DS702 Assignment 1

Release Date: 16 January 2022
Due Date: 30 January 2022

- Submit your answers as an electronic copy on Moodle (pdf, jupyter notebook).
- No unapproved extension of deadline is allowed. For emergencies and sickness, extensions must be requested as soon as possible.
- Cite your sources if you are taking help (papers, websites, students etc.).
- Plagiarism is strictly prohibited. Negative mark will be assigned for plagiarism.
- Remember to comment your code. And your answers should be detailed.

1 Theoretical Questions

1.1 Map-Reduce

Suppose we execute the word-count MapReduce program on a large repository such as a copy of the Web. We shall use 100 Map tasks and some number of Reduce tasks.

(a) Suppose we do not use a combiner at the Map tasks. Do you expect there to be significant skew in the times taken by the various reducers to process their value list? Why or why not?

(b) If we combine the reducers into a small number of Reduce tasks, say 10 tasks, at random, do you expect the skew to be significant? What if we instead combine the reducers into 10,000 Reduce tasks?

1.2 Algorithms using MapReduce

In the form of relational algebra implemented in SQL, relations are not sets, but bags; that is, tuples are allowed to appear more than once. There are extended definitions of union, intersection, and difference for bags, which we shall define below. Write MapReduce algorithms (pseudocode) for computing the following operations on bags R and S:

(a) Bag Union, defined to be the bag of tuples in which tuple t appears the sum of the numbers of times it appears in R and S.

(b) Bag Intersection, defined to be the bag of tuples in which tuple t appears the minimum of the numbers of times it appears in R and S.

(c) Bag Difference, defined to be the bag of tuples in which the number of times a tuple t appears is equal to the number of times it appears in R minus the number of times it appears in S. A tuple that appears more times in S than in R does not appear in the difference.

1.3 Finding Similar Items

(a) Compute the Jaccard similarities of each pair of the following three sets: {1, 2, 3, 4}, {2, 3, 5, 7}, and {2, 4, 6}

(b) Compute the Jaccard bag similarity of each pair of the following three bags: {1, 1, 1, 2}, {1, 1, 2, 2, 3}, and {1, 2, 3, 4}.

2 Spark

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 db.txt.
- 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 = 10 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.
- Note: The exact number of recommendations per user could be less than 10. If a user has less than 10 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.

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.
- Your top 10 recommendations for **user ID 11** should be: 27552, 7785, 27573, 27574, 27589, 27590, 27600, 27617, 27620, 27667.

What to submit:

- Your code (with description).
- Include in the description the recommendations for the users with following user IDs: 1211, 9993, 19978, 24211, 4774, 2017, 10709, 9206, 24435, 24444.