PROJECT TWO

Social Network Mining

Group Members:

Junyu Wang (UID: 405729050)

Zhaoxi Yu (UID: 005432230)

Wayne Zhang (UID: 905845716)

UCLA

ECE 232: Large Scale Social and Complex Networks

Contents

List of Figures & Tables

1 Facebook Network

Question 1

The Facebook network is shown below.

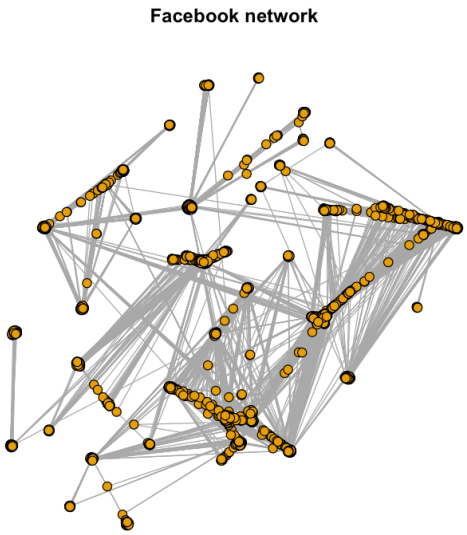


Figure 1-1 Generated Facebook Network

Question 1.1

The Facebook network has 4039 nodes and 88234 edges.

Question 1.2

The Facebook network is always connected.

Question 2

The diameter of the Facebook network is 8.

Question 3

The figure below shows the degree distribution of the Facebook network. The average degree of this network is 43.691.

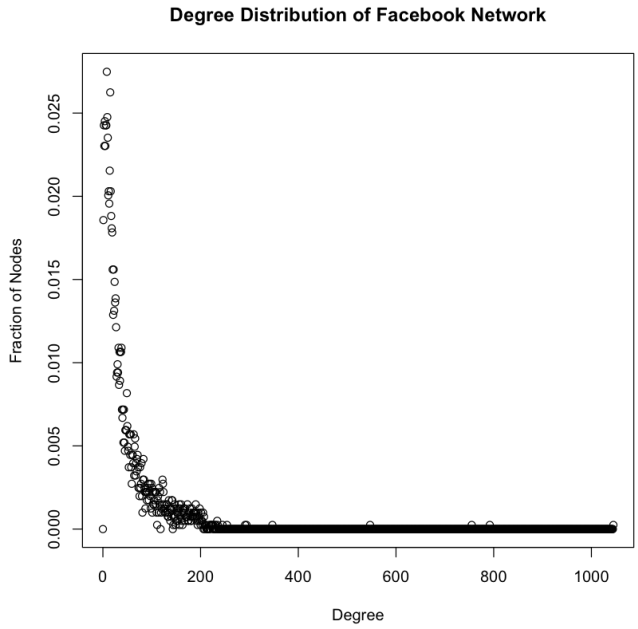


Figure 1-2 Degree Distribution of the Facebook Network

Question 4

The log-log scale degree distribution of the Facebook network is plotted below.

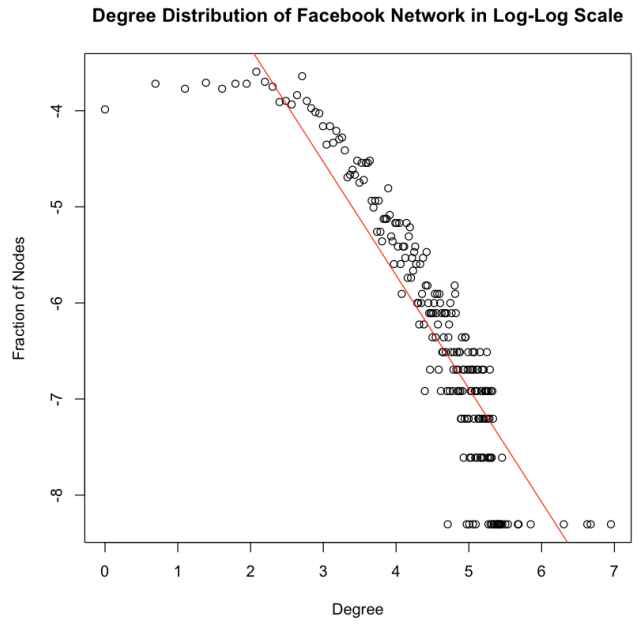


Figure 1-3 Degree Distribution in Log-Log Scale (base e)

We applied linear regression on the log-log data, the estimated slope is -1.1802.

Question 5

The following figure shows the personalized network of user 1. This graph has 348 nodes and 2866 edges.

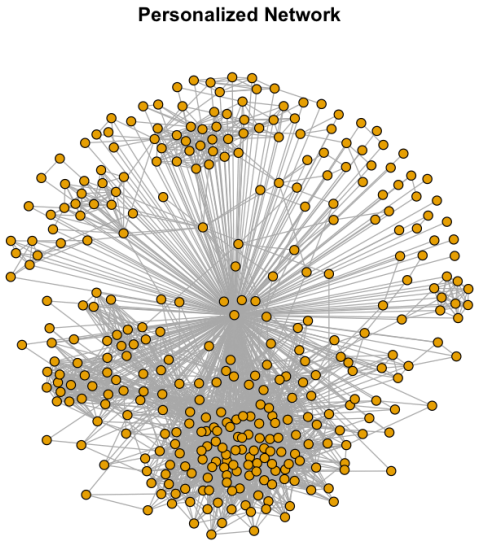


Figure 1-4 Personalized Network of user 1

Question 6

The diameter of this personalized network with ID = 1 is 2. The trivial upper bound for the diameter is 2 and the lower bound is 1.

Question 7

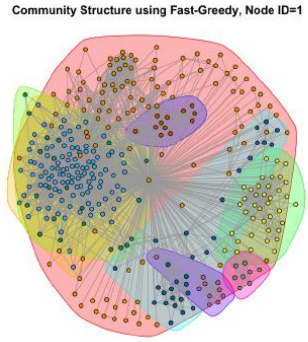
The diameter of graph is defined as the maximum distance between the pair of vertices. In Question 6, the diameter of the personalized network of user 1 is 2, which means that there is a pair of nodes that are not directly connected and the distance between them is 2. Furthermore, this is the longest distance in the graph. The realistic meaning of the diameter in this Facebook network is that there exist two users who are not mutual friends, but they share the common friend (center user). In other words, not all users in the network knows each other.

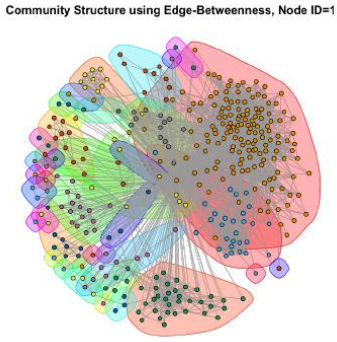
In the case that the diameter is equal to 1, the personalized network is a complete graph, every pair of distinct vertices is connected by an edge, which means every user in this network knows each other.

