VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE4069- Social Media Analytics

Social Ego Network Analysis

project report

By

TEAM-20

Abhishek Kasam – 19MIA1081

M. Sree Reddy - 19MIA1057

A. Arvind Reddy - 19MIA1089

T. Venkat Reddy – 19MIA1104

Submitted to

Dr. Priyadarshini R
Assistant Professor
SCOPE, VIT Chennai

School of Computer Science and Engineering

January 2023

TABLE OF CONTENTS

ABSTRACT

Ego networks models describe the social relationships of an individual (ego) with its social peers (alters). The structural properties of ego networks are known to determine many aspects of the human social behavior, such as willingness to cooperate and share resources. Due to their importance, we have investigated if Online Social Networks fundamentally change the structures of human ego networks or not. In this project we provide a comprehensive and concise compilation of the main results we have obtained through this analysis. Specifically, by analyzing several datasets in Facebook and Twitter, we have found that OSN ego networks show the same qualitative and quantitative properties of human ego networks in general, and therefore that, somewhat counter-intuitively, OSNs are just "yet another" social communication means which does not change the fundamental properties of personal social networks. Moreover, in this project we also survey the main results we have obtained studying the impact of ego network structures on information diffusion in OSNs. We show that, by considering the structural properties of ego networks, it is possible to accurately model information diffusion both over individual social links, as well at the entire network level, i.e., it is possible to accurately model information "cascades". Moreover, we have analyzed how trusted information diffuses in OSNs, assuming that the tie strength between nodes (which, in turn, determines the structure of ego networks) is a good proxy to measure the reciprocal trust. Interestingly, we have shown that not using social links over a certain level of trust drastically limits information spread, up to only 3% of the nodes when only very strong ties are used. However, inserting even a single social relationship per ego, at a level of trust below the threshold, can drastically increase information diffusion. Finally, when information diffusion is driven by trust, the average length of shortest paths is more than twice the one obtained when all social links can be used for dissemination. Other analyses in the latter case have highlighted that also in OSNs users are separated by about 6 (or less) degrees of separation. Our results show that when we need trustworthy "paths" to communicate in OSNs, we are more than twice as far away from each other

INTRODUCTION

Social network analysis is a rapidly growing field that seeks to understand the social structures and patterns of interaction within and between groups. At the core of this field are the concepts of ego networks, which refer to the set of social ties that a given individual has with others.

Ego networks are one of the key concepts to study the microscopic properties of personal social networks. Different definitions of ego network exist in the literature, corresponding to different approaches in analyzing them. In this paper, an ego network is formed of a single individual (ego) and the other users directly connected to it (alters). This model gives particular emphasis to the impact of the ego cognitive constraints on the personal social networks and, in the rest of the text, we will refer to it as the 'Ego-Network Model'. Another possible definition of ego network also considers the links between alters, possibly even excluding the links between them and the ego. This is typically used to analyze the topological features of the local social context in which the ego is immersed. Techniques that have been used to this end are based on complex network indices, such as density, connectivity (e.g., Burt's 'Structural Holes') or ego betweenness measures. Two important perspectives on ego networks are the first-order and second-order egocentric network analyses.

First-order ego network analysis focuses on the direct connections between an individual and their immediate contacts or alters. This approach provides insight into the individual's position within their social network, as well as the nature of the relationships they have with their immediate peers. Second-order ego network analysis, on the other hand, examines the connections between an individual's alters, rather than just their direct ties to the ego. This approach can help to uncover broader patterns of social interaction and influence that may not be immediately apparent through first-order analysis.

Overall, the study of ego networks and the use of both first-order and second-order network analysis techniques has become an increasingly important tool for social scientists seeking to understand the complex social structures and relationships that shape our world.

LITERATURE SURVEY

A literature survey is an essential step in social ego network analysis to identify gaps in existing research and to determine the most appropriate methodological approach for a particular research question.
We did literature survey for our project by taking five research papers into consideration which focuses on social ego network analysis.

“Analysis of Ego Network Structure in Online Social Networks” by Valerio Arnaboldi et.al. In this paper they find that the properties of OSN ego networks have a strong similarity with those found in offline ego networks. Namely, the typical number of circles in the structure of virtual ego networks is, on average, equal to 4 and the average scaling factor between the concentric circles of the social structure is near to 3, as found in real environments. Moreover, the sizes of the circles, i.e., the number of social relationships of each type, is remarkably similar to those existing in offline social networks. Notably, the average size of the OSN ego networks is very close to the well-known Dunbar’s number, which denotes the average size of ego networks in offline social networks.

“Ego Networks in Twitter: An Experimental Analysis” by Marco Conti et.al. r. The results indicate that Twitter presents social structures qualitatively similar to that found by Dunbar in offline ego networks and by ourselves in a similar study on Facebook. This suggests that Twitter (as we have previously shown for Facebook) does not fundamentally change the structure of human ego networks, which is instead determined by other characteristics of the human socializing process, such as the maximum number of cognitive resources dedicated to social activities.

“Ego Network Structure in Online Social Networks and its Impact on Information Diffusion” by Massimiliano La Gala et.al. The results on ego network structures in OSNs, we performed an information diffusion analysis assessing the impact of the different ego network rings (i.e., portion of each circle not containing the other nested circles) on the process. Specifically, we performed a correlation analysis on Twitter data to assess the relation between direct contact frequency (of Twitter replies) and the frequency of retweets passing through social links. The results indicate that the two measures are highly correlated, with links in the internal ego network layers showing the highest correlation. As a further refinement of the analysis, we classified the alters of each ego network into “socially relevant users” and “other users”, and we calculated the correlations for these classes separately. Interestingly, the correlations of both classes are higher when taken separately rather than analyzing them together. This could indicate the presence of two separate processes governing the diffusion of information for the two classes of alters.

“Online Social Networks and information diffusion: The role of ego networks” by Andrea Passarella et.al. In this paper they have presented our most recent work on the characterization of the structural properties of social relationships in OSNs, and how they depend on human cognitive and time constraints. From our analyses, we have seen that the properties of ego networks in OSNs are compatible with those found in offline environments. This indicates that the hierarchical structure of concentric layers of alters around the ego is consistent among different social environments, and is not influenced by the use of a particular communication medium. This is a clear indication that human cognitive and time constraints shape social relationships not only in offline environments, but also in OSNs, in contrast to the conventional wisdom that OSNs are able to improve our social capacity and allow us to maintain a much larger number of relationships than is possible “offline”.

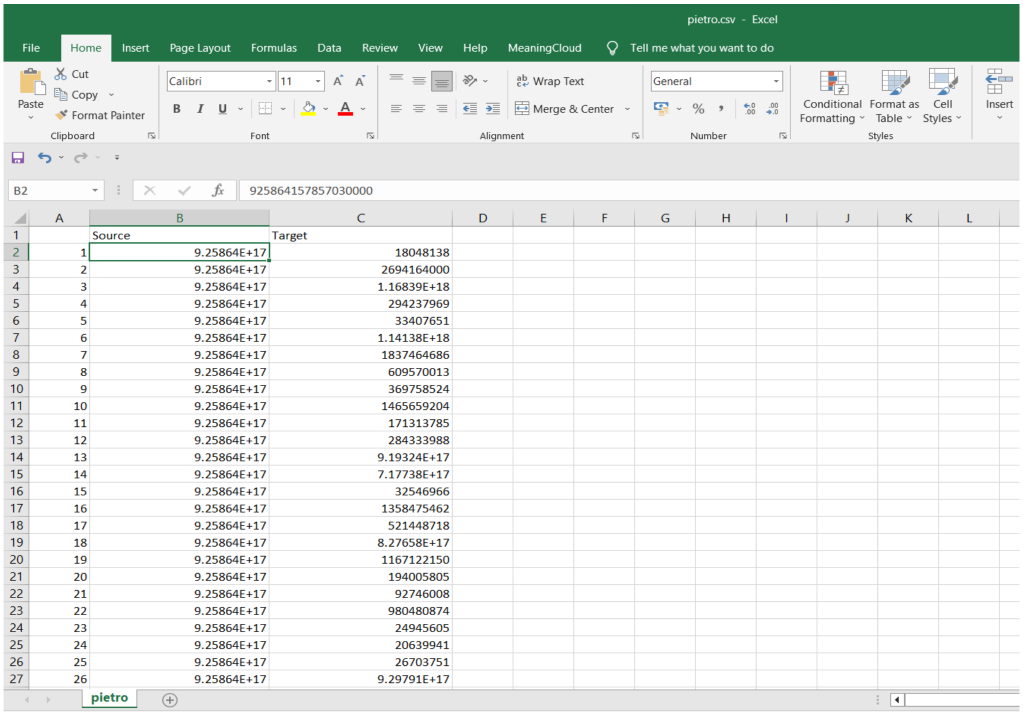
"ACO-based clustering for ego network analysis" by David Camacho et.al. In this paper finally, although both ACO algorithms provides good results, we have found some Ego Networks extracted from Facebook that present problems to any community finding algorithm. All the algorithms taken into account in this work, the ACO-based and the ones extracted from the literature, provides poor results when facing these Ego Networks. Analyzing in detail these Ego Networks, we found that these networks are composed by profiles with low number of characteristics in the user profiles, and the networks are composed by low number of edges (which means a low interactions of the users). This fact suggests a future work that will be focused on the creation of an hybrid ACO algorithm based on both: profile and topology information, that could take advantages form both sources of data.

OBJECTIVES

- In this project the ultimate objective is to perform social network analysis on the network developed due to the main considered user who is treated as EGO.
- We also perform Community detection to analyse the modularity in the EGO network.
- We show that, by considering the structural properties of ego networks, it is possible to accurately model information diffusion both over individual social links, as well at the entire network level, i.e., it is possible to accurately model information “cascades”.

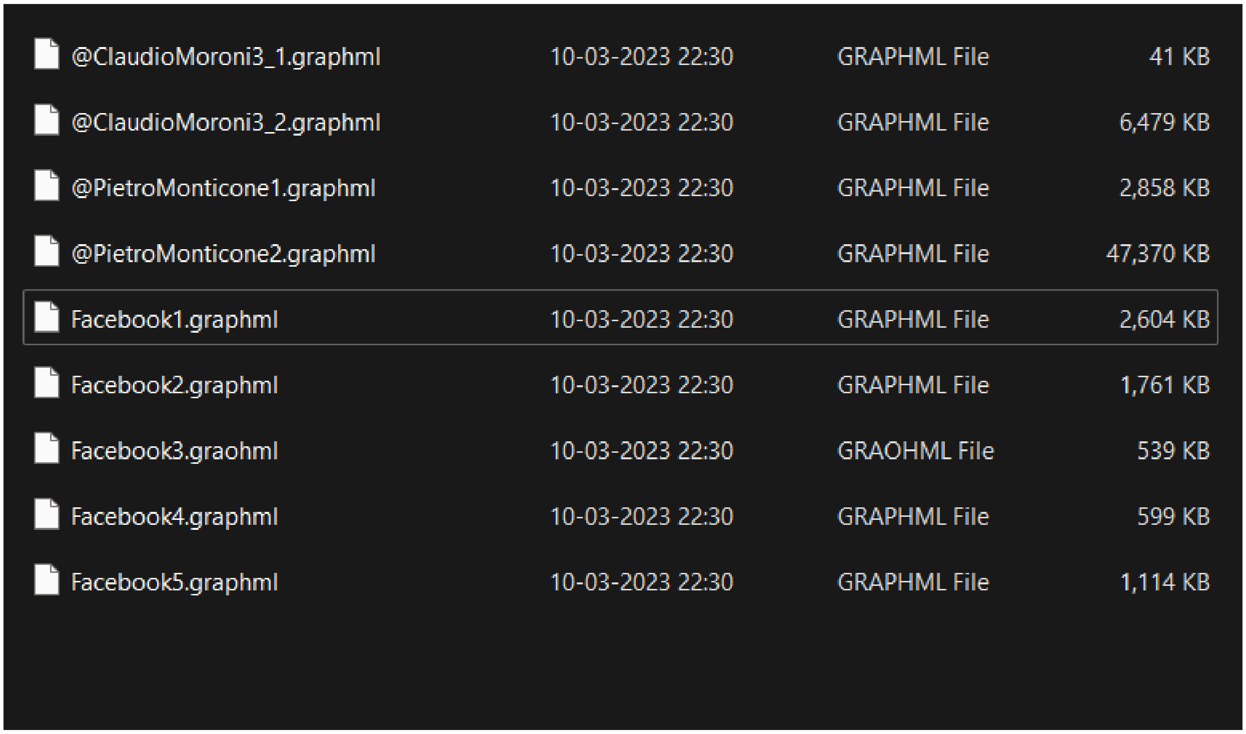
DATASET

- We have taken our dataset from Facebook and twitter [GRAPH DATA COLLECTION]
- In the form of edge lists [source and target] (csv) and graphml formats



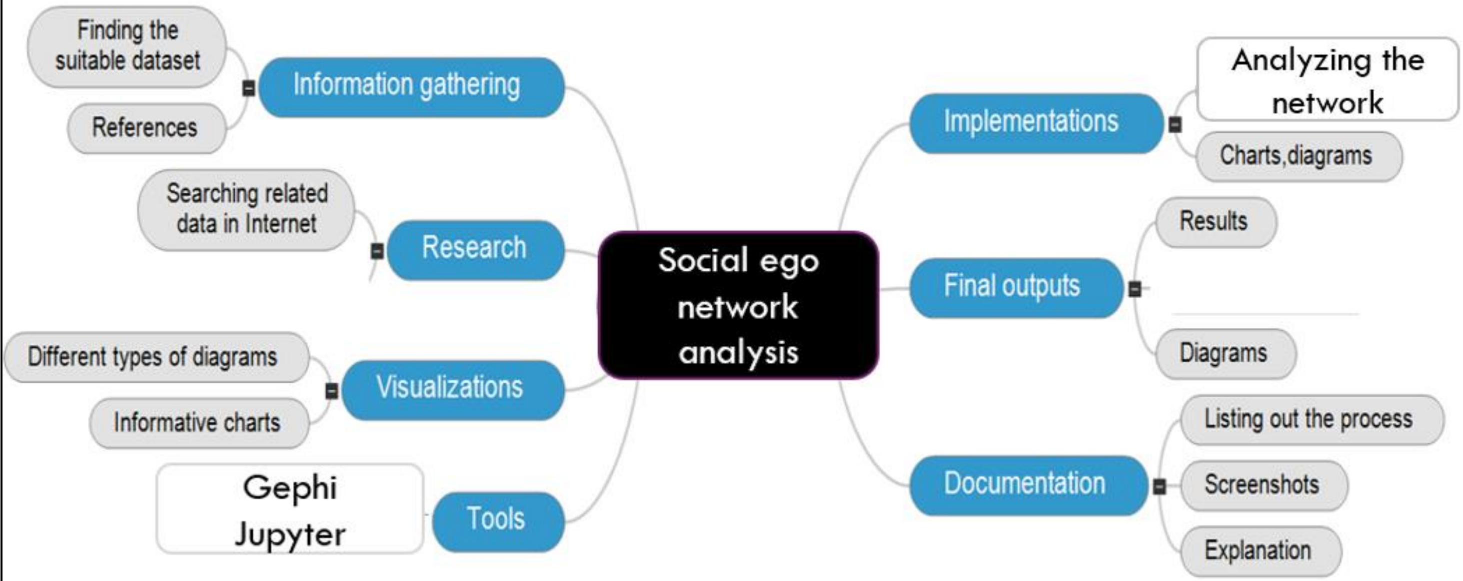
B2 925864157857030000

	A	B	C
1		Source	Target
2	1	9.25864E+17	18048138
3	2	9.25864E+17	2694164000
4	3	9.25864E+17	1.16839E+18
5	4	9.25864E+17	294237969
6	5	9.25864E+17	33407651
7	6	9.25864E+17	1.14138E+18
8	7	9.25864E+17	1837464686
9	8	9.25864E+17	609570013
10	9	9.25864E+17	369758524
11	10	9.25864E+17	1465659204
12	11	9.25864E+17	171313785
13	12	9.25864E+17	284333988
14	13	9.25864E+17	9.19324E+17
15	14	9.25864E+17	7.17738E+17
16	15	9.25864E+17	32546966
17	16	9.25864E+17	1358475462
18	17	9.25864E+17	521448718
19	18	9.25864E+17	8.27658E+17
20	19	9.25864E+17	1167122150
21	20	9.25864E+17	194005805
22	21	9.25864E+17	92746008
23	22	9.25864E+17	980480874
24	23	9.25864E+17	24945605
25	24	9.25864E+17	20639941
26	25	9.25864E+17	26703751
27	26	9.25864E+17	9.29791E+17



```
@ClaudioMoroni3_1.graphml     10-03-2023 22:30     GRAPHML File        41 KB
@ClaudioMoroni3_2.graphml     10-03-2023 22:30     GRAPHML File     6,479 KB
@PietroMonticone1.graphml     10-03-2023 22:30     GRAPHML File     2,858 KB
@PietroMonticone2.graphml     10-03-2023 22:30     GRAPHML File    47,370 KB
Facebook1.graphml             10-03-2023 22:30     GRAPHML File     2,604 KB
Facebook2.graphml             10-03-2023 22:30     GRAPHML File     1,761 KB
Facebook3.graohml             10-03-2023 22:30     GRAOHML File       539 KB
Facebook4.graphml             10-03-2023 22:30     GRAPHML File       599 KB
Facebook5.graphml             10-03-2023 22:30     GRAPHML File     1,114 KB
```

