Assignment 1: Friends Recommendation Engine

Natasa Farmaki - DS3517018

Stratos Gounidellis - DS3517005

Course: Data Mining Techniques

Professor: Y.Kotidis

Assistant: I.Filippidou

Athens, March 2018

INTRODUCTION

Facebook is a popular free social networking website that allows registered users to create profiles, upload photos and videos, send messages and keep in touch with friends, family and colleagues. A key characteristic of social networks is the ability to create connections between users and it is a main source of information for data analysts. It is also the main source of attractiveness of new users and constitutes a motivation to increase the activity - use of the application. Constitutes, Facebook pays a lot of attention in the creation of an efficient link prediction system that encourages the creation of new “friendships”.

The aim of this assignment is to implement different metrics to construct a recommendation system and evaluate their efficiency. More specifically, in order to create the friend recommendation engine topology based methods as proposed in the literature were used. The representation of Facebook environment will be done using a graph that has users as nodes and “friendships” as edges.

The python code is available in the following link: https://thinkingtea.github.io/TEArepo/

LINK PROJECTION

Recommendation systems are implemented using link prediction algorithms. The aim of these methods is to predict the likelihood of a future association between two nodes, knowing that there is no association between the nodes in the current state of the graph. Link prediction task can be divided into two categories. The first category is to predict that the new link will appear in future time. The second category is to forecast hidden unknown link in the space.

The easiest framework of link prediction algorithm is based on the similarity of the algorithm. For every pair of nodes, we assign a similarity function, which is the similarity function between nodes. Then sorting the nodes pair in accordance with the function values from the largest to smallest, the greater the value of the similarity function, the greater the probability of the link in the nodes.

Link prediction however should not only be seen as a tool useful for friends’ recommendation in social media. It could also b proved useful in detecting interactions among proteins, and consequently it could be used in general in bioinformatics. Furthermore, in marketing and e-commerce sector could also be used in the sense that suitable products or services could be identified for the relevant – suitable customers.

SIMILARITY METRICS

Therefore, the key decision now is what method will we use in order to produce te similarity function. The main approaches used for this purpose are Friend-of-Friend and Path-based methods.

Friend-of-Friend methods

FoF Algorithm:
The FoF algorithm derives from the fact that if two users in the social network share many common friends, they may have a great chance of becoming friends in the future. This algorithm is also known as “Common-Neighbors”. The similarity function for this method is the number of common friends of a user A and a user B.

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

As social networks continue to grow, the initial Common-Neighbors model proliferated into several improved algorithms, such as Jaccard coefficient and Adamic /Adar.

Jaccard coefficient

The Jaccard coefficient measures the probability that both A and B have a feature f, for a randomly selected feature f that either A or B has. If we take “features” here to be friends, then this measure captures the intuitively appealing notion that the proportion of friends of A, who are also friends with B (and vice versa) is a good measure of the similarity of A and B. In other words, 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.

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

Adamic and Adar Function

This measure refines the simple counting of common features between A, B by weighting rarer features more heavily. Thus, the common neighbor of a pair of nodes with few neighbors contributes more to the Adamic and Adar score value than this with large number of relationships. In real-world social network, if a common acquaintance of two people has more friends, then it is less likely that he or she will introduce the two people to each other than in the case when he or she has only few friends.

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

Path-based methods

Differing from the neighbor-based FoF approach, calculating the shortest path is the basic idea of the Path-based methods.

In this assignment, we additionally used a simple shortest-path similarity measure assuming that the distance between two connected nodes is equal to one (1). We also tried a combination of the two approaches.

Combination of common neighbors and distance

For the calculation of this measure, we divided the number of common neighbors with the distance between the node of user A and node of user B.

Combination of Jaccard Similarity and distance

For the calculation of this measure we divided Jaccard coefficient with the distance between the node of user A and node of user B.

