CS 579: Online Social Network Analysis
Project I – Instagram Social Media Data Analysis
Prof. Kai Shu

YASH PATEL(A20451170) **NIRAV SONI(A20435538)**

Objective:

In the project we will show how to web crawl Instagram users' following data using selenium, web driver Manager and python. We are using graph algorithms to extract data from Instagram. We used a Gephi tool for visualizing the data as histogram plot and measuring different kinds of Centrality like Degree Distribution, Eigenvector, Closeness and Betweenness. Also, we calculated Diameter, Average Path Cost of the graph.

Outline:

1. **Data Collection:**

   From June 2020, Facebook changed all Instagram APIs and we do not get a list of following users. And due to privacy policies, we switch to web crawling to access the data of Instagram accounts.

   We are using python as a programming language and Selenium, web driver manager and json as python libraries. Json is used to write an adjacency list and list of all users into a json file. Selenium is used to automate crawling data including access to the HTML tags and provide form filling and mouse clicking. Web driver manager is used to access the web browser as a helper library of selenium.

   We made a list of 300 instagram users in all_insta_accounts.json file and an adjacency list as a HashMap to store the following users in graph_edges.json file. In this HashMap user is a key and value are a list of following users.
   We implemented the BFS algorithm with little change to get level wise data. Here, change is to create a graph instead of a tree using the root user.

   Pseudo code for this script is as below.

   check all_insta_accounts.json and graph_edges.json data files are present or not in the project directory :

   If not present then take instagram's official account in all_insta_accounts list and empty adjacency list.

If present then initialize a list named all_insta_accounts with an all_insta_accounts.json file and an adjacency list named graph_edges with a graph_edges.json.
(the purpose is to get the following list of a single user, It takes around 5 minutes and if the script is run for a very long time Instagram stops responding and also for private profiles. So, use try catch block everywhere needed and when we get an error from Instagram, the process continues crawling data for other users in a list. Also, when we run the script again, we do not need to start with the root user again and we can skip those users whose following users' list is found.)

Get an access of web browser using web driver manager
Selenium opens Instagram webpage in a browser then login into instagram account using provided credentials.

initialize current_level with all_insta_accounts.

Use threshold of 300 users in all_insta_accounts list and repeat until reach the threshold and after reaching this threshold expansion of graph will be stopped:

Initialize an empty list named next_level.
Repeat to explore every single user in the current_level :
if the following users' list of this current_user is not present in an adjacency list:

- Selenium opens the user's profile in the browser then clicks on the following button and returns a list of 20 following users.
- Get a list of 20 following users of a current_user and a list of 20 following users of every following user.
- Rank these following users on basis of how many following users of every following user are present in all_Insta_accounts list by this time. (The purpose behind this is to get more follow back data which makes the graph denser.)
- Generate a random number (RN) between 1 to 10 to get a random degree of a current_user which gives a list of top RN following users of a current_user.
- Add these following users of a current_user in next_level list and put key-value pairs (current_user, list of following users) in the adjacency list.

Replace current_level with next_level.
Take the union of all_Insta_accounts list with current_level list.

Repeat to explore every single user in the lowest level of the bfs tree which is stored in current_level:

Get a list of following users of a current_user and intersect with all_insta_accounts to get following users which are only present in all_insta_accounts.

(Here, the purpose is to get following users of the lowest level's users of BFS tree which are present in higher levels of BFS tree which make our graph denser.)

Store all_insta_accounts list in all_insta_accounts.json file and adjacency list in graph_edges.json file.

For the Gephi tool we used edge list in csv file format.
Pseudo code to get an edge list is as below.

Initialize an empty edge_list.

Iterate all user accounts(key) in the adjacency list graph_edges:

Get the following users' list using a key from graph_edges.

