Assignment 2 SNCAS

Van Nguyen s3726266

October 2023

1 Part I

1.1 Question 1.1

Graphs that have clustering coefficient (average) of 0 usually translate to having few characteristics such as not having clusters or triangles.

Logically speaking we can just then look into basic graph types and look for ones that do not have any clusters or triangles.

Those who fit these criteria can be: **acyclic graph**, **complete bipartite graph** and **line graph** as well.

1.2 Question 1.2

Networks having 0 average clustering coefficient mean that the network is not likely to have many triangles and clusters whereas networks having average clustering coefficient close to 1 or 1 show that almost every or every node is connected to neighboring node. Basically likelihood of triangles and clusters is high.

If the nodes have same average clustering coefficient that is not 0 nor 1 we can assume that the network is evenly distributed. Meaning that there is some clusters and triangles but they are of same size and evenly dispersed throughout the network. And since it's less than 1 it is not one big cluster (not all nodes are connected together)

1.3 Question 1.3

I think there can be two similar but principally different methods how to prove this.

First method doesn't really provide mathematical solution but uses more of common sense in math. By definition average in mathematics means a typical value of said data set. An average is calculated by calculating sum of all values divided by number of counted values. In our case we count clustering coefficient of all nodes and then dividing it by number of nodes. Our hypothesis lie in fact that there exist a node that is of average clustering value of whole network or higher. If we look at distribution of values in our network the average sits in middle therefore there must exist some values for the average to be in middle.

Second method is mathematical proof. We can prove that there is existence of node that has clustering coefficient equal or higher than the average clustering coefficient of whole network by using contradiction method. So we start with assumption that there is no node with higher or equal average clustering coefficient than of average.

1. First we calculate the average clustering coefficient of node for whole network. Sum of all nodes and their coefficients divided by number of nodes.

$$G(C) = \frac{1}{n} \sum_{v \in G} c_v \tag{1}$$

2. We are assuming that there is no node higher or equal in coefficient value than average.

$$c(v) < C(G) \tag{2}$$

3. 3) For all nodes 4) Simplify and combine

$$\sum_{v \in V} c(v) < \sum_{v \in V} C(G) \tag{3}$$

$$\sum_{v \in V} c(v) < |V| \cdot C(G) \tag{4}$$

The part $\sum_{v \in V} c(v)$ can be further "simplified" into C(G) as it is same as average clustering coefficient of whole network. In other words the left side of sum is sum of clustering coefficients of all nodes which is same as C(G) as per (1).

4. Therefore it is:

$$C(G) < |V| * C(G) \tag{5}$$

Which contradict itself. Some number cannot be smaller than the same number it can only be equal. Some number cannot be smaller than other number if that second number is being multiplied by another number (which we know isn't zero in fact always at least 1). Since our hypothesis is false we can assume the opposite is true. That there indeed a node that that has higher or equal average clustering coefficient of whole network.

1.4 Question 1.4

-

1.5 Question 1.5

Degree centrality of a node is based on its importance in the network by the amount of in or out "connections". Whereas betweenness centrality is how often node lies in shortest path between 2 or more nodes.

We can assume that if we were to rank the network where degree centrality and betweenness centrality are equal it would result in specific structure of the network.

First would be **line graph**. A graph where all nodes are in line. Only the first and last node have one neighbor. The rest have same amount of neighbors (2 neighbors). Meaning that they will all have same degree of centrality. At the same time the betweenness centraility is same for all nodes as they are always in the shortest path of first and last node.

The second one would be **star graph**. It satisfies the condition that all nodes except central one have same betweenness centrality as they have to go through middle one. For degree centrality the central node has the highest one and other nodes have same as well. But this graph perhaps does not satisfy the condition of *"node ranking produced by degree centrality is equal to that of betweenness centrality"*.

