CS37300 Homework 5

Due date: Wednesday December 4, 11:59pm.

In this programming assignment you will use python to implement an association rule algorithm and apply it to it on the *Yelp* dataset. Use the `yelp5.csv` dataset, which is a subset of the *Yelp* data from previous homeworks with only discrete attributes and no missing values. Instructions below detail how to turn in your code and assignment on data.cs.purdue.edu.

Note: You can use any standard python libraries available on the CS computers (e.g., numpy) but don't use methods from machine learning libraries (e.g., scikit-learn).

Dataset Details

Use the `yelp5.csv` dataset that consists of $5,432$ rows and 11 discrete attributes.

Algorithm Details

Implement the Apriori algorithm and apply it to the data. Use all the 11 discrete attributes and consider patterns involving all the unique attribute values.

Code specification

Your python script should take three arguments as input.

1. *trainingDataFilename*: corresponds to a subset of the Yelp data (in the same format as `yelp5.csv`) that should be used as the *training set* for your algorithm.
2. *minsup*: the value of the *minsup* threshold to use in the algorithm.
3. *minconf*: the value of the $minconf$ threshold to use in the algorithm.

Your code should read in the training sets from the csv file, discover the association rules with the provided thresholds and then output the number of discovered frequent itemsets and association rules for each pattern size (starting with size of 2).

Name your file association-rules.py. Then the input and output should look like this:

```
$ python association-rules.py train-set.csv 0.25 0.75
...
FREQUENT-ITEMS 2 60
FREQUENT-ITEMS 3 123
...
ASSOCIATION-RULES 2 66
ASSOCIATION-RULES 3 249
...
```

Assignment

1. (5 pts) The *Apriori* algorithm uses a generate-and-count strategy for deriving frequent itemsets. Candidate itemsets of size $k+1$ are created by joining a pair of frequent itemsets of size k. A candidate is discarded if any one of its subsets is found to be infrequent during the pruning step. Suppose the algorithm is applied to data set below with $minsup = 30\%$:

Trans ID	*Items*
1	{a,b,d,e}
2	{b,c,d}
3	{a,b,d,e}
4	{a,c,d,e}
5	{b,c,d,e}
6	{b,d,e}
7	{c,d}
8	{a,b,c}
9	{a,d,e}
10	{b,d}

   (a) Draw an itemset lattice representing the data above (with $\{\emptyset\}$ at the bottom and at $\{a, b, c, d, e\}$ at the top). Label each node of the lattice with the following letter(s):
      (i) **N**: If the itemset is not considered to be a candidate itemset by the *Apriori* algorithm. Note that there are two reasons for an itemset to not be considered as a candidate: (1) it is not generated at all during the candidate generation step, or (2) it is generated during the candidate generation step but is subsequently removed during the candidate pruning step because one of its subsets is found to be infrequent.
      (ii) **F**: If the candidate is found to be frequent by the *Apriori* algorithm.
      (iii) **I**: If the is found to be infrequent after support counting.

   (b) What is the percentage of frequent itemsets (with respect to all itemsets in the lattice)?

   (c) What is the pruning ratio of the *Apriori* algorithm on this data set? (The pruning ratio is defined as the percentage of all itemsets that are not considered to be candidates, either because they are not generated or because they are pruned before support is measured.)

   (d) What is the false alarm rate (i.e., percentage of candidate itemsets that are found to be infrequent after performing support counting?)

2. (5 pts) Consider the `yelp5.csv` data. In your programming assignment, you will construct itemsets based on the unique attribute values in each of the 11 discrete attributes. Given this:

   (i) How many frequent itemsets are possible?

   (ii) How many association rules are possible?

   (iii) Describe how the support and confidence thresholds will help to prune this space.

(iv) Which threshold will have a larger impact on the efficiency of the association rule algorithm?

3. (20 pts) Implement the Apriori association rule algorithm You only need to consider rules with a single variable in the consequent. Track how many frequent itemsets and association rules are discovered during the algorithm's search for patterns.

4. (10 pts) Apply the Apriori association rule algorithm to the `yelp5.csv` data. Use cutoff thresholds of $minsup = 25\%$ and $minconf = 75\%$.

   (i) List the 20 discovered association rules with `goodForGroups=1` or `goodForGroups=0` as a consequent, and largest support values, to characterize the data. Report the rules themselves in an interpretable form (e.g., refer to the original attribute name and value), along with their numerical scores (i.e., support and confidence).

   (ii) Given your prior knowledge of the patterns in this data from previous homeworks, discuss whether any of the discovered rules are *interesting*. Why or why not?

5. (10 pts) Evaluate the efficiency of the algorithm.

   (i) Report how many frequent itemsets ($|I|$) and association rules ($|R|$) are discovered when the algorithm is run on the full `yelp5.csv` data with cutoff thresholds of $minsup = 25\%$ and $minconf = 75\%$.

   (ii) Now keep $minconf = 75\%$ and rerun the algorithm while varying $minsup = [10\%, 30\%, 50\%]$. Report the values of $|I|, |R|$ for each of the different $minsup$ values.

   (iii) Now keep $minsup = 25\%$ and rerun the algorithm while varying $minconf = [40\%, 60\%, 80\%]$. Report the values of $|I|, |R|$ for each of the different $minconf$ values.

Submission Instructions:

For the coding implementations, submit your code as a Python program (Python 3.6). For all other parts write your answers in a single PDF. Name this yourusername_hw5.pdf.

As usual, label the plots with the question number. Your homework must contain your name and Purdue ID at the start of the file. If you omit your name or the question numbers for the plots, you will be penalized.

To submit your assignment, log into data.cs.purdue.edu (physically go to the lab or use ssh remotely) and follow these steps:

1. Make a directory named yourusername_hw5 (all letters in lower case) and copy your PDF file and Python file inside it. To do it remotely, use:
   `scp ./path/to/your-file.pdf your-id@data.cs.purdue.edu:./remote/path/from-home-dir/`

2. Go to the directory containing yourusername_hw5 (e.g., if the files are in `/homes/dan/danhw5`, go to `/homes/dan`), and execute the following command:
   `turnin -c cs373 -p hw5 yourusername_hw5`
   (e.g. Dan would use: `turnin -c cs373 -p hw5 dan_hw5` to submit his work)

3. To overwrite an old submission, simply execute this command again.

4. To verify the contents of your submission, execute the following command:
   `turnin -v -c cs373 -p hw5`
   Do not forget the `-v` option, else your submission will be overwritten with an empty submission.

   Note: Make sure that you test your code on data.cs.purdue.edu before submitting.