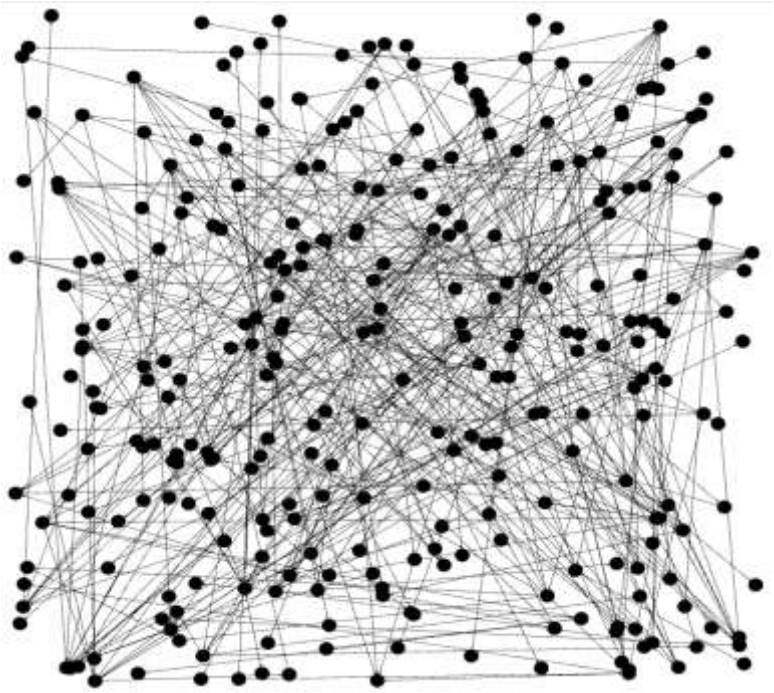
Iterate all following users in the following users' list:

append a directed edge (Source, Target) = (user, following user) in edge_list.

Write edge_list in edge_list.csv file.

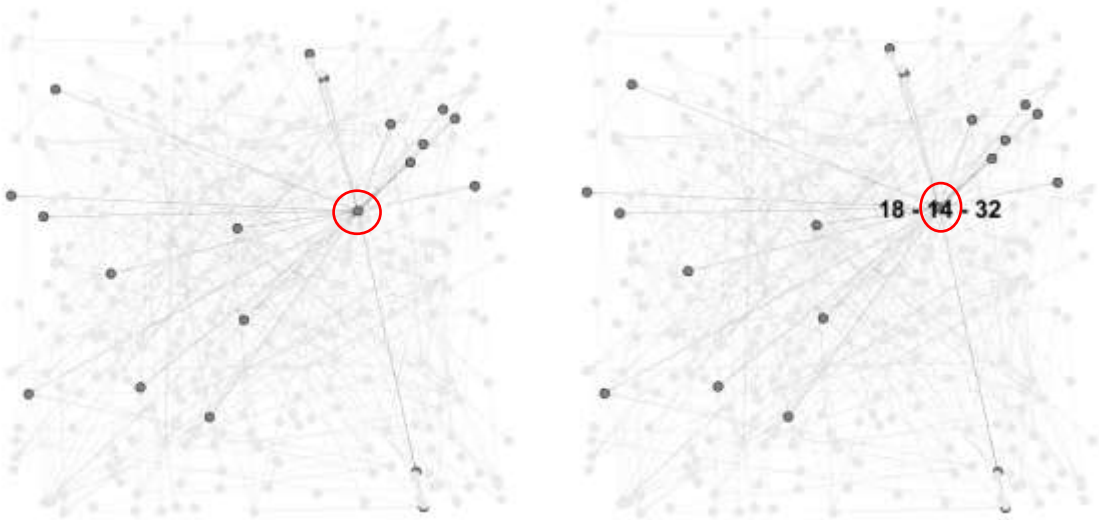
2. Data Visualization:

We visualized Instagram users' graph with the following relationship using the Gephi tool. The file we used is edge_list.csv which was generated using a python script as mentioned before.
This file is having 300 Instagram users and 529 following relationships(edges).



2(a). Instagram Users' Graph Representation

Above graph 2(a), black nodes represent Instagram user and edges between them represent following relationship.



2(b), (c) follower following graphs with in-degree, out-degree, and total degree

Here in above two graphs 2(b)(c) node without red circle are the followers and following users of a user with red circle. In right graph we can see in degree, out-degree, and total degree of user with red circle.

2(d). Bi-directional Connection

Here the graph 2(d) shows bi-directional connection between two users and the relationship both are following each other.

2(e). Directional Connection

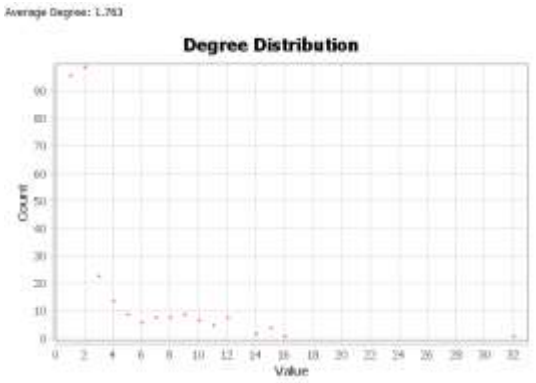
Here the graph 2(e) shows directional relationship between two users and in this relationship only left user following right one.

