CS 579: Online Social Network Analysis

Project I - Social Media Data Analysis

Submitted to :

Prof. Kai Shu

Submitted From:

Rahul Kolli (A20519819)

Kruthi Kanukuntla (A20505890)

I. INTRODUCTION

The objective of the report is to crawl the data from one of the social media network and analyse the data. For data crawling and analysis we chose Twitter. Social Network Analysis (SNA) can be utilized to examine the links between Twitter users through their friendship network on the platform. The data about these relationships can be obtained by looking at who follows whom on Twitter. This data can then be used to create a network representation of the connections between users. Further analysis of this network can reveal hidden patterns and structures, such as the existence of communities or identification of influential users.

II. DATA COLLECTION

The first step in conducting Social Network Analysis on a Twitter friendship network is gathering information about the connections between users. We accomplished this through python library, **Tweepy** which provides a simple way to interact with the API and help in fetching the data. To use Tweepy, we authenticated with the API using our credentials(**API Keys**) and then retrieved the friendship data for our user by calling the friends method of the API object. We stored our API Keys in a **configuration file** (config.ini) and used that for authentication.

We focused on collecting the followed list for the user. We first collected followed list data for our user and then collected followed list for fetched followed users.

Code for collecting the data for our user:

```
#Fetching Following list for user
user_list = [me.id]
following = []
following_list = []
try:
    for user in user_list:
        following= api.get_friend_ids(user_id="1620263260552400897")
        #print(following)
except tweepy.errors.TweepyException:
    following=[]

#Limiting fetch for following users to only 10
if(len(following)>0):
    minimum=min(len(following),10)
    for i in range(minimum):
        following_list.append(following[i])

#print(following_list)
#follower_list.append(followers)

#Loading the list data to dataframe
df = pd.DataFrame(columns=['source','target']) #Empty DataFrame
df['target'] = following_list #Set the list of followers as the target column
df['source'] = me.id #Set my user ID as the source

#print(df)
```

- This code gathers the Twitter user IDs that are being followed by our user with user ID "1620263260552400897".
- We **limit** the collection to a maximum of **10** user IDs, to reduce the nodes collection to 500 and adds them to a list,
- Then use the pandas library to create a **DataFrame** with columns labelled as **source** and **target**.
- The **source** column is populated with our **main user ID**, and the **target** column is populated with the **collected Twitter user IDs**.

Code for collecting the data for followed users:

```
user_list = list(df['target']) #Use the list of following we extracted in the code above i.e. my 10 followings
for userID in user_list:
    #print(userID)
    following = []
    following_list = []

    # fetching the user
    user = api.get_user(user_id=userID)
    #print(user.id)
    try:
       following = api.get_friend_ids(user_id=userID)
    except tweepy.errors.TweepyException:
        continue
    #Limiting fetch for following users to only 50
    if(len(following)>0):
        minimum=min(len(following),50)
        for i in range(minimum):
            following_list.append(following[i])

    #Loading the list data to dataframe
    temp = pd.DataFrame(columns=['source', 'target'])
    temp['target'] = following_list
    temp['source'] = userID
    df = df.append(temp)

    #print("updating file")
    df.to_csv("networkOfFollowing.csv")
```

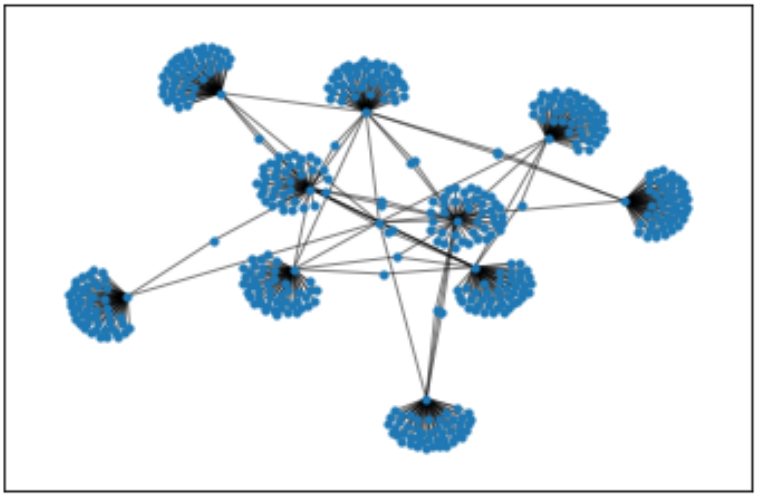
- This code loops through the list of Twitter user IDs previously extracted for our user, it gets the list of user IDs that they are following.
- The collection is **limited** to a maximum of **50 user IDs** for each user to reduce the nodes collection to 500, and these are added to a temporary dataframe.
- The source column of the temporary dataframe is populated with the user ID from the outer loop, and the target column is populated with the Twitter user IDs of those being followed.
- Each temporary dataframe is appended to the main dataframe, creating a graph representation of the following network.
- Finally, the main dataframe is saved as a **CSV** file named **networkOfFollowing.csv**.
- In other words, this code is constructing a network graph of the users that the original user follows, along with the users that each of those users follow, up to a maximum of 50 followed users per user.