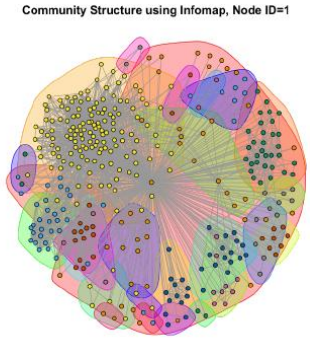
Question 8

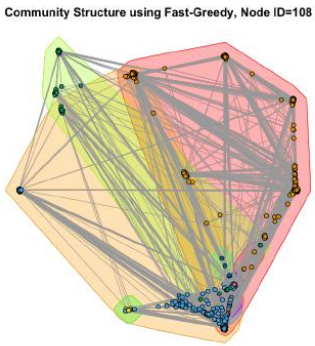
A core node is defined as the nodes that have more than 200 neighbors. There are 40 core nodes in the Facebook network. The average degree of those core nodes is 279.375.

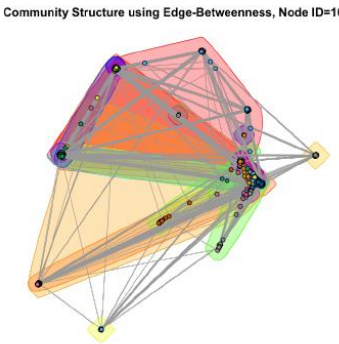
Question 9

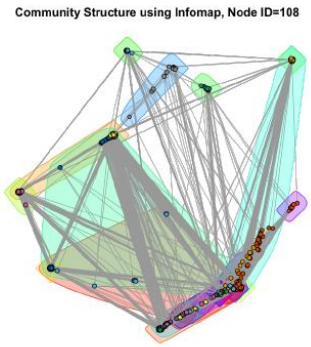


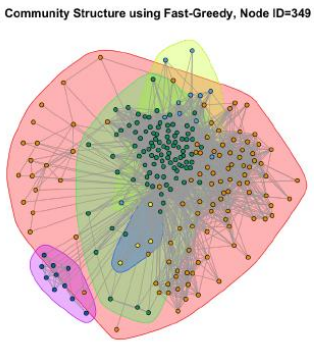


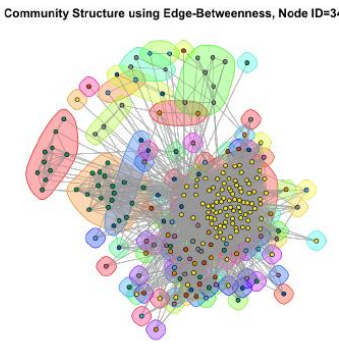


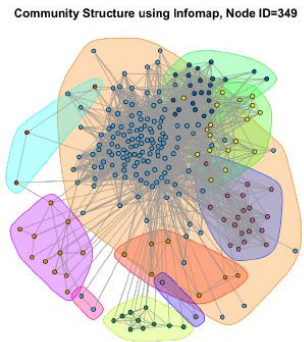












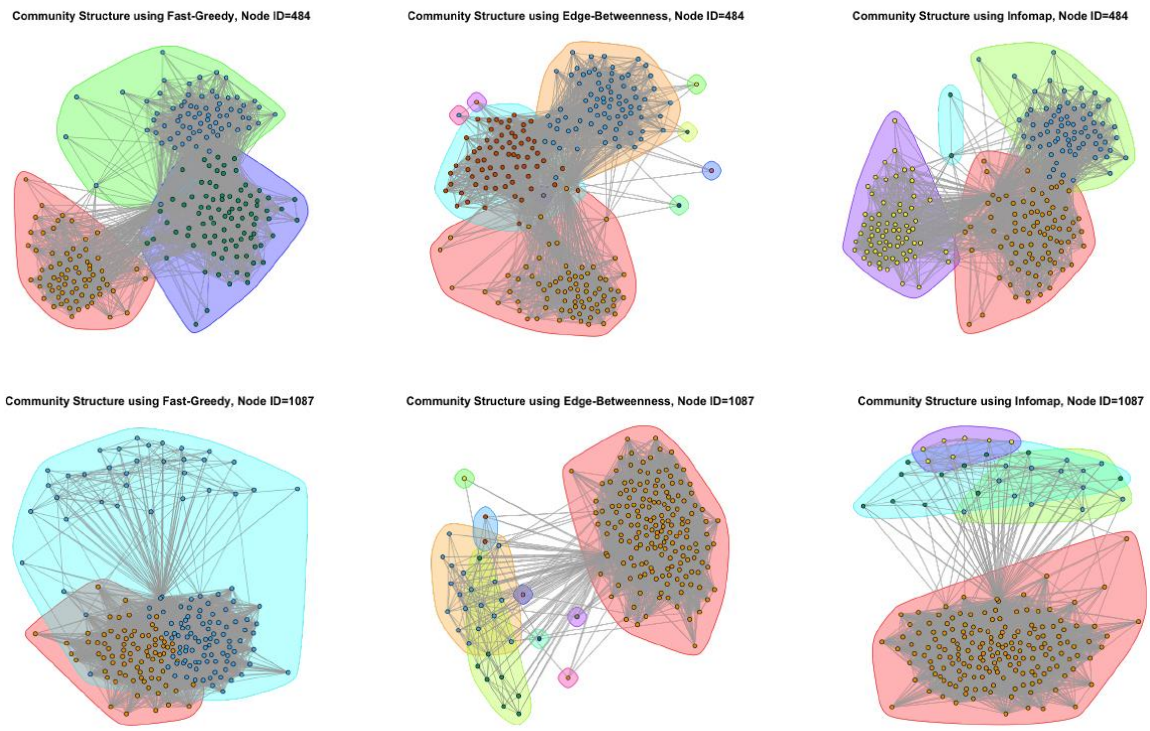


Figure 1-5 Community structures for node 1, 108, 349, 484, 1087

The 15 figures above respectively show the community structures of the personalized networks of node ID 1, 108, 349, 484, 1087 using Fast-Greedy, Edge-Betweenness, and Infomap community detection algorithms.

The following table reports the modularity scores of each network via different algorithms.

Table 1-1 Modularity scores using 3 community detection algorithms

Core Node ID	Fast-Greedy	Edge-Betweenness	Infomap
1	0.413101	0.353302	0.389118
108	0.435929	0.506755	0.508249
349	0.251715	0.133528	0.095464
484	0.507002	0.489095	0.515643
1087	0.145531	0.027624	0.026905

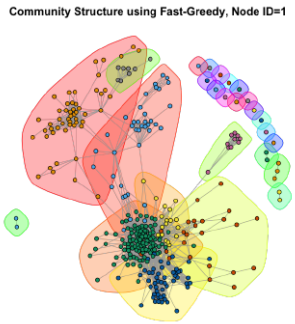
From the figures and table, we can find that:

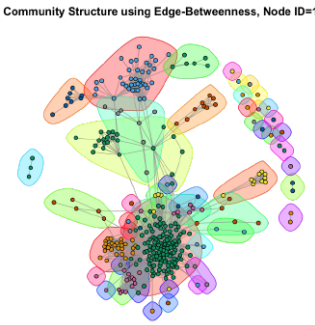
1. The core node 484 has the largest average modularity, while the core node 1087 has the smallest modularity on average.
2. Fast-Greedy algorithm performs best on average among 3 community detection algorithms.