3. Network Measures Calculation:

In network measure calculation we are calculation centrality to know that how much influential or important and individual node is. In centrality calculation we considering neighbor nodes.
Here we calculated several centrality measures. Which are mention below:

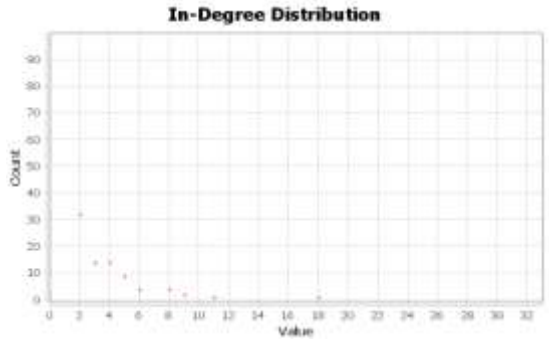
3.1 Degree Centrality:

It ranks the node based on connection (edges). More connect edges show more degree which means higher in rank. Here we have directed graph for Instagram following users, so higher the indegree shows more influential and higher the out-degree shows less influential and vice versa.

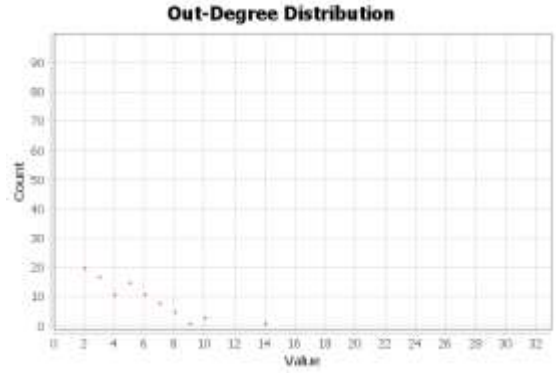


Here the left histogram represents degree distribution of Instagram network. Here avg degree for Instagram network is 1.763.

Here we can see that highest degree is 32 with count 1 and lowest degree is 1 with count more than 90. But we have directed graph, so this degree distribution does not give us precise centrality.

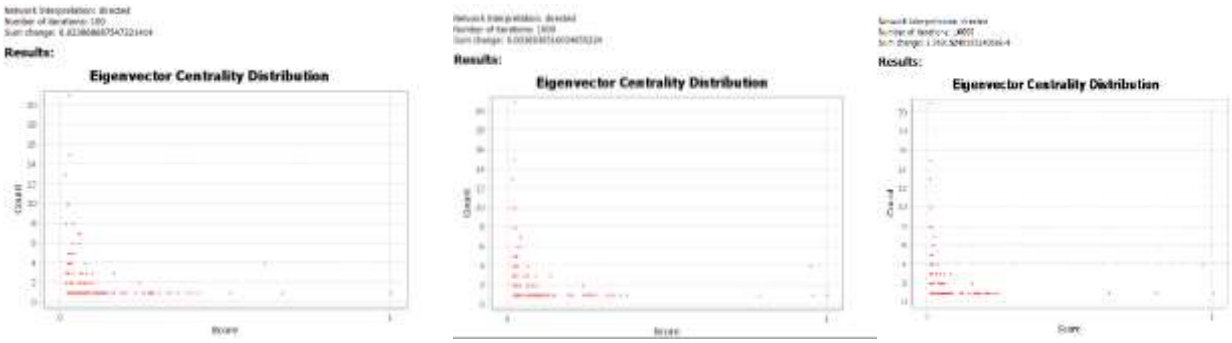


More In-degree means more important the node is and vice versa. Here, we can see that highest in degree is 18 with count less than 5 and lowest indegree is 2 with counts above 30. So, we can say that very less influential nodes in this graph.



More out-degree means more friendly with others and less important node. Here in this graph highest out-degree is 14 with count less than 5 and lowest is 2 with counts around 20.

