COMS E6111 Advanced Database Systems
Fall 2012

Project 3

Due Date: Monday November 26 at 5 p.m. EST
(same deadline for both on-campus and CVN students)

For this project you will **extract association rules** from an interesting **data set of your choice**. Therefore, your project will consist of two main components: (1) *defining your data set of choice* and (2) *implementing the a-priori algorithm for finding association rules*.

Data Set Definition

For this project, you will use the official New York City data sets that are available at the NYC Open Data site, at https://data.cityofnewyork.us/. You can access the data sets at https://data.cityofnewyork.us/browse?limitTo=datasets. There is a wide variety of data sets, and an important part of this project is that you explore these data sets and pick one or more for you to use in the project, as follows:

1. You can base your project on just one data set from the above web site, or you can combine multiple such data sets (for example, by joining them over a common attribute, such as zip code) into a larger data set. Either way, your association rule mining algorithm will operate over an individual file, which we will refer to as **your INTEGRATED-DATASET file** in the rest of this description. **IMPORTANT:** Make sure you pick NYC Open Data data set(s) from which we can derive interesting association rules.
2. Your INTEGRATED-DATASET file should then **always** be a single file, which could correspond to just one data set from NYC Open Data or to multiple data sets joined together.
3. Your INTEGRATED-DATASET file should be formatted as a CSV file, which you will include in your submission. Note that the NYC Open Data files can be downloaded in a variety of formats. Regardless of the format of the original data set(s) that you used to generate your INTEGRATED-DATASET file, the INTEGRATED-DATASET file should be a single CSV file, so you will need to map the original data set(s) that you use into a single CSV file if needed.
4. The INTEGRATED-DATASET file should consist of (a) **at least 1000 rows** if this file corresponds to an individual NYC Open Data data set, or (b) **at least 10,000 rows** if this file corresponds to the combination of multiple such data sets.
5. Each row in your INTEGRATED-DATASET will be interpreted as a "market basket" and each attribute of each row, intuitively, will correspond to an "item." You will identify association rules from this file (see below) using this interpretation of the rows and attributes in the file.

You do not need to submit any code or scripts that you use to generate the INTEGRATED-DATASET file, or the original NYC Open Data data sets. However, you need to submit:

1. A single CSV file containing your INTEGRATED-DATASET file.
2. A detailed description in your README file (see below) explaining: (a) which NYC Open Data data set(s) you used to generate the INTEGRATED-DATASET file; (b) what (high-level) procedure you used to map the original NYC Open Data data set(s) into your INTEGRATED-DATASET file. The explanation should be detailed enough to allow us to recreate your INTEGRATED-DATASET file exactly from scratch from the NYC Open Data site.

Association Rule Mining Algorithm

You should write and submit either a **Java** or a **Python** program to find association rules in your INTEGRATED-DATASET file, where each row in your file corresponds to one "market basket" and each attribute of each row corresponds to one "item" (see above). Specifically, you should write a program to do the following:

1. Accept as input the name of a file from which to extract association rules; we will input here the name of your INTEGRATED-DATASET file. You can assume that we will only test your program with your INTEGRATED-DATASET file, so you can implement variations of the a-priori algorithm that are a good fit for your data (see below). In this case, **you must explain in the README file precisely what variation(s) you have implemented and why** (see item 3 below for more details on what variations are acceptable).
2. Prompt the user for a minimum support *min_sup* and a minimum confidence *min_conf*, which are two values between 0 and 1. These values must be specified in the command line (and not, for example, using JOptionPane.showInputDialog()). So we should be able to call your program, for example, as:
   sh run.sh INTEGRATED-DATASET.csv 0.3 0.5
   which specifies *min_sup=0.3* and *min_conf=0.5*.
3. Compute all the "*large (i.e., frequent) itemsets*," using *min_sup* as your support threshold. The large itemsets have support **greater than or equal to** *min_sup*. You should use the a-priori algorithm described in **Section 2.1 of the Agrawal and Srikant paper in VLDB 1994** (see class schedule) to compute these large itemsets. You do **not** need to implement the "subset function" using the hash tree as described in Section 2.1.2. However, you must implement the version of a-priori in Section 2.1.1, which we discussed in class briefly but is slightly more sophisticated than the version that we covered in detail in class. ***Note:*** *Your program has to* ***compute all the large itemsets from scratch every time the program is run****; you cannot "precompute" anything ahead of time, but rather all computations have to happen each time your program is run.* You are welcome to implement variations of the a-priori algorithm that are a good fit for your data, as discussed above (e.g., to account for item hierarchies, as we discussed in class, or numerical items). **IMPORTANT NOTE: These variations have to be at least as "sophisticated" as the description of a-priori in Section 2.1 in general, and in Section 2.1.1 in particular** (i.e., your variations cannot be more primitive than the algorithm as described in these sections of the paper). A good place to start to search for relevant variations of the original algorithm is Rakesh Agrawal's publications, http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/a/Agrawal:Rakesh.html. Implementing such a variation is strictly optional; if you decide to implement such a variation, you must so indicate in the README file, explaining precisely what variation(s) you have implemented and why.
4. For each of the large itemsets, build all possible association rules and identify those that have a confidence of at least *min_conf*. Generate only association rules with one item on the right hand side and with at least one item on the left hand side. We will call the rules with confidence **greater than or equal to** *min_conf*