Except from these methods, we calculated a **baseline method** in order to observe the accuracy of extracting friend recommendations randomly. Note that all approaches calculate similarity metrics which means that the greater the measure the more probable it is for the two nodes to be connected. The only exception is the simple path-based method, because we choose the friend/node with the smallest distance. Additionally, in case of ties in friendship score the node with the smallest nodeID is shown first.

DATA DESCRIPITION AND GRAPH CREATION

The data used in this assignment are Facebook data that were collected from survey participants using this Facebook app. The dataset consists of 'circles' (or “friends” lists') from Facebook. Facebook data was collected from survey participants using this Facebook app. 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 undirected (each edge counts as friendship for both nodes).

In order to construct and manipulate the graph we used Networkx library in python. We first converted the graph to undirected. Then, we created functions that calculate the top k friend recommendations using a baseline algorithm, common neighbors' algorithm, Jaccard coefficient, Adamic and Adar score, shortest path distance, a combination of Jaccard and distance, a combination of the number of common neighbors and the distance. To make the calculation of the distances faster we created an adjacency matrix containing all the distances between the nodes.

MEASURES TO EVALUATE METRICS PERFORMANCE

The evaluation of the recommendation system will be done in two parts. In the first part, we compare the similarity of the different methods applied. In the second part, we evaluate which scoring function recommends the best links.

Similarity of the metrics

In order to compare the performance of the methods we first need to examine if the scoring functions give different recommendations for specific users. Considering forty (40) Facebook users with an id that is a multiple of one hundred (100), we compute and print the number of Facebook users who have the same first 10 friend recommendations under each scoring function, and the number of Facebook users who have different first ten (10) friend recommendations under the each scoring function.

Further to this for every pair of functions, we compute the similarity percentage of the recommended friend lists for the forty (40) users. That is for each user we compute the number of common recommendations between two methods divided by the number of recommendations for both method in total. Then to calculate the average similarity between the algorithms, we get the average for every pair of methods by dividing by forty (40) in order to get the similarity between each pair.

Evaluate Performance

To evaluate the actual performance of a method we implement the following steps for each method one hundred times for each method:

1. We randomly choose a real friend connection (F1, F2) and remove the friendship from the graph.
2. We then compute one hundred (100) friend recommendations for F1 and F2.
3. We determine the rank of F1 in F2's list of recommended friends and the rank of F2 in F1's list of recommended friends. If either of these does not exist, we discard the F1-F2 pair from the experiment. Note that when a pair is discarded it is removed from all the methods not just from the one it failed. Furthermore, the total number of pairs taken for the experiment are one hundred (100) in total after discarding the ones, which failed.
4. If the pair is not removed, we average these two numbers for each similarity function and thus we get the "rank", also known as "index" or "position".
5. Finally, we re-add the connection removed in the first step back in the graph.

After this process is finished, we calculate and print as a table the average rank of these 100 trials for every method. The process is implemented for five (5) and twenty (20) times and we compute the average and the standard deviation for the two experiments in order to get more accurate results.

EXPERIMENTAL RESULTS

Indicative outputs

Common neighbors

Current facebook user: 107

	nodeid	number_of_common_neighbors
0	513	19
1	400	18
2	559	18
3	373	17
4	492	17
5	500	17
6	378	16
7	436	16
8	431	15
9	514	15

Current facebook user: 1126

	nodeid	number_of_common_neighbors
0	916	113
1	1238	103
2	1750	99
3	1230	98
4	1004	94
5	1791	94
6	1530	90
7	1172	89
8	1570	85
9	1597	84

Current facebook user: 14

	nodeid	number_of_common_neighbors
0	2	9
1	17	9
2	140	9
3	111	8
4	137	7
5	162	7
6	19	6
7	333	6
8	44	5
9	243	4

Current facebook user: 35

	nodeid	number_of_common_neighbors
0	46	2
1	68	2
2	99	2
3	131	2
4	175	2
5	177	2
6	225	2
7	227	2
8	278	2
9	321	2

