Project 1

Mining Frequent $k+$ Itemsets

Data Mining

Instructor: Arjun Mukherjee

1. Pattern Mining

We learnt about Association rule and pattern mining. Refer to slides/lecture notes given in class and algorithm details in [Liu, 2007]. In this project we will be implementing a pattern mining algorithm based on the Apriori Algorithm (Figure 2.2, 2.3 in [Liu, 2007]).

2. Dataset

Our transaction database is a set of reviewers from Amazon.com. Specifically, reviewer ids are our items. The transaction is a set of reviewer ids. Specifically, all reviewer ids which were used to post a review on that product. The format of the transaction database follows the standard format seen in class. Each line represents a transaction. For a given transaction, the items (reviewer ids) are separated by a space character. The dataset is available at:
http://www2.cs.uh.edu/~arjun/courses/dm/hw_proj/proj_1.zip .You are required to find frequent patterns in this dataset.

Pre-processing: For pattern mining, one usually requires frequent itemset candidate generation and that is done based on the total ordering of elements of previous lower order frequent itemsets (e.g., candidate-gen function in Figure 2.3 of [Liu, 2007]). Since comparing strings is more expensive than integers, you will be at an advantage if you map all unique reviewer ids to intergers and convert the transaction database above to set of transaction with integers as items.

3. Implementation

You are required to mine $k+$ frequent itemsets given a minimum support count (min_sup). Essentially, you are required to implement the Algorithms in Figure 2.2, 2.3 in [Liu, 2007] with slight changes in the format. A frequent k itemset refer to an itemset whose size is k (i.e., it has k elements) and the support of that itemset (no. of times that itemset appears in the transaction database) exceeds min_sup. Frequent $k+$ itemsets refers to all frequent itemsets (i.e., itemsets appearing more than min_sup times in the data) which have sizes greater than k.

Helpful Hints:
Pay special care in utilizing the downward closure property and other algorithmic tricks detailed in the class and the book. Remember you are not required to generate all frequent itemsets/patterns. You are only required to retrieve frequent itemsets which have k or more items in them and k is specified. Don't attempt to set the min_sup to $<= 2$ for the transaction database in Section 1. There will be profusely many itemsets/patterns than your machine could handle! Try smaller test datasets to make sure your implementation is correct.

4. Examples and sample Input/Output

We provide several examples based on the transaction database give in Section 1. For a given value of min_sup and k, your program should retrieve/output all itemsets that are frequent with sizes greater than k. Write each itemset as space delimited in a new line. Further, beside each itemset, you should report its support (frequency) in parenthesis as they appear in the following examples.

i. $min\_sup = 4,\ k = 3$

This would yield all itemsets appearing at least 4 times and containing at least 3 elements. Some of results of this query would include the following itemsets:

```
A37787I8C184FW AWE8HU0AZKASV A3UIATN5XW74NQ (4)
A3Y9BX5AS769T AWE8HU0AZKASV A3UIATN5XW74NQ (5)
AZ7I5GAJZA3JO A28R83ADQPMF2X A2GKW94L6HRND7 A2IE7YPWUYZAXS (4)
```

The first itemset is a frequent 3 itemset having a support count of 4 (i.e., appears 4 times in the data, satisfies the $min\_sup = 4$ and hence frequent). The second itemset is a frequent 3 itemset with a support of 5 (i.e., it appears 5 times in the transaction database). The last itemset is a frequent 4 itemset with a support of 4. Notice that itemsets appearing less than min_sup=4 times

should not be retrieved for the above query. There are 53 frequent $3+$ itemsets with min_sup=4. The above three are only a few of the total 53 frequent $3+$ itemsets retrieved for $min\_sup = 4,\ k = 3$.