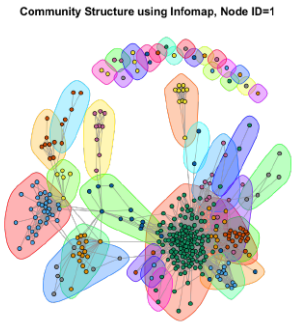
3. The performances of Edge-Betweenness and Infomap is similar in most cases.

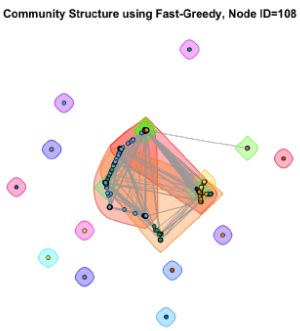
Question 10

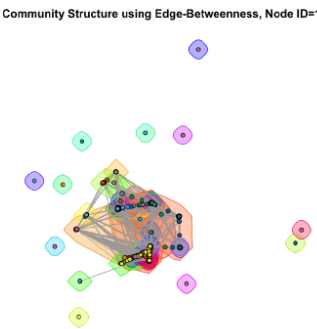
The 15 plots below respectively show the community structures of the personalized networks of node ID 1, 108, 349, 484, 1087 using Fast-Greedy, Edge-Betweenness, and Infomap, but without core nodes.

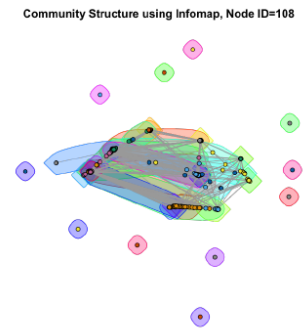


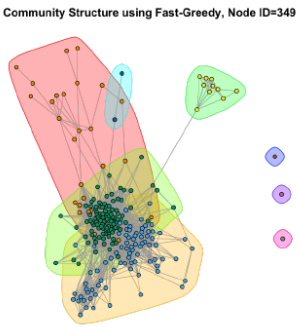


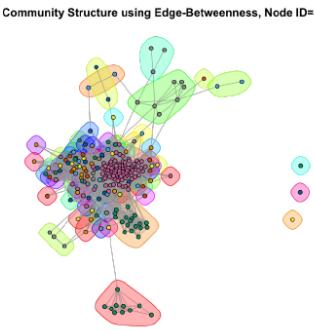


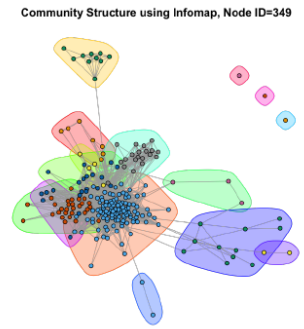












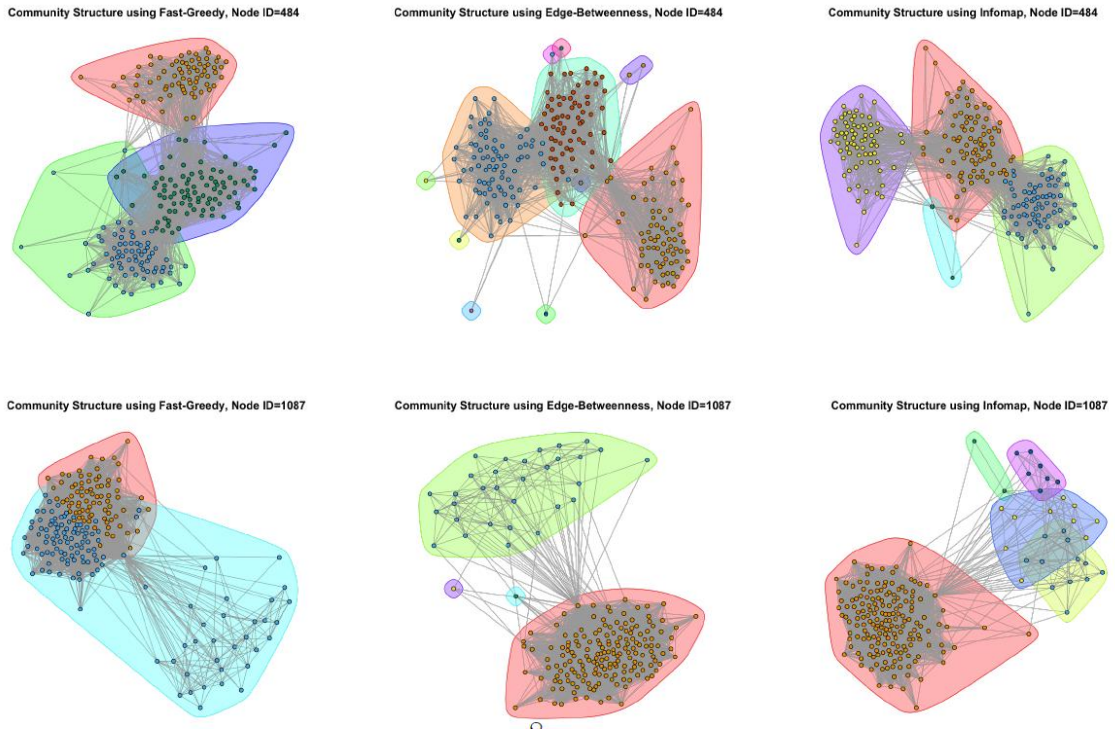


Figure 1-6 Community structures for node 1, 108, 349, 484, 1087

The following table records the modularity scores of the community structures of the modified personalized networks with the modularity scores of the community structures of the personalized networks in Question 9.

Question 11

The expression should be (1.1). Because in the personalized network of a core node, every friend (neighbor) of a non-core node is also a friend of the core node, or the core node itself. And every node in this personalized network is a friend of the core node, or the core node itself. According to the definition, the Embeddedness of a node is defined as the number of mutual friends a node shares with the core node. So, the number of mutual friends is the number of the neighbor of the non-core node minus one (core node itself).

$$Embeddedness = Degree\,(noncorenode\,) - 1 \tag{1.1}$$

Question 12

To calculate the distribution of embeddedness, we use (1.1) to calculate every non-core node's embeddedness and store all values in a list. To calculate the distribution of

dispersion, we iterate every node in the graph to collect dispersion one by one. For each node, we perform the following 2 steps. 1) Find its mutual friends with the core node, which are its neighbors. 2) Do a nested for loop, to calculate every two nodes' distance and sum them up. Then we can get the distribution. And we perform these two calculations on every personalized graph.