2 Part II

2.1 Question 2.1

-

3 Part III

3.1 Question 3.1

This task can be summarized to few key points:

- Load the data set
- Parse data set line by line
- Split the line word by word
- Check if the word starts with @ (signifying that user is mentioning somebody)
- Check if he is not mentioning himself
- If he is not mentioning himself add him to adjacency list (by source user)
- Create .csv file using source, target, weight (user, mention, count)

We start by loading the data set. Then we loop through the data set row by row. We specify definition of the row which is timestamp, user and the tweet itself. Then we loop through each row and split the whole row word by word. Those words are stored in list where we loop the list again and we check for if the word starts with @ which would mean that the word is a mention of a user. If we successfully check that it starts with @ then we check if the user is not mentioning himself by comparing the checked word with current username. If then this condition passes we add mention user to a adjacency list. If the user is mentioned again by same user we simply increase the count by 1. The output is then written into the output file with label such as source, target and weight following user, mention, count pattern.

That was first iteration and then I had to tweak some things. For example after the first iteration I imported the csv file to check if the csv can be successfully put into Gephi. Which it does but after running statistics I found out that the second most mentioned user is blank. Meaning @ without any specific user. Another condition had to be added.

Now instead of checking if

```
word != username
```

we instead check

```
if tag != "@" and tag != "@" + username.
```

This checks for @ (tag) being at least 1 character long (basically as long as it's not just @).

3.2 Question 3.2

For these calculations we could mostly re-use the code from previous assignment with slight changes since the data set differs at some places.

Metric	**Value**
Number of nodes	77693
Number of edges	83558
Number of SCCs	77693
Number of WCCs	6343
Density	0.0000138430
Average node clustering coefficient	0.0000
Approx. distance distribution	-
Approx. average distribution	11.2815

Table 1: Values for 3.2 question

For the first two values we simply take .csv file we created from question 3.1 and create graph using nx.DiGraph function from networkx. Unlike previous assignment this required handling weights separately. After that we simply just built-in function like *number_of_edges, number_of_nodes*.

For number of SCCs and WCCs we similarly use networkx functions. For number of SCCs we get same value as number of nodes 77693 when checking sizes of SCCs were all 1. Meaning that each own node was its own SCC.

In another words there is no connectivity between nodes. No forming of larger strongly connected components. For WCC we get more interesting results. The number of WCC is 6343 and the sizes hover around 2 or 3 but the biggest one being 51442. Occasional spikes in double digits usually no bigger than 30.

Density is 0.0000138430 which means that the network is very sparse have a relatively few connections. The density can be calculated by dividing number of edges divided by (number of nodes * number of nodes - 1).

For indegree and outdegree distribution we similiary use *in_degree, out_degree* functions. Similarly to previous assignment we use bin spaces to better visualize the results on graph.

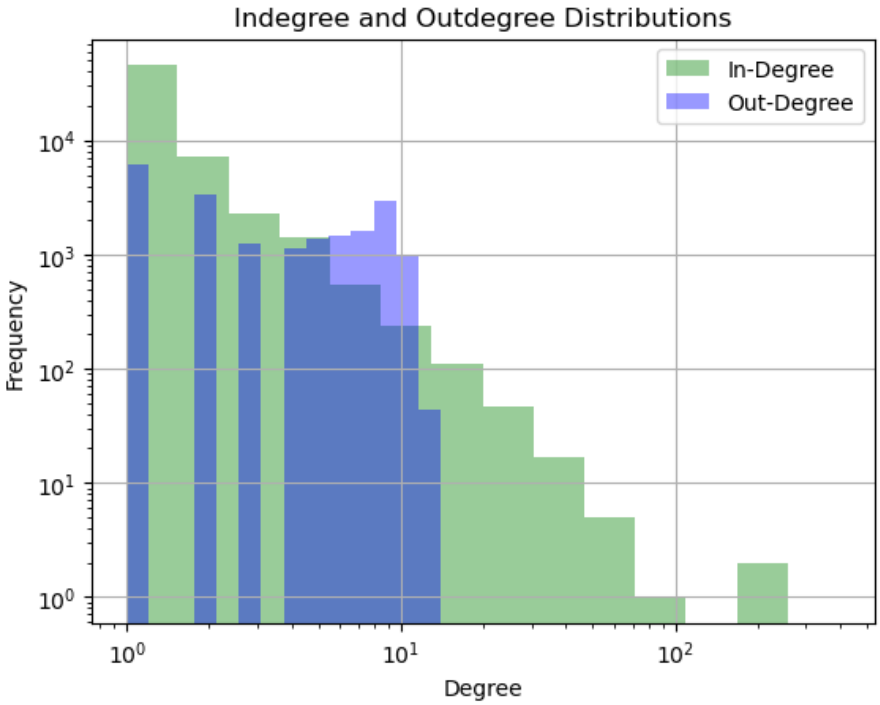

Figure 1: Indegree and outdegree distributions

For average node clustering coefficient I was surprised to find out that the result is 0 even though I tried to calculate with 20 decimal precision. Meaning the network is simply just not producing any triangles or clusters.