WORKFLOW



METHODOLOGIES

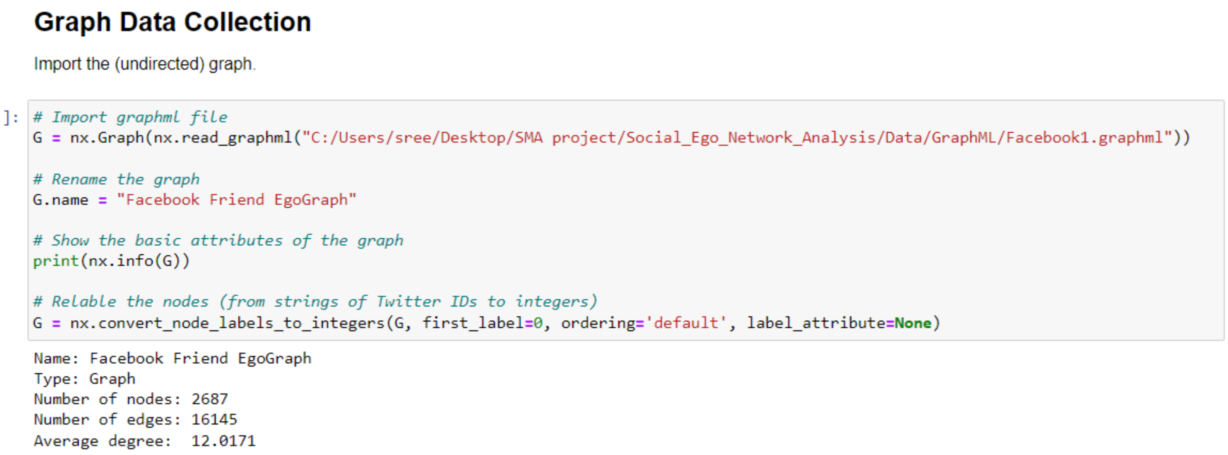
- **Loading the data.**

  Loading social ego network data into Jupyter Notebook is an essential step in conducting network analysis. There are different formats of social ego network data, such as adjacency matrix and edge list.

  To load an adjacency matrix, first, you should save the matrix in a CSV file. Then, use the pandas library to read the CSV file and create a data frame.

  To load an edge list, you can use the networkx library to create a graph. The edge list should have two columns representing the source and target nodes.

  Once the social ego network data is loaded, you can use different network analysis tools to analyze the structure and properties of the network.

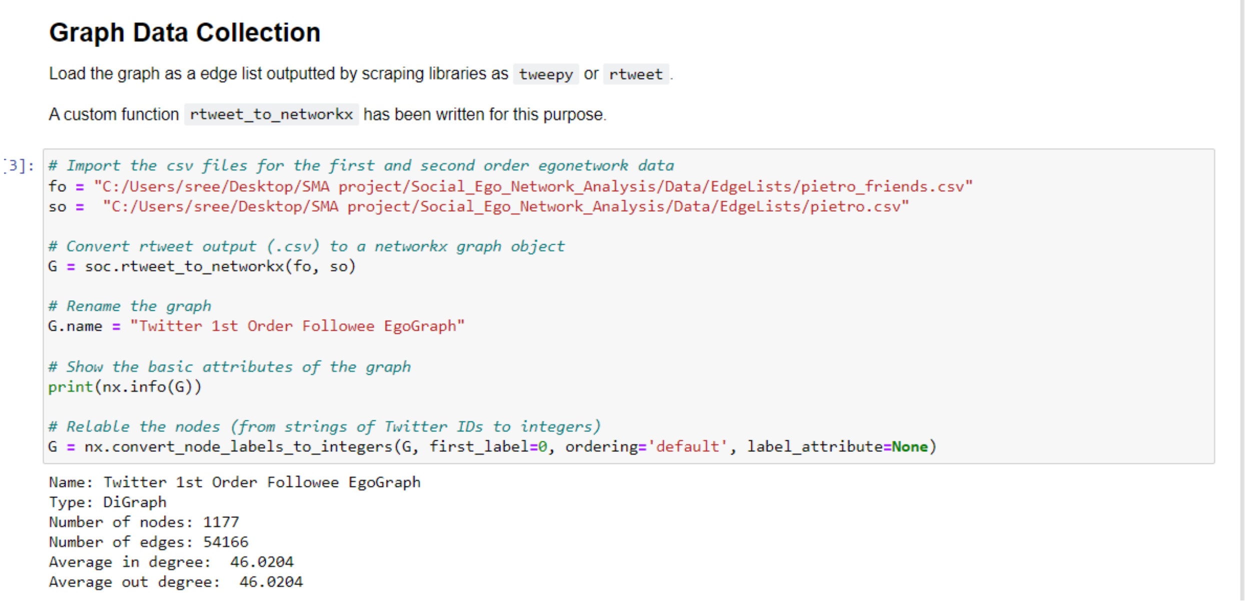


Graph Data Collection

Import the (undirected) graph.

```
# Import graphml file
G = nx.Graph(nx.read_graphml("C:/Users/sree/Desktop/SMA project/Social_Ego_Network_Analysis/Data/GraphML/Facebook1.graphml"))

# Rename the graph
G.name = "Facebook Friend EgoGraph"

# Show the basic attributes of the graph
print(nx.info(G))

# Relable the nodes (from strings of Twitter IDs to integers)
G = nx.convert_node_labels_to_integers(G, first_label=0, ordering='default', label_attribute=None)
```

```
Name: Facebook Friend EgoGraph
Type: Graph
Number of nodes: 2687
Number of edges: 16145
Average degree:  12.0171
```



Graph Data Collection

Load the graph as a edge list outputted by scraping libraries as `tweepy` or `rtweet`.

A custom function `rtweet_to_networkx` has been written for this purpose.

```
# Import the csv files for the first and second order egonetwork data
fo = "C:/Users/sree/Desktop/SMA project/Social_Ego_Network_Analysis/Data/EdgeLists/pietro_friends.csv"
so =  "C:/Users/sree/Desktop/SMA project/Social_Ego_Network_Analysis/Data/EdgeLists/pietro.csv"

# Convert rtweet output (.csv) to a networkx graph object
G = soc.rtweet_to_networkx(fo, so)

# Rename the graph
G.name = "Twitter 1st Order Followee EgoGraph"

# Show the basic attributes of the graph
print(nx.info(G))

# Relable the nodes (from strings of Twitter IDs to integers)
G = nx.convert_node_labels_to_integers(G, first_label=0, ordering='default', label_attribute=None)
```

```
Name: Twitter 1st Order Followee EgoGraph
Type: DiGraph
Number of nodes: 1177
Number of edges: 54166
Average in degree:  46.0204
Average out degree:  46.0204
```

- **Graph Data visualization**

Graph data visualization is an essential step in exploring and communicating social ego network data in Jupyter Notebook. Visualization allows us to gain insights into the structure, properties, and dynamics of the network, and to communicate our findings to others.

There are different libraries and packages available in Python for graph data visualization, such as matplotlib, seaborn, and networkx.

we can create simple ego network with nodes and edges and visualizes it using the nx.draw() function. The with_labels=True argument adds labels to the nodes, and the plt.show() function displays the graph. In addition to the basic visualization, we can customize the visualization by changing the node and edge colors, sizes, and shapes, and by adding labels and legends. Different graph visualization techniques, such as node-link diagrams, matrix plots, and heatmaps, can also be used depending on the research question and data structure.

- **Graph Data analysis**

  - **Degree distribution**

    Degree distribution is a fundamental concept in network analysis that describes the distribution of the number of edges or connections per node in a network. Degree distribution is often used to characterize the structure and properties of a network, such as its connectivity, heterogeneity, and resilience.

    In an **undirected network**, the degree of a node is the number of edges that connect to it. The degree distribution in an undirected network is the probability distribution of the degree values across all nodes in the network. The degree distribution is typically plotted as a histogram or a probability density function.

In a **directed network**, the degree of a node can be split into in-degree, which is the number of edges pointing towards the node, and out-degree, which is the number of edges pointing away from the node. The **in-degree distribution** is the probability distribution of the in-degree values across all nodes in the network, while the **out-degree distribution** is the probability distribution of the out-degree values.

The degree distribution in a network can have different shapes, depending on the network topology and the degree of heterogeneity. For example, in a random network, the degree distribution follows a Poisson distribution, which means that the degree values are evenly distributed around the average degree. In contrast, in a scale-free network, the degree distribution follows a power law distribution, which means that a few nodes have a high degree while most nodes have a low degree.

The analysis of degree distribution and its different forms, such as the un-directed degree distribution, in-degree distribution, and out-degree distribution, is essential in understanding the properties and dynamics of social ego networks, such as the influence of different nodes, the spread of information, and the emergence of communities.

- **power law distribution**

Power law distribution is a common pattern observed in many complex systems, including social ego networks. In a power law distribution, the frequency of a value (in this case, node degree) is proportional to a power of that value. In other words, the probability of a node having a degree k is proportional to k^(-γ), where γ is the power law exponent.

In social ego networks, power law distributions of degree (in-degree or out-degree) have been observed in various contexts, such as online social networks, citation networks, and collaboration networks. The power law exponent γ typically ranges between 2 and 3, indicating that a few nodes have a very high degree, while most nodes have a low degree.

The presence of power law distribution in social ego networks implies that some nodes are more important than others, as they have a disproportionate influence on the network's structure and dynamics. These highly connected nodes are often referred to as "hubs," and they play a crucial role in information diffusion, network resilience, and the emergence of communities.

However, it is important to note that not all social ego networks follow a power law distribution, and other types of distributions, such as the exponential distribution, may also be observed. Moreover, the power law distribution can be affected by different factors, such as network size, density, and homophily, and caution should be taken when interpreting the results of power law analysis in social ego networks.

- **centrality metrics:**

Degree centrality:

Degree centrality is a measure of node importance in a social ego network that is based on the number of connections or edges that a node has with other nodes in the network. A node with a high degree centrality is connected to many other nodes and can be seen as influential or popular in the network.

There are two types of degree centrality measures: in-degree centrality and out-degree centrality. In-degree centrality is based on the number of incoming connections to a node, while out-degree centrality is based on the number of outgoing connections from a node.

To calculate the degree centrality of a node, we divide its degree (in-degree or out-degree) by the maximum possible degree in the network. The maximum possible degree is n-1 for an undirected network and n-1 for a directed network, where n is the number of nodes in the network. The degree centrality measure is easy to compute and provides a simple way of identifying important nodes in a social ego network.

Closeness centrality:

Closeness centrality is a measure of node importance in a social ego network that is based on the distance or shortest path between a node and all other nodes in the network. A node with a high closeness centrality is located close to other nodes and can be seen as influential or central in the network.

To calculate the closeness centrality of a node, we first calculate the shortest path between that node and all other nodes in the network. We then sum the length of all these paths and divide by the total number of nodes in the network minus one. Finally, we take the reciprocal of this value to obtain the closeness centrality measure.

The closeness centrality measure reflects the average distance or time it takes for a node to reach all other nodes in the network. It captures the idea that nodes that are closer to other nodes in the network are more likely to be important or influential, as they can communicate or exchange information more easily.

Closeness centrality is a useful measure in social ego network analysis, as it can be used to identify nodes that are geographically or socially central in the network, to explore the accessibility or reachability of nodes in the network, and to compare the centrality of nodes across different networks or time periods.

Betweenness centrality:

Betweenness centrality is a measure of node importance in a social ego network that is based on the number of times a node acts as a bridge or intermediary between other nodes in the network. A node with a high betweenness centrality is situated on many shortest paths between other nodes and can be seen as influential or central in the network.

To calculate the betweenness centrality of a node, we first calculate all the shortest paths between pairs of nodes in the network. We then count the number of times that a node appears on a shortest path between two other nodes, and divide this number by the total number of shortest paths in the network. Finally, we normalize this value by dividing by the maximum possible betweenness centrality.

The betweenness centrality measure reflects the importance of a node in connecting different parts of the network, and capturing the idea that nodes that act as bridges or bottlenecks are more likely to be important or influential, as they control the flow of information or resources between different parts of the network.

Betweenness centrality is a useful measure in social ego network analysis, as it can be used to identify nodes that act as "brokers" or "gatekeepers" in the network, to explore the flow of information or resources in the network, and to compare the centrality of nodes across different networks or time periods.

Katz centrality:

Katz centrality is a measure of node importance in a social ego network that takes into account both the number of connections a node has and the importance of those connections. The Katz centrality measure assigns higher centrality scores to nodes that have connections to other nodes with high centrality scores.

To calculate the Katz centrality of a node, we sum the contributions of all paths of different lengths between that node and all other nodes in the network. The contribution of each path is given by a parameter α raised to the power of the length of the path, multiplied by the centrality score of the end node of the path. The parameter α is a damping factor that reduces the importance of longer paths.

The Katz centrality measure reflects the importance of a node in a network based on its ability to connect to other important nodes. It captures the idea that nodes that have connections to other important nodes are themselves likely to be important or influential.

Katz centrality is a useful measure in social ego network analysis, as it can be used to identify nodes that are well-connected to other important nodes in the network, to explore the flow of influence or resources in the network, and to compare the centrality of nodes across different networks or time periods.

Eigenvector centrality:

Eigen vector centrality is a measure of node importance in a social ego network that takes into account not only the number of connections a node has, but also the centrality of the nodes it is connected to. The eigen vector centrality measure assigns higher centrality scores to nodes that are connected to other highly central nodes.