III. DATA VISUALIZATION

Once we have collected data from Twitter using the API and stored it in the csv file . We read the csv file and used **NetworkX** to plot the network graph and calculate network measures.

When we executed the code the network generated had **484 nodes** and **509 edges**.

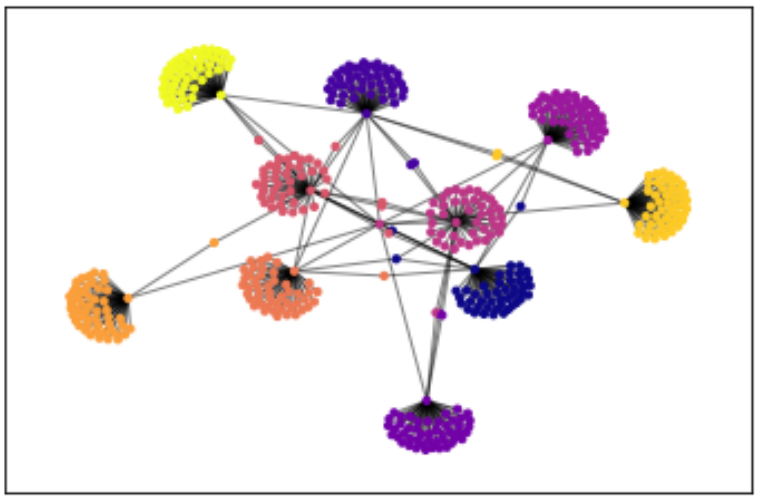
NETWORK GRAPH FOR DATA COLLECTED:



Modularity is a metric used in Social Network Analysis to evaluate the structure and organization of a network graph. It gives a value between 0 and 1 that indicates the degree to which the network can be divided into distinct communities. By analysing the modularity of a network, we can gain insights into the relationships between nodes and the overall structure of the network, such as identifying influential nodes and communities, and understanding the overall structure of the network.

The **modularity** calculated for our data was **0.8312226678143129**, which is closer to 1 and means that the nodes can be divided easily into distinct communities.

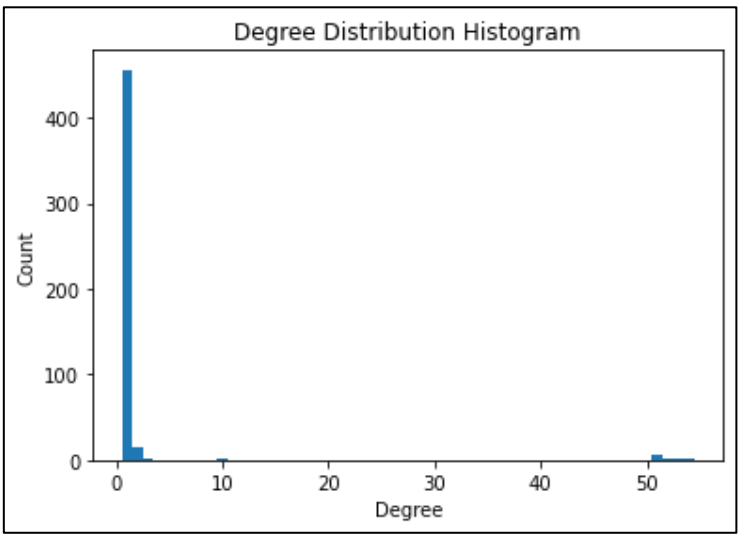
NETWORK GRAPH FOR DATA COLLECTED WITH PARTITION OF COMMUNITIES:



IV. NETWORK MEASURES CALCULATION

The degree distribution of a network graph provides an understanding of the network structure by giving information on the frequency of connections each node has. By analysing the degree distribution, we can gain valuable insight into the organization and arrangement of the network as a whole.

