threshold from the candidate frequent singleton list C1. Then we put the truly frequent singletons into a dictionary L, as the first set L[0]. Since there are many itemsets (100,000) in this database, we use 1% as the support degree, which gives a support threshold of 1,000. We use k= 5 as the maximum number of items in a candidate itemset, which means that we only concern about the itemsets that include 1-5 items in each itemset.

Thirdly, we created a filter loop. In each loop,

1) Create a new k-itemset by combining a (k-1)-itemset with a 1-itemset (e.g. pair= singleton + singleton, triple=pair + singleton);
2) For each new k-itemset, get the values of (k-1)-itemset and 1-itemset, and calculate the intersection of the two values. Then we can get the new value for the new k-itemset, e.g., if we use "frozenset({25}) :{0,2}" and "frozenset({52}) :{0,3}" to create a new 2-itemset "frozenset({25, 52}" , the value of "frozenset({25, 52})" will be {0,2}{0,3} ={0}, then we get "frozenset({25, 52}: {0}" . If the intersection of the two values= , e.g. "frozenset({a,b}: ", this new k-itemset will be neglected.
3) Delete the non-frequent items with a frequency lower than the support threshold(support <1,000) from the candidate frequent k-itemsets list C_k(k=2, 3, 4, 5) using the function "filter_remove";
4) Put the frequent k-itemsets into list L, into the L[k-1] position.

As a result, the number of frequent 1-itemsets with support 1000.0 = 375; the number of frequent 2-itemsets with support 1000.0 = 9; the number of frequent 3-itemsets with support 1000.0 = 1. The results of the frequent k-itemsets list are shown in the following table:

	Frequent itemsets	Support (frequency)
1- itemset	({966})	3921
	({978})	1141
	({104})	1158
	({143})	1417
	({569})	2835
	({620})	2100
	({798})	3103
	({185})	1529
	({214})	1893
	({350})	3069
2-itemsets	({368, 682})	1193
	({368, 829})	1194
	({825, 39})	1187
	({704, 825})	1102
	({704, 39})	1107
	({227, 390})	1049
	({722, 390})	1042
	({217, 346})	1336
	({829, 789})	1194
3-itemsets	({704, 825, 39})	1035

Part 2. Discover association rules with a certain confidence

The association rules I ➡j means that 2 or more items(I (i_1,i_2,...I_k) and j) usually being included in the same basket. We generated new sets (genset) based on the length of the input frequent itemset that is being split up to make association rules. For example, for the 2-itemset {25,52}, we can split this into three sets, namely the set itself and 2 sets of one item each. For the association rules I ➡j, the new set (genset) corresponds to "I", and the input frequent k-itemsets correspond to I∗ j.

So we calculate the confidence of a rule by using

$$Confidence = \frac{Support\ of\ \ input\ frequent\ k-itemsets}{Support\ of\ new\ set\ (genset)}$$

The association rules which have high confidence threshold(>0.5) are shown in the following table:

Association rules	Confidence
[704] ➡ [825]	0.6142697881828316
[704] ➡ [39]	0.617056856187291
[227] ➡ [390]	0.577007700770077
[704] ➡ [825, 39]	0.5769230769230769
[704, 825] ➡ [39]	0.9392014519056261
[704, 39] ➡ [825]	0.9349593495934959
[825, 39] ➡ [704]	0.8719460825610783

Conclusion:

It is very important for a supermarket to know people who bought items {a, b, c, …} will also tend to buy other items {d, e,…}. And a sale transaction dataset always includes too much data to be calculated, so we need to implement some smart algorithm to find the association rules I ➡j. The A-priori algorithm is based on the monotonicity of support: if a set of items appears x times in a database of itemsets, every subset will appear at least x times. By implementing the 2 steps: 1)finding frequent k-itemsets and 2)discovering association rules with a certain confidence, we found that there are seven association rules in the database "T10I4D100K.dat". So we know people who bought items (e.g., {[704]},will also tend to buy some other items (e.g., {[825]} in a certain probability (e.g., 0.614). But further analysis is need to implement because a rules I ➡j has a high confidence maybe just because the item j is just purchased very often, but not has a strong correlation with items I.