To calculate the eigen vector centrality of a node, we start with an initial centrality score for each node in the network. We then update the centrality scores iteratively by taking a weighted average of the centrality scores of the nodes that are connected to each node. The weights of the average are determined by the strength of the connections between the nodes. This process continues until the centrality scores converge to a stable value.

The eigen vector centrality measure reflects the importance of a node in a network based on its connections to other highly central nodes. It captures the idea that nodes that are connected to other highly central nodes are themselves likely to be important or influential.

Eigen vector centrality is a useful measure in social ego network analysis, as it can be used to identify nodes that are well-connected to other important nodes in the network, to explore the flow of influence or resources in the network, and to compare the centrality of nodes across different networks or time periods.

- **page rank**

  PageRank is a measure of node importance in a social ego network that was originally developed by Google as part of its search algorithm. The PageRank measure assigns higher centrality scores to nodes that have many incoming links from other important nodes.

  To calculate the PageRank of a node in a social ego network, we start with an initial uniform distribution of scores across all nodes in the network. We then iteratively update the scores by taking a weighted average of the scores of the nodes that are connected to each node. The weights of the average are determined by the number and importance of the incoming links to each node. This process continues until the scores converge to a stable value.

The PageRank measure reflects the importance of a node in a network based on the quality and quantity of the incoming links to that node. It captures the idea that nodes that are linked to by other important nodes are themselves likely to be important or influential.

PageRank is a useful measure in social ego network analysis, as it can be used to identify nodes that have many high-quality incoming links, to explore the flow of influence or resources in the network, and to compare the centrality of nodes across different networks or time periods.

- **connectivity**

  Connectivity is a measure of the connectedness of a social ego network, reflecting the extent to which nodes in the network are linked to one another. In a social ego network, connectivity can be assessed at different levels, including the overall connectivity of the network, the connectivity of individual nodes, and the connectivity of different subgroups or communities within the network.

  The overall connectivity of a social ego network can be described in terms of metrics such as the density of the network, which reflects the proportion of possible connections that are present between nodes. A dense network has many connections between nodes, while a sparse network has relatively few connections. Other measures of overall connectivity include the clustering coefficient, which reflects the degree to which nodes tend to be connected to one another in groups or clusters, and the average path length, which reflects the typical number of steps required to traverse the network between any two nodes. Individual node connectivity in a social ego network can be assessed using centrality measures such as degree centrality, eigen vector centrality, or PageRank, as discussed in previous questions. These measures reflect the extent to which a node is linked to other nodes in the network, and can be used to identify nodes that are particularly well-connected or influential.

  Finally, the connectivity of subgroups or communities within a social ego network can be assessed using measures such as modularity or community detection algorithms. These measures identify groups of nodes that are more densely connected to one another than to nodes outside the group, and can be used to explore the structure and dynamics of social ego networks at different scales.

- **clustering**

  Clustering is a key concept in social ego network analysis, reflecting the extent to which nodes in a network tend to be connected to one another in clusters or groups. Clustering can be assessed at different levels, including the overall clustering of the network, the clustering of individual nodes, and the clustering of subgroups or communities within the network.

  The overall clustering of a social ego network can be assessed using metrics such as the clustering coefficient, which reflects the degree to which nodes tend to be connected to one another in clusters or triangles. A high clustering coefficient indicates that nodes in the network tend to be closely connected to one another, while a low clustering coefficient indicates that nodes are more sparsely connected.

  Individual node clustering in a social ego network can be assessed using measures such as the local clustering coefficient, which reflects the degree to which a node's neighbors are connected to one another. Nodes with high local clustering coefficients tend to be located in densely connected subgraphs or clusters, while nodes with low local clustering coefficients tend to be located in more sparsely connected regions of the network.

- **shortest path**

  The average shortest path is a key metric in social ego network analysis, reflecting the typical distance between nodes in a network. It is calculated as the average number of steps required to traverse the network between any two nodes.

  In a social ego network, the average shortest path can provide insights into the ease of communication or information flow between individuals. A smaller average shortest path indicates that nodes in the network are closer together, which can facilitate communication and the spread of information. In contrast, a larger average shortest path indicates that nodes are further apart, which can hinder communication and the spread of information.

  The average shortest path can be calculated using algorithms such as Dijkstra's algorithm or Floyd-Warshall algorithm. These algorithms determine the shortest path between two nodes in the network by calculating the minimum number of steps required to traverse the network between the nodes.

- **comparisons**

  G with Random Erdos-Renyi graph

  A common approach to assessing the structure of a social ego network is to compare it to a randomly generated graph known as an Erdos-Renyi graph. An Erdos-Renyi graph is a type of random graph in which nodes are connected to one another with a fixed probability.

  G with random Albert-Barabasi graph

  Another approach to assessing the structure of a social ego network is to compare it to a different type of random graph known as an Albert-Barabasi graph. An Albert-Barabasi graph is a type of random graph that is generated by a process called preferential attachment, in which new nodes tend to be connected to existing nodes with high degree.

  G with random watts-strogatz graph

  A third approach to assessing the structure of a social ego network is to compare it to a different type of random graph known as a Watts-Strogatz graph. A Watts-Strogatz graph is a type of random graph that is generated by a process called rewiring, in which existing edges are randomly rewired with a low probability.

- **Degree assortativity of a network**

  Degree assortativity is a measure of the extent to which nodes in a network tend to be connected to other nodes with similar degree. In other words, it is a measure of the correlation between the degrees of neighboring nodes in the network.

  A network is said to be degree assortative if nodes with high degree tend to be connected to other nodes with high degree, and nodes with low degree tend to be connected to other nodes with low degree. Conversely, a network is said to be degree disassortative if nodes with high degree tend to be connected to nodes with low degree, and vice versa.

  Degree assortativity is typically measured using the Pearson correlation coefficient between the degrees of neighboring nodes in the network. This coefficient ranges from -1 (indicating perfect disassortativity) to 1 (indicating perfect assortativity), with 0 indicating no correlation.

- **Stochastic SIR Epidemic on Static Network**

A Stochastic SIR Epidemic on a Static Network is a mathematical model used to simulate the spread of a contagious disease or infection within a population. The model is based on the Susceptible-Infected-Recovered (SIR) framework, which divides the population into three categories: those who are susceptible to the disease, those who are infected, and those who have recovered and are no longer susceptible.

The model assumes that the population is represented as a static network, with individuals represented as nodes and connections between individuals represented as edges. The model also assumes that the spread of the disease is stochastic, meaning that the likelihood of an individual becoming infected depends on a combination of their susceptibility to the disease and the probability of transmission from infected individuals.

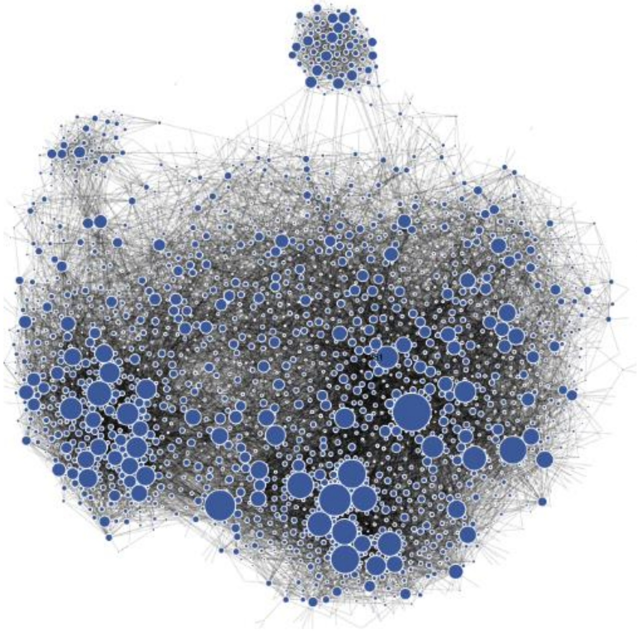
The model proceeds through a series of discrete time steps, during which infected individuals can transmit the disease to susceptible individuals with a certain probability. Infected individuals then recover after a certain amount of time and become immune to the disease. The simulation continues until all individuals in the population have either recovered or died.

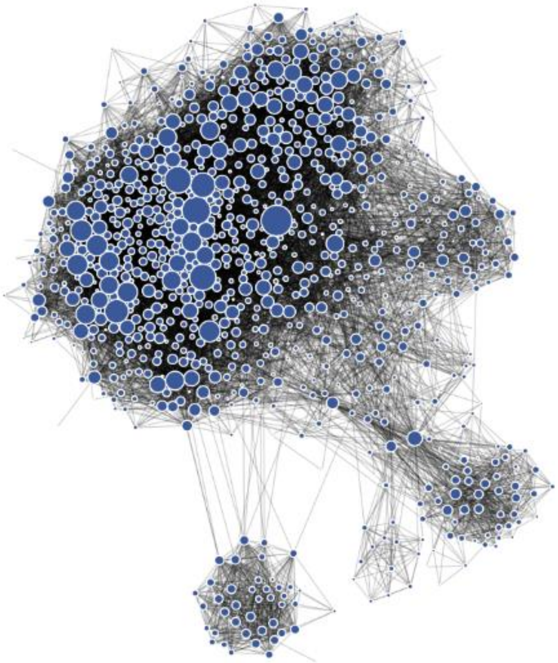
One important application of the Stochastic SIR Epidemic model is in the field of public health, where it is used to study the spread of infectious diseases and evaluate the effectiveness of different control measures. For example, the model can be used to study the impact of vaccination programs or quarantine measures on the spread of a disease, and to identify which interventions are most effective at reducing the overall impact of the disease on the population.

RESULTS

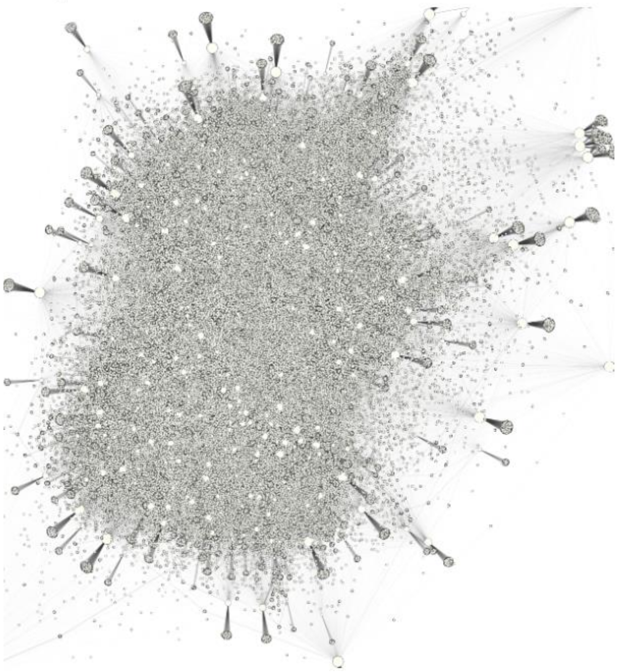
DATA VISUALIZATION

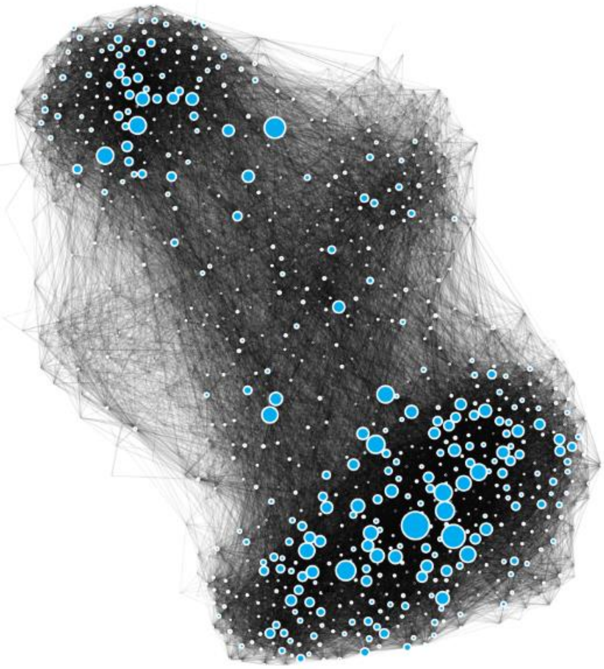
FACEBOOK





TWITTER

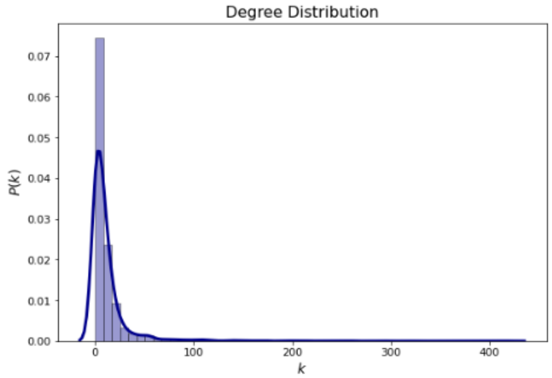




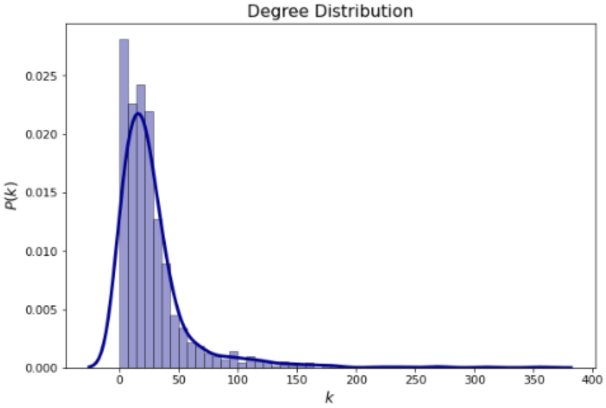
DEGREE DISTRIBUTION

FACEBOOK

User1 and user2



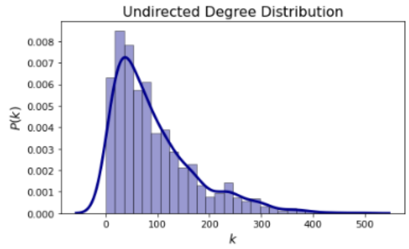
```
Mean =  12.017119464086342
Var =  624.7581736154269
```

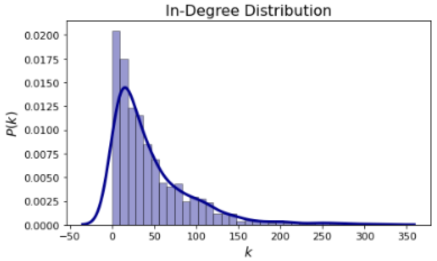


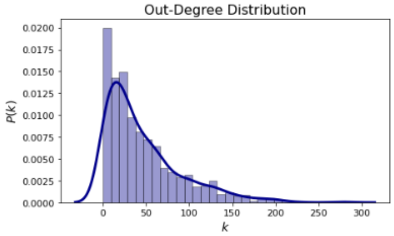
```
Mean =  27.939393939393938
Var =  1154.9700642791554
```

TWITTER

First order EGO







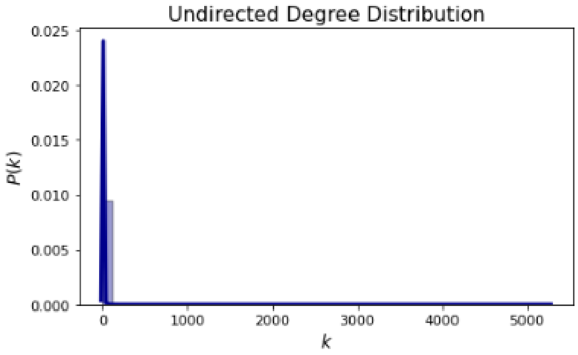
```
Mean =  92.04078164825829
Var =  6195.66018902369
```

```
Mean =  46.020390824129144
Var =  2305.0259223621247
```

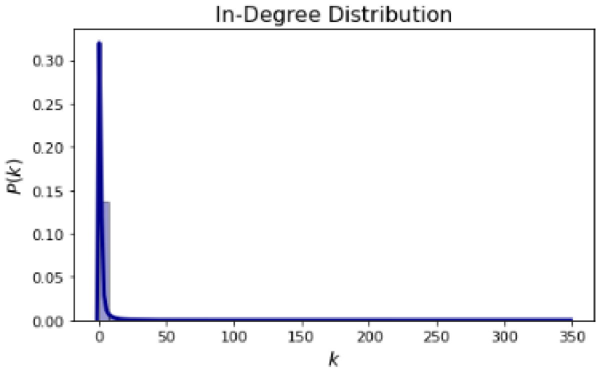
```
Mean =  46.020390824129144
Var =  1939.9231186238067
```

As expected $\langle k_{in} \rangle = \langle k_{out} \rangle$.

