CS 5683: Big Data Analytics

Project-1: A Simple Movie Recommender with Distributed Association Rules Analysis

Total Points: 50; Due date: Sept. 15, 2020

Association rules are frequently used for Market Basket Analysis by retailers to understand the shopping/browsing behavior of their customers. This information can then be used for different purposes such as engaging customers, cross-selling and up-selling of products, sales promotions, loyalty programs, store design, discount plans, and many other.

Application in movie recommendations: A simple movie recommender system display movie recommendations for the customer 'X' based on their watch history and other customers' watch history which is similar to X's. X would most likely revisit platforms like Netflix or Amazon Prime only if they get interesting movie recommendations to spend their weekend. A very simple recommendation method is to give movies that are most frequently watched by other customers given that other customers also watched a subset of movies in X's watch history.

Suppose we want to recommend movies to customers based on the movies they have already watched online. Write *A-Priori* algorithm using a Spark program to find movies which are frequently watched together by other customers. Fix the support to *s = 3* (i.e. movies need to occur at least 3 times to be considered frequent) and find itemsets of size 2, 3, and 4. With frequent itemsets, generate association rules that has only one item on the right hand-side of the rule (i.e) {X} → {Y} for frequent 2-itemset, {X,Y} → Z, {X,Z} → {Y}, and {Y, Z} → {X} for frequent 3-itemset and so on and so forth. Generate rules with confidence threshold *c = 60%.* Sort all association rules by decreasing order of confidence. Save the sorted association rules to a file.

Use the MovieLens movie ratings data set[1] (*vies.txt*) given with this instruction. Each line in the data represent a list of movies that a user watched and given a rating. On each line each movie is converted into an ID for the ease of computation. The movie IDs are separated by the special-characer **::** (two colons).

Tips for your Spark program:

1. One PySpark job to find both frequent itemsets and association rules
2. All computations should be done only with RDDs. If you collect() results and do computations, you are not utilizing Spark at all
3. Make use of the broadcast() wisely. Broadcast is very essential for the A-Priori algorithm

[1] DOI=http://dx.doi.org/10.1145/2827872

4. Except for frequent item mining (first pass of A-Priori), other frequent itemset mining should be in a recursive function or a ‘for’ loop. DO NOT WRITE 3 functions to generate frequent 2-itemsets, frequent 3-itemsets, and frequent 4-itemsets
5. Your Spark program should be memory efficient. Follow the A-Priori steps as it is. There are other memory inefficient programs available in web
6. Make a wise decision to choose whether to persist and write the intermediate results (frequent itemsets, for example)

Expected Output Format:

The Spark program should generate only one output file – association rules. If you write any intermediate outputs, your program should delete them before its termination.

ID1,ID5 → ID2; Confidece=98.45%

ID234,ID712,ID1 → ID193; Confidence=94.2%

(NOTE: ID*** should be replaced with ids from the input file)

Bonus Task: Association rules simply get association of movies present in the data that could be interesting to the user and they do not provide real recommendations. In the bonus task, we will complete this gap by actually recommending movie names given a user’s watch history. The dictionary of movie ID-movie name is given in *mappings.txt*. For each watch history in *get_recommendations.txt* get 3 recommendations (*movie names*) based on high confidence in association rules from the above task. **You do not need to use PySpark for this bonus task.**

Grading Rubric:

Read data and generate frequent items: 10 points

Recursive/For-loop logic: 5 points

Candidate itemset generation: 5 points

Frequent itemset generation: 5 points

Handling intermediate results: 5 points

Association rules generation: 10 points

Efficient programming: 5 points

Program documentation and output: 5 points

Bonus: 10 points (optional)

What to submit in Canvas?

Submit only your complete python notebook (.ipynb) file. We will use Google Colab to test and grade your work. Type your name on the first cell of the notebook. Add sufficient documentations in your program – the reader should understand what your logic tries to achieve from the documentation.

Simple reminder: You can discuss the logic and implementation details with your friends. But you should write your own program and documentation. *We will not grade your work if we suspect cheating.* Give references wherever necessary, if you are using a logic from any internet sources.