Jaccard Coefficient

Current facebook user: 107

	nodeid	jaccard_coef
0	513	0.017040
1	400	0.016364
2	559	0.016216
3	492	0.015554
4	500	0.015302
5	373	0.015152
6	436	0.014692
7	378	0.014665
8	515	0.013749
9	514	0.013711

Current facebook user: 1126

	nodeid	jaccard_coef
0	916	0.487069
1	1750	0.440000
2	1230	0.435556
3	1530	0.422535
4	1004	0.419643
5	1238	0.417004
6	1172	0.400901
7	1791	0.400000
8	1789	0.374429
9	1597	0.371681

Current facebook user: 14

	nodeid	jaccard_coef
0	2	0.562500
1	140	0.529412
2	17	0.473684
3	162	0.437500
4	111	0.380952
5	333	0.352941
6	44	0.312500
7	137	0.291667
8	19	0.240000
9	243	0.210526

Current facebook user: 35

	nodeid	jaccard_coef
0	321	0.666667
1	11	0.500000
2	12	0.500000
3	15	0.500000
4	18	0.500000
5	37	0.500000
6	43	0.500000
7	74	0.500000
8	114	0.500000
9	209	0.500000

Adamic and Adar function

Current facebook user: 107

	nodeid	adamic_adar_similarity
0	513	9.724953
1	400	9.307010
2	559	8.818876
3	500	8.528682
4	492	8.373501
5	373	8.368363
6	378	7.852810
7	436	7.813676
8	524	7.441609
9	514	7.437235

Current facebook user: 1126

	nodeid	adamic_adar_similarity
0	916	53.556246
1	1238	48.753746
2	1750	46.496906
3	1230	45.811896
4	1004	44.041808
5	1791	43.959211
6	1530	41.808500
7	1172	41.579656
8	1570	40.006415
9	1597	39.146933

Current facebook user: 14

	nodeid	adamic_adar_similarity
0	2	6.843324
1	140	6.736628
2	17	6.736628
3	111	5.964252
4	162	5.214517
5	137	5.073642
6	333	4.598281
7	19	4.190559
8	44	3.471818
9	243	2.722615

Current facebook user: 35

	nodeid	adamic_adar_similarity
0	46	1.320278
1	68	1.320278
2	99	1.320278
3	131	1.320278
4	175	1.320278
5	177	1.320278
6	225	1.320278
7	227	1.320278
8	278	1.320278
9	321	1.320278

Graph Distance similarity

Current facebook user: 107

	nodeid	graph_distance_similarity
0	1	2.0
1	2	2.0
2	3	2.0
3	4	2.0
4	5	2.0
5	6	2.0
6	7	2.0
7	8	2.0
8	9	2.0
9	10	2.0

Current facebook user: 1126

	nodeid	graph_distance_similarity
0	0	2.0
1	58	2.0
2	136	2.0
3	171	2.0
4	348	2.0
5	353	2.0
6	363	2.0
7	366	2.0
8	376	2.0
9	389	2.0

Current facebook user: 14

	nodeid	graph_distance_similarity
0	1	2.0
1	2	2.0
2	3	2.0
3	4	2.0
4	5	2.0
5	6	2.0
6	7	2.0
7	8	2.0
8	9	2.0
9	10	2.0

Current facebook user: 35

	nodeid	graph_distance_similarity
0	1	2.0
1	2	2.0
2	3	2.0
3	4	2.0
4	5	2.0
5	6	2.0
6	7	2.0
7	8	2.0
8	9	2.0
9	10	2.0

Weighted distance (FoF) similarity

Current facebook user: 107

	nodeid	graph_distance_similarity
0	513	9.5
1	400	9.0
2	559	9.0
3	373	8.5
4	492	8.5
5	500	8.5
6	378	8.0
7	436	8.0
8	431	7.5
9	514	7.5

Current facebook user: 1126

	nodeid	graph_distance_similarity
0	916	56.5
1	1238	51.5
2	1750	49.5
3	1230	49.0
4	1004	47.0
5	1791	47.0
6	1530	45.0
7	1172	44.5
8	1570	42.5
9	1597	42.0