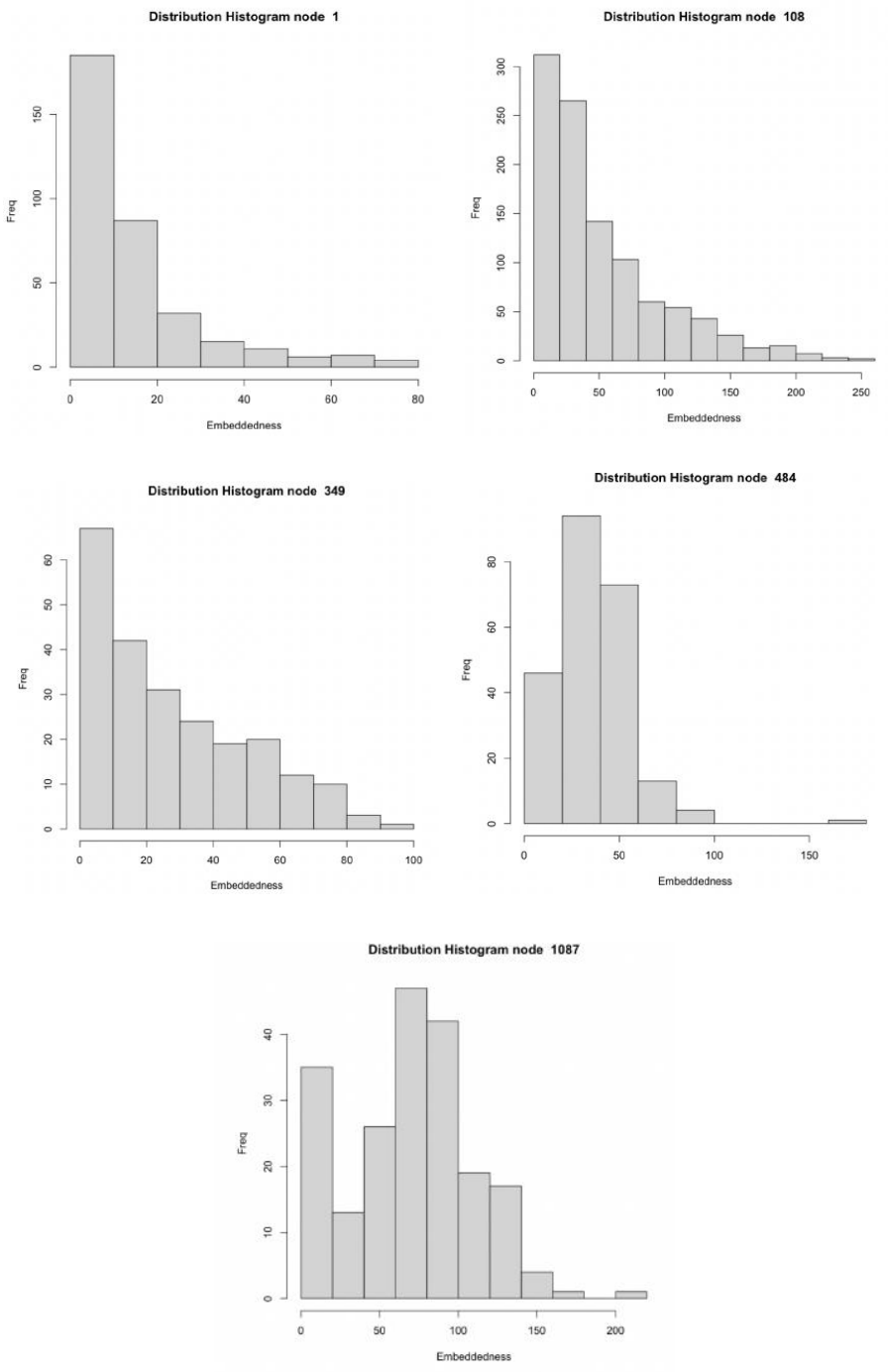


Figure 1-7 Embeddedness for node 1, 108, 349, 484, 1087

Question 13

For this problem, we create a function to calculate dispersion of every node in each personalized graph. So, we get the index of the node with max dispersion. Then we run fast greedy algorithm to plot the community structure, core node, and the node with max dispersion.