Second order EGO



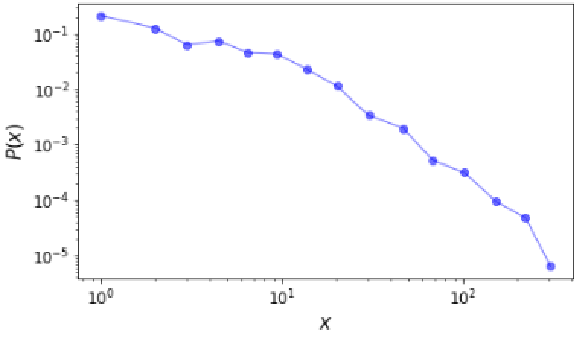
```
Mean =  4.552138072612988
Var =  5755.073795626525
```



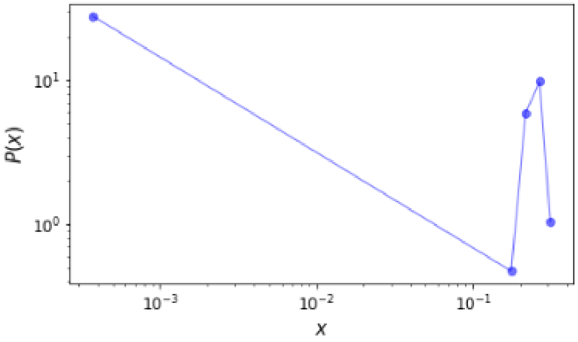
```
Mean =  2.276069036306494
Var =  33.98333234701862
```

FACEBOOK

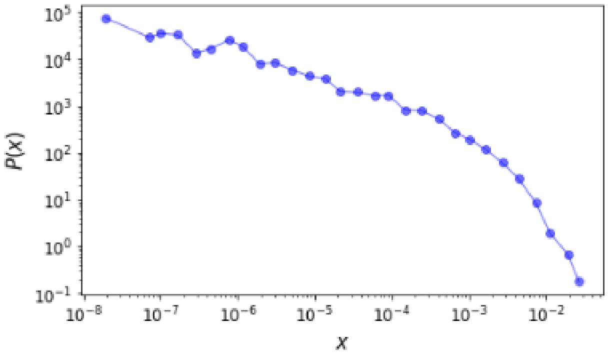
Degree centrality



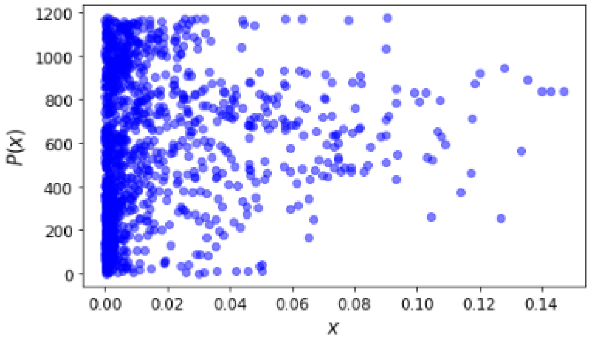
closeness centrality



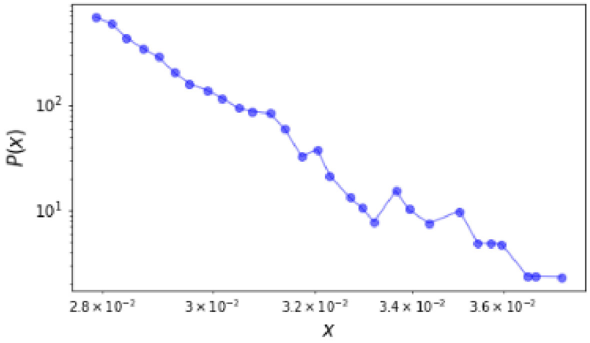
Betweeness centrality



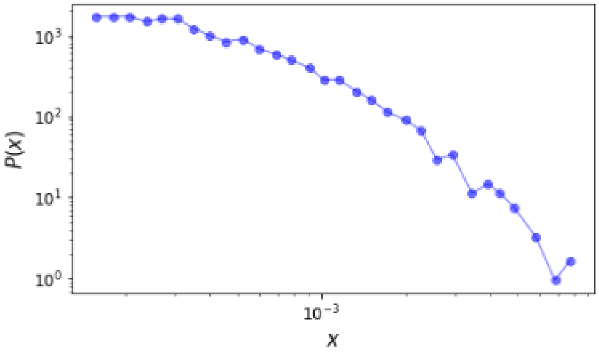
eigenvector centrality



Katz centrality

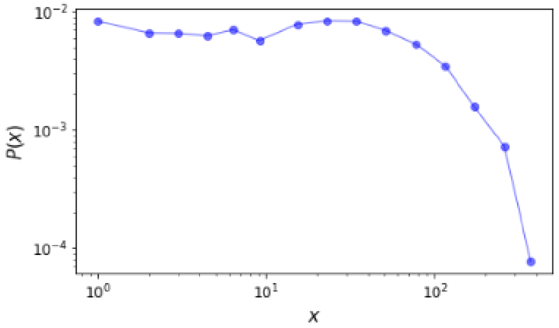


page rank

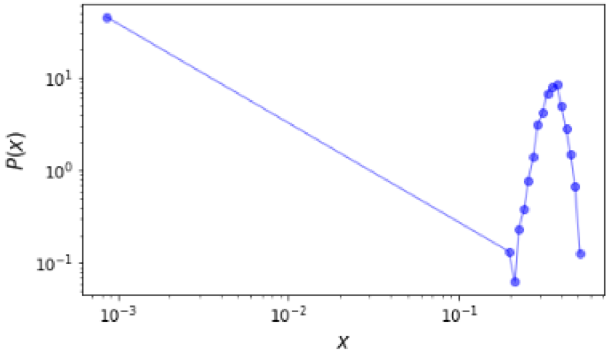


TWITTER

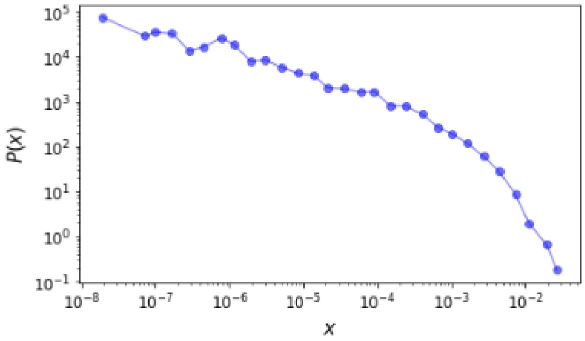
Degree centrality



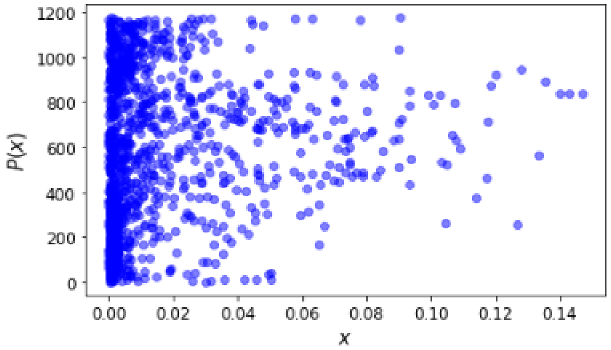
closeness centrality



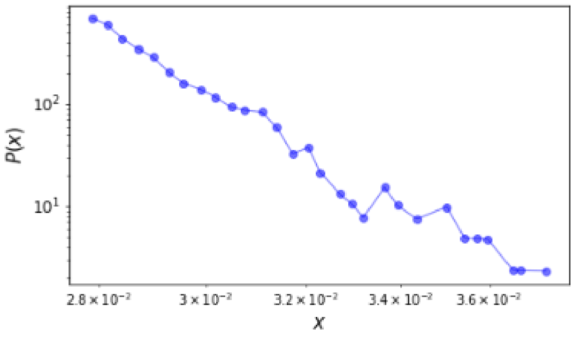
Betweeness centrality



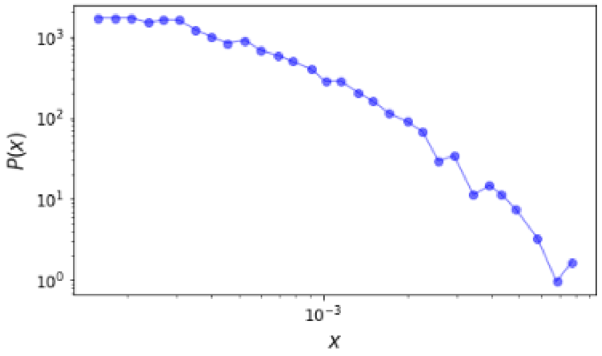
eigenvector centrality



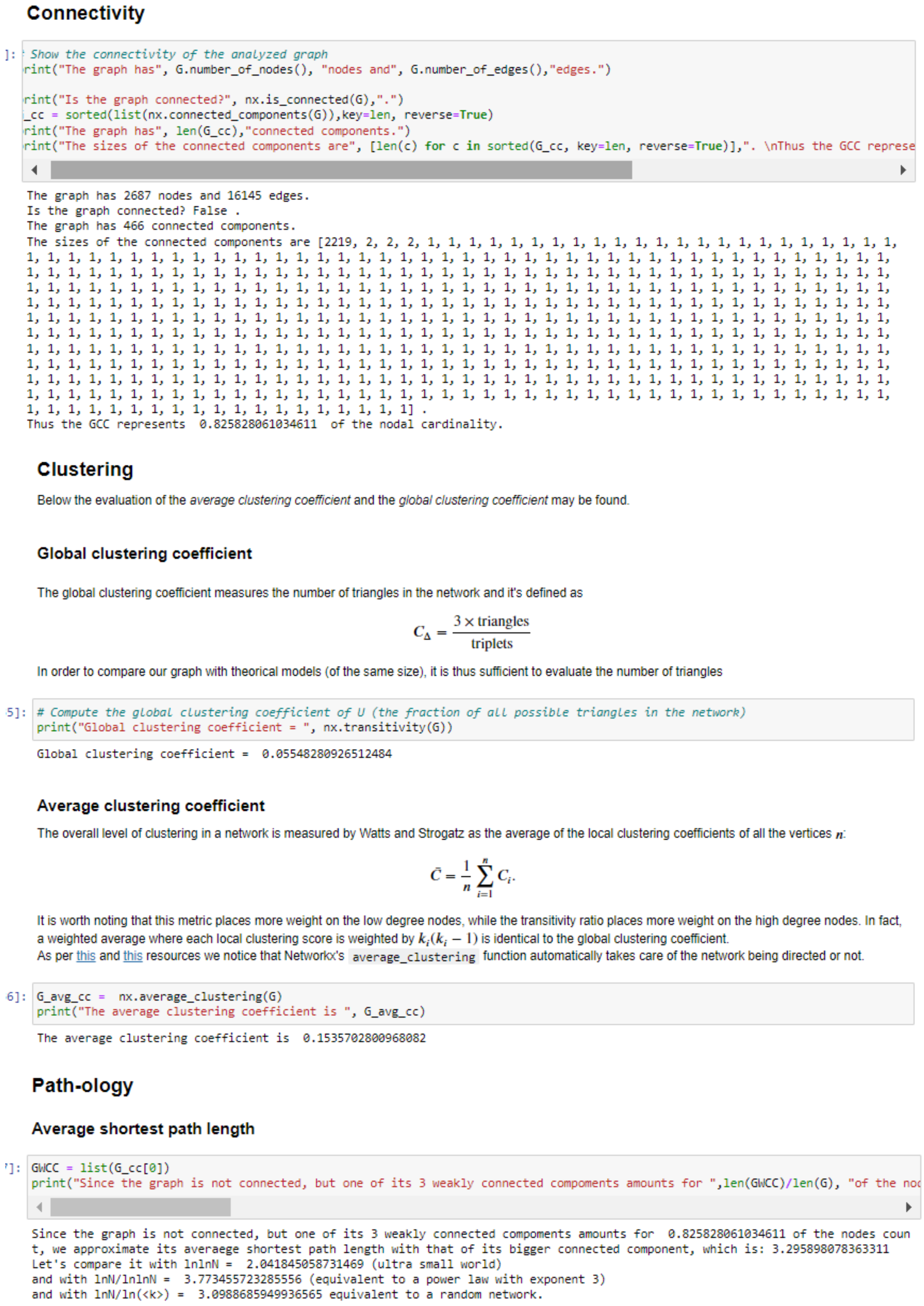
Katz centrality



page rank



FACEBOOK



Connectivity

```
]: ‡ Show the connectivity of the analyzed graph
   rint("The graph has", G.number_of_nodes(), "nodes and", G.number_of_edges(),"edges.")

   rint("Is the graph connected?", nx.is_connected(G),".")
   _cc = sorted(list(nx.connected_components(G)),key=len, reverse=True)
   rint("The graph has", len(G_cc),"connected components.")
   rint("The sizes of the connected components are", [len(c) for c in sorted(G_cc, key=len, reverse=True)],". \nThus the GCC represe
```

```
The graph has 2687 nodes and 16145 edges.
Is the graph connected? False .
The graph has 466 connected components.
The sizes of the connected components are [2219, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] .
Thus the GCC represents  0.825828061034611  of the nodal cardinality.
```

Clustering

Below the evaluation of the *average clustering coefficient* and the *global clustering coefficient* may be found.

