Apriori algorithm

~ Implement the apriori algorithm for finding association rules ~

Professor: Sang-Wook Kim
Department of Computer Science
2012003716 Kwangil Cho
2017. 03. 30

0. Introduction

Apriori is an **algorithm for frequent item set mining and association rule learning over transactional databases**. It proceeds by **identifying the frequent individual items** in the database and **extending them to larger and larger item sets** as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database

1. Objectives and Environment

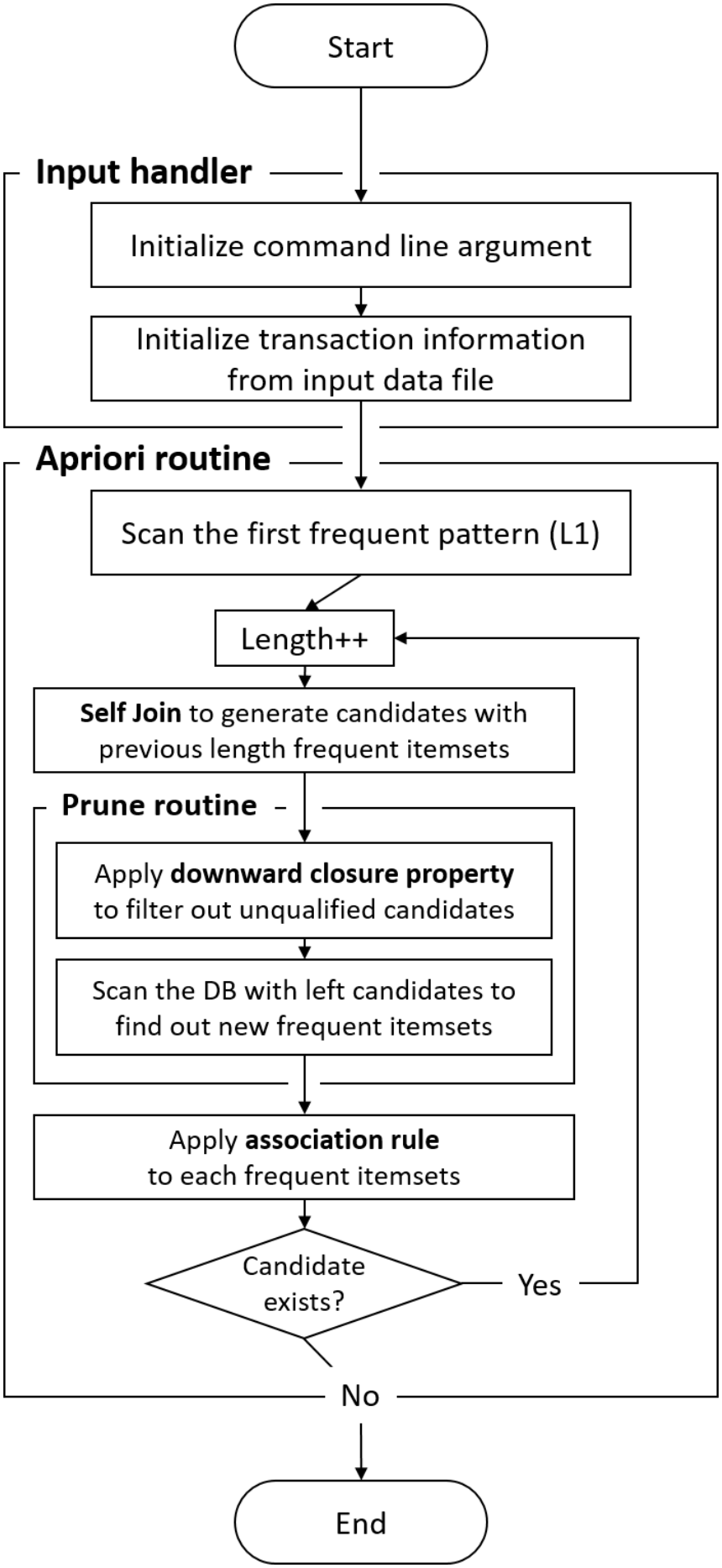
The goal of this project is to find frequent patterns by applying apriori algorithm with given DB transaction data and minimum support and apply association rule to frequent patterns.

