CS412: Introduction to Data Mining **Fall 2015**

Assignment 3

Due: 10/22/2015 11:59pm

General Instruction

- Errata: After the assignment is released, any further corrections of errors or clarifications will be posted at the Errata page at Piazza. Please watch it.
- Feel free to talk to other members of the class when doing the homework. We are more concerned about whether you learn how to solve the problem than whether you solve it entirely on your own. You should, however, write down the solution yourself.
- Please use Piazza first if you have questions about the homework. Also feel free to send us e-mails and come to office hours.
- For each question, you will **NOT** get full credit if you only give a final result. Necessary calculation steps and reasoning are required.
- For each question, you should show the necessary calculation steps and reasoning—not only final results. Keep the solution brief and clear.
- For a good balance of cognitive activities, we label each question with an activity type:
  - **L1 (Knowledge)** Definitions, propositions, basic concepts.
  - **L2 (Practice)** Repeating and practicing algorithms/procedures.
  - **L3 (Application)** Critical thinking to apply, analyze, and assess.

Assignment Submission

- Please submit your work before the due time. **We do NOT accept late homework!**
- Please submit your answers electronically via Compass (`http://compass2g.illinois.edu`). Contact TAs if you have technical difficulties in submitting the assignment.
- Please **type** your answers in an **Answer Document**, and submit it in PDF. **Handwritten answers are not acceptable**.
- This assignment consists of four written assignments and one large Machine Problem (MP). Your answers to all questions (including MP) should be included in one Answer Document, named as `NetId_assign3_answer.pdf`.
- The four written questions do not require programming at all. The last part is the first MP (not a mini one) we have in this course. It consists of several programming assignments, which usually take more time than written assignments, so please start early.
- Find detailed submission guidelines for codes and results in the requirements of MP.

Question 1 (12 points)

Based on the tiny database of 5 transactions in Table 1, use the Apriori algorithm to find the frequent patterns with relative $min\_sup = 0.6$.

Purpose

- Get a better understanding as well as hands-on experience of the Apriori algorithm.

Requirements

- For this question, you are required to simulate the basic Apriori algorithm and write down all intermediate as well as final results. No programming is needed.
- Use the abbreviations we give you (C1, L1...) to denote which list you are writing about. You may use a table to contain all lists, or just write them one by one.
- For each itemset you write down in the Li lists, put its corresponding absolute support after it, with a colon between them, such as $L1 = \{m : 4, \ldots\}$.
- Do not forget to write down the actions (pruning, self-joining or db-scanning) you take to generate F1 and C2.

Trans.	Items
1	b,d,f,g,l
2	f,g,h,l,m,n
3	b,f,h,k,m
4	a,f,h,j,m
5	d,f,g,j,m

Table 1: A tiny transaction database

a. $(2', L1)$ List all candidate 1-itemsets (C1). What will you do to get rid of non-frequent 1-itemsets (*Choose one from pruning, self-joining and db-scanning*)? List all frequent 1-itemsets (L1).

b. $(2', L1)$ What will you do to generate all candidate 2-itemsets (C2) (*Choose one from pruning, self-joining and db-scanning*)? List all itemsets in C2.

c. $(1', L1)$ Take the same action you choose for computing L1 from C1, and list all frequent 2-itemsets (L2) computed from C2.

d. $(3', L1)$ To generate all candidate 3-itemsets (C3) from L2, what is the extra action you need to consider besides the one you take from L1 to C2 in sub-question b (*Choose one from pruning, self-joining and db-scanning*)? List all itemsets in C3. Are they all frequent?

e. $(2', L2)$ Is there any frequent 4-itemset? Why?

f. $(2', L3)$ According to your simulation, what part of Apriori involves the heaviest computation? Under what circumstances will this be extremely bad? *Hints: For the first question, you can still choose from the three steps hinted before and explain. For the second question, think about the impact of min_sup.*

Question 2 (13 points)

Based on the same database Question 1, use the Frequent Pattern Growth algorithm with relative $min\_sup = 0.4$ to find the frequent patterns.

