Athens University of Economics and Business

MSc in Business Analytics

Data Mining Techniques – Assignment 1

Deadline: 20/5/2024

Group assignment (groups of up to 3 people).

The assignment corresponds to 25% of the total grade of the course.

Discussions between groups are recommended, but collaborating on the actual solutions is considered cheating and will be reported.

There will be no extension of the assignment deadline

Professor: Y.Kotidis (kotidis@aueb.gr)

Assistant: I.Filippidou(filippidoui@aueb.gr)

Assignment 1 Recommending friends

When you sign into Facebook, it suggests friends. In this assignment, you will write a program that reads Facebook data and makes friend recommendations. In order to create your recommendation engine you will use topology based methods proposed in the literature. In details the steps that this assignment requires are described below.

1) Dataset

For this assignment you will need to handle graph datasets and in specific social network graphs. You need to download ego-Facebook dataset from the link below:

https://snap.stanford.edu/data/egonets-Facebook.html

This dataset consists of 'circles' (or “friends” lists') from Facebook. The dataset includes node features (profiles), circles, and ego networks. This dataset consists of 4039 nodes (users) and 88234 edges (friendships between users). The graph is connected and directed (each edge counts as friendship for both nodes). For this assignment you need to transform the graph to undirected by adding the missing edges (e.g. for an edge from node1 to node2 you need to add an edge from node2 to node1 in the file). Also keep the node ids from the file, notice id count start from 0. From the available files you will need to download only facebook_combined.txt.gz , which combines all the necessary information (nodes and connections) for this assignment. In this file each line contains one edge (friendship) between two user ids.

Before working with Facebook dataset, and in order to be able to test your code manually you can create your own sample graph. It is always a good idea to test your code on a dataset that is small enough for you to manually compute the results. For this reason create a small graph (8-10 nodes) and add edges as you wish in order to form a connected graph. Then manually compute the results and compare it to your program output each time you wish to test your code.

2) Recommending friends using Common neighbors (friend-of-friend (FoF) method)

For this assignment you need to implement a recommendation system using the common neighbors' method. This method derives from the fact that if two users in the social network share many common friends, they may have a great chance to become friends in the future. This algorithm is also called as "Common-Neighbors". In this method if non-friend Y is your friend's friend, then maybe Y should be your friend too. If person Y is the friend of many of your friends, then Y is an even better recommendation. The best friend recommendation is the person with whom you have the largest number of mutual friends. In order to score each of the friendship suggestions between two nodes that are not already connected, for your algorithm you will use the score function below:

$$\boldsymbol{score(A, B) = |N_A \cap N_B|}$$

Where N_A denotes the neighbors of node A.

For the assignment you need to print a list containing the first 10 friend recommendations and their corresponding score, as determined by FoF method for nodes:

nodeID: 107 (high degree node)

nodeID: 1126

nodeID: 14

nodeID: 35

If there are fewer than 10 recommendations, print all the recommendations.

In the case of ties in friendship score you should output the node with the smallest nodeID (based on the ids given from the dataset).

3) Recommending friends using Jaccard coefficient

Another approach that you need to implement in this assignment is to recommend friends based on jaccard coefficient. The jaccard coefficient is a statistic used to compare similarities. The Jaccard coefficient measures the similarity between sample sets and is defined as the size of the intersection divided by the size of the union. In order to score each of the friendship suggestions between two nodes that are not already connected, for your algorithm you will use the score function below:

$$\boldsymbol{score(A,B) = \frac{|N_A \cap N_B|}{|N_A \cup N_B|}}$$

Again for jaccard coefficient scoring function you need to print a list containing the first 10 friend recommendations and their corresponding score, for nodes:

nodeID: 107 (high degree node)

nodeID: 1126

nodeID: 14

nodeID: 35

If there are fewer than 10 recommendations, print all the recommendations.

In the case of ties in friendship score you should output the node with the smallest nodeID (based on the ids given from the dataset).

