COMS E6111-Advanced Database Systems (index.html)
Spring 2022

Project 3

Due Date: Tuesday April 26, 5 p.m. ET

Teams

You will carry out this project in **teams of two**. You can do the project with your same teammate as for Project 2 and you are also welcome to switch teammates if you wish. In this case, please be considerate and **notify your Project 2 teammate immediately**. If you want to form a new team but can't find a teammate, please follow these steps:

- Post a message on the Ed Discussion board asking for a teammate—**the best way**.
- Send email to Akhil (mailto:ak4590@columbia.edu) right away (and definitely **before Tuesday, April 5, at 5 p.m. ET**) asking Akhil to pair you up with another student without a teammate. Akhil will make a best effort to find you a teammate.

You do not need to notify us of your team composition. Instead, you and your teammate will indicate your team composition when you submit your project on Gradescope (click on "Add Group Member" after one of you has submitted your project). You will upload your final electronic submission on Gradescope exactly once per team, rather than once per student.

Important notes:

- Please do not wait until the day before the deadline to start working on the project, just to realize then that your teammate formed a new team. It is your responsibility to start working on the project and spot any problems with your teammate early on.
- You can do this project by yourself if you so wish. Be aware, however, that you will have to do exactly the same project as two-student teams will.
- Please check the Class Policies webpage (proj-policies.html) for important information about what kinds of collaboration are allowed for projects, and how to compute the number of available grace late days for a team.

Description

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://opendata.cityofnewyork.us/ (https://opendata.cityofnewyork.us/). You can access the data sets at https://opendata.cityofnewyork.us/data/

(https://opendata.cityofnewyork.us/data/). There is a wide variety of data sets, and an **important part of this project is that you will 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 you can derive association rules that are revealing, surprising, useful, or helpful. (These are of course subjective properties of the rules; use your best judgment and explore the data to figure out if the data set(s) lead to the extraction of such association rules or not.)
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 (http://en.wikipedia.org/wiki/Comma-separated_values), 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 **at least 1,000 rows**.
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 used 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 and (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 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. So we should be able to call your program, for example, as:
   ```
   run.sh INTEGRATED-DATASET.csv 0.01 0.5
   ```
   which specifies *min_sup=0.01* and *min_conf=0.5*.
3. Compute all the "**frequent (i.e., large) itemsets**," using *min_sup* as your support threshold. The frequent 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 (schedule.html)) to compute these frequent 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 the a-priori algorithm described in Section 2.1.1, which is slightly more sophisticated than the version that we covered in detail in class. Your program has to **compute all the frequent 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 Chapter 6 of the "Mining of Massive Datasets" textbook (http://infolab.stanford.edu/~ullman/mmds/ch6.pdf) (3rd. edition, 2019). 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 frequent itemsets, build all possible association rules and identify those that have a confidence of at least *min_conf*. Generate only association rules with exactly one item on the right side and with at least one item on the left side, where the right-side item does **not** appear on the left side (i.e., don't consider trivial association rules). We will call the rules with confidence greater than or equal to *min_conf* as the *"high-confidence rules."*
5. Output the **frequent itemsets** and the **high-confidence association rules** to a file named **output.txt**: in the first part of this file, for the frequent itemsets, each line should include one itemset, within square brackets, and its support, separated by a comma (e.g., *[item1,item2,item3,item4], 7.4626%*). The lines in the file should be listed in decreasing order of their support. In the second part of the same output.txt file, for the high-confidence association rules, each line should include one association rule, with its support and confidence (e.g., *[item1,item3,item4] => [item2] (Conf: 100%, Supp: 7.4626%)*). The lines in the file should be listed in decreasing order of their confidence.

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:

```
==Frequent 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 to Submit and When

Your Project 3 submission will consist of the following four components, which you should submit on Gradescope by Tuesday April 26 at 5 p.m. ET:

1. Your well commented **Python source code** with your implementation (see below), which should run on your Google Cloud VM, set up as detailed here (gc-setup.html).
2. A single CSV file containing your **INTEGRATED-DATASET** file.
3. A **README** file including the following information:
    a. Your name and Columbia UNI, and your teammate's name and Columbia UNI.
    b. A list of all the files that you are submitting.
    c. A clear description of how to run your program. Note that your project must **compile/run in a Google Cloud VM set up exactly following our instructions (gc-setup.html)**; if you need more memory, please indicate so clearly in the README file and specify what configuration we should use for your Google Cloud VM. Provide all commands necessary to install the required software and dependencies for your program.
    d. 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 compelling (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.
    e. A clear description of the internal design of your project; in particular, if you decided to implement variations of the original a-priori algorithm (see above), you must explain precisely what variations you have implemented and why.
    f. The command line specification of a compelling sample run (i.e., a *min_sup*, *min_conf* combination that produces association rules that are revealing, surprising, useful, or helpful; see above). Briefly explain why the results are indeed compelling.

   g. Any additional information that you consider significant.

4. A text file named **example-run.txt** with the output of the compelling sample run of point 3f, listing all the frequent itemsets as well as association rules for that run, as discussed in the Association Rule Mining Algorithm section above.

To submit your project, please follow these steps:

1. Create a directory named proj3.
2. Copy the source code files into the proj3 directory, and include all the other files that are necessary for your program to run.
3. Tar and gzip the proj3 directory, to generate a single file proj3.tar.gz.
4. **Submit on Gradescope exactly four files:**
    - Your proj3.tar.gz file with your code,
    - Your uncompressed INTEGRATED-DATASET file,
    - Your uncompressed README file, and
    - Your uncompressed example-run.txt file.

Grading for Project 3

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 example-run.txt files will also determine part of your grade for the project.