Purpose

- Get a better understanding as well as hands-on experience of the FP-Growth algorithm.

Requirement

- For this question, you are required to simulate the basic FP-Growth algorithm. No programming is needed.
- You are required to generate some tables and figures. You can use any software to do that. Only for sub-question b, you are also allowed to draw the Header Table and FP-tree by hand and then scan or take picture. But be sure to make the picture clear in your Answer Document.
- For sub-question a, generate a table to present the results.
- For sub-question b, put the Header Table and FP-tree side by side, preferably with Header Table on the left (just like those in the slides). Use a colon to separate an item and its corresponding count in the FP-tree.
- For sub-question c, for each of the items, write down its Conditional Pattern Base followed by the frequent patterns computed based on it. In order to generate the correct frequent patterns, please first generate the Conditional FP-trees. But you do not need to show the Conditional FP-trees in the Answer Document.

a. $(2', L2)$ Generate an ordered list of frequent items based on the raw transaction database. *Hints: Reorder items within each transaction according to their frequencies in the whole database.*

b. $(5', L2)$ Generate Header Table and FP-tree based on the frequent item list. Link nodes to the corresponding positions in the Header Table.

c. $(4', L2)$ Generate Conditional Pattern Bases and Conditional FP-trees for items m, h, b, j based on the FP-tree, and list the frequent patterns computed based on each of the Conditional FP-trees. (You only need to show the Conditional Pattern Bases and the frequent patterns.)

d. $(2', L3)$ Why do we order the items in each transaction by their frequency before constructing the FP-tree? *Hints: Think about the purpose of FP-tree and how this order will affect its structure.*

Question 3 (10 points)

Based on the frequent patterns computed in Question 1, find closed patterns, maximal patterns and association rules.

Purpose

- Get a better understanding of closed patterns, maximal patterns and association rules.

Requirement

- For this question, you will be doing some counting. No programming is needed.
- For sub-question c, show the relative support and confidence of each association rule you find.

a. $(3', L1)$ List **all** closed patterns among the frequent patterns computed in Question 1.

b. $(3', L1)$ List **all** maximal patterns among the frequent patterns computed in Question 1.

c. $(2', L2)$ List **at least 2** association rules with $min\_conf = 0.6$ from the frequent patterns computed in Question 1.

d. $(2', L3)$ Under what circumstances should we prefer maximal patterns than closed patterns? What about the other way around? *Hints: Think about the differences between the two special frequent patterns.*

Question 4 (15 points)

This is a set of **true** or **false** questions. Please answer the following questions.

Purpose

- Have a better understanding of some basic concepts about frequent pattern mining.

Requirement

- For each sub-question, choose **true (T)** or **false (F)** and provide a brief explanation of your choice. You will not get credit without explanation.

a. $(3', L3)$ A max-pattern must be a closed pattern. If yes, briefly explain your idea; otherwise, give a counter example.

b. $(3', L2)$ At each step of Apriori, we will generate all non-repeated $(k+1)$-itemsets joined by each pair of k-itemsets that agree on $(k-1)$ items, and then test the frequency of those $(k+1)$-itemsets on the transaction database to remove all infrequent ones.

c. $(3', L2)$ For FP-Growth, in order to mine all frequent patterns, we have to recursively generate conditional frequent bases and conditional FP-trees until the FP-tree generated has only one node.

d. $(3', L1)$ CLOSET is based on Apriori, while MaxMiner is based on FP-Growth.

e. $(3', L3)$ To measure the correlations between frequent items, *Lift* is always better than *Confidence* (as defined in association rule). Use examples to analyze.