I have been working on the following environment:

- Operating System: Ubuntu 14.05 LTS 64-bit
- Compilation: g++ compiler
- Language: C/C++ with C++ standard 11
- Source code editor: Vim
- Source code version control: Git
- Followed HYU coding convention faithfully

2. Program Structures

The flowchart of the program is as follows:



(0) The program expects 4 arguments(executable file, minimum support, input data file, output data file) for executing apriori algorithm. When the arguments are successfully ready, input handling routine works. First, input handler sets minimum support that has been entered as global variable. Because it is the probability, data type should be double type. Second, handler opens input and output files.

(1) When the environment settings are done, handler reads all of transaction information line by line from the input data file. At this time, transaction data which is a set of integer is made and entered into a global list. A minimum support count can be derived from minimum support and whole data size (transaction size). The input handling procedure is done.

(2) Before executing an apriori algorithm iterative routine, the first frequent pattern should be mined. 1L(length) frequent patterns are found at this step. Scanning DB and counting the support of item generates 1L frequent patterns with all different kind of items.

(3) Each time the program performs the apriori algorithm routine, the length variable is incremented by one.

(4) To find out next length frequent patterns, candidates which are the ingredients of frequent pattern should be generated. By selfjoining which is implemented by combination of whole itemsets the program can get unrefined candidates.

(5) Candidates generated in selfjoining step are not sufficient to be the frequent pattern. By pruning with downward closure property, meaningless candidates are removed. Downward closure property guarantees that if the subsets of candidate with previous length are not frequent, the candidate is not frequent pattern.

(6) After filtering out the dirty candidates by downward closure, scan DB and figure out the support of 'real' candidates. Therefore, the program gets new frequent patterns with increased length.

(7) Apply association rule to each frequent pattern level by level(length by length). Make all of the subsets excluding the empty set and the identical set and figure out each corresponding associative set. Measures support and confidence by scanning DB from the perspective of conditional probability. Print out each association rule.

(8) Continue the iteration from (3) to (7) while there exists the candidate to be mind.

(9) If there is no candidate to be dealt with terminates the program successfully.

3. Execution

```
ki@ubuntu:~/DataMining/assignment1$ make
g++ -g -Wall -O3 -std=c++11 -I./include/ -o ./bin/apriori src/apriori.cc  -L./lib/ -lpthread
ki@ubuntu:~/DataMining/assignment1$ ./bin/apriori 4 input.txt output.txt
ki@ubuntu:~/DataMining/assignment1$
```