Global clustering coefficient

The global clustering coefficient measures the number of triangles in the network and it's defined as

$$C_\Delta = \frac{3 \times \text{triangles}}{\text{triplets}}$$

In order to compare our graph with theorical models (of the same size), it is thus sufficient to evaluate the number of triangles

```
5]: # Compute the global clustering coefficient of U (the fraction of all possible triangles in the network)
    print("Global clustering coefficient = ", nx.transitivity(G))
```

```
Global clustering coefficient =  0.05548280926512484
```

Average clustering coefficient

The overall level of clustering in a network is measured by Watts and Strogatz as the average of the local clustering coefficients of all the vertices n:

$$\bar{C} = \frac{1}{n}\sum_{i=1}^{n} C_i.$$

It is worth noting that this metric places more weight on the low degree nodes, while the transitivity ratio places more weight on the high degree nodes. In fact, a weighted average where each local clustering score is weighted by $k_i(k_i - 1)$ is identical to the global clustering coefficient.
As per this and this resources we notice that Networkx's `average_clustering` function automatically takes care of the network being directed or not.

```
6]: G_avg_cc =  nx.average_clustering(G)
    print("The average clustering coefficient is ", G_avg_cc)
```

```
The average clustering coefficient is  0.1535702800968082
```

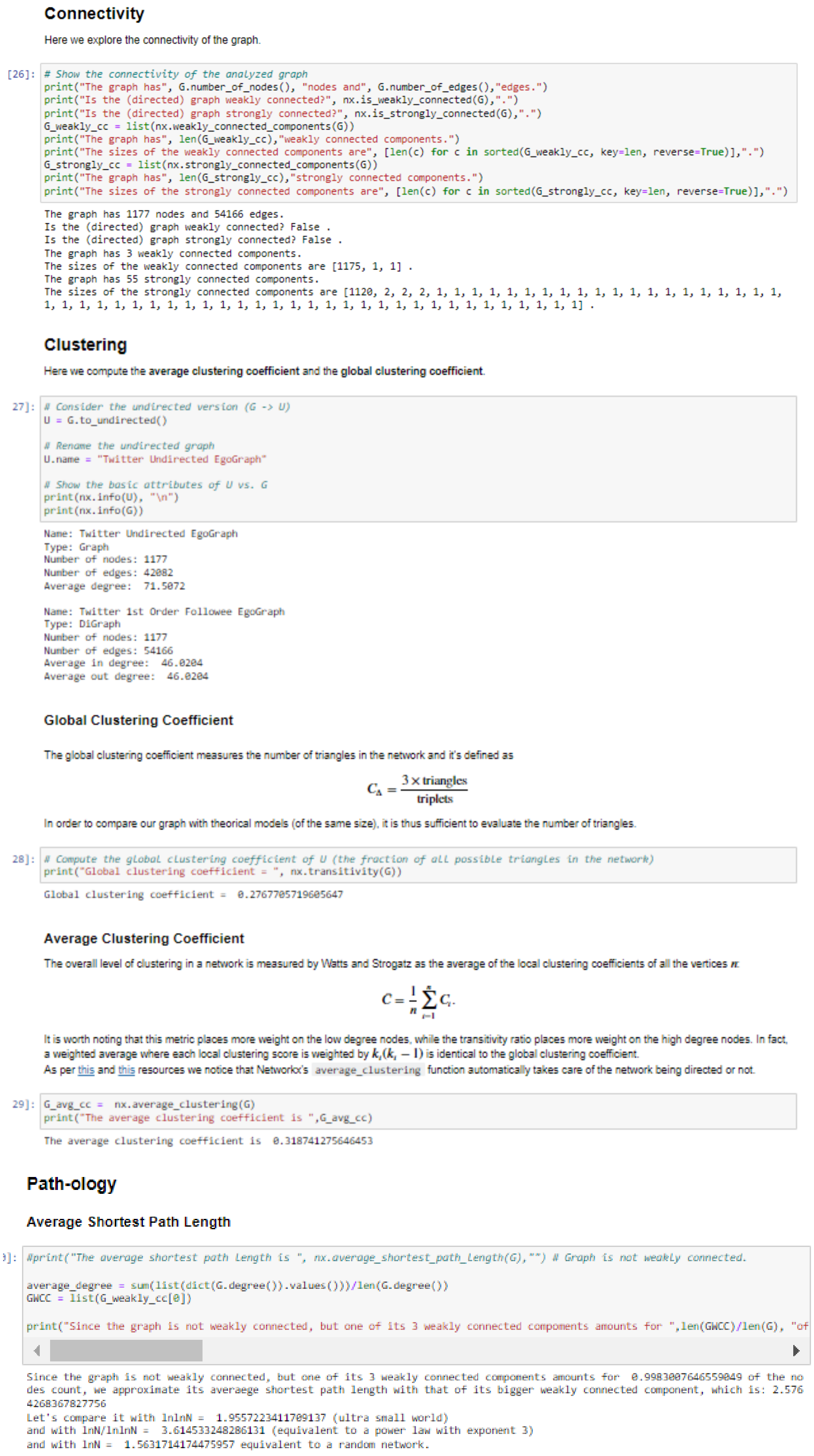
Path-ology

Average shortest path length

```
']: GWCC = list(G_cc[0])
    print("Since the graph is not connected, but one of its 3 weakly connected compoments amounts for ",len(GWCC)/len(G), "of the noc
```

```
Since the graph is not connected, but one of its 3 weakly connected compoments amounts for  0.825828061034611 of the nodes coun
t, we approximate its averaege shortest path length with that of its bigger connected component, which is: 3.295898078363311
Let's compare it with lnlnN =  2.041845058731469 (ultra small world)
and with lnN/lnlnN =  3.773455723285556 (equivalent to a power law with exponent 3)
and with lnN/ln(<k>) =  3.0988685949936565 equivalent to a random network.
```

TWITTER



Connectivity

Here we explore the connectivity of the graph.

```
[26]: # Show the connectivity of the analyzed graph
print("The graph has", G.number_of_nodes(), "nodes and", G.number_of_edges(),"edges.")
print("Is the (directed) graph weakly connected?", nx.is_weakly_connected(G),".")
print("Is the (directed) graph strongly connected?", nx.is_strongly_connected(G),".")
G_weakly_cc = list(nx.weakly_connected_components(G))
print("The graph has", len(G_weakly_cc),"weakly connected components.")
print("The sizes of the weakly connected components are", [len(c) for c in sorted(G_weakly_cc, key=len, reverse=True)],".")
G_strongly_cc = list(nx.strongly_connected_components(G))
print("The graph has", len(G_strongly_cc),"strongly connected components.")
print("The sizes of the strongly connected components are", [len(c) for c in sorted(G_strongly_cc, key=len, reverse=True)],".")
```

```
The graph has 1177 nodes and 54166 edges.
Is the (directed) graph weakly connected? False .
Is the (directed) graph strongly connected? False .
The graph has 3 weakly connected components.
The sizes of the weakly connected components are [1175, 1, 1] .
The graph has 55 strongly connected components.
The sizes of the strongly connected components are [1120, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] .
```

Clustering

Here we compute the **average clustering coefficient** and the **global clustering coefficient**.

```
27]: # Consider the undirected version (G -> U)
U = G.to_undirected()

# Rename the undirected graph
U.name = "Twitter Undirected EgoGraph"

# Show the basic attributes of U vs. G
print(nx.info(U), "\n")
print(nx.info(G))
```

```
Name: Twitter Undirected EgoGraph
Type: Graph
Number of nodes: 1177
Number of edges: 42082
Average degree:  71.5072

Name: Twitter 1st Order Followee EgoGraph
Type: DiGraph
Number of nodes: 1177
Number of edges: 54166
Average in degree:  46.0204
Average out degree:  46.0204
```

Global Clustering Coefficient

The global clustering coefficient measures the number of triangles in the network and it's defined as

$$C_\Delta = \frac{3 \times \text{triangles}}{\text{triplets}}$$

In order to compare our graph with theorical models (of the same size), it is thus sufficient to evaluate the number of triangles.

```
28]: # Compute the global clustering coefficient of U (the fraction of all possible triangles in the network)
print("Global clustering coefficient = ", nx.transitivity(G))
```

```
Global clustering coefficient =  0.2767705719605647
```

Average Clustering Coefficient

The overall level of clustering in a network is measured by Watts and Strogatz as the average of the local clustering coefficients of all the vertices n:

$$C = \frac{1}{n} \sum_{i=1}^{n} C_i.$$

It is worth noting that this metric places more weight on the low degree nodes, while the transitivity ratio places more weight on the high degree nodes. In fact, a weighted average where each local clustering score is weighted by $k_i(k_i - 1)$ is identical to the global clustering coefficient.
As per this and this resources we notice that Networkx's `average_clustering` function automatically takes care of the network being directed or not.

```
29]: G_avg_cc =  nx.average_clustering(G)
print("The average clustering coefficient is ",G_avg_cc)
```

```
The average clustering coefficient is  0.318741275646453
```

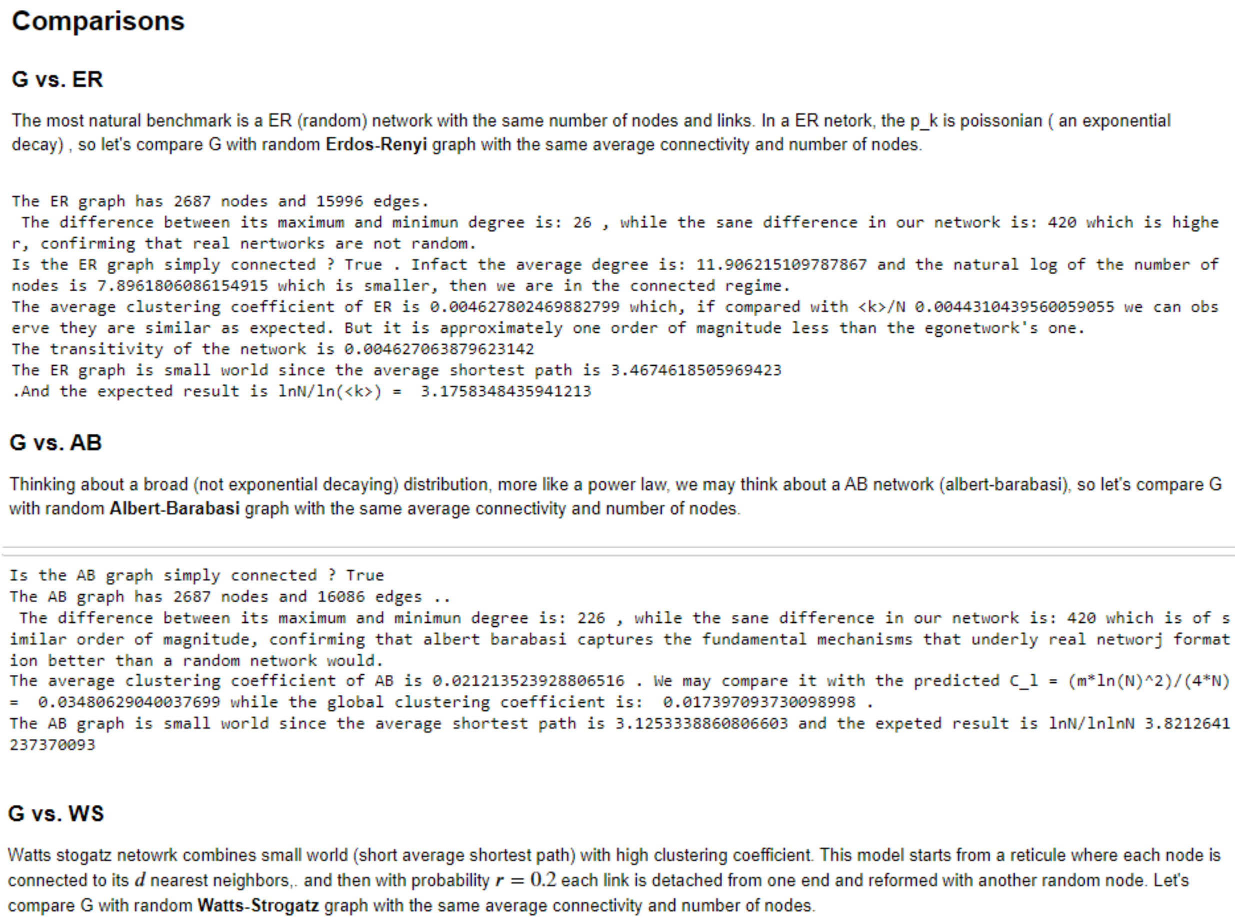
Path-ology

Average Shortest Path Length

```
]: #print("The average shortest path length is ", nx.average_shortest_path_length(G),"") # Graph is not weakly connected.

average_degree = sum(list(dict(G.degree()).values()))/len(G.degree())
GWCC = list(G_weakly_cc[0])

print("Since the graph is not weakly connected, but one of its 3 weakly connected compoments amounts for ",len(GWCC)/len(G), "of
```

```
Since the graph is not weakly connected, but one of its 3 weakly connected compoments amounts for  0.9983007646559049 of the no
des count, we approximate its averaege shortest path length with that of its bigger weakly connected component, which is: 2.576
4268367827756
Let's compare it with lnlnN =  1.9557223411709137 (ultra small world)
and with lnN/lnlnN =  3.614533248286131 (equivalent to a power law with exponent 3)
and with lnN =  1.5631714174475957 equivalent to a random network.
```

FACEBOOK



Comparisons

G vs. ER

The most natural benchmark is a ER (random) network with the same number of nodes and links. In a ER netork, the p_k is poissonian (an exponential decay) , so let's compare G with random **Erdos-Renyi** graph with the same average connectivity and number of nodes.

```
The ER graph has 2687 nodes and 15996 edges.
 The difference between its maximum and minimun degree is: 26 , while the sane difference in our network is: 420 which is highe
r, confirming that real nertworks are not random.
Is the ER graph simply connected ? True . Infact the average degree is: 11.906215109787867 and the natural log of the number of
nodes is 7.8961806086154915 which is smaller, then we are in the connected regime.
The average clustering coefficient of ER is 0.004627802469882799 which, if compared with <k>/N 0.004431043956059055 we can obs
erve they are similar as expected. But it is approximately one order of magnitude less than the egonetwork's one.
The transitivity of the network is 0.004627063879623142
The ER graph is small world since the average shortest path is 3.4674618505969423
.And the expected result is lnN/ln(<k>) =  3.1758348435941213
```

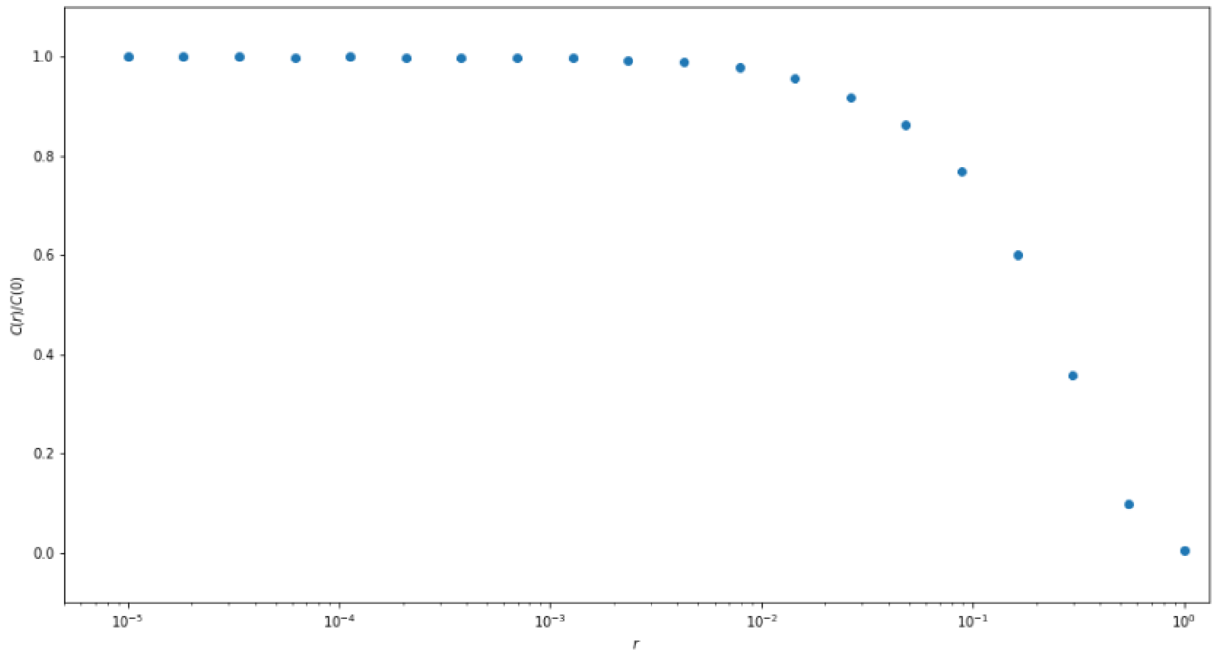
G vs. AB

Thinking about a broad (not exponential decaying) distribution, more like a power law, we may think about a AB network (albert-barabasi), so let's compare G with random **Albert-Barabasi** graph with the same average connectivity and number of nodes.

```
Is the AB graph simply connected ? True
The AB graph has 2687 nodes and 16086 edges ..
 The difference between its maximum and minimun degree is: 226 , while the sane difference in our network is: 420 which is of s
imilar order of magnitude, confirming that albert barabasi captures the fundamental mechanisms that underly real networj format
ion better than a random network would.
The average clustering coefficient of AB is 0.021213523928806516 . We may compare it with the predicted C_l = (m*ln(N)^2)/(4*N)
=  0.03480629040037699 while the global clustering coefficient is:  0.017397093730098998 .
The AB graph is small world since the average shortest path is 3.1253338860806603 and the expeted result is lnN/lnlnN 3.8212641
237370093
```