Machine Problem (MP, 50 points)

Computing frequent patterns by hand is so tedious - this is where computers come into use! In this MP, given preprocessed data of the paper titles collected from computer science conferences from 5 domains, you are required to 1) implement a frequent pattern mining algorithm to mine frequent patterns from each of the 5 domains, so as to find 'meaningful' patterns for each domain; 2) mine closed/maximal patterns based on the frequent patterns you find, so as to understand the different definitions and applications of closed/maximal patterns; 3) find association rules using Weka. You can find `data.zip` from the course website.

Purpose

- Get deeper understanding and working experience of Apriori and FP-growth algorithms through implementation.
- Explore how frequent pattern mining can be applied to text data to discover meaningful phrases and summarize topics within documents.
- Get hands-on experience of mining association rules using Weka.

Requirement

- This is the first MP (not a mini one) we have in this course. It consists of several programming tasks, which usually take more time than written assignments, so please start early.
- This is an individual assignment, which means it is OK to discuss with your classmates and TAs regarding the methods, but it is not OK to work together or share code.
- Similar libraries or programs of frequent pattern mining algorithms can be found online, but you are prohibited from using these resources directly. This means you can not include external libraries, or modify existing programs, since the purpose of this MP is to help you go through frequent pattern mining step by step.

- You can use Java/C++/Python/Matlab as your programming language. No other languages are allowed.
- For your answers in the Answer Document, you should include 1) brief explanation about the algorithms you use in Step 1 and Step 2; 2) answers to all questions in *Question to ponder*; 3) a list of the names of your source files and their corresponding steps (the steps are listed below). Put all of them after the answers to the written assignments.
- Put all your codes in a separate folder with the name `NetId_assign3_codes`. Do not use sub-folders inside this folder. All of your codes should have been successfully compiled before submission. Do not include files other than the codes you write. Put a single `readme.txt` file in the code folder to briefly describe the functionalities of your codes and how to run them.
- Put the results you generate in another folder with the name `NetId_assign3_results`. Create sub-folders with names: `patterns` (which should include files `pattern-0.txt`∼`pattern-4.txt`), `closed` (which should include files `closed-0.txt`∼`closed-4.txt`) and `max` (which should include files `max-0.txt`∼`max-4.txt`).
- Your Answer Document, with the name `NetId_assign3_answer.pdf`, should be at the same level as your code folder and result folder. Compress these (two folders and one file) three together into a zip file, and name it `NetId_assign3.zip`. Submit this zip file through Compass2g.
- Copying source code from other students will give you 0 grade. We will run plagiarism detection software. Please do not cheat.

Description of Data Preprocessing

- We use paper titles collected from conferences in computer science of 5 domains: Data Mining (DM), Machine Learning (ML), Database (DB), Information Retrieval (IR) and Theory (TH). The raw data is named as `paper_raw.txt`. Each line contains two columns, the `PaperID` and the `Title` of a paper, separated by a tab. Recall the example in class. You can consider each line in the file as one transaction. Each term in the title is then equivalent to an item in a transaction. We provide this file to give you a basic idea about what the task is and what data you are using. You will not work on this file directly.
- For this assignment, we have pre-processed the raw data by removing stop words, converting the words to lower cases, and lemmatization. The results are in `paper.txt`. In this file, each line is a list of terms. Terms are separated by one space. Again, this file is for you to understand the task, and you will not work on this file directly.
- To make computation easier, we generate a vocabulary from `paper.txt`, and name it as `vocab.txt`. Each line in this file has two columns: the first column is the term index and the second column is a unique term extracted from `paper.txt`; columns are separated by Tab. Each term in `paper.txt` appears exactly once in `vocab.txt`. With

this vocabulary, we can always map between each term and its corresponding unique indexing number. You do not need to know how this vocabulary is generated, but you do need this file to show mined patterns as required in Step 1 and Step 2 of the MP.