as the *"high-confidence rules."*

5. Output the **large itemsets** to a file named "**output.txt**": each line should include one itemset, within square brackets, and its support, separated by a comma (e.g., *[item1,item2], 7.4626%*). The lines in the file should be listed in decreasing order of their support. Output also **in the same output.txt file** the **high-confidence association rules**, in decreasing order of confidence, reporting the support and confidence of each rule (e.g., *[item1] => [item2] (Conf: 100%, Supp: 7.4626%)*).

As a "toy" example from class, consider the following INTEGRATED-DATASET file, which is a CSV with four "market baskets":

pen,ink,diary,soap
pen,ink,diary
pen,diary
pen,ink,soap

As a reminder, note that spaces are considered part of the fields in a CSV file. If we run your program with *min_sup*=0.7 and *min_conf*=0.8, the program should produce a file output.txt with the following information:

==Large itemsets (min_sup=70%)
[pen], 100%
[diary], 75%
[diary,pen], 75%
[ink], 75%
[ink,pen], 75%

==High-confidence association rules (min_conf=80%)
[diary] => [pen] (Conf: 100.0%, Supp: 75%)
[ink] => [pen] (Conf: 100.0%, Supp: 75%)

What You Should Submit

1. **Your well commented Java or Python code** with your implementation;
2. A **Makefile** file explaining how we should compile and run your code on a CS machine running Linux (e.g., on the **clic.cs.columbia.edu** machines);
3. A **single CSV file** containing **your INTEGRATED-DATASET file**;
4. A **README** file including the following information:
    a. Your name and your partner's name;
    b. A list of all the files that you are submitting;
    c. A **detailed description** explaining: (a) which NYC Open Data data set(s) you used to generate the INTEGRATED-DATASET file; (b) what (high-level) procedure you used to map the original NYC Open Data data set(s) into your INTEGRATED-DATASET file; (c) what makes your choice of INTEGRATED-DATASET file interesting (in other words, justify your choice of NYC Open Data data set(s)). The explanation should be detailed enough to allow us to recreate your INTEGRATED-DATASET file exactly from scratch from the NYC Open Data site.
    d. A clear description of how to run your program (note that your project must compile/run under

Linux in your CS account);

e. A clear description of the internal design of your project; in particular, if you decided to implement variation(s) of the original a-priori algorithm (see above), you must explain precisely what variation(s) you have implemented and why.
f. The **command line specification** of an interesting sample run (i.e., a *min_sup*, *min_conf* combination that produces interesting results). Briefly explain why the results are interesting.
g. Any additional information that you consider significant.

5. A **text file named "example-run.txt" with the output of the interesting sample run of point 4f**, listing all the large itemsets as well as association rules for that run, as discussed in the *Association Rule Mining Algorithm* section above.

How to Submit

1. Create a directory named **<your-UNI>-proj3**, where you should replace **<your-UNI>** with the Columbia UNI of one teammate (for example, if the teammate's UNI is **abc123**, then the directory should be named **abc123-proj3**).
2. Copy the source code files into the **<your-UNI>-proj3** directory, and include all the other files that are necessary for your program to run.
3. Copy your **Makefile**, **INTEGRATED-DATASET**, **README**, and **example-run.txt** files into the **<your-UNI>-proj3** directory.
4. **Tar** and **gzip** the **<your-UNI>-proj3** directory, to generate **a single file <your-UNI>-proj3.tar.gz**, which is the file that you will submit.
5. Login to Courseworks at https://courseworks.columbia.edu/ and select the site for our class.
6. Select "Assignments."
7. Upload your **<your-UNI>-proj3.tar.gz** file under **"Project 3."**

IMPORTANT NOTE 1: Your Java or Python program can use any standard classes/libraries that you might find useful.

IMPORTANT NOTE 2: We will grade your project on how interesting your data set definition is, as well as on the overall correctness of your implementation of the association rule mining algorithm. The **README** and **Makefile** files will also determine part of your grade for the project.

Good luck!

Luis Gravano