Homework 1

CSCI 4907/6444: Big Data and Analytics

Fall 2023

The deadline for homework assignment 1 is **09/26/2023 11:59 pm**. The total number of points for this assignment is 10. Make sure that you upload a *single* zip file for all of the questions in the assignment on blackboard. Create two folders `q1` and `q2` and put all the related content (code, output files, writeups, etc.) in the corresponding folder. Do not include the input data in your submissions. Name you folder `lastname_firstname_hw1` and compress it to create a zip file. Upload your zip file to blackboard.

Academic Integrity

Honor Code: Students may discuss the solutions at a high level with one another. Note that each student must write down their solutions/code independently to show they understand the solution well enough in order to reconstruct it by themselves. Students should clearly mention the names of all the other students who were part of their discussion group.

Important: Using code or solutions obtained from the web is considered an honor code violation. You may use printed or online sources for understanding how to use specific Python or Spark functions but you may not search for code or theoretical solutions. We check all the submissions for plagiarism. Please include the following text excerpt in your submission zip file (in the main folder, next to folders `q1` and `q2`):

```
I [insert your name here], affirm that this is my own work,
I attributed where I used the work of others, I did not fa-
cilitate academic dishonesty for myself or others, and I used
only authorized resources for this assignment, per the GW Code
of Academic Integrity.  If I failed to comply with this sta-
tement, I understand consequences will follow my actions.  Con-
sequences may range from receiving a zero on this assignment
to expulsion from the university and may include a transcript
notation.
```

1 Programming with Spark (5 pts)

In this question, you will improve your Spark programming skills and practice composing an algorithm into map and reduce functions.

Write a Spark program that implements a simple "People You Might Know" social network friendship recommendation algorithm. The key idea is that if two people have a lot of mutual friends, then the system should recommend that they connect with each other.

Data:

- Associated data file is in q1.
- The file contains the adjacency list and has multiple lines in the following format:

  `<User><TAB><Friends>`

Here, `<User>` is a unique integer ID corresponding to a unique user and `<Friends>` is a comma separated list of unique IDs corresponding to the friends of the user with the unique ID `<User>`. Note that the friendships are mutual (i.e., edges are undirected): if A is friend with B then B is also friend with A. The data provided is consistent with that rule as there is an explicit entry for each side of each edge.

Algorithm: Let us use a simple algorithm such that, for each user U, the algorithm recommends N=20 users who are not already friends with U, but have the most number of mutual friends in common with U.

Output:

- The output should contain one line per user in the following format:

  `<User><TAB><Recommendations>`

  where `<User>` is a unique ID corresponding to a user and `<Recommendations>` is a comma separated list of unique IDs corresponding to the algorithm's recommendation of people that `<User>` might know, ordered in decreasing number of mutual friends.
- Even if a user has less than 20 second-degree friends, output all of them in decreasing order of the number of mutual friends. If a user has no friends, you can provide an empty list of recommendations. If there are recommended users with the same number of mutual friends, then output those user IDs in numerically ascending order.

Pipeline sketch: Please provide a clear description of how you used Spark to solve this problem. The description does not have to be long, but it should be clear enough to demonstrate your strategy for solving the problem.

Note: Please implement and run your algorithm on all users. In your writeup, please include only the recommendations for the users with following user IDs: 924, 8941, 8942, 9019, 9020, 9021, 9022, 9990, 9992, 9993.

Tips:

- Before submitting a complete application to Spark, you may go line by line, checking the outputs of each step. Command `.take(X)` should be helpful, if you want to check the first X elements in the RDD.
- For sanity check, your top 10 (you should provide top 20) recommendations for **user ID 11** should be:
  `27552,7785,27573,27574,27589,27590,27600,27617,27620,27667`.
- The execution may take a while. Our implementations took around 10 minutes.
- The default memory assigned to the Spark runtime may not be enough to process this data file, depending on how you write your algorithm. If your Spark job fails with a message starting as:

  ```
  17/12/28 10:50:35 INFO DAGScheduler: Job 0 failed: sortByKey at
  FriendsRecomScala.scala:45, took 519.084974 s Exception in
  thread "main" org.apache.spark.SparkException: Job aborted due
  to stage failure: Task 0 in stage 2.0 failed 1 times, most
  recent failure: Lost task 0.0 in stage 2.0 (TID 4, localhost,
  executor driver)
  ```

  then you'll very likely need to increase the memory assigned to the Spark runtime. For Google Colab, you can use the following lines if you run into this issue:

  ```
  from pyspark import SparkConf
  conf=SparkConf().set("spark.executor.memory", "8g")
  ```

What to submit

1. Upload your .ipynb file (colab/jupyter notebook) to blackboard (as a part of the zip file for the whole assignment). Make sure you write comments for your code and cite any references that you used.
2. Include the output text file (containing recommendations) in your submission.
3. Include in your writeup a short paragraph sketching your spark pipeline. Demonstrate your understanding by explaining your stratgey, including what exactly your map and reduce functions are doing and what happens in the sorting by key stage.
4. Include in your writeup the recommendations for the users with following user IDs: 924, 8941, 8942, 9019, 9020, 9021, 9022, 9990, 9992, 9993.