- Recall we have papers from 5 domains. We want you to mine frequent patterns for each of the 5 domains so as to find 'meaningful' patterns of each domain. However, the terms of 5 domains are mixed together in `paper_raw.txt` and `paper.txt`. In order to separate them, we apply LDA with 5 topics to assign one topic to each term. Then we re-organize the terms and create one file `topic-`i`.txt` for each topic i, where $i = 0, 1, 2, 3, 4$. Each line in file `topic-`i`.txt` corresponds to one paper title in the dataset. File `topic-`i`.txt` only contains paper titles with at least one term assigned with topic i by LDA, and within each paper title in `topic-`i`.txt`, terms assigned with topics other than topic i are removed. Note that we do not know which `topic-`i`.txt` corresponds to which of the 5 domains now, and we will use frequent pattern mining to figure it out. These `topic-`i`.txt` are the files you are going to work on to mine frequent patterns.

Step 1: $(25', L3)$ **Mining frequent patterns for each topic.**

In this step, you need to implement a frequent pattern mining algorithm. We recommend you to choose from Apriori and FP-Growth. Note that you need to figure out *min_sup* by yourself.

You need to run your code on 5 files corresponding to 5 topics (`topic-0.txt`∼`topic-4.txt`). Your output files should be put into one directory named as `patterns`. The i-th file is named as `pattern-`i`.txt`. The output should be of the form ([s] (space) [t1 (space) t2 (space) t3 (space) ...]). Frequent patterns in each file should be sorted from high to low by *Support*.

A sample output is given in data.zip with the name `sample.txt`. Note that in `pattern-`i`.txt` files, the term indexes should be mapped back to words based on `vocab.txt`. This requirement is the same for the next step.

Question to ponder A: How do you choose min_sup for this task? Explain your criteria; any reasonable choice will be good.

Step 2: $(20', L3)$ **Mining closed/maximal patterns.**

In this step, you need to implement an algorithm to mine closed patterns and maximal patterns. You can write the code based on the output of Step 1, or implement a specific algorithm to mine closed/maximal patterns, such as CLOSET, MaxMiner, etc.

The output should be of the same form as the output of Step 1. Closed patterns are put into a directory named `closed`. The i-th file is named as `closed-`i`.txt`. Max patterns are put into a directory named `max`. The i-th file is named as `max-`i`.txt`. Within each file, the term indices should be mapped back to the words.

Question to ponder B: Can you figure out which topic corresponds to which domain based on patterns you mine? Write down your observations.

Question to ponder C: Compare the results of frequent patterns, closed patterns and maximal patterns, is there any difference? If so, what kind of patterns give more satisfying results? Write down your analysis.

Step 3: $(5', L2)$ **Mining association rules by Weka.**

In this step, you will use Weka to mine association rules. In class, we have demonstrated how to do this. You can download the slides *Weka-Associate* from the course schedule page or watch the online lecture video to review the process. To make things easier, we also provide detailed procedures here.

(a) Generate `.arff` files.

We have provided you with the code `featureGenerator.py`. Put it in the same folder as `topic-`i`.txt` and `vocab.txt`. In the command line, type `python featureGenerator.py`. It will generate one `.arff` file corresponding to each `topic-`i`.txt` file. (If you are MacOS/Linux user, you can easily do this. Otherwise, you can use the EWS machines with Linux OS on campus.)

(b) Get Association rules by Weka.

i. Download Weka from here and install.
ii. Double click on Weka application icon.
iii. Choose *Explorer*.
iv. Click on *Preprocess* tag.
v. Click on *Open file*, choose `topic-i.arff` , click on *Choose* and wait until Weka finishes loading the data.
vi. Click on *Associate* tag. Click on *Choose* and select *FPGrowth* as the mining algorithm.
vii. Click on the parameter line. You need to modify *lowerBoundMinSupport* and *minMetric* to generate at least 10 association rules. You may keep the default values for other parameters. (*Hint: lowerBoundMinSupport is the min_sup and minMetric is the min_conf of the association rules.*)
viii. Copy and paste the association rules generated by Weka.

Question to ponder D: What are the differences between phrases which satisfy only the min_sup criterion and phrases (association rules) which satisfy both min_sup and min_conf criteria? Compare the results of Step 1 and Step 3 and write down your observations.