ii. $min\_sup = 5,\ k = 3$

This would yield all itemsets appearing at least 5 times (i.e., 5 or more) and containing at least 3 elements. We have exactly 4 such itemsets. The complete result set for the above query is given as follows:

```
A9TJYY7P2R280 A2S9IDC1IZH7WN AYFQ8ML2PYZ1D (5)
A3PXX92YUMGMBG A3UIATN5XW74NQ AWE8HU0AZKASV (5)
A2J96R1J6MDMEV A3UIATN5XW74NQ AWE8HU0AZKASV (5)
A3Y9BX5AS769T A3UIATN5XW74NQ AWE8HU0AZKASV (5)
```

iii. $min\_sup = 8,\ k = 2$

This would return the following frequent $2+$ itemsets

```
A20JYIHL1W1U54 A7Y6AVS576M03 (10)
AO2V6YDDYQ2CR A5JLAU2ARJ0BO (8)
A2AEZQ3DGBBLPR A231WM2Z2JL0U3 (8)
A231WM2Z2JL0U3 A5JLAU2ARJ0BO (11)
```

Notice that there are no frequent 3 itemsets with min_sup = 8 in our data. And since there are no frequent 3 itemsets, we don't have any frequent itemsets with larger sizes (based on the downward closure property), thus the query $min\_sup = 8,\ k = 3$ would yield no itemsets. Similarly, the following queries too yields no frequent itemsets:

$min\_sup = 4, \quad k = 5$
$min\_sup = 6, \quad k = 3$

iv. $min\_sup = 7,\ k = 2$

This must have what we found in the query (iii) above (due to the downward closure property) and also retrieves some other itemsets appearing 7 times. The complete result set for this query is given below:

```
A20JYIHL1W1U54 A7Y6AVS576M03 (10)
AO2V6YDDYQ2CR A5JLAU2ARJ0BO (8)
A39KIFR965H0CX A1MJMYLRTZ76ZX (7)
A1C9C1QOQB94RT A15S4XW3CRISZ5 (7)
A1E7QLJVWNFOZY A2ZM9BGE3K3SY2 (7)
A2AEZQ3DGBBLPR A231WM2Z2JL0U3 (8)
A3PPXFIVMJ9MV8 AUPUZ14778SD7 (7)
A231WM2Z2JL0U3 A5JLAU2ARJ0BO (11)
```

v. $min\_sup = 10,\ k = 2$

This is a subset of query (iii) as we have increased the min_sup. The result is given below:

```
A20JYIHL1W1U54 A7Y6AVS576M03 (10)
A231WM2Z2JL0U3 A5JLAU2ARJ0BO (11)
```

vi. $min\_sup = 75,\ k = 1$

This yield itemsets having at least 1 element and appearing at least 75 time sin the data. It turns out we only have 9 frequent 1-itemsets as follows:

```
A2A10ZSC2RH4RG (629)
A25HBO5V8S8SEA (159)
AT6CZDCP4TRGA (87)
A2CL818RN52NWN (81)
A2B7BUH8834Y6M (147)
A2AEZQ3DGBBLPR (91)
A2ZM9BGE3K3SY2 (92)
A231WM2Z2JL0U3 (209)
A5JLAU2ARJ0BO (323)
```

5. Full Output and Demo

You will be evaluated on two aspects in this homework when you demo it. In the actual demo new values of min_sup and k will be given and you are required to run your implementation and produce the result, i.e, generate all frequent $k+$ itemsets appearing at least min_sup number of times in the data. Make sure to provide a README file with your implementation which details on how to run your code with the specified inputs. You are also required to make sure that your results are generated in the same format as the examples provided in the project codebase. More specifically, your code should take command line arguments as follows:

$> project_executable min_sup k input_transaction_file_path output_file_path

If you cannot code via command line arguments, at least ask the parameters (min_sup, k, input_transaction_file_path, output_file_path) via console input. Also you should NOT print the results in console. The output should be present in a flat text (using the filename given in the output_file_path parameter) bearing the structure: each line is a frequent item-set delimited by space with the support count in brackets. These guidelines are hard and no alterations will be possible as output will be evaluated using automated scripts. **If your output does not abide by the instructions, you risk your score as our automated checking will not be able to evaluate your results.**