3.2 Eigenvector Centrality:



3.2 (a), (b), (c) Eigenvector Centrality using 100,1000,10000 parameters

In degree centrality we calculate importance of a node based on their connected edges, but importance is not depending on that it likewise relies upon neighbor nods significance. Eigenvector centrality gives importance of node based on their impotent friends.

Here we took 3 distinct estimations of parameters 100,1000 and 10000 3.2(a), (b), (c) respectively. In every iteration we got a blunder 0.023, 0.00369 and 1.96e-4. As we increment no of iteration, mistake is diminishing which offers more exact answer. In degree distribution we considered that to be degree goes up tallies go down. Be that as it may, here eigenvalue centrality circulation changes more than degree dispersion.

In above distribution 3.2, degree in range of 0 to 32 and counts were in range of 0 to 30 with 9 data points. In eigen value distribution score in range of 0 to 1 and counts were in range of 0 to 30 with much more data points than degree distribution.

In nutshell, in degree distribution users are gathered at one data point but in eigenvector distribution these users are distributed in different scored data points which shows more precision in eigenvalue centrality.

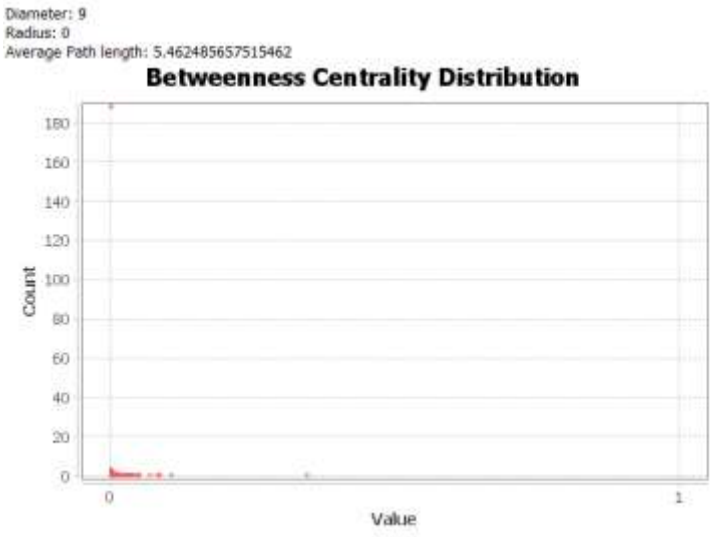
3.3 Diameter:

The diameter of a diagram is the greatest eccentricity of any vertex in the graph. That is, it is the best separation between any pair of vertices. To discover the distance across of a graph, first locate the most limited way between each pair of vertices. The greatest length of any of these paths is the diameter of the graph.
Here the diameter is 9 which means max length of shortest path between any pair of users is 9. So, any user can cover every other user with a shortest path with max length of 9.

3.4 Average Path Cost:

Average path length is defined as the average number of steps along the shortest paths for all possible pairs of nodes. Here the average path cost is 5.46.

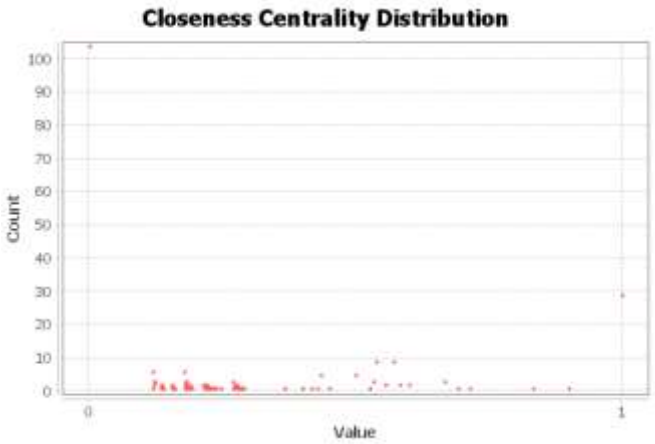
3.5 Betweenness Centrality:



3.5(a)

Betweenness centrality measures the extent to which a vertex lies on paths between other vertices. Vertices with high betweenness may have considerable influence within a network by virtue of their control over information passing between others.
As the value of betweenness centrality goes higher it means more important and vice-versa. Here 3.5(a) for all the users, centrality lies between 0 to 0.5, more than 180 users have lower centrality near to value 0 which means they are not much important in the network. And the only one user with highest centrality which is most important user.

3.6 Closeness Centrality:



3.6(a)

