EECS E6893 Big Data Analytics - Fall 2019

Homework Assignment 2: Friends Recommendation, GraphFrame

Due Friday, October 18th, 2019 by 5:00pm

TL;DR

1. Implement and run in Spark
2. Process data with Spark Dataframe, and perform graph analysis
3. Submit your codes and report to Canvas

Question 1 (40%)

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.

Input:

The input 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><Recommendations>
Where <User> is a unique ID corresponding to a user and <Recommendations> is a list of unique IDs corresponding to the algorithm’s recommendation of people that <User> might know, ordered in decreasing number of mutual friends. If a user has less than 10 second-degree friends, output all of the, in decreasing order of the number of mutual

friends. If a user has no friends, providing 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.

We've provided you with starter code friends.py on Canvas. Feel free to change the code if needed, or paste the code into a Jupyter notebook. Take a screenshot of your code and results in your report.

(1) Provide the screenshots of the code. (15%)

(2) Provide the screenshots of the recommendations result for the users with the following user IDs: [924, 8941, 8942, 9019, 49824, 13420, 44410, 8974, 5850, 9993] (25%)

Question 2 Graph Analysis (60%)
Implement Connected Components and PageRank

Connected Components (30%)
The connected components algorithm labels each connected component of the graph with the ID of its lowest-numbered vertex. For example, in a social network, connected components can approximate clusters.

Provide screenshots for the question (1) - (3). Use the dataset in Question 1.

(1) How many clusters / connected components in total for this dataset? (10%)

(2) How many users in the top 10 clusters? There are different number of users in each clusters, so rank them and give the top 10 clusters with the largest amount of users. (10%)

(3) What are the user ids for the cluster which has 25 users? Basically, list out all the 25 user IDs in that cluster. (10%)

Page rank (30%)
PageRank measures the importance of each vertex in a graph, assuming an edge from u to v represents an endorsement of v's importance by u. For example, if a Twitter user is followed by many others, the user will be ranked highly.

For the questions (4) - (6), use the dataset from Question 1.

(4) Provide a list of 10 important Users in this network. Who is the most important one? Order by the "PageRank" value. Provide screenshots of this answer. (10%)

(5) By using different parameter setting for PageRank, is there any difference? This is an open question, you can try as many as you want. Provide the screenshots of your tests (5%)

(6) Why this User become the most important one? What are the possible reasons? This is an open question, basically, understand how PageRank works. You can also use the result from the connected component to explain it. (5%)

For the question (7), this is a written question. Use Figure (1). No code needed.

(7) PageRank Calculation (10%)
Given the graph and formula below, calculate 5 ID's PageRank until convergence. (For each iteration, the values keep 2 decimals.)
Assume the initial PageRanks is 0.2; N =5; Tolerance / Margin of error = 0.1; d = 0.85;
P_j are the sources of incoming edges, the vertices that point to P_i; Take ID1 as an example, P_i is ID1, and P_j is ID2 and ID3;
$L(P_j)$ is the number of outgoing edges. Take ID2 as an example, L(ID2) = 2

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

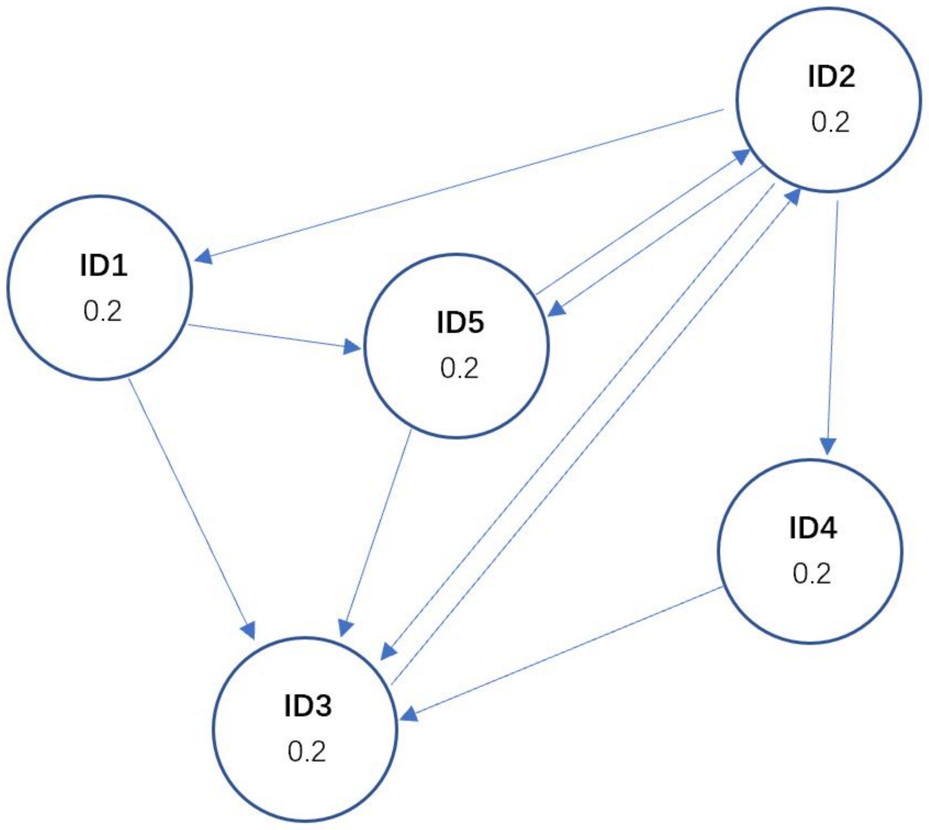

Figure (1)