Approximated distance distribution of the giant component can by seen in below figure 2. For approximation I used 20000 samples as it represents 50% of the smaller data set.

For average we just simply sum distances we got and divide it lenght of distances (how many distances we have). The result it then 11.2815.

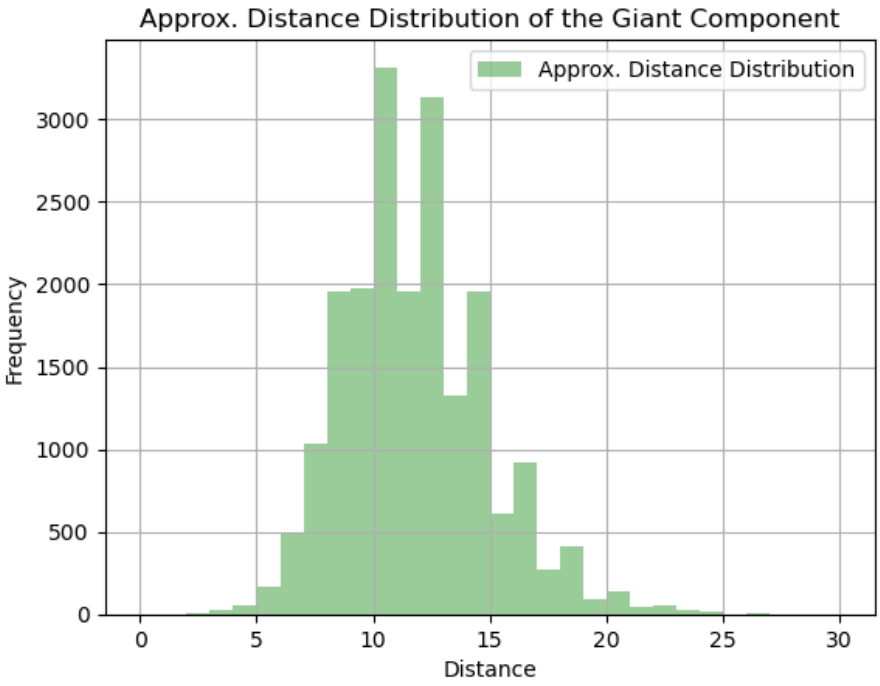

Figure 2: Distance distribution of GC

3.3 Question 3.3

First we calculate the centraility measures using built-in function such as: *degree_centrality, betweenness_centraility and closeness_centrality*. Then we sort the nodes where key to sorting is degree of given centrality/betweenness/ closeness from highest to lowest. After that we take just top 20 users.

I have encountered a problem where it would still take in empty mentions (@). First I've tried to solve it again redefining graph with condition to check whether the mention is empty or not. Unfortunately that did not work so instead of taking top 20 users I took 21 and manually removed empty @.

Directionality is taken in account by using directed graph. Meaning data set imputed keeps "in memory" who mentions who.

For comparing there are couple methods I found online [1]. And the easiest to apply was Spearman Rank which is included in scipy package and these are the results:

Spearman Rank Correlation (Degree vs. Betweenness): -0.21654135338345865

Spearman Rank Correlation (Degree vs. Closeness): 1.0

Spearman Rank Correlation (Betweenness vs. Closeness): -0.21654135338345865

We can easily interpret these result as such. Correlation between Degree and Closeness centrality is 1 meaning that they are identical. Nodes with high degree centrality are same nodes that are high in closeness centrality.

Whereas the comparison of degree and closeness with betweeness yield negative values. Meaning that they are not similarly. Meaning that the nodes are not same as it was in previous case.