DEGREE DISTRIBUTION FOR OUR DATA:

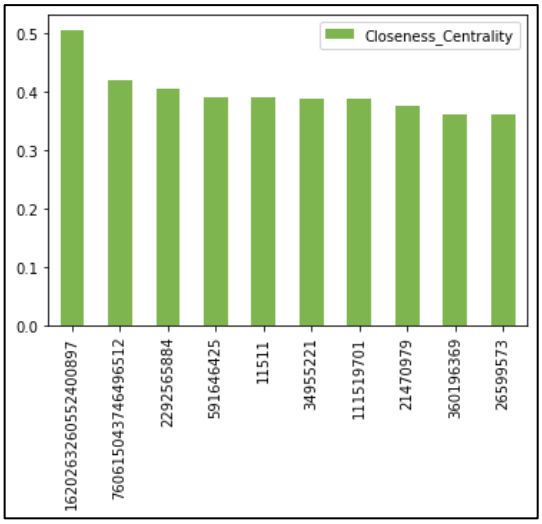


We can observe from the histogram that majority of the nodes in our data have degree 1.

A. BETWEENNESS CENTRALITY

The importance of a node in a network can be measured using betweenness centrality, which is based on the node's ability to control or influence information flow between other nodes. Nodes with high betweenness centrality serve as bridges between different parts of the network and are crucial for maintaining overall connectivity. By analysing betweenness centrality, we can identify key nodes that are essential for communication, transportation, or other networked systems, and also detect potential vulnerabilities that need to be addressed to improve network efficiency, resilience, or security. Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes.

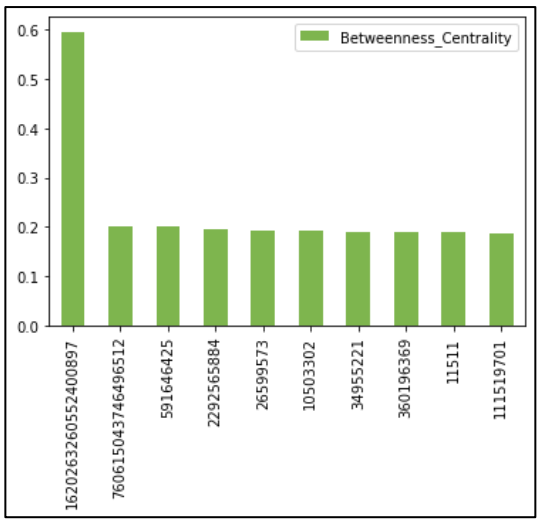
We took top 10 nodes and plot them as a bar graph which helps in understanding most influential node.



B. CLOSENESS CENTRALITY

Closeness centrality is a network metric that measures the average distance between a node and all other nodes in the network, identifying nodes that are located close to other nodes and thus important for overall connectivity. Nodes with high closeness centrality are critical for efficient information, resource, and people spread, and detecting them can help identify network vulnerabilities. Closeness centrality is helpful in understanding network efficiency, resilience, and critical node identification.

We took top 10 nodes and plot them as a bar graph which helps in understanding most influential node.



V. CONCLUSION

Nodes with high betweenness centrality or closeness centrality are critical for the overall connectivity and efficiency of the network, and identifying them can help inform strategies for improving network performance, resilience, and security. In our case from our network measure calculations and network graph we can conclude that most influential and crucial node is **"1620263260552400897".**

VI. REFERENCES:

[1] https://www.geeksforgeeks.org/network-centrality-measures-in-a-graph-using-networkx-python/

[2] https://www.kaggle.com/code/rahulgoel1106/network-centrality-using-networkx

[3] https://uwaterloo.ca/networks-lab/blog/post/network-analysis-metaknowledge

[4] https://towardsdatascience.com/how-to-download-and-visualize-your-twitter-network-f009dbbf107b