ID2222 Data Mining Homework 2

Frequent Itemsets and Association Rules

Kevin Dalla Torre
kevindt@kth.se

Luís Santos
lmpss@kth.se

November 2020

1 Fist subproblem

We solved this problem by using a class Apriori that has 4 methods.

1.1 algorithm

The method called algorithm is the apriori algorithm itself. The only parameter it contains is *verbose* which if True prints the information we consider useful for presentation. First the method reads the data and stores the items as integers in the matrix *basket_list* and at the same time counts the items using the method first_pass. After that the method filters out all the items which do not have at least support s. The support s is a variable that we pass on initialization for our class Apriori. This will be L1. Then starting with k=1 and while the size of the L_k is not 0 it increases k by 1 and repeat the rest of the algorithm. If there is no frequent itemset for a certain size k, there will be no frequent itemset for larger k which is why we stop here. The next step is to create all the candidates C_k of size k. This step uses a method apriori_gen (see below). It then iterates over all the transactions (baskets). So, for every transaction t in basket_list it takes all the itemsets of length k which appear on the candidates C_k and adds a count of 1 to all of these possible candidates. It finds the candidates contained in t by using the method subset (see below). The last step of the while loop in the method algorithm is to check all of the candidates which had a count of at least s (the minimum support) and filter out the ones that did not have a count $\geq s$. This quantity will be L_k. As mentioned above the while loop is then repeated until the length of L_k is 0 and then it will return a dictionary with the k values used by the while loop as keys and the corresponding L_k:s as values. This will contain all the frequent item sets for the given data file and minimum-support.

1.2 first_pass

The method first_pass take an item as input. It adds 1 to the count for that item and then returns the item itself.

1.3 apriori_gen

This method has the parameters L (which is L_(k-1)) and k which is the set size of the candidates that we want. What this method achieves is to join L_(k-1) with L_k in the following way. For every combination of 2 members in L_(k-1) we check if all items except the last item are the same and if the last item of the first member is smaller than the last item of the second member. This last condition is to ensure that there will not be duplicates. If the members satisfy the conditions, it creates the candidates by adding the last item of the second member to the whole set of the first member. This will give us candidates with size k. Then for every candidate it checks if all its subsets with size k-1 are in L_(K-1). If all the subsets are in L_(k-1) it keeps the candidate otherwise it removes the candidate. Then return the candidates C_k.

1.4 subset

This method has the parameters Ck(candidates), t (the transaction) and k (set size of the candidates). It looks at all the possible combinations of set size k in t. It adds the combinations that are in C_k to a list that it returns.

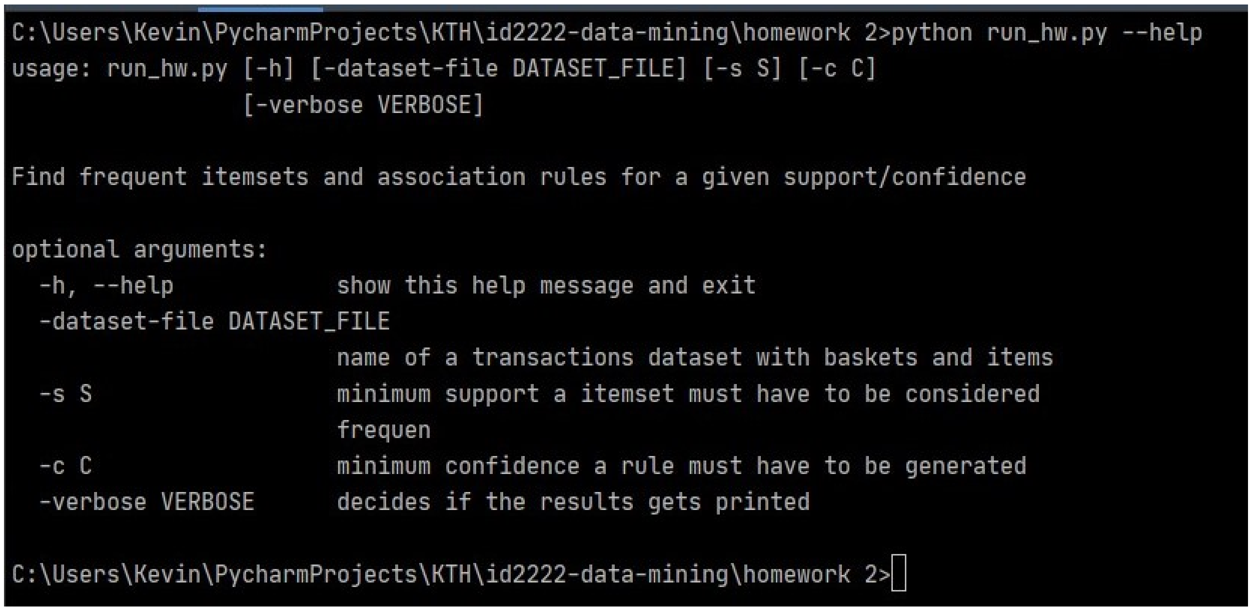
2 Second Sub problem

This second step generates association rules with confidence of at least c from the itemsets found in the first step. We have a class called AssociationRules with one single method find. The method find has 4 parameters which are

