Data Mining

Lab Assignment #1

Goal

Practice the use and implementation of data mining methods to **discover the frequent itemsets** in varied-scale synthetic datasets.

Steps

I. Use a data generator to generate synthetic datasets.

1. Download the synthetic data generator (IBMGenerator) from IBM Almaden Quest Project.
   - Link: https://github.com/zakimjz/IBMGenerator
   - Note: the data generator is in C++.
2. Verify its correctness (through code trace and other testing).
   - Please note particularly the **Data format** and the use of **Command line options**.
3. Generate the following datasets for use later (**Three** configurations):
   - Set average transaction length =10 (items) and
     A. Set ntrans (number of trans)= **1000**, nitems (number of items) = **500**
     B. Set ntrans = **100,000**, nitems = **1000**
     C. Set ntrans = **1,000,000**, nitems = **1000**

II. Mining the datasets generated in Step I using Apriori algorithm:

1. You need to use the open-sourced Apriori program (in C++ or Python) **we provide you** to mine the datasets generated in Step I.
   - **Please download either one of the C++ or Python version Apriori source code from E3.**
   - TAs had already modified original code to print out the frequent itemset with support on the screen. Please read the following references (original source) for the use of command line options and input/output:
     - C++: https://github.com/bowbowbow/Apriori (developed by Seungwon)
     - Python: https://github.com/asaini/Apriori (developed by Asaini)
   - **Note: Once you have selected the version of the Apriori code, all the following modifications/comparisons with Apriori must be done on this same one.**
2. Next, you need to **modify the Apriori source code as obtained above** to complete the following tasks:
   - **Task 1: Mining all Frequent Itemset**
     - Your program must output 2 files (in **.txt**):

(a) Result file 1 (itemset list):

- Print out all frequent itemsets **with support (take percentage % and round to the first decimal place)**.
- Please **sort** those frequent itemsets according to support (from large to small).
- File format (each frequent itemset): [support]\t[item_set]\n

(b) Result file 2 (statistics file):

- Print out the total number of frequent itemsets.
- Print out the number of candidates generated **before and after pruning, respectively,** in each iteration of Apriori.
- File format (The index of iteration starts from 1):

  [total number of frequent itemsets]\n ➡ only one row.

  [index_of_iteration]\t[the number of candidates (**before pruning**)]\t[the number of candidates (**after pruning**)]\n

- Count the computation time for this task (task independent).

- **Task 2: Mining all <u>Frequent Closed Itemset</u>**
  - The aim for this task is to mine all **Frequent Closed Itemset** with computation as fast as possible.
  - Your program must <u>output</u> 1 file (in **.txt**):

    (a) Result file 1 (itemset list):

    - Print out the total number of frequent closed itemset.
    - Print out all frequent closed itemset **with support (take percentage % and round to the first decimal place)**.
    - Please **sort** those frequent closed itemsets according to support (from large to small).
    - File format:

      [total number of frequent closed itemsets]\n ➡ only one row.

      [support]\t[item_set]\n

  - Count and show the whole computation time for this task (task independent). **If you conduct any post-processing procedures, the time for post-processing should be counted in too.**
  - Show the **ratio** of computation time compared to that of Task 1: $\frac{Task2}{Task1} \times 100\%$

3. You need to try **different settings** of minimum support (%) in the above two tasks:
   - Configuration for dataset A: {5.0, 7.5, 10.0}
   - Configuration for datasets B and C: {2.5, 5.0, 7.5}
   - If you have time, you can also try other settings by yourself. (This part is optional, you may not get any bonus point.)

III. Conduct the mining task (**only Task 1**) in Step II using a **non-Apriori algorithm** with the goal to **get performance improvements as higher as possible**:

1. You may implement by yourself or use an open-sourced one as the base for modification.
   - Please note that clear citation is needed if you use an open-source code.
2. The algorithm should be able to handle all datasets in Step I and as efficient as possible.
3. Verify the correctness of the mined results with comparisons to Step II.
4. The mining task you need to do are same as Step II (**only need to do Task 1**).
   - Constraints (The following should be **same** as Step II):
     - Programming language
     - Running Machine/platform
     - Dataset configuration
5. The main scoring criteria in this step: the performance improvements (**the percentage of speedup:** $\frac{(Step\ \text{II} - Step\ \text{III})}{Step\ \text{II}} \times 100\%$)

Requirements

1. A Report:
   (1) Explain how to run your code in Step II and III.
   (2) Step II
   - Report on the mining algorithms/codes:
     - The modifications you made for Task 1 and Task 2
     - The restrictions (e.g., the scalability, etc)
     - Problems encountered in mining
     - Any observations/discoveries you want to share
   - Paste the screenshot of the computation time
     - Note: You need to try different settings of minimum support.
   - For Task 2, you also need to show the ratio of computation time compared to that of Task 1 in your report.
   - Paste the screenshot of your code modification for Task 1 and Task 2 with comments and explain it.
     - You need to explain it in clear and well-structured ways.
     - If you didn't explain the code, you cannot get any points in this subitem.

   (3) Step III
   - Descriptions of your mining algorithm
     - Relevant references
       - If you use an open-sourced code as the base for modification, the clear citation is needed.
     - Program flow
     - Anything you want to share
   - Differences/Improvements in your algorithm

- Explain the main differences/improvements of your algorithm compared to Apriori.
- If you use open-source code as the base, you should describe the modifications your made on the original algorithm.
- You need to explain it in clear and well-structured ways.

- Computation time
  - Compare the computation time of **Task 1** with Step II (give **the percentage of speedup:** $\frac{(Step\ \text{II} - Step\ \text{III})}{Step\ \text{II}} \times 100\%$)
  - Paste the screenshot of the computation time
  - Note: You need to try different settings of minimum support as used in Step II.
- Discuss the **scalability** of your algorithm in terms of the size of dataset (i.e., the rate of change on computing time under different data size; the largest dataset size the algorithm can handle, etc).

2. Uploads

(1) Zip your contents in one file, including
- Report (.pdf)
  - Name the file as HW1_Report.pdf
- Output files in Step II and III
  - Task 1
    - Result file 1 (itemset list): {step(2 or 3)}_task1_{dataset(A, B or C)}_{min_support}_result1.txt
    - Result file 2 (statistics file): {step(2 or 3)}_task1_{dataset(A, B or C)}_{min_support}_result2.txt
  - Task 2
    - Result file 1 (itemset list): **step2**_task2_{dataset(A, B or C)}_{min_support}_result1.txt
- Source code in Step II and III
- Datasets (.data)

(2) Name the zip file as DM_HW1_{your student id}_{your name}.zip. (e.g., DM_HW1_310XXX_王大明.zip)
- If the zip file name has format error or the report is not in pdf, there will be a punishment.

Grading

- Step II: 65 %
- Step III: 35 %

Important Date

- Deadline: 11/4 (Fri.) 23:59:59

Penalty

- Format error
  - The zip file name has format error (-5%)
  - The report is not in pdf. (-5%)
  - The output file has format error. (-10%)
- Late submission
  - If your work is submitted within one day after the deadline, a penalty of 20 percentage marks will be applied.
  - If your work is submitted within two days after the deadline, a penalty of 50 percentage

marks will be applied.

- If your work is submitted over two days after the deadline, you will get score of 0 in this homework.