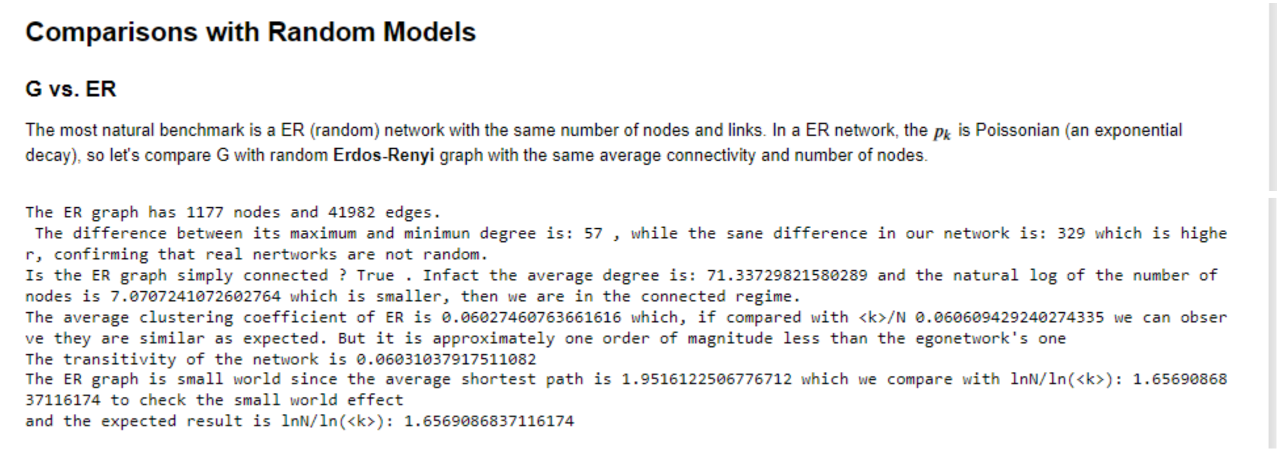
G vs. WS

Watts stogatz netowrk combines small world (short average shortest path) with high clustering coefficient. This model starts from a reticule where each node is connected to its d nearest neighbors,. and then with probability $r = 0.2$ each link is detached from one end and reformed with another random node. Let's compare G with random **Watts-Strogatz** graph with the same average connectivity and number of nodes.



```
best rewiring rate =  0.5455594781168515
best_avg_cc =  0.06767705171854975 ( 0.08589322837825845 apart from G's one)
```

TWITTER

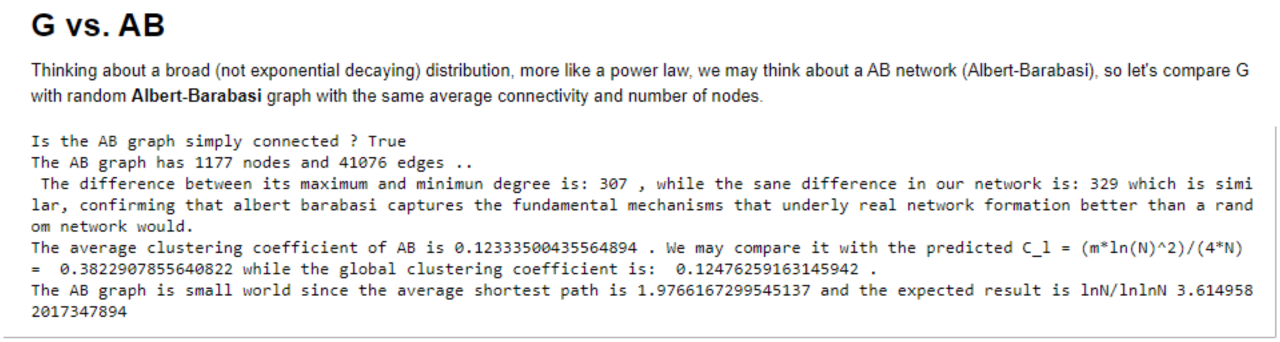


Comparisons with Random Models

G vs. ER

The most natural benchmark is a ER (random) network with the same number of nodes and links. In a ER network, the p_k is Poissonian (an exponential decay), so let's compare G with random **Erdos-Renyi** graph with the same average connectivity and number of nodes.

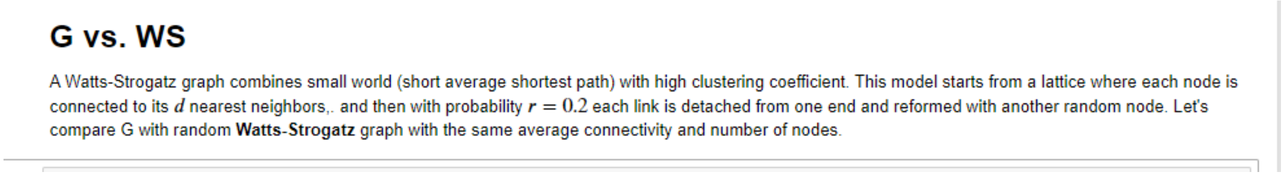
```
The ER graph has 1177 nodes and 41982 edges.
 The difference between its maximum and minimun degree is: 57 , while the sane difference in our network is: 329 which is highe
r, confirming that real nertworks are not random.
Is the ER graph simply connected ? True . Infact the average degree is: 71.33729821580289 and the natural log of the number of
nodes is 7.0707241072602764 which is smaller, then we are in the connected regime.
The average clustering coefficient of ER is 0.06027460763661616 which, if compared with <k>/N 0.060609429240274335 we can obser
ve they are similar as expected. But it is approximately one order of magnitude less than the egonetwork's one
The transitivity of the network is 0.06031037917511082
The ER graph is small world since the average shortest path is 1.9516122506776712 which we compare with lnN/ln(<k>): 1.65690868
37116174 to check the small world effect
and the expected result is lnN/ln(<k>): 1.6569086837116174
```



G vs. AB

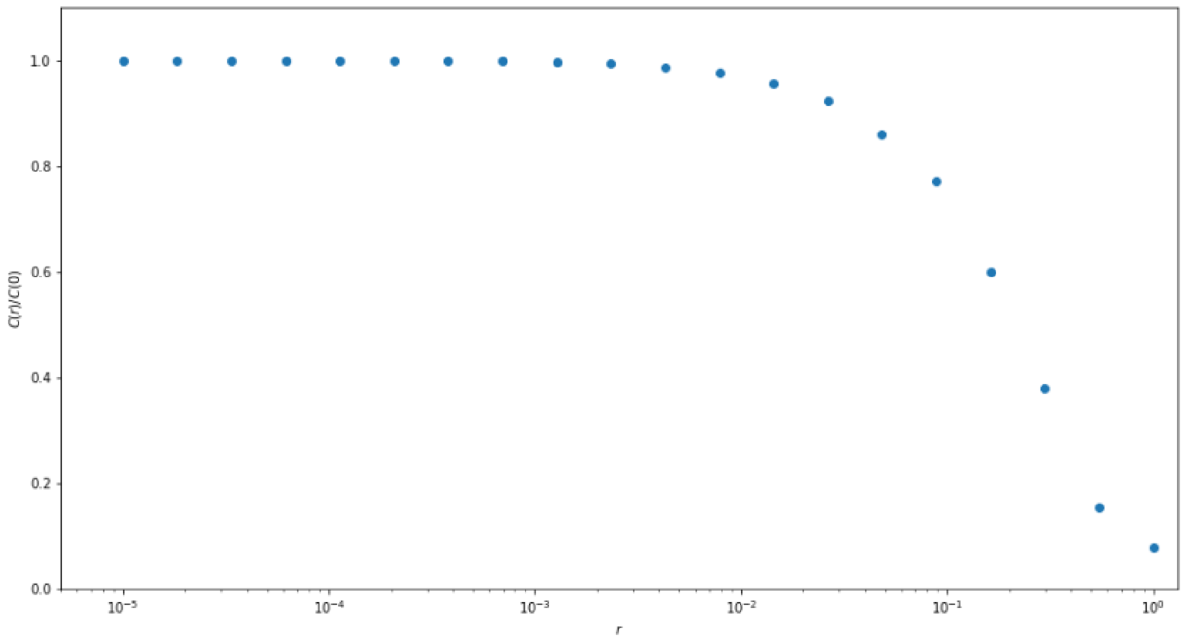
Thinking about a broad (not exponential decaying) distribution, more like a power law, we may think about a AB network (Albert-Barabasi), so let's compare G with random **Albert-Barabasi** graph with the same average connectivity and number of nodes.

```
Is the AB graph simply connected ? True
The AB graph has 1177 nodes and 41076 edges ..
 The difference between its maximum and minimun degree is: 307 , while the sane difference in our network is: 329 which is simi
lar, confirming that albert barabasi captures the fundamental mechanisms that underly real network formation better than a rand
om network would.
The average clustering coefficient of AB is 0.12333500435564894 . We may compare it with the predicted C_l = (m*ln(N)^2)/(4*N)
=  0.3822907855640822 while the global clustering coefficient is:  0.12476259163145942 .
The AB graph is small world since the average shortest path is 1.9766167299545137 and the expected result is lnN/lnlnN 3.614958
2017347894
```



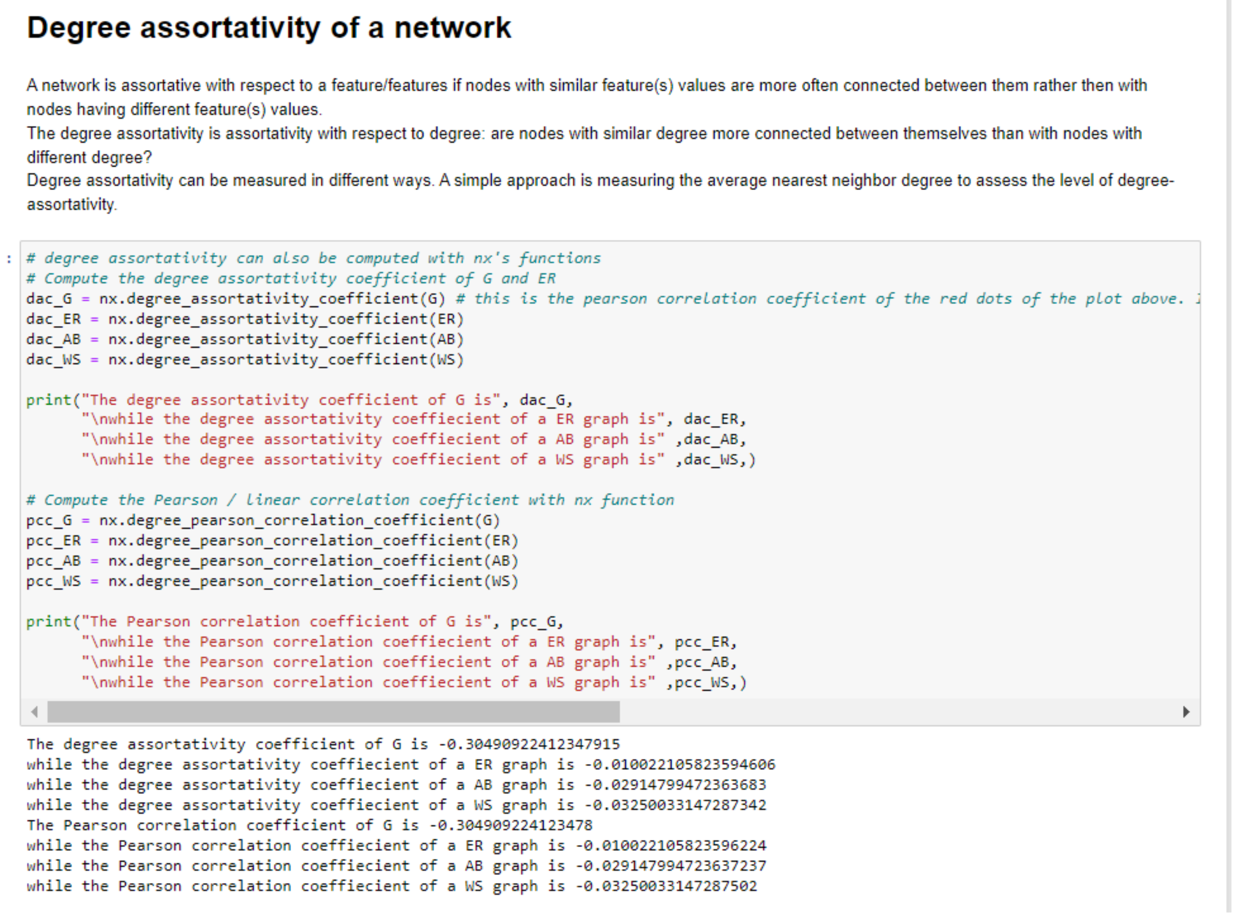
G vs. WS

A Watts-Strogatz graph combines small world (short average shortest path) with high clustering coefficient. This model starts from a lattice where each node is connected to its d nearest neighbors,. and then with probability $r = 0.2$ each link is detached from one end and reformed with another random node. Let's compare G with random **Watts-Strogatz** graph with the same average connectivity and number of nodes.



```
best rewiring rate =  0.2976351441631319
best_avg_cc =  0.2796770525477853 ( 0.039064223098667694 apart from G's one)
```

FACEBOOK



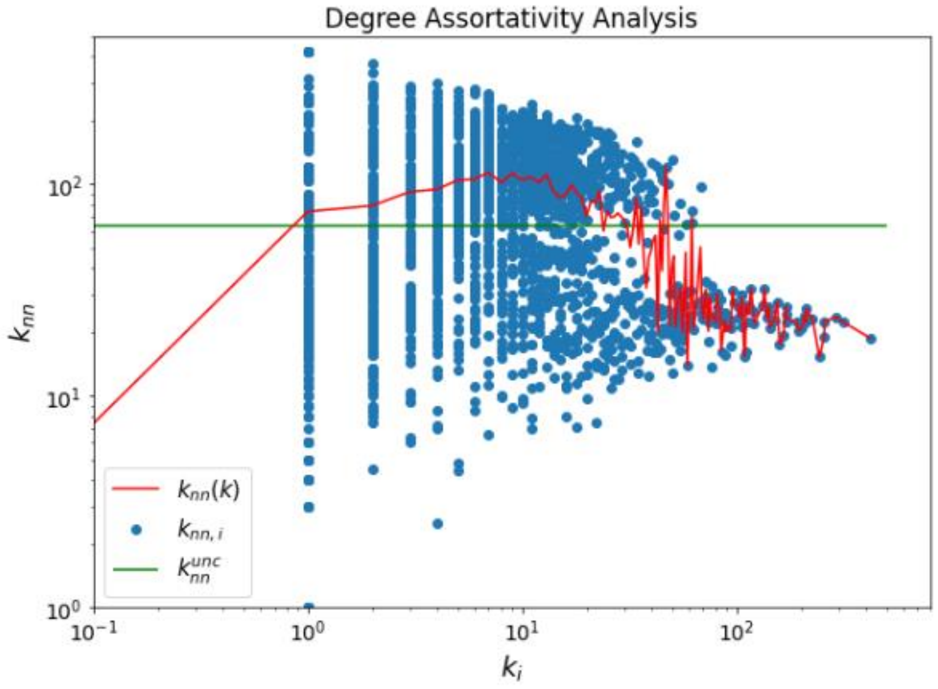
Degree assortativity of a network

A network is assortative with respect to a feature/features if nodes with similar feature(s) values are more often connected between them rather then with nodes having different feature(s) values.
The degree assortativity is assortativity with respect to degree: are nodes with similar degree more connected between themselves than with nodes with different degree?
Degree assortativity can be measured in different ways. A simple approach is measuring the average nearest neighbor degree to assess the level of degree-assortativity.