L, c, *verbose* and *option*. L is the dictionary of frequent itemsets found in step 1 and c is the minimum confidence (threshold) that we want the rules to have. If *verbose* is set it will print out useful information for the presentation. The parameter option lets you choose between two solutions: option 1 is the real solution to this assignment (the one we submit) and option 2 is just our own experiment outside of the assignment. The task was to find association rules $X \rightarrow Y$ and in option one we are only considering the cases where size(Y)=1. In option 2 we experiment and try to find the solution for size(Y)$\geq$ 1 (of course size(X)+size(Y) will be the same size as option 1). Here we will only explain option 1 since that is the relevant task for this assignment. For a frequent itemset (X+Y) (itemset of size k) and a subset X (itemset of size k-1) of (X+Y) we divide the count of X+Y by the count of X. If the fraction of (X+Y)/X is at least c then we add the rule $X \rightarrow Y$ (Y is the item in the itemset (X+Y) that was not in the itemset X) to a list of rules. For a frequent itemset(X+Y) this is repeated for every subset X. We apply this procedure for every frequent itemset found in step 1 that has the size of at least 2.

3 How to run

Our homework solution can be run via the command `python run_hw.py`. Check Figure 1 for the command-line parameters, as well as a short description of the functionality of each parameter. Check Figure 2 to observe an example run, as well as to obtain the default values of the parameters. We used the sales transaction dataset as provided to us in the homework statement. To run our Python script, call the script `run_hw.py` with the optional command line arguments.



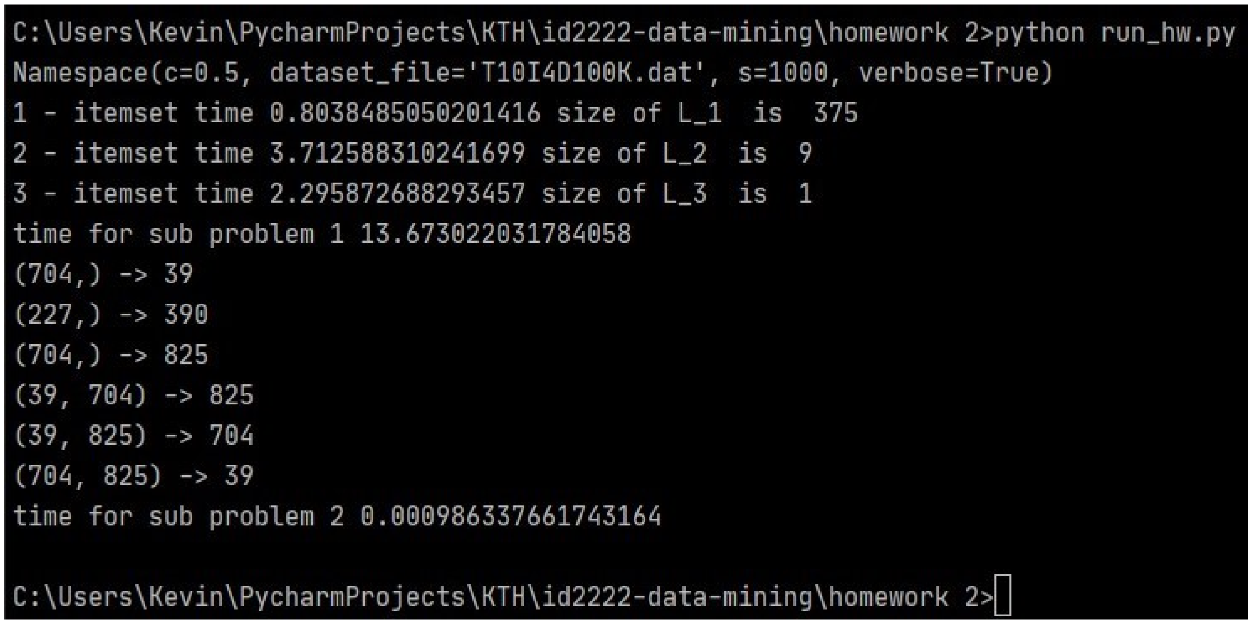
```
C:\Users\Kevin\PycharmProjects\KTH\id2222-data-mining\homework 2>python run_hw.py --help
usage: run_hw.py [-h] [-dataset-file DATASET_FILE] [-s S] [-c C]
                 [-verbose VERBOSE]

Find frequent itemsets and association rules for a given support/confidence

optional arguments:
  -h, --help            show this help message and exit
  -dataset-file DATASET_FILE
                        name of a transactions dataset with baskets and items
  -s S                  minimum support a itemset must have to be considered
                        frequen
  -c C                  minimum confidence a rule must have to be generated
  -verbose VERBOSE      decides if the results gets printed

C:\Users\Kevin\PycharmProjects\KTH\id2222-data-mining\homework 2>
```

Figure 1: Command-line parameters of our solution.



```
C:\Users\Kevin\PycharmProjects\KTH\id2222-data-mining\homework 2>python run_hw.py
Namespace(c=0.5, dataset_file='T10I4D100K.dat', s=1000, verbose=True)
1 - itemset time 0.8038485050201416 size of L_1  is  375
2 - itemset time 3.712588310241699 size of L_2  is  9
3 - itemset time 2.295872688293457 size of L_3  is  1
time for sub problem 1 13.673022031784058
(704,) -> 39
(227,) -> 390
(704,) -> 825
(39, 704) -> 825
(39, 825) -> 704
(704, 825) -> 39
time for sub problem 2 0.000986337661743164

C:\Users\Kevin\PycharmProjects\KTH\id2222-data-mining\homework 2>
```

Figure 2: Default values of command-line parameters for an example run.