3.4 Question 3.4

Using nodes from largest weakly connected components and sub graph made from graph containing all the edges and nodes we can detect community using Louvain community detection algorithm (*best_partition*). Output is then written into output file to further analyze.

There are exactly 161 communities with average of 319.515 members. Top 5 communities are communities with ID of 2, 48, 6, 78 and 19. Community 2 having 2446 members and community 19 having 796 members. If we

compare rest of the communities to the amount of nodes in the network there is no so much significance. Most of them are very fragmented and low in number and in comparison to amount of total nodes very negligible.

3.5 Question 3.5

First I imported the parsed data set from Question 3.1. Then I started experimenting with different layouts. First image's layout is purely Yifan Hu. For community detection I did calculate modularity (with weights and using randomize) then applied default scheme to visualize communities themselves. For node sizes I chose degree centrality as it seemed important to just count number of edges that node has.

For second image I applied extra Force Atlas 2 to have a different view on the whole network. Also used some extra expansion layout as well.

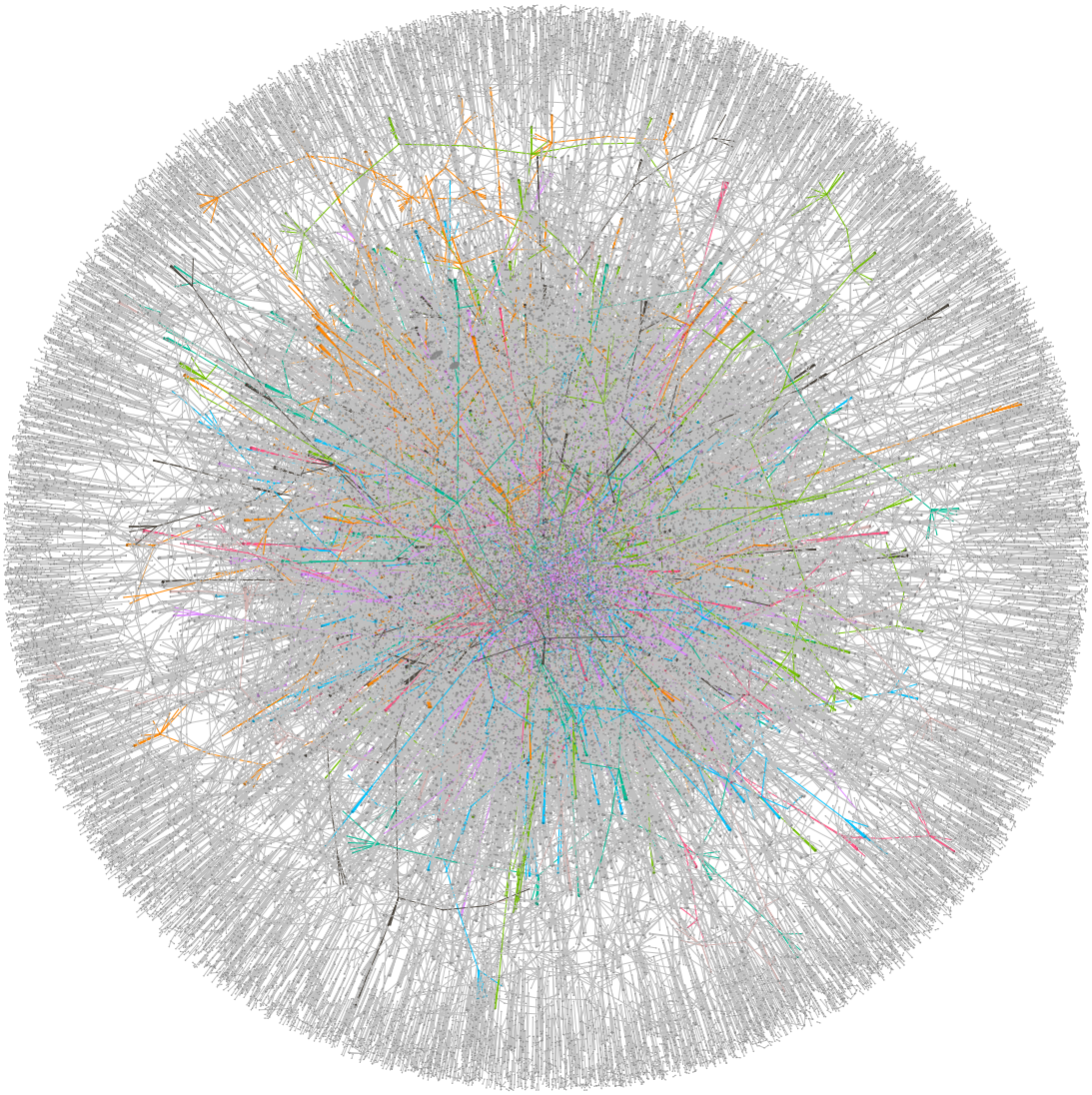

Figure 3: Yifan Fu

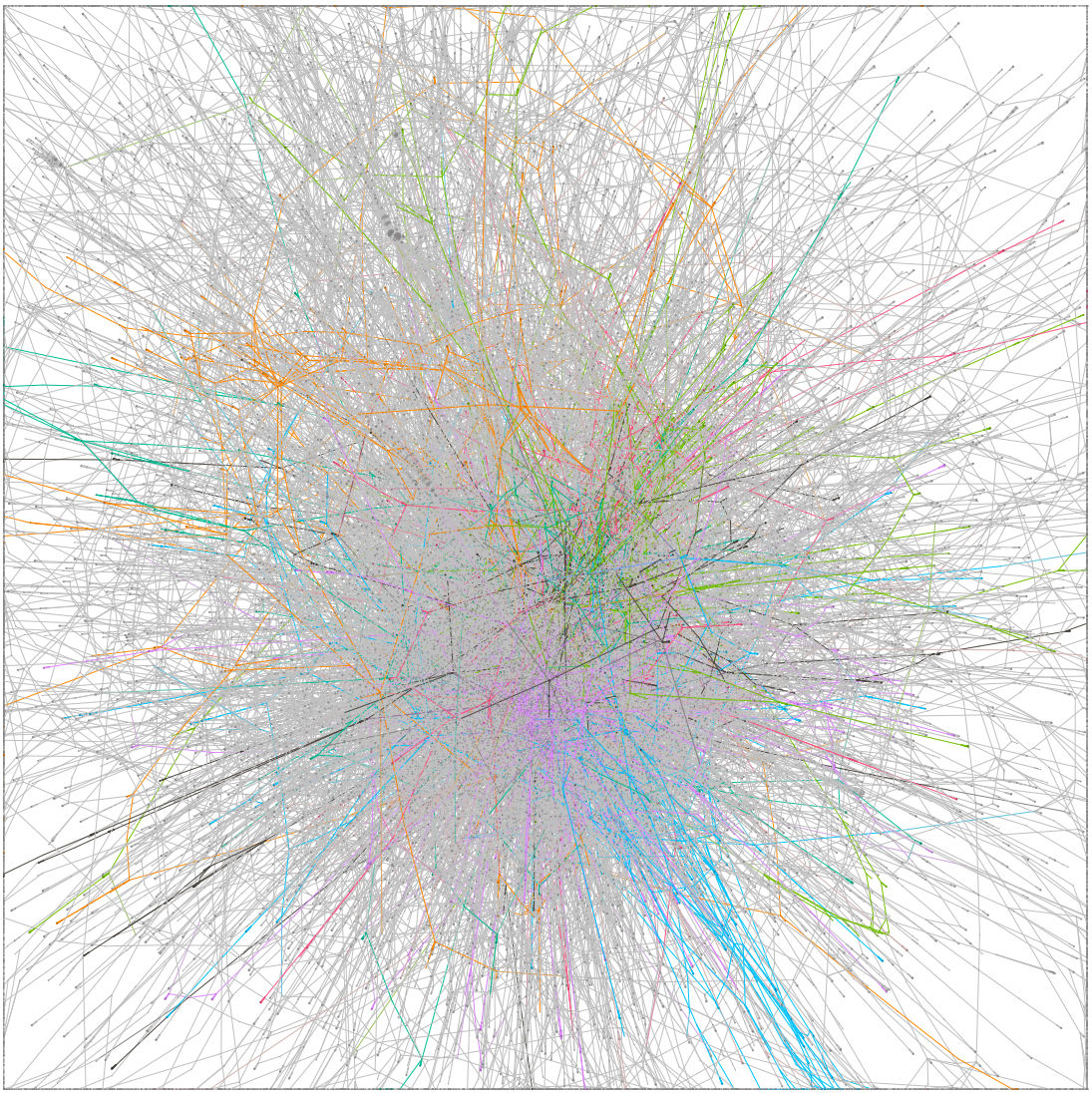

Figure 4: Yihan Fu and Force Atlas 2

3.6 Question 3.6

A simple plotting into histogram from adjency.csv. In the first iteration it seemed like only weights of 1 and 2 existed in the network but after manually counting them with a script I found out there is more than just weights of 1 and 2s so I had to apply bin with logarithmic scale.

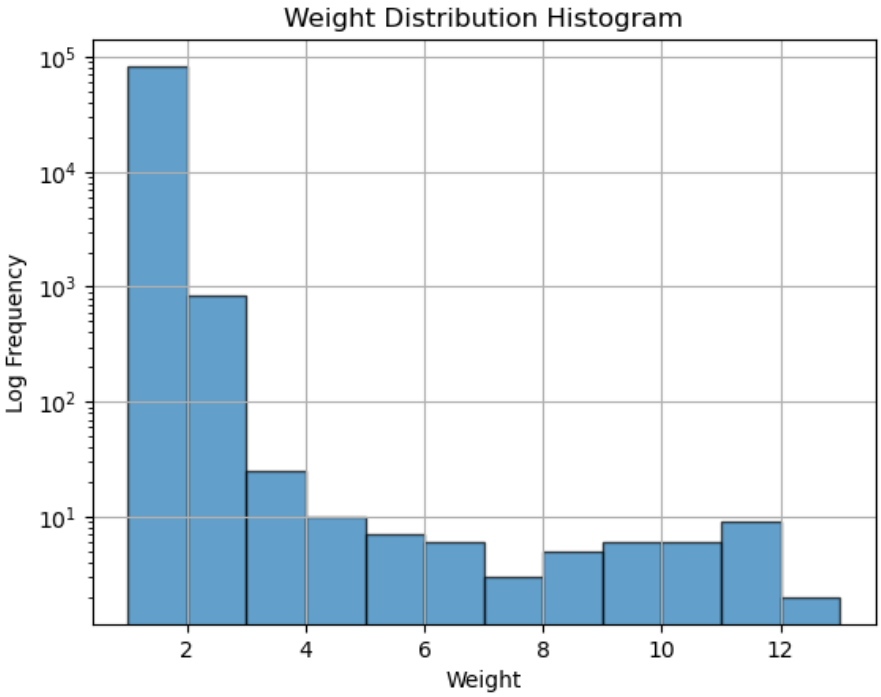

Figure 5: Enter Caption

3.7 Question 3.7

Metric	**Value**
Number of nodes	398771
Number of edges	456112
Number of SCCs	398771
Number of WCCs	34199
Density	0.00000287
Average node clustering coefficient	0.0000
Approx. distance distribution	-
Approx. average distribution	10.5927

Table 2: Values for 3.7 question

The procedure for calculation these values are pretty much same so I will just discuss the results.

For the sizes of SCC there is almost no difference as most of the values are at 1. For the sizes of WCCs we can observe an increase presumably of the same node. The highest value of 287865.

Density got even smaller or rather even more precise since we could calculate it across more nodes and edges.

For the histogram of in and out degrees we can see more values being filled out for out-degree. But visual representation of in-degree is almost similar to the of a small data set. (Figure 6)

Average node clustering coefficient is with no change.