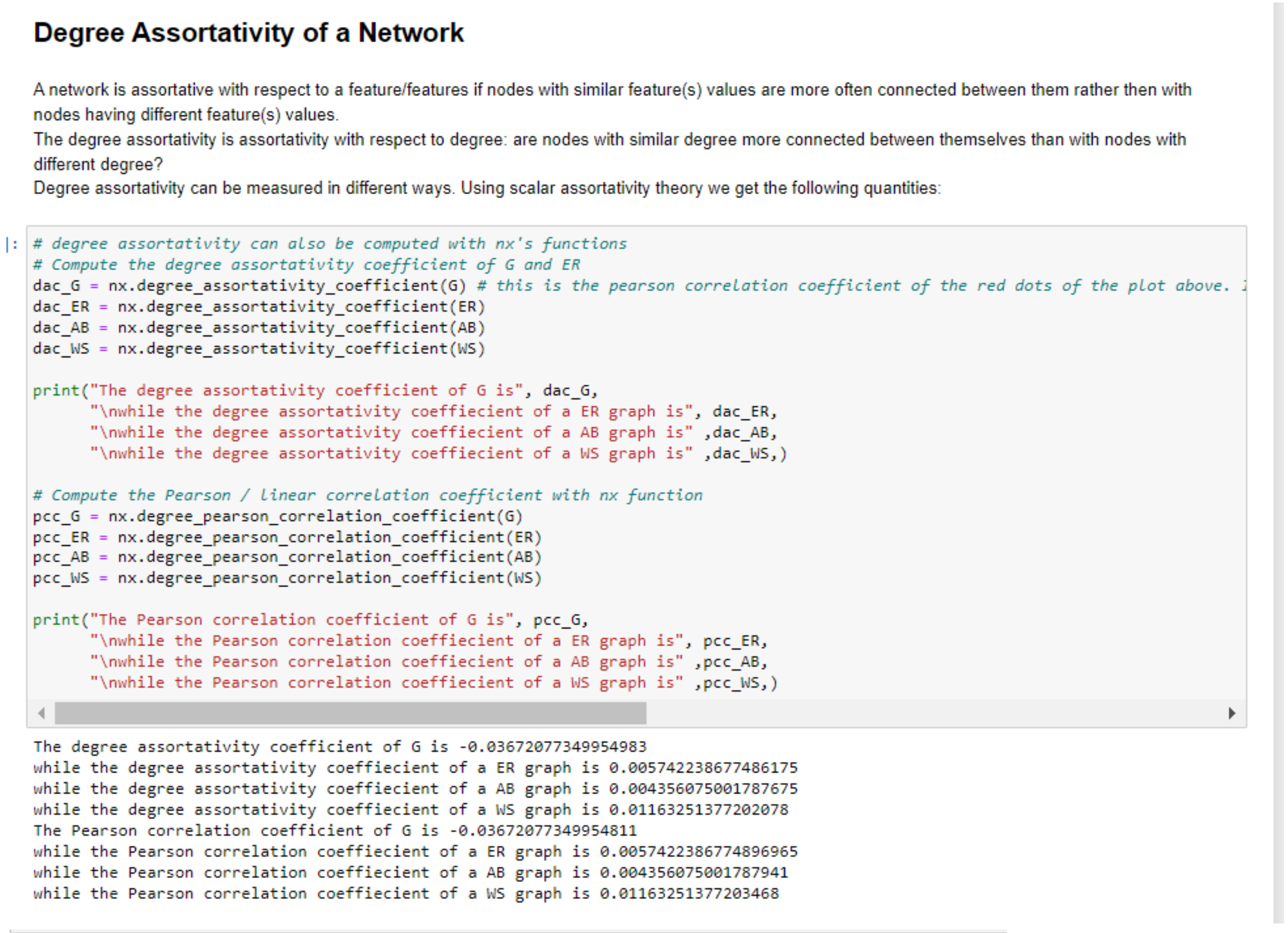
```
# degree assortativity can also be computed with nx's functions
# Compute the degree assortativity coefficient of G and ER
dac_G = nx.degree_assortativity_coefficient(G) # this is the pearson correlation coefficient of the red dots of the plot above. i
dac_ER = nx.degree_assortativity_coefficient(ER)
dac_AB = nx.degree_assortativity_coefficient(AB)
dac_WS = nx.degree_assortativity_coefficient(WS)

print("The degree assortativity coefficient of G is", dac_G,
      "\nwhile the degree assortativity coeffiecient of a ER graph is", dac_ER,
      "\nwhile the degree assortativity coeffiecient of a AB graph is" ,dac_AB,
      "\nwhile the degree assortativity coeffiecient of a WS graph is" ,dac_WS,)

# Compute the Pearson / linear correlation coefficient with nx function
pcc_G = nx.degree_pearson_correlation_coefficient(G)
pcc_ER = nx.degree_pearson_correlation_coefficient(ER)
pcc_AB = nx.degree_pearson_correlation_coefficient(AB)
pcc_WS = nx.degree_pearson_correlation_coefficient(WS)

print("The Pearson correlation coefficient of G is", pcc_G,
      "\nwhile the Pearson correlation coeffiecient of a ER graph is", pcc_ER,
      "\nwhile the Pearson correlation coeffiecient of a AB graph is" ,pcc_AB,
      "\nwhile the Pearson correlation coeffiecient of a WS graph is" ,pcc_WS,)
```

```
The degree assortativity coefficient of G is -0.30490922412347915
while the degree assortativity coeffiecient of a ER graph is -0.010022105823594606
while the degree assortativity coeffiecient of a AB graph is -0.02914799472363683
while the degree assortativity coeffiecient of a WS graph is -0.03250033147287342
The Pearson correlation coefficient of G is -0.304909224123478
while the Pearson correlation coeffiecient of a ER graph is -0.010022105823596224
while the Pearson correlation coeffiecient of a AB graph is -0.029147994723637237
while the Pearson correlation coeffiecient of a WS graph is -0.03250033147287502
```

```
k_unc =  64.0061319293899
```



TWITTER



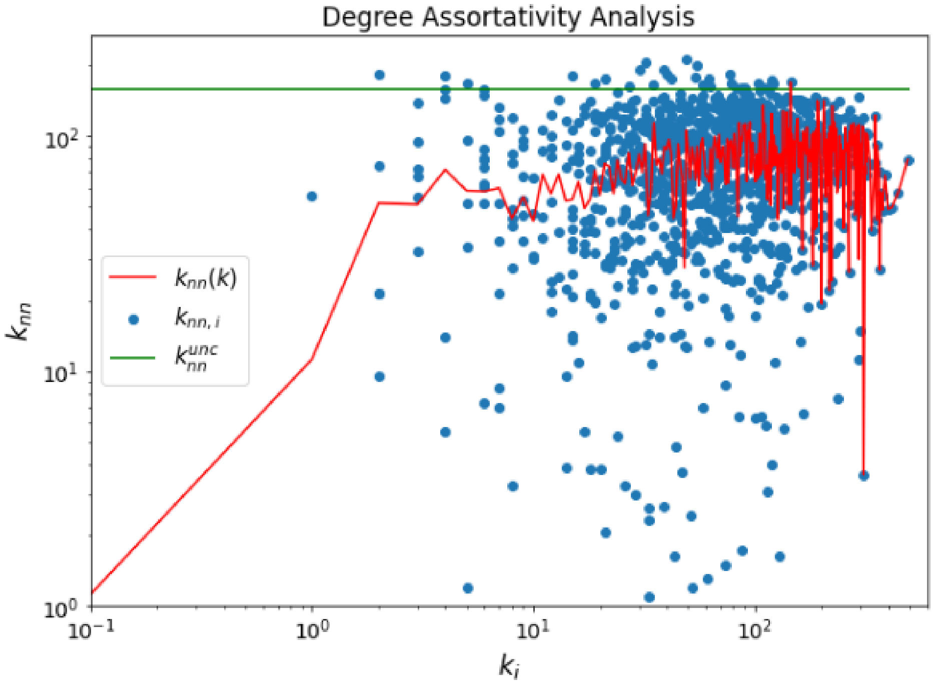
Degree Assortativity of a Network

A network is assortative with respect to a feature/features if nodes with similar feature(s) values are more often connected between them rather then with nodes having different feature(s) values.
The degree assortativity is assortativity with respect to degree: are nodes with similar degree more connected between themselves than with nodes with different degree?
Degree assortativity can be measured in different ways. Using scalar assortativity theory we get the following quantities:

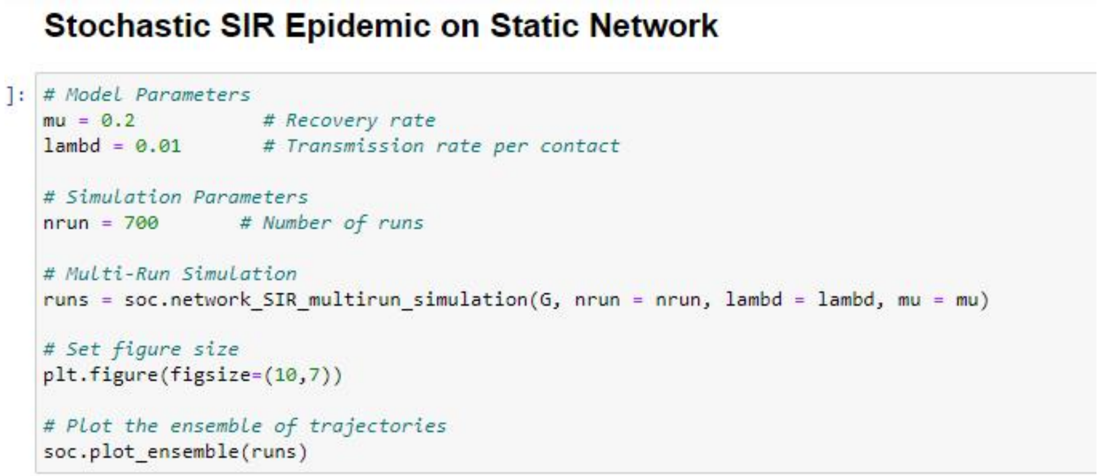
```
# degree assortativity can also be computed with nx's functions
# Compute the degree assortativity coefficient of G and ER
dac_G = nx.degree_assortativity_coefficient(G) # this is the pearson correlation coefficient of the red dots of the plot above. I
dac_ER = nx.degree_assortativity_coefficient(ER)
dac_AB = nx.degree_assortativity_coefficient(AB)
dac_WS = nx.degree_assortativity_coefficient(WS)

print("The degree assortativity coefficient of G is", dac_G,
      "\nwhile the degree assortativity coeffiecient of a ER graph is", dac_ER,
      "\nwhile the degree assortativity coeffiecient of a AB graph is" ,dac_AB,
      "\nwhile the degree assortativity coeffiecient of a WS graph is" ,dac_WS,)

# Compute the Pearson / linear correlation coefficient with nx function
pcc_G = nx.degree_pearson_correlation_coefficient(G)
pcc_ER = nx.degree_pearson_correlation_coefficient(ER)
pcc_AB = nx.degree_pearson_correlation_coefficient(AB)
pcc_WS = nx.degree_pearson_correlation_coefficient(WS)

print("The Pearson correlation coefficient of G is", pcc_G,
      "\nwhile the Pearson correlation coeffiecient of a ER graph is", pcc_ER,
      "\nwhile the Pearson correlation coeffiecient of a AB graph is" ,pcc_AB,
      "\nwhile the Pearson correlation coeffiecient of a WS graph is" ,pcc_WS,)
```

```
The degree assortativity coefficient of G is -0.03672077349954983
while the degree assortativity coeffiecient of a ER graph is 0.005742238677486175
while the degree assortativity coeffiecient of a AB graph is 0.004356075001787675
while the degree assortativity coeffiecient of a WS graph is 0.01163251377202078
The Pearson correlation coefficient of G is -0.03672077349954811
while the Pearson correlation coeffiecient of a ER graph is 0.005742238677489696
while the Pearson correlation coeffiecient of a AB graph is 0.004356075001787941
while the Pearson correlation coeffiecient of a WS graph is 0.01163251377203468
```

```
k_unc =  159.35507513938634
```