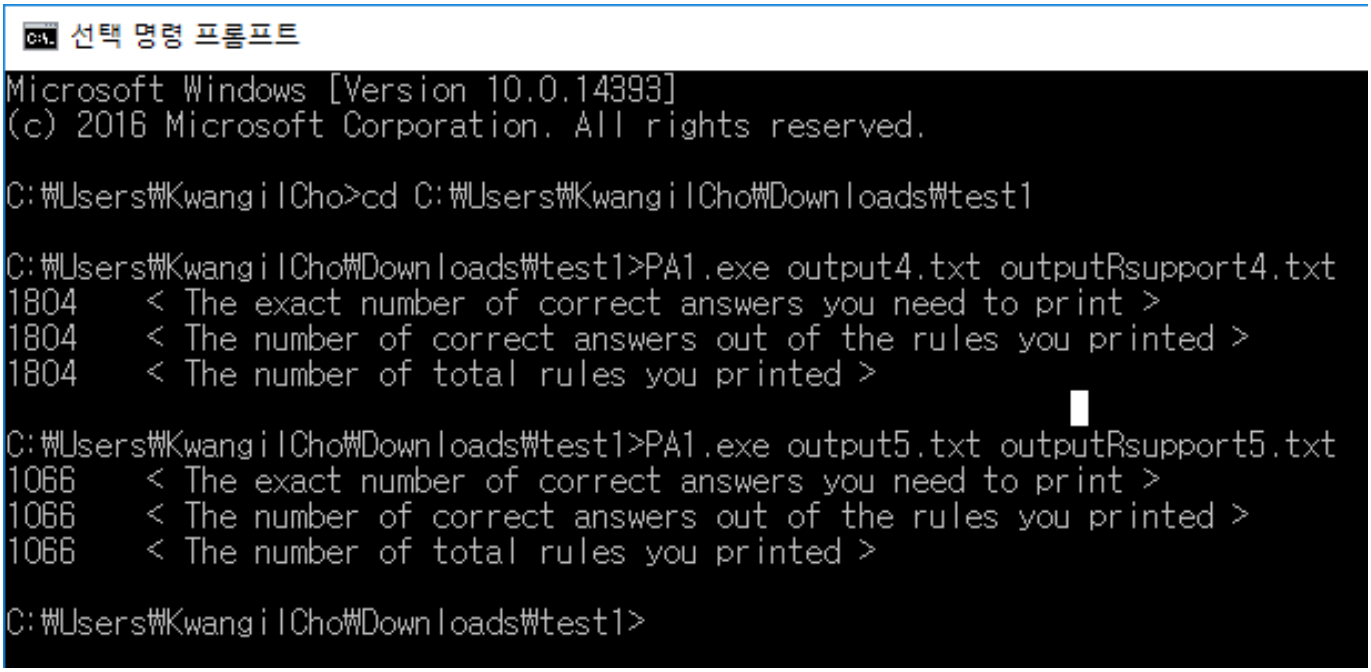
```
1781 {3,16}    {8,19}    5.60      22.22
1782 {8,16}    {3,19}    5.60      18.54
1783 {3,8,16}  {19}      5.60      23.33
1784 {19}      {3,8,16}  5.60      23.33
1785 {3,19}    {8,16}    5.60      77.78
1786 {8,19}    {3,16}    5.60      42.42
1787 {3,8,19}  {16}      5.60      90.32
1788 {16,19}   {3,8}     5.60      47.46
1789 {3,16,19} {8}       5.60      100.00
1790 {8,16,19} {3}       5.60      66.67
1791 {5}       {8,15,16} 4.00      15.87
1792 {8}       {5,15,16} 4.00      8.85
1793 {5,8}     {15,16}   4.00      31.75
1794 {15}      {5,8,16}  4.00      14.29
1795 {5,15}    {8,16}    4.00      43.48
1796 {8,15}    {5,16}    4.00      32.26
1797 {5,8,15}  {16}      4.00      80.00
1798 {16}      {5,8,15}  4.00      9.43
1799 {5,16}    {8,15}    4.00      32.79
1800 {8,16}    {5,15}    4.00      13.25
1801 {5,8,16}  {15}      4.00      43.48
1802 {15,16}   {5,8}     4.00      28.99
1803 {5,15,16} {8}       4.00      68.97
1804 {8,15,16} {5}       4.00      46.51
 NORMAL >> output.txt                          < text < utf-8[unix] < 7,216 words < 100% :1804/1804 :  1
```

(1) Build and execute the program with 4% of minimum support.

```
ki@ubuntu:~/DataMining/assignment1$ ./bin/apriori 5 input.txt output.txt
ki@ubuntu:~/DataMining/assignment1$
```

```
1052 {8,16,18} {3}       8.00      83.33
1053 {3}       {8,16,19} 5.60      18.67
1054 {8}       {3,16,19} 5.60      12.39
1055 {3,8}     {16,19}   5.60      21.71
1056 {16}      {3,8,19}  5.60      13.21
1057 {3,16}    {8,19}    5.60      22.22
1058 {8,16}    {3,19}    5.60      18.54
1059 {3,8,16}  {19}      5.60      23.33
1060 {19}      {3,8,16}  5.60      23.33
1061 {3,19}    {8,16}    5.60      77.78
1062 {8,19}    {3,16}    5.60      42.42
1063 {3,8,19}  {16}      5.60      90.32
1064 {16,19}   {3,8}     5.60      47.46
1065 {3,16,19} {8}       5.60      100.00
1066 {8,16,19} {3}       5.60      66.67
 NORMAL >> output.txt                          < text < utf-8[unix] < 4,264 words < 100% :1066/1066 :  1
```

(2) Build and execute the program with 5% of minimum support.



(3) Test with TA's testing program with output4 and output5 files. Successfully passes the test.

4. Epilogue

Through this project, I was able to learn how to find frequent patterns that are the basis of data mining. The process of applying the apriori algorithm and getting the right result was a lot worrying than thought. Thanks to professor and assistant who prepared for class and practices.

5. Reference

- Wikipedia: Apriori algorithm (https://en.wikipedia.org/wiki/Apriori_algorithm)