For approximated distance distribution there is again no change either. Or any significant one at least. (Figure 7)

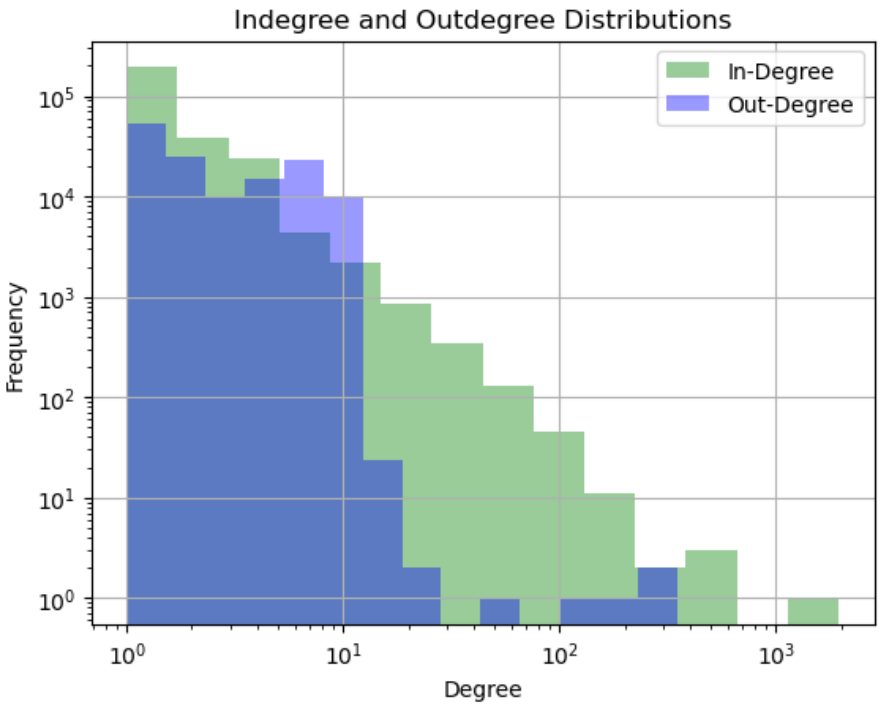

Figure 6: In and Out degree distribution

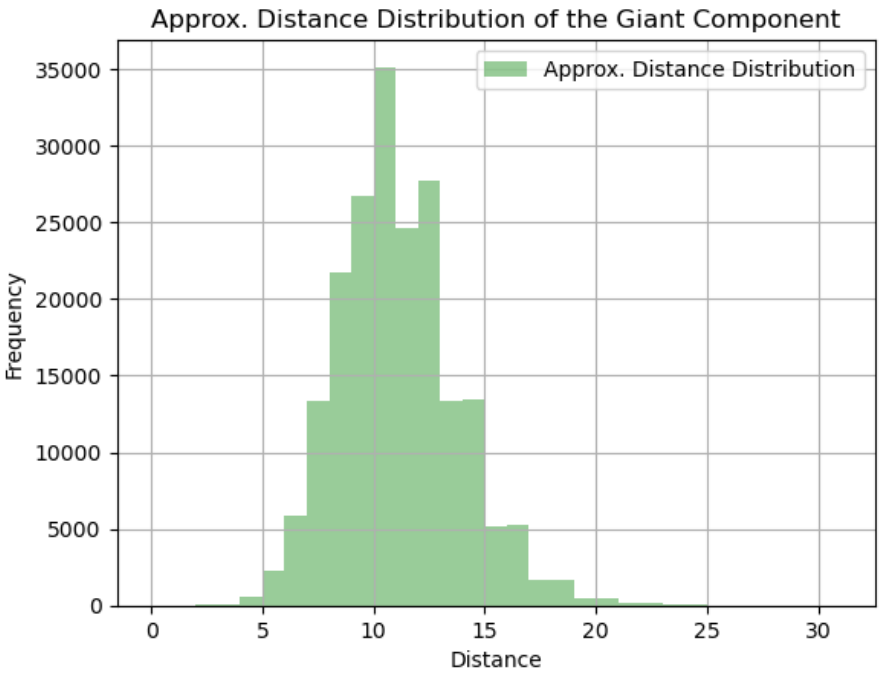

Figure 7: Approx. distance distribution of GC large-dts

And for average distance in giant component the values is 10.5927 which has slightly decreased if we were to compare it to the calculation from smaller data set.

3.8 Sources

[1] https://www.degruyter.com/document/doi/10.1515/phys-2018-0122/html?lang=en