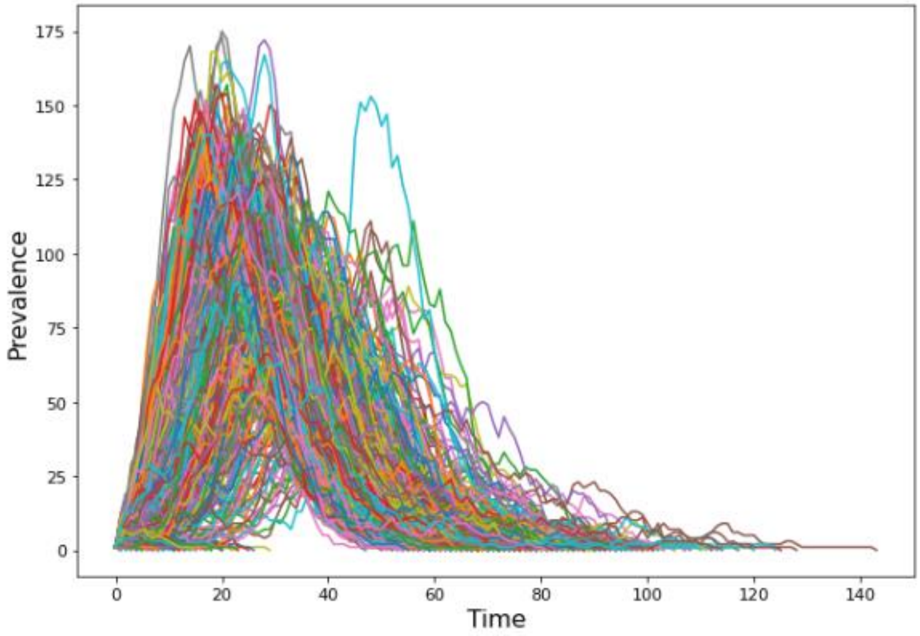


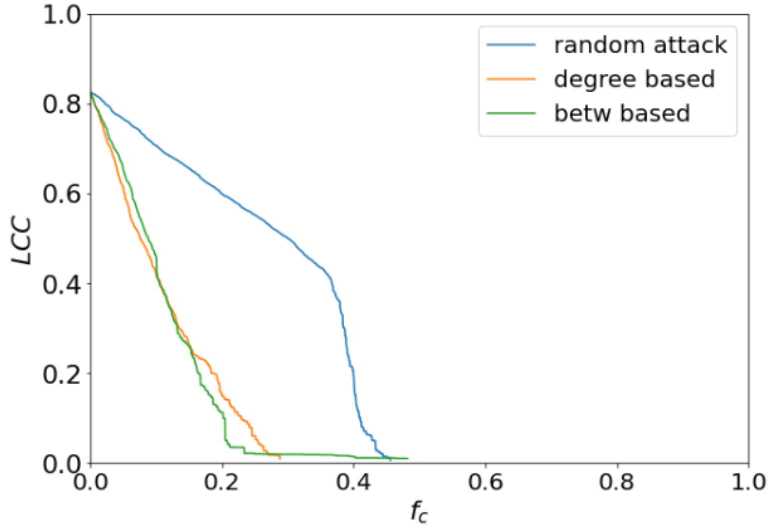
FACEBOOK



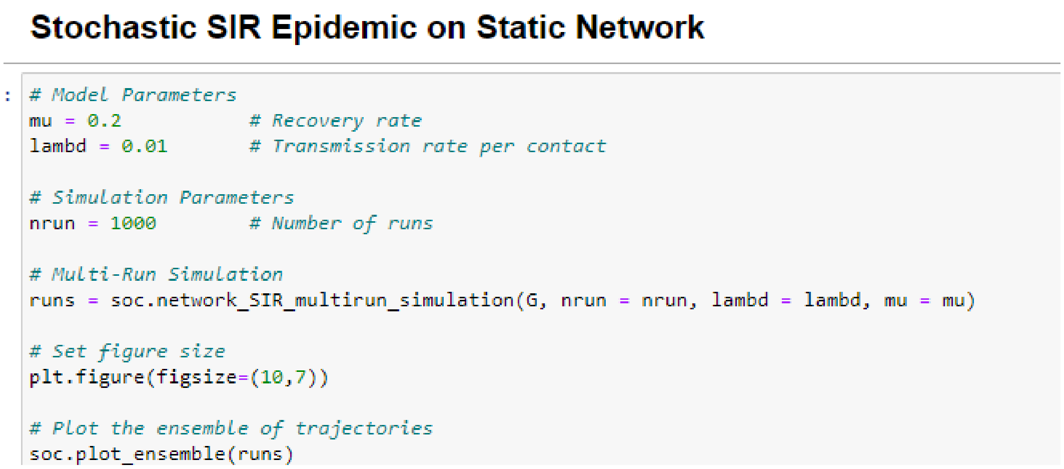
Stochastic SIR Epidemic on Static Network

```
]: # Model Parameters
   mu = 0.2           # Recovery rate
   lambd = 0.01       # Transmission rate per contact

   # Simulation Parameters
   nrun = 700        # Number of runs

   # Multi-Run Simulation
   runs = soc.network_SIR_multirun_simulation(G, nrun = nrun, lambd = lambd, mu = mu)

   # Set figure size
   plt.figure(figsize=(10,7))

   # Plot the ensemble of trajectories
   soc.plot_ensemble(runs)
```



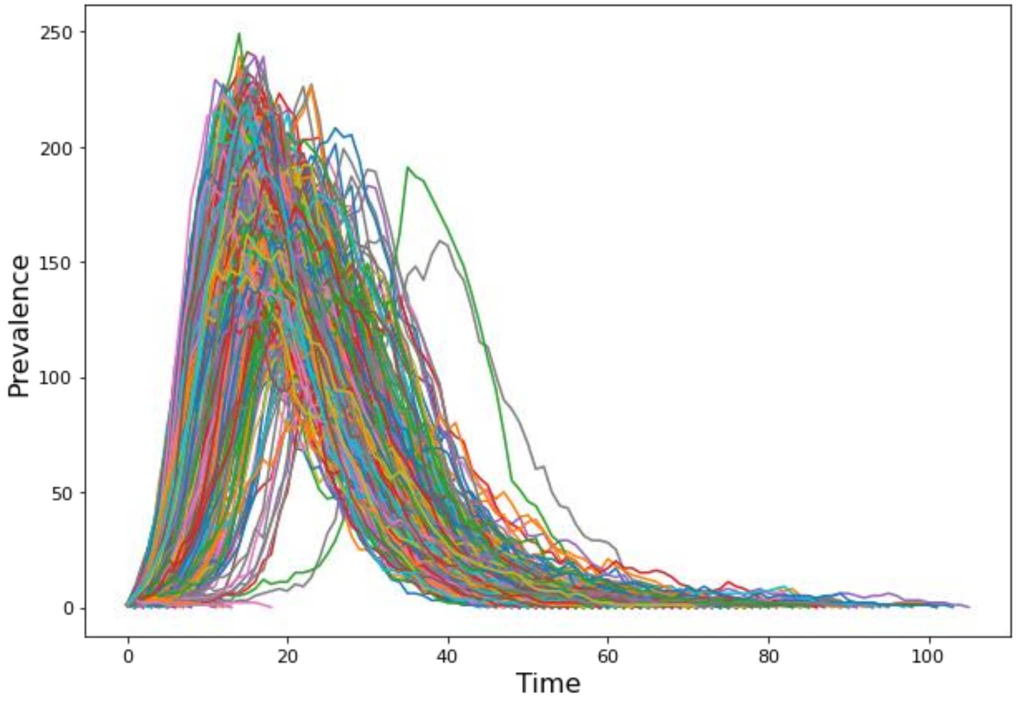


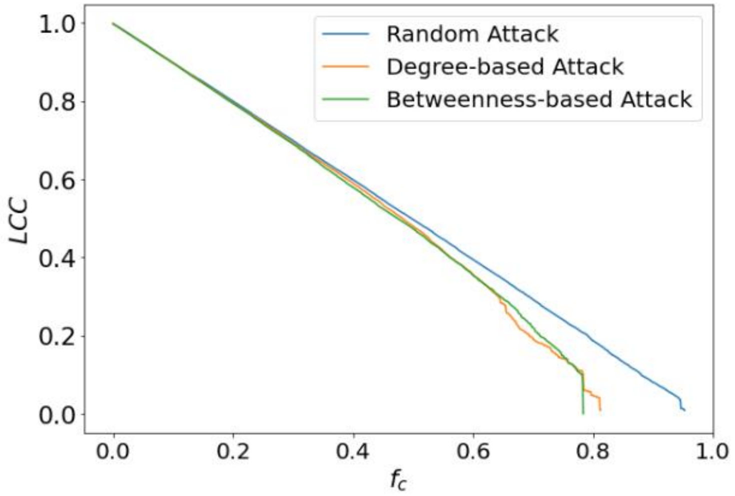
TWITTER



Stochastic SIR Epidemic on Static Network

```
# Model Parameters
mu = 0.2            # Recovery rate
lambd = 0.01        # Transmission rate per contact

# Simulation Parameters
nrun = 1000         # Number of runs

# Multi-Run Simulation
runs = soc.network_SIR_multirun_simulation(G, nrun = nrun, lambd = lambd, mu = mu)

# Set figure size
plt.figure(figsize=(10,7))

# Plot the ensemble of trajectories
soc.plot_ensemble(runs)
```





TWITTER

Community Detection

Modularity | `Gephi`

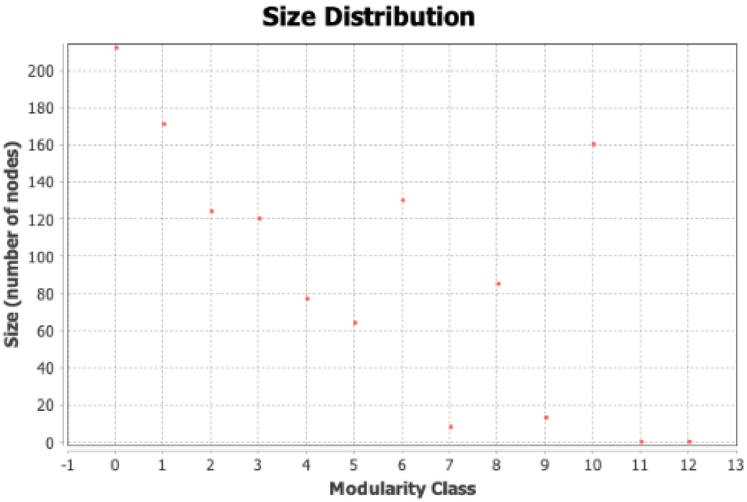
- **Algorithm**: Vincent D Blondel et al. Fast unfolding of communities in large networks, *Journal of Statistical Mechanics: Theory and Experiment* (2008).
- **Resolution**: R. Lambiotte et al.Barahona Laplacian Dynamics and Multiscale Modular Structure in Networks, *arXiv pre-print* (2009).

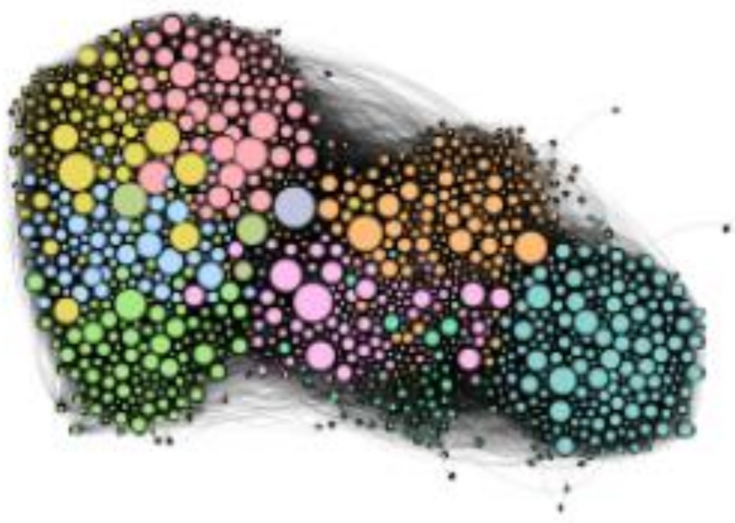
Inputs

- Randomize: On
- Use edge weights: On

Low Resolution (0.5)

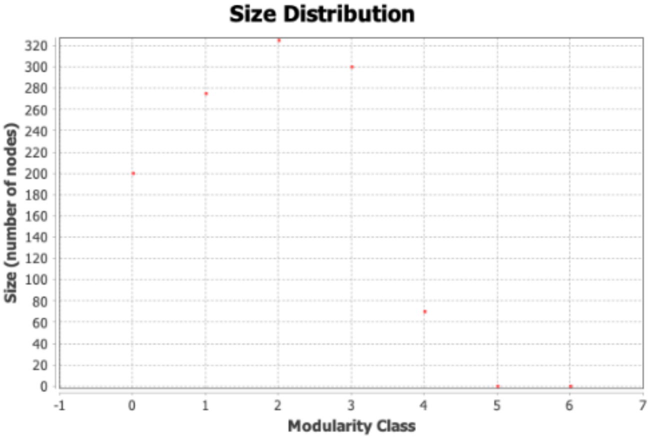
- Modularity: 0.390
- Modularity with resolution: 0.131
- Number of Communities: 13

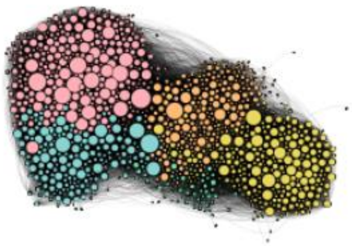




Medium Resolution (1.2)

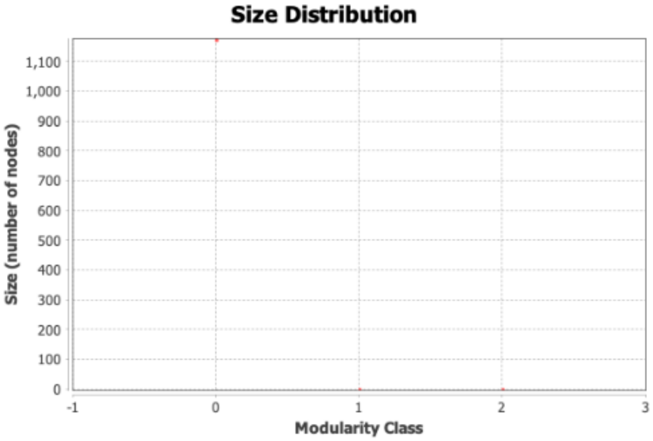
- Modularity: 0.442
- Modularity with resolution: 0.582
- Number of Communities: 7

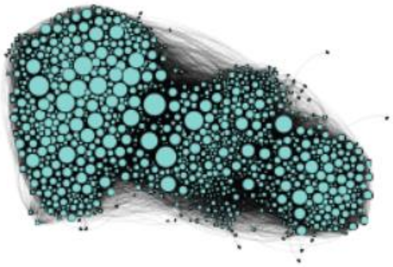




High Resolution (3.9)

- Modularity: 0.000
- Modularity with resolution: 2.900
- Number of Communities: 3





CONCLUSION

In conclusion, the social ego network analysis provides a powerful tool for understanding the structure and dynamics of social networks. By analyzing the properties of the network and the centrality measures of nodes within it, we can gain insights into the roles that individuals play within the network, the patterns of communication and influence, and the overall health and resilience of the network.

Throughout our project, we applied various methods and techniques to analyze a social ego network dataset, including degree distribution, centrality measures, clustering, connectivity, and comparison with other types of networks. Our analysis revealed important insights into the structure of the network, including the presence of power-law degree distribution, high levels of clustering, and a relatively small average shortest path.

Furthermore, we compared our network to other types of networks, including random Erdos-Renyi graphs, Albert-Barabasi graphs, and Watts-Strogatz graphs. This comparison allowed us to understand the unique features and properties of our network, and to identify areas for future research and analysis.

Overall, our project highlights the importance of social ego network analysis in understanding the complex and interconnected nature of social networks. By applying these tools and techniques, we can gain a deeper understanding of the social structures that shape our world and develop more effective strategies for communication, influence, and collaboration within these networks.

REFERENCES

[1] Arnaboldi, V., Conti, M., Passarella, A. and Pezzoni, F., 2012, September. Analysis of ego network structure in online social networks. In 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing (pp. 31-40). IEEE.

[2] Conti, M., Passarella, A. and Pezzoni, F., 2013, April. Ego networks in twitter: an experimental analysis. In 2013 Proceedings IEEE INFOCOM (pp. 3459-3464). IEEE.

[3] M., Passarella, A. and Pezzoni, F., 2016. Ego network structure in online social networks and its impact on information diffusion. Computer Communications, 76, pp.26-41.

[4] Arnaboldi, V., Conti, M., Passarella, A. and Dunbar, R.I., 2017. Online social networks and information diffusion: The role of ego networks. Online Social Networks and Media, 1, pp.44-55.

[5] Gonzalez-Pardo, A., Jung, J.J. and Camacho, D., 2017. ACO-based clustering for Ego Network analysis. Future Generation Computer Systems, 66, pp.160-170.