2 Association Rules (5 pts)

Association Rules are frequently used for Market Basket Analysis (MBA) by retailers to understand the purchase behavior of their customers. This information can be then used for many different purposes such as cross-selling and up-selling of products, sales promotions, loyalty programs, store design, discount plans and many others.

Evaluation of item sets: Once you have found the frequent itemsets of a dataset, you need to choose a subset of them as your recommendations. Commonly used metrics for measuring significance and interest for selecting rules for recommendations are:

1. **Confidence** (denoted as conf(A $\rightarrow$ B)): Confidence is defined as the probability of occurrence of B in the basket if the basket already contains A:
$$conf(A \rightarrow B) = Pr(B|A),$$
where $Pr(B|A)$ is the conditional probability of finding item set B given that item set A is present.

2. **Lift** (denoted as lift(A $\rightarrow$ B)): Lift measures how much more "A and B occur together" than "what would be expected if A and B were statistically independent":
$$lift(A \rightarrow B) = \frac{conf(A \rightarrow B)}{S(B)}$$
where $S(B) = \frac{Support(B)}{N}$ and N = total number of transactions (baskets).

3. **Conviction** (denoted as conv(A $\rightarrow$ B)): Conviction compares the "probability that A appears without B if they were independent" with the "actual frequency of the appearance of A without B":
$$conv(A \rightarrow B) = \frac{1 - S(B)}{1 - conf(A \rightarrow B)}$$

(a) [0.5 pts]

A drawback of using confidence is that it ignores Pr(B). Why is this a drawback? Explain why lift and conviction do not suffer from this drawback.

(b) [0.5 pts]

A measure is symmetrical if measure(A $\rightarrow$ B) = measure(B $\rightarrow$ A). Which of the measures presented here are symmetrical? For each measure, please provide either a proof that the measure is symmetrical, or a counterexample that shows the measure is not symmetrical. You have to show every step of the proof in order to get credit.

(c) [0.5 pts]

Perfect implications are rules that hold 100% of the time (or equivalently, the associated conditional probability is 1). A measure is desirable if it reaches its maximum achievable value for all perfect implications. This makes it easy to identify the best rules. Which of the above measures have this property? You may ignore $0/0$ but not other infinity cases. Also you may find it easy to explain by an example.

Application in product recommendations: The action or practice of selling additional products or services to existing customers is called cross-selling.

Giving product recommendation is one of the examples of cross-selling that are frequently used by online retailers. One simple method to give product recommendations is to recommend products that are frequently browsed together by the customers.

Suppose we want to recommend new products to the customer based on the products they have already browsed online. Write a program using the A-priori algorithm to find products which are frequently browsed together. Fix the support to s =100 (i.e. product pairs need to occur together at least 100 times to be considered frequent) and find itemsets of size 2 and 3.

Use the online browsing behavior dataset from browsing.txt in q2. Each line represents a browsing session of a customer. On each line, each string of 8 characters represents the ID of an item browsed during that session. The items are separated by spaces. Some lines contain duplicste items. Removing or ignoring duplicates should not impact your results.

Note: for parts (d) and (e), the writeup will require a specific rule ordering but the program need not sort the output. We are not giving partial credits to coding when results are wrong. However, two sanity checks are provided and they should be helpful when you progress: (1) there are 647 frequent items after 1st pass ($|L1| = 647$), (2) the top 5 pairs you should produce in part (d) all have confidence scores greater than 0.985. See detailed instructions below.

(d) [1.75 pts]

Identify pairs of items (X, Y) such that the support of {X, Y} is at least 100. For all such pairs, compute the confidence scores of the corresponding association rules: $X \Rightarrow Y$, $Y \Rightarrow X$. Sort the rules in decreasing order of confidence scores and list the top 5 rules in the writeup. Break ties, if any, by lexicographically increasing order on the left hand side of the rule. (You need not use Spark for parts d and e of question 2)

(e) [1.75 pts]

Identify item triples (X, Y, Z) such that the support of {X, Y, Z} is at least 100. For all such triples, compute the confidence scores of the corresponding association rules: $(X, Y) \Rightarrow Z$, $(X, Z) \Rightarrow Y$, $(Y, Z) \Rightarrow X$. Sort the rules in decreasing order of confidence scores and list the top 5 rules in the writeup. Order the left-hand-side pair lexicographically and break ties, if any, by lexicographical order of the first then the second item in the pair.

What to submit

Upload all the code on blackboard and include the following in your writeup:

1. Explanation for 2(a).
2. Proofs and/or counterexamples for 2(b).

3. Explanation for 2(c).
4. Top 5 rules with confidence scores [2(d)].
5. Top 5 rules with confidence scores [2(e)].