Current facebook user: 14

	nodeid	graph_distance_similarity
0	2	4.5
1	17	4.5
2	140	4.5
3	111	4.0
4	137	3.5
5	162	3.5
6	19	3.0
7	333	3.0
8	44	2.5
9	243	2.0

Current facebook user: 35

	nodeid	graph_distance_similarity
0	46	1.0
1	68	1.0
2	99	1.0
3	131	1.0
4	175	1.0
5	177	1.0
6	225	1.0
7	227	1.0
8	278	1.0
9	321	1.0

Weighted distance (Jaccard) similarity

Current facebook user: 107

	nodeid	distance_jaccard_similarity
0	513	0.008520
1	400	0.008182
2	559	0.008108
3	492	0.007777
4	500	0.007651
5	373	0.007576
6	436	0.007346
7	378	0.007333
8	515	0.006874
9	514	0.006856

Current facebook user: 1126

	nodeid	distance_jaccard_similarity
0	916	0.243534
1	1750	0.220000
2	1230	0.217778
3	1530	0.211268
4	1004	0.209821
5	1238	0.208502
6	1172	0.200450
7	1791	0.200000
8	1789	0.187215
9	1597	0.185841

Current facebook user: 14

	nodeid	distance_jaccard_similarity
0	2	0.281250
1	140	0.264706
2	17	0.236842
3	162	0.218750
4	111	0.190476
5	333	0.176471
6	44	0.156250
7	137	0.145833
8	19	0.120000
9	243	0.105263

Current facebook user: 35

	nodeid	distance_jaccard_similarity
0	321	0.333333
1	11	0.250000
2	12	0.250000
3	15	0.250000
4	18	0.250000
5	37	0.250000
6	43	0.250000
7	74	0.250000
8	114	0.250000
9	209	0.250000

Metrics' similarity comparison

Metrics	Similarity percentage
FoF - Jaccard	0.4460914880613022
Jaccard - Adamic	0.45298849327487084
FoF - Adamic	0.8567927170868345
FoF - Distance	0.00625
FoF - Weighted Distance (FoF)	0.025
FoF - Weighted Distance (Jaccard)	0.020454545454545454
Jaccard - Distance	0.00625
Jaccard - Weighted Distance (FoF)	0.020454545454545454
Jaccard - Weighted Distance (Jaccard)	0.025
Adamic - Distance	0.00625
Adamic - Weighted Distance (FoF)	0.025
Adamic - Weighted Distance (Jaccard)	0.020454545454545454
All metrics (average similarity)	0.15924886123222035

In the forty trials, the six algorithms never return the same set of recommendations.

Algorithms' performance evaluation

	algorithm	average_rank	stdev_rank
0	Common Neighbors (FoF)	13.85875	1.698487
1	Jaccard Coefficient	13.54875	1.520829
2	Adamic and Adar function	13.03550	1.644221
3	Graph Distance	1.00000	0.000000
4	Weighted Graph Distance (FoF)	2.34900	0.375991
5	Weighted Graph Distance (Jaccard)	2.49900	0.511871
6	Baseline	97.70000	8.845055

As it is shown in the above figure, the path-based algorithms seem to outperform. According to that specific evaluation method and for this specific network the graph distance metric seems to be the best metric for the friend recommendation engine as it has the highest average rank and lowest standard deviation. It should also be mentioned that for the baseline approach since the experiment failed most of the times we manually set the rank to one hundred (100), if there are one hundred fails in each iteration. This can become clearer through the source code.

CONCLUSIONS

The link prediction problem in social networks has many potential applications. Examples include predicting forthcoming links from users to items in recommendation systems, friendship suggestions and discovering spurious connections. Here in order to create the friend recommendation engine topology based methods as proposed in the literature were used.

Our experiments showed that the path-based algorithms perform better. They present higher average rank in the respective evaluation measure and lower standard deviation. However, those methods are more computationally intensive as an adjacency matrix should be computed first and then a distance matric should be computed. Therefore, this could constitute a drawback in larger graphs.