4) Recommending friends using Adamic and Adar function

Adamic and Adar proposed a weighted function for scoring similarity between users (nodes) in a network. The more similar a user is to another, the more likely a friendship will occur in the network. In order to score each of the friendship suggestions between two nodes Adamic and Adar suggested the following weighted scoring function:

$$\boldsymbol{score(A,B) = \sum_{C \in N_A \cap N_B} \frac{1}{\log |N_C|}}$$

This scoring function evaluate the likelihood that user A is linked to user B, by summing the number of neighbors the two users have in common. Items that are unique to a few users are weighted more than commonly occurring items. The weighting scheme uses the inverse log frequency of their occurrence.

Again for Adamic and Adar scoring function you need to print a list containing the first 10 friend recommendations and their corresponding score, for nodes:

nodeID: 107 (high degree node)

nodeID: 1126

nodeID: 14

nodeID: 35

If there are fewer than 10 recommendations, print all the recommendations.

In the case of ties in friendship score you should output the node with the smallest nodeID (based on the ids given from the dataset).

5) Evaluation of the recommendation system

In order to evaluate your friendship recommendation system for each scoring function you will first need to examine if the scoring functions give different recommendations for specific users. Considering only those 40 Facebook users with an id that is a multiple of 100, you have to compute and print for every pair of functions (FoF-jaccard, Jaccard-Adamic, FoF-Adamic) the similarity percentage (number of common recommendation in the top-10 friend list) for every of the 40 users. Then you need to compute the average similarity between the algorithms.

As a second step of the evaluation you also need to implement the following algorithm that can forecast if the recommendations are going to be accepted from the users.

1. Randomly choose a real friend connection; call the two friends F1 and F2.
2. Remove their friendship from the graph.
3. Compute friend recommendations for F1 and F2 (10 recommendations).
4. Determine the rank of F1 in F2's list of recommended friends. Determine the rank of F2 in F1's list of recommended friends. If either of these does not exist (e.g., F1 is not recommended as one of F2's friends), discard the F1-F2 pair from your experiment. Otherwise, average these two numbers for each similarity function. The "rank" is also known as the "index" or "position".
5. Don't forget to put their friendship back in the graph.

For each scoring function, perform the above experiment 100 times.

Compute the average rank of the correct recommendation within the list of recommendations. (If the correct recommendation does not appear within the list of recommendations, ignore that trial when computing the average rank.)

To prevent different random choices from skewing your results, use the *same* random choices for all similarity functions. Another way of saying this is that each time you make a random choice, you should evaluate all three recommendation systems using that choice. Then go on to the next choice. Every run of your program will produce slightly different average ranks, but your program should be consistent in terms of which method is better.

5) Assignment Bonus

In order to gain up to 10% grade bonus for this assignment you can implement any different evaluation function that have comparable or better results to the three functions required for this assignment. Your results should be evaluated in the same way as the three above scoring functions. You can choose any topology based method described in the literature or create your own. Provide any additional experiments or bibliography that is needed.

Assignment handout:

1) A pdf file(or your jupyter notebook) describing in detail:
    a. Any processing and conversion was made to the original data and the reasons it was necessary. Specifically, you should indicate the process you used in order to transform the graph to undirected.
    b. The techniques you used in order to implement the three scoring functions and the evaluation method required for this assignment. Moreover for each scoring function you need to present the first 10 friend recommendations and their corresponding score, as determined in the description of each algorithm.
    c. For the evaluation of your results, first you have to present if the scoring functions give different recommendations for specific users and compute the similarity percentage as well as the average similarity between the algorithms. Then using the evaluation method described above, you should evaluate which scoring function recommends the best links. Present the tests you have done and justify your conclusions. The presentation should contain charts with comparative measurements and comment on these results.
2) The program/script you implemented for your recommendation system. Implementation can be done in any programming language and should be accompanied by the necessary comments and remarks.
3) The pdf with the presentation of your work as well as the required programs/scripts should be uploaded to moodle until the assignment deadline.