Data Mining
Fall 2024

Lab Assignment #1

Goal

Practice the use and implementation of data mining methods to **discover the frequent itemset** 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 the **Data format** and the use of **Command line options (e.g., numerical unit)**.

     Ex:

     Setting a datasetA with number of transactions = 1000, transaction length = 10, number of items = 600

     ./gen lit -ntrans 1 -tlen 10 -nitems 0.6 -fname datasetA -ascii
3. Generate the following datasets for use later (**Three** configurations):
   - Set average transaction length =10 (items) and
     A. Set number of transactions = **1,000**, number of items = **600**
     B. Set number of transactions = **100,000**, number of items = **500**
     C. Set number of transactions = **500,000**, number of items = **500**
   - Except for the specified variables, all other configurations will use the model's **default settings**, requiring no further adjustments.

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

1. You need to use the open-sourced Apriori program (in Python) **we provide you** to mine the datasets generated in Step I.
   - **Please download 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:
     - https://github.com/asaini/Apriori (developed by Asaini)
2. Next, you need to **modify the Apriori source code as obtained above** to complete the

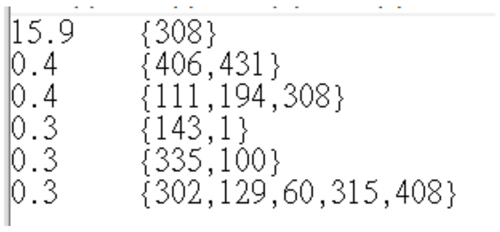
following tasks:

- **Task 1: Mining all <u>Frequent Itemset</u>**
  - ■ Your program must <u>output</u> 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

      Ex:



```
15.9    {308}
0.4     {406,431}
0.4     {111,194,308}
0.3     {143,1}
0.3     {335,100}
0.3     {302,129,60,315,408}
```

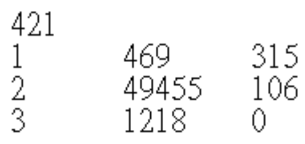
    (b) Result file 2 (statistics file):

    - ◆ 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

      Ex:



```
421
1       469     315
2       49455   106
3       1218    0
```

  - ■ 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**):

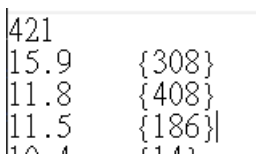
    (a) Result file 1 (itemset list):

- 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

  Ex:



```
421
15.9    {308}
11.8    {408}
11.5    {186}
```

- 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: {0.3, 0.6, 0.9}
   - Configuration for datasets B: {0.2, 0.4, 0.6}
   - Configuration for datasets C: {0.5, 1.0, 1.5}
   - If you have time, you can also try other settings by yourself. (This part is optional with no extra bonus point.)

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

1. You may implement it yourself or use an open-sourced one as the base for modification.
   - Please note that a 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 efficiently 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 the **same** as Step II):
     - Programming language

       (a) Please use **Python** to implement the algorithm in Step II.
     - Running Machine/platform
     - Dataset configuration
5. After verifying the correctness, you need to show the performance (on efficiency)

improvements and give some analysis in your report.

(the percentage of speedup: $\frac{(Computation\ time\ of\ Step\ \text{II} - Computation\ time\ of\ Step\ \text{III})}{Computation\ time\ of\ Step\ \text{II}} \times 100\%$)

IV. Performance ranking on Step III:

1. The points of this step will be determined by your **ranking of computation efficiency** (on the testing dataset) compared among classmates. The shorter the computation time, the higher rank you will get.
2. The scale of the testing dataset is the same as **dataset C**.
3. TA will run your code of Step III to mine the testing dataset and count the computation time.
   - TA would **use the command line to run your code**. The format of command line options must be the same as the description we mentioned in **Requirements 1-(1)**.
   - The output format must be the same as the result file 1 of Task 1 in Step II.
   - **Note: If you have the wrong mining results or wrong format on the output file, you will not get any score on this part.**
   - A toy dataset (the scale is same as the testing dataset) and a verification code will be provided to let you verify the correctness of the output format and mining results. (Details will be announced later on E3.)

Requirements

請注意，每一項均有分配分數；助教會根據下方準則為標準，並依據撰寫的品質來給分（非有寫就滿分）。

1. A Report:

(1) Explain how to run your code in Step II and III.

- The command line options for your code only require the dataset path (-f), the minimum support (-s), and the output file path (-p) as input. If you have any other parameters, **you must set a default value.**

Ex:

```
python ./step3.py -f datasetA.data -s 0.01 -p ./testing
```

- If we cannot run your code correctly, you will not get the points in step IV.
- The output file of your Python code must match your command line. If TA modifies the dataset path and provides minimum support when running your code, **the name of the output file also needs to be modified accordingly.**

Ex: If your command line looks like the example above, your output should be step2_task1_datasetA_0.01_result.txt .

(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 (算分)
- 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
- 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:** $\frac{(\mathit{Computation\ time\ of\ Step}\ \mathrm{II} - \mathit{Computation\ time\ of\ Step}\ \mathrm{III})}{\mathit{Computation\ time\ of\ Step}\ \mathrm{II}} \times 100\%$)
    - 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 dataset size (i.e., the rate of change on computing time under different data sizes, 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
    - **TA will use the code of Step III to conduct performance ranking (Step IV).**
    - **Name the file as Step2.py and Step3.py, respectively.**
- 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.

```
DM_HW1_{student id}_{your name}.zip
├ HW1_Report.pdf
├ Step2.py
├ Step3.py
├ datasetA.data
├ datasetB.data
├ datasetC.data
└ {Result Files}.txt
```

Grading

- Step II: 65 %
- Step III: 25 %
- Step IV: 10 %

Important Date

- Deadline: **2024/10/30 (Wed.) 23:59:59**

Penalty

- Command line error
    - Cannot run your code through command line options correctly (-10% for Step IV)
- Format error
    - The zip file name has format error (-5%)
    - The report is not in pdf. (-5%)
- Late submission
    - If you submit your work within one day after the deadline, you will get **80%** of the original score.
    - If you submit your work within two days after the deadline, you will get **50%** of the original score.
    - If you submit your work more than two days after the deadline, you will get **zero score** on this homework.