CSC4008 Assignment 1

Start: Sep 22 at 14:00

End: Oct 13 at 23:59

Important Notes

1. The assignment is an individual project, to be finished on one's own effort.
2. The work must be submitted before the deadline. Late submissions within 2 days will apply 20% penalty. Late submissions after 2 days will not be accepted.
3. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, 30%-100% marks will be deducted.
4. Please read the document carefully. No argument will be accepted on issues that have been specified clearly in the documents.

1 Sales Analysis

You are provided with historical sales data for 45 stores located in different regions in US - each store contains a number of departments. The company also runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of which are the Super Bowl, Labor Day, Thanksgiving, and Christmas.

Data: There are two files:

- `stores-data-set.csv` contains anonymized information about the 45 stores, indicating the type and size of store.
- `sales-data-set.csv` contains Historical sales data, which covers to 2010-02-05 to 2012-11-01. Within this tab you will find the following fields: `Store` - the store number, `Dept` - the department number, `Date` - the week, `Weekly_Sales` - sales for the given department in the given store, `IsHoliday` - whether the week is a special holiday week.

Output:

You need to write the Spark program to output:

(a) The total sales for each store type. [10 pts (5 pts for code)] *Note that the result should have 3 rows (header is not included)*

(b) Average sales on Holidays vs. Non-Holidays, and determine if sales are generally higher during holidays. [10 pts (5 pts for code)] *Note that the result should have 2 rows (header is not included)*

2 People You Might Know

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. For example, A and E in Figure 1 have 3 mutual friends (i.e., B, C, D), which is a high number in this toy example. But, if the two users are already friends, the system should not recommend them to each, even if they share mutual friends. For example, E and F have one mutual friend G, but they are already friends with each other.

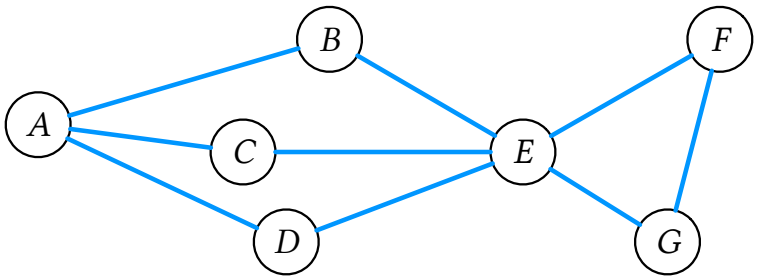

Figure 1: A social network toy example.

Data:

- Associated data file is `ego-facebook.txt` in `A1/`.
- The file contains the edge list and has multiple lines in the following format:
  `<User1><Whitespace><User2>`
  Here, `<User1>` and `<User2>` are unique integer IDs corresponding to two unique user, respectively. The pair denotes `<User1>` and `<User2>` are friends. Note that the friendships are mutual (i.e., edges are undirected): if A is friend with B then B is also friend with A. For the friendship between A and B, we only have one edge in the data.

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.

Pipeline sketch: Please provide a description of how you used Spark to solve this problem. Don't write more than 3 to 4 sentences for this: we only want a very high-level description of your strategy to tackle this problem.

Tips:

- Use Google Colab to use Spark seamlessly, e.g., copy and adapt the setup cells from Colab 0.
- 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 recommendations for **user ID 1571** should be: 35, 247, 716, 719, 1526, 1527, 1528, 1529, 1530, 1531.
- The execution may take a while.
- You can also create a toy test dataset (e.g., using Figure 1) to help you debug the program.

What to submit

You need to submit the following three files to BlackBoard.

1. A short writeup contains
    - Q1(a): The total sales for the 3 types, respectively. (5 pts)
    - Q1(b): The average sales on Holidays vs. Non-Holidays. (5 pts)
    - Q2: A short paragraph sketching your spark pipeline. (12 pts)
    - Q2: The recommendations for the users with following user IDs: 10, 152, 288, 603, 714, 1525, 2434, 2681. (6 pts for each, 48 pts in total)
2. Your code for Q1. (10 pts)
3. Your code for Q2. (20 pts)