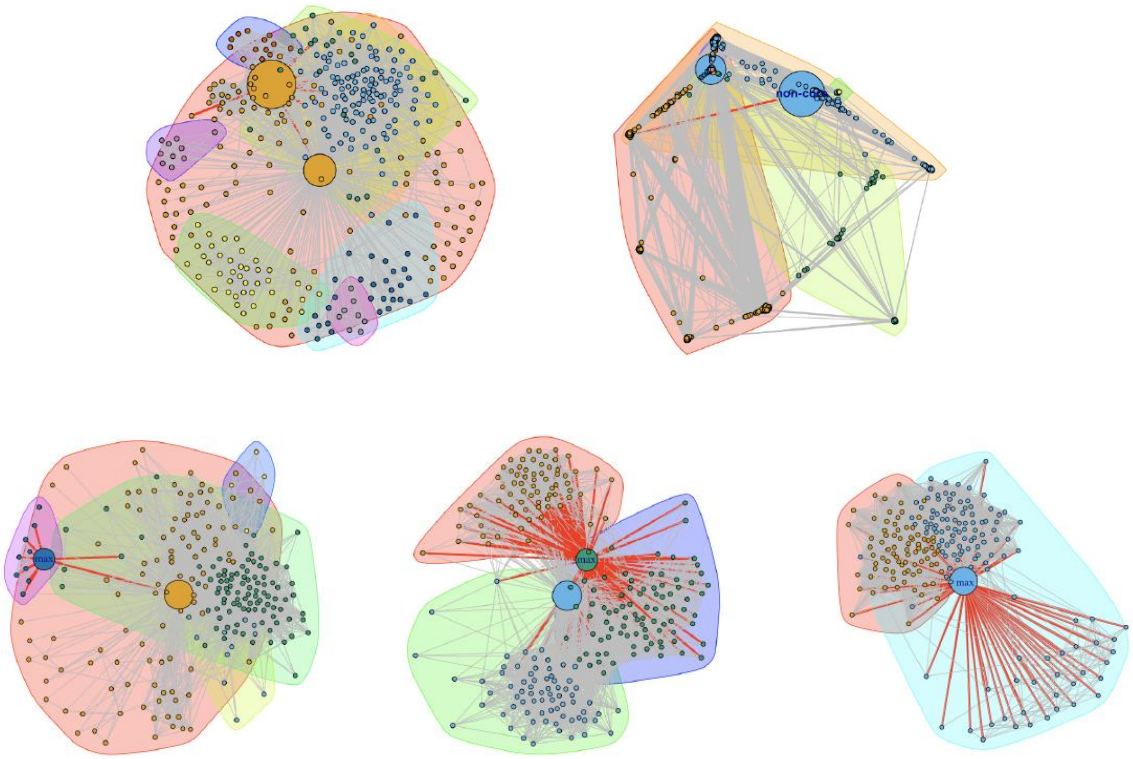


Figure 1-8 Highlighted community structure for Node 1, 108, 349, 484, 1087

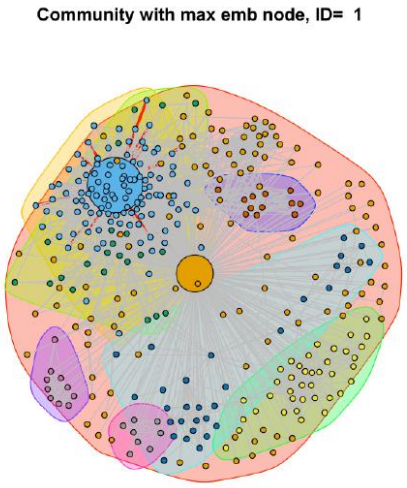
Question 14

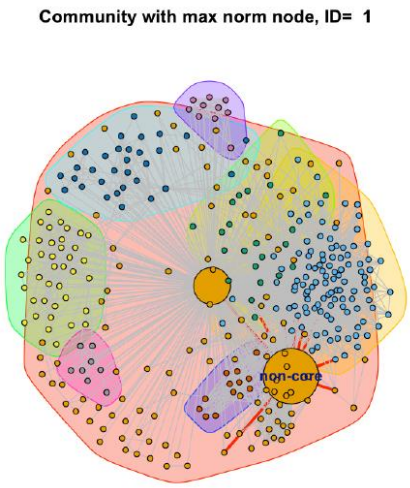
Like Question 13, we create functions to calculate the embeddedness and dispersion/embeddedness for each node. Then we get the index of each node and draw the graph. On each row below, the left graph contains the max embeddedness node, and the right graph contains the max embeddedness/dispersion node.

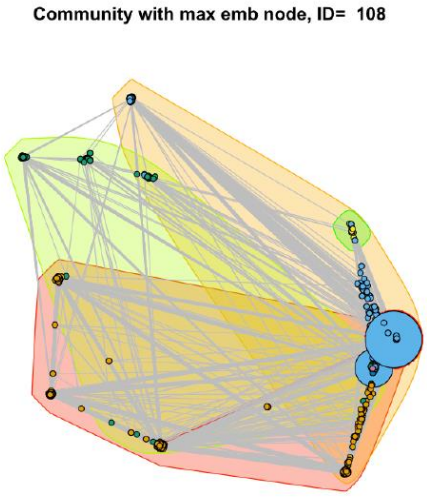
The report of the results has been shown in . Max embedded records the index of node with max embeddedness. Max disp records the index of the node with max dispersion. Max dispersion/embedded records the index of the node with max dispersion/embeddedness.

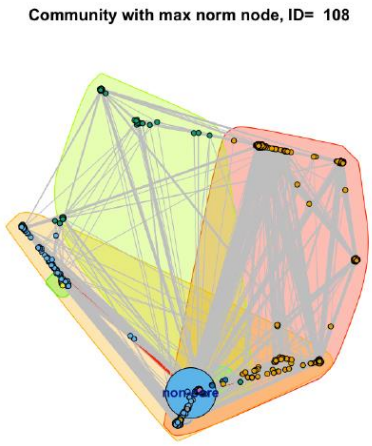
Table 1-2 Report of Question 13 & 14 results

Core Node id	max embedded	max disp	max dispersion/embedded
1	node 57	node 6	node 6
108	node 1889	node 172	node 172
349	node 377	node 199	node 199
484	node 108	node 108	node 108
1087	node 108	node 108	node 108









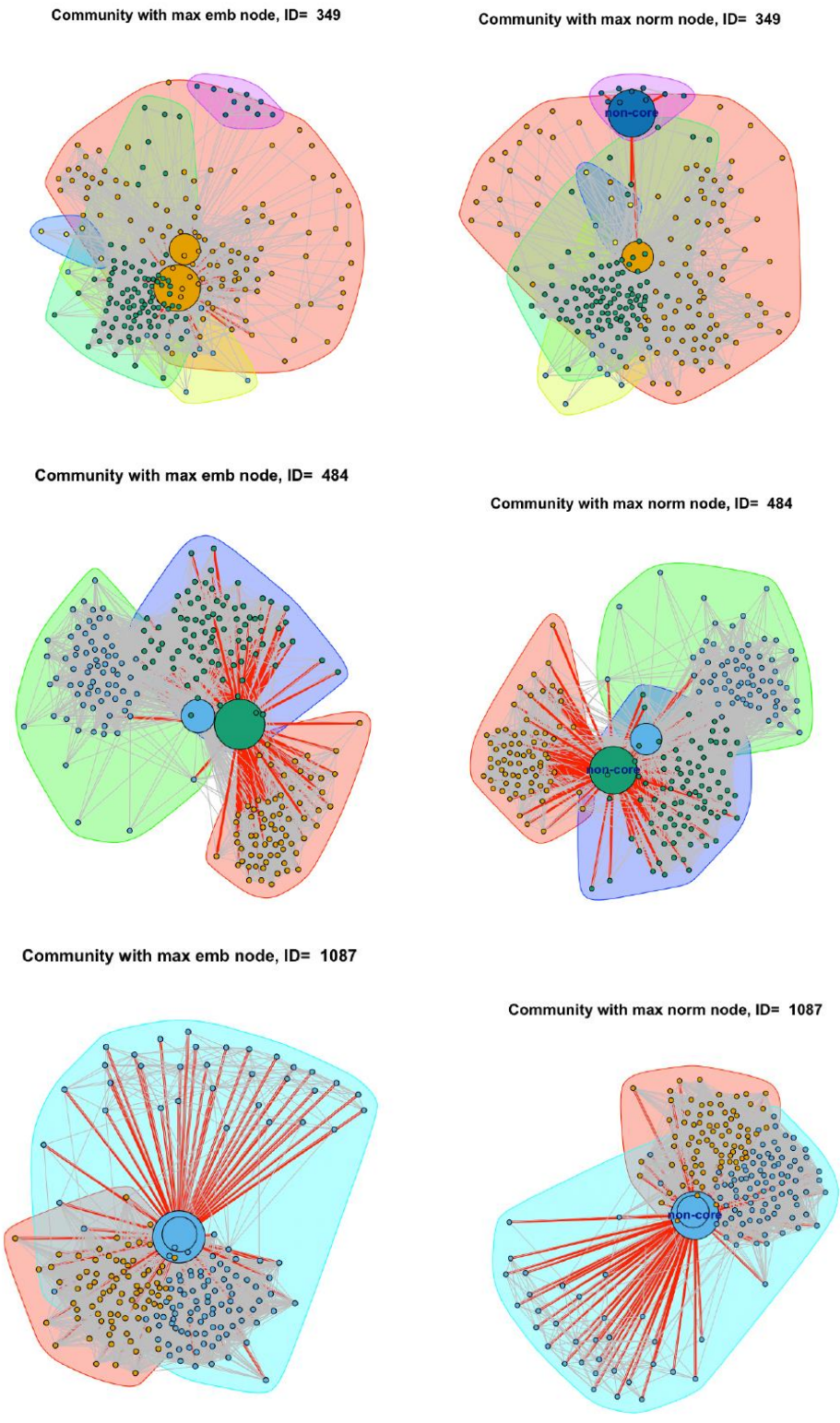


Figure 1-9 Highlighted community structure for Node 1, 108, 349, 484, 1087

Question 15

Both embeddedness and dispersion could be measurements of the closeness of the tie between two nodes. However, they are from different dimensions. If two nodes have a high embeddedness value, it means they have a lot of mutual friends in this personalized network. If two nodes have a high dispersion value, it means their mutual friends have long distances among them. The fact is that they are both important metrics to evaluate the "closeness".

In Question 13, we try to find the node with max embeddedness. The node that we found is the node that shares the most mutual friends with the core node. In the Facebook network, it seems like this node could be a popular person in this connection graph because this is a personalized graph of the core node, and this node has a lot of direct connections with other nodes in this graph. However, from this result, we cannot tell the closeness of the relationship between this node and the core node as embeddedness does not reveal direct information about this.

For the dispersion, it measures the distance between each pair of nodes in their mutual friends. In real life, if two people have many mutual friends, and their mutual friends tend to have long-distance, it means that this node and the core node may have a close relationship. Let's say, if two people are in a close relationship, they tend to know each other's friends, but their friends may not get a chance to know each other. Therefore, two nodes with high dispersion tend to have a close relationship. Then in Question 14, we calculate the node with dispersion/embeddedness value. The value is proportional to the dispersion and inversely proportional to embeddedness. As a result, according to our above analysis, the non-core node with the highest dispersion/embeddedness tends to have the closest relationship with the core node in this personalized graph.

Question 16

According to the personalized graph code, we select all the nodes with degree 24. The length of Nr is 11.

We use each different algorithm and run 10 iterations on each node. Then we collect data in table below. It shows different average accuracy of each node.

Table 1-3 Results of different friend recommending algorithm

	common	jaccard	adamic
	<dbl>	<dbl>	<dbl>
497	0.3566667	0.1365584	0.2538745
579	0.9909091	0.9547619	0.9857143
601	0.9365476	0.8386111	0.8606410
616	0.8529365	0.8316667	0.8004762
619	0.3976190	0.5025000	0.4620238
628	0.9875000	0.9246032	1.0000000
644	0.9421032	0.8622222	0.8371789
659	0.9857143	0.9888889	0.9466667
660	1.0000000	0.9708333	1.0000000
662	0.8854365	0.9080952	0.8841234
663	0.9732143	0.9450000	0.9666667

Then we calculate the average value by each row (each algorithm) and get the results:

1. Common Neighbors measure: 0.846240653286108
2. Adamic Adar measure: 0.817942310215038
3. Jaccard measure: 0.805794634658271

We can make the conclusion that Common Neighbors measure is the best algorithm based on given scenario.

2 Google+

Question 18

There are 57 personal networks in total. We go through all the .circles files in the Google+ dataset and count the number of files with more than two lines. User with more than two circles owns a personal network.

Question 19

From this point, we focus on three user nodes, we label them as:

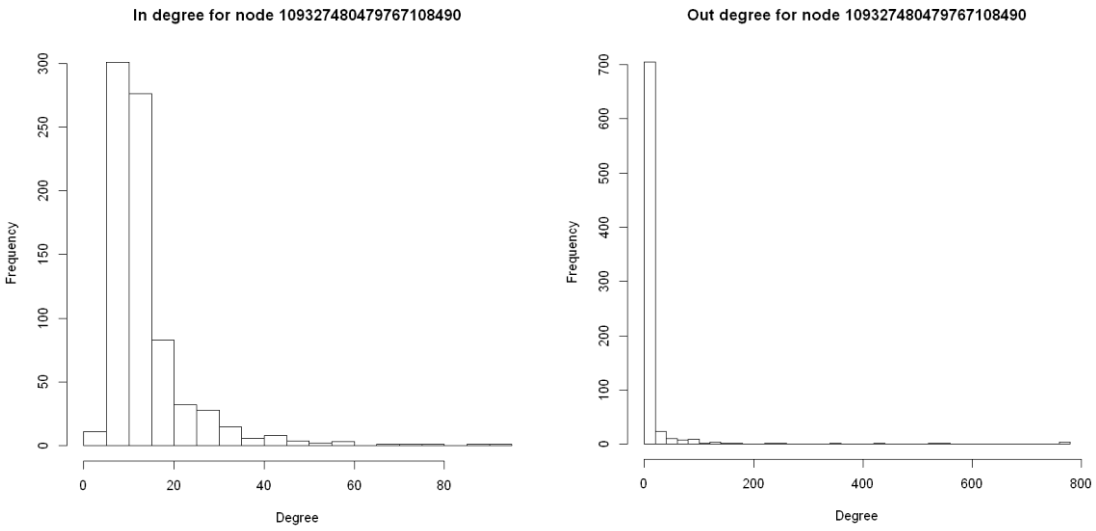
Node A	109327480479767108490
Node B	115625564993990145546
Node C	101373961279443806744

We design a *get_graph* function to read all the .edges files and construct graphs of ego network for different users. Figure 2-1 shows the in-degree and out-degree for 3 target user nodes. Mathematicaly, the in-degree of node i in a network is given by

$$\deg(i)^{\text{in}} = \sum_j A_{ij} \tag{2.1}$$

Where A_{ij} is the node-node adjacency matrix. Similarly, the out-degree is given by

$$\deg(i)^{out} = \sum_j A_{ji} \tag{2.2}$$



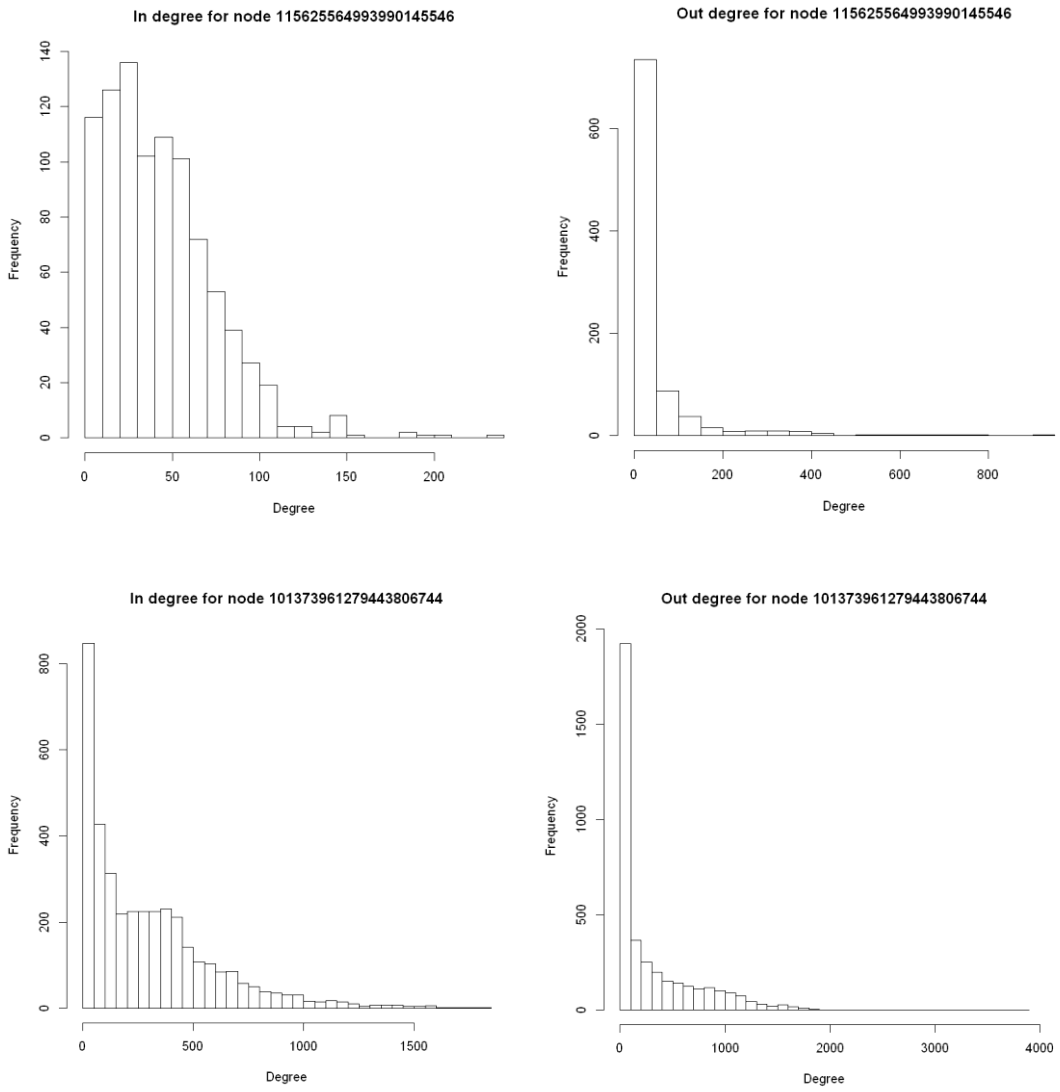


Figure 2-1 In and out-degree distributions of 3 personalized networks from Google+ dataset

From Figure 2-1 we can see that all the degree distributions roughly follow the power-law curve. However, from the y axis we can see that node C's ego network has the most edges, followed by node A. The out-degree distributions of the three nodes are more like each other than the in-degree distributions. Within each node, the in and out degree are significantly different.

Question 20

Using the *cluster_walktrap* function from *igraph*, we extract the community structure of each personal network using walktrap community detection algorithm. We plot the three clusters and give their modularity as follows:

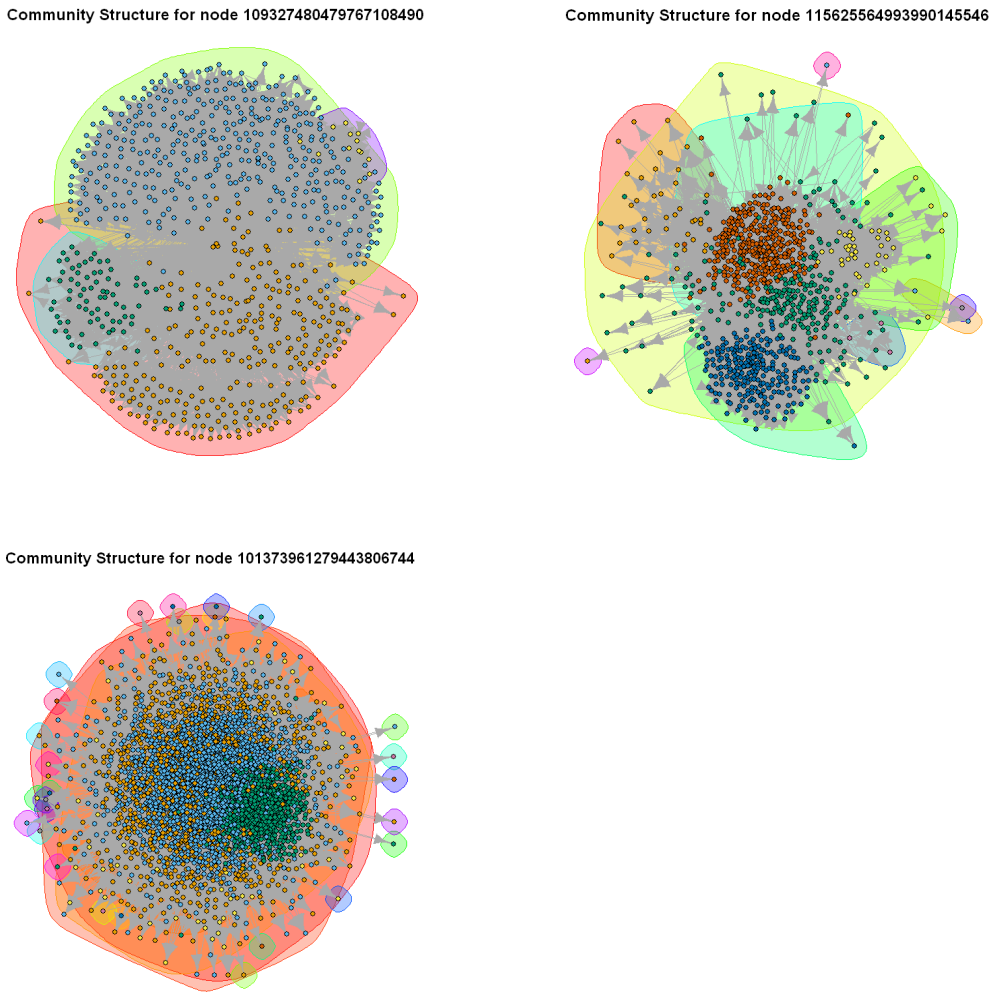


Figure 2-2 Community structure of three personalized networks from Google+ dataset

Node A	109327480479767108490	M = 0.252765
Node B	115625564993990145546	M = 0.319473
Node C	101373961279443806744	M = 0.191090

We can see that node C has the lowest modularity score, meaning that the network is least capable of being divided into densely packed modules with strong intra-connections and sparse inter-connections among communities. This is clearly visible from Figure 2-2, where most of the nodes surrounding node C are grouped in the center GCC. The above results also apply to node B who has the highest modularity.

Mathematically, modularity is given by

$$Q(P) = \sum \frac{1}{2m} \sum_{i,j \in C_i} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \tag{2.3}$$

As node C's ego network has more edges than the other two, m increase, forming more inter-connections in the graph and $Q(P)$ drops.

Question 21

Homogenity: A perfectly homogeneous clustering is one where each cluster has data-points belonging to the same class label. Homogeneity describes the closeness of the clustering algorithm to this perfection.

Completeness: A perfectly complete clustering is one where all data-points belonging to the same class are clustered into the same cluster. Completeness describes the closeness of the clustering algorithm to this perfection.

The term homogeneous is different from completeness in the sense that while talking about homogeneity, the base concept is of the respective cluster which we check whether in each cluster does each data point is of the same class label. While talking about completeness, the base concept is of the respective class label which we check whether data points of each class label is in the same cluster.

Question 22

The homogenity and completeness values for the 3 personalized networks are given below.

node	id	modularity	homo	complete
Node A	109327480479767108490	0.252765	0.85188	0.32987
Node B	115625564993990145546	0.319473	0.45189	-3.42396
Node C	101373961279443806744	0.191090	0.00386	-1.50423

Table 2-1 Homogeneity and Completeness scores of three personalized networks in Google+

We discuss three ego networks seperately:

- Node A: The network has a high homogeneity score, meaning that in each cluster/community, most users belong to the same circle. The completeness score is between 0 and 1, indicating that some users from the same circle have not been grouped in the same cluster. This result can be observed in Figure 2-2, where the members of the personalized network of node A can be well seperated into different clusters with occasional overlaps.

- Node B: This network has a lower homogenity score than node A, meaning that more users from different circles can be grouped in the same cluster. The completeness score is negative, or $H(K|C) > H(K)$. This can happen when some of the circles have not been assigned to any community and have been lumped together (each community has a few circles). In other words, a negative completeness points towards a large mismatch between the number of circles and number of communities.
- Node C: This network has the lowest homogenity score and negative completeness score. Similar results can be derived from the previous discussions.

3 Cora data

Question 23

Question 24

Question 25

We start by dealing with the dataset. We extract the nodes of the graph from *cora.content* and the edges of the graph from *cora.cites*. Each node is a data frame containing a unique id, one of the 7 labels and a bool array of 1433 features/words.

We then find the GCC of the original graph and relabel the node ids in the GCC. Within each label class, we choose 20 nodes as the training set, resulting in 140 starting nodes. For each of the starting node, we start a random walk for 100 steps and repeat 1000 times, yielding 1000 ending nodes (duplication is possible). Each node has 7 counters with 7 labels. Each time it becomes the ending node from the random walk, the counter corresponding to the label of the starting node is increased. In the end, the counter with the maximum count determines the label of the node.

In the random walk process, we have two customized properties:

(i) teleportation takes the random walker to one of the seed documents of that class (with a uniform probability of 1/20 per seed document). The teleportation probability varies in {0, 0.1, 0.2}.

(ii) the probability of transitioning to neighbors is proportional to the cosine similarity between the text features of the current node and the next neighboring node.

Based on these properties, we discuss our results in four sections:

Section	Teleportation	TFIDF
A		
B	✔	
C		✔
D	✔	✔

The table below shows the overall results of the four sections.

Table 3-1 Overall Accuracy and F1 scores

Section	Teleportation%	TFIDF	Accuracy%	F1%
A	0	NONE	37.3	34.0
B	0	NONE	37.5	34.6
	10	NONE	71.3	70.2
	20	NONE	69.1	68.0
C	0	SIMILARITY	38.4	35.6
D	0	SIMILARITY	40	36.8
	10	SIMILARITY	70.5	69.4
	20	SIMILARITY	69.2	68.0

A. W.o. teleportation, w.o. tfidf

The detailed report is shown in the table below

Table 3-2 Section A report

	precision	recall	f1-score	support
0	0.36	0.57	0.44	285
1	0.46	0.51	0.48	406
2	0.53	0.30	0.38	726
3	0.65	0.59	0.62	379
4	0.12	0.16	0.14	214
5	0.10	0.21	0.13	131
6	0.21	0.17	0.19	344
accuracy			0.37	2485
macro avg	0.35	0.36	0.34	2485
weighted avg	0.41	0.37	0.38	2485

We can see that without teleportation and tfidf, the prediction accuracy is only 37%. A solid explanation is that the random walk goes too far away from the starting node, mistakenly labeling nodes that are not as relevant to be the same class as the starting node. This can be proved by the fact that there is no unvisited node even with only 100 steps and 1000 random walks.

B. W. teleportation, w.o. tfidf

The detailed report is shown in the tables below

Table 3-3 Section B report, tp=10%

	precision	recall	f1-score	support
0	0.71	0.72	0.72	285
1	0.83	0.89	0.86	406
2	0.86	0.56	0.67	726
3	0.75	0.79	0.77	379
4	0.59	0.83	0.69	214
5	0.44	0.84	0.58	131
6	0.62	0.62	0.62	344
accuracy			0.71	2485
macro avg	0.69	0.75	0.70	2485
weighted avg	0.74	0.71	0.71	2485

Table 3-4 Section B report, tp=20%

	precision	recall	f1-score	support
0	0.60	0.72	0.65	285
1	0.84	0.88	0.86	406
2	0.86	0.53	0.66	726
3	0.71	0.75	0.73	379
4	0.60	0.83	0.70	214
5	0.44	0.83	0.58	131
6	0.61	0.57	0.59	344
accuracy			0.69	2485
macro avg	0.67	0.73	0.68	2485
weighted avg	0.72	0.69	0.69	2485

We can see that the prediction accuracy drastically increases with 10% teleportation probability, and slightly decrease when we increase the probablity to 20%. We also notice that there are only 23 unvisited nodes with 10% probability but up to 87 unvisited nodes with 20% probability. As we maintain the number of steps and times of the random walk, higher teleportation probability leads to closer ending nodes to the starting nodes and more unvisited nodes, which have a negative affect on the prediction accuracy.

C. W.o. teleportation, w. tfidf

In the previous sections, the transition matrix is simply the normalized adjacency matrix, only restoring the connection information of nodes. From this point, we calculate the transition matrix based on cosine similarity of the two nodes' text feature array. We take the following steps in our calculation:

1. Transform the original bool feature array into tfidf feature array.
2. For each neighboring nodes, we calculate their cosine similarity.
3. We normalize the result by all the neighboring similarities of a particular node.

The detailed report is shown in the table below

Table 3-5 Section C report

	precision	recall	f1-score	support
0	0.36	0.58	0.44	285
1	0.45	0.53	0.49	406
2	0.55	0.28	0.37	726
3	0.69	0.58	0.63	379
4	0.12	0.17	0.14	214
5	0.11	0.26	0.16	131
6	0.30	0.22	0.25	344
accuracy			0.38	2485
macro avg	0.37	0.38	0.36	2485
weighted avg	0.44	0.38	0.39	2485

To our surprise, we failed to notice any significant accuracy gain using tfidf. We believe this may be due to the innate similary between a particular node and all its neighboring nodes. In other words, having tfidf or not does not significantly affect the transition probability, leaving a similar accuracy as Section A.

D. W. teleportation, w. tfidf

The detailed report is shown in the table below

Table 3-6 Section D report, tp=10%

	precision	recall	f1-score	support
0	0.70	0.71	0.71	285
1	0.84	0.90	0.87	406
2	0.87	0.53	0.66	726
3	0.75	0.78	0.76	379
4	0.54	0.82	0.65	214
5	0.43	0.84	0.57	131
6	0.62	0.63	0.63	344
accuracy			0.71	2485
macro avg	0.68	0.74	0.69	2485
weighted avg	0.74	0.71	0.71	2485

Table 3-7 Section D report, tp=20%

	precision	recall	f1-score	support
0	0.58	0.71	0.64	285
1	0.85	0.88	0.87	406
2	0.86	0.54	0.66	726
3	0.74	0.76	0.75	379
4	0.58	0.81	0.68	214
5	0.44	0.83	0.58	131
6	0.60	0.58	0.59	344
accuracy			0.69	2485
macro avg	0.67	0.73	0.68	2485
weighted avg	0.73	0.69	0.69	2485

As we can see in Table 3-1, after adding tfidf with teleportation, the accuracy roughly remains the same. We have given a possible explanation of why tfidf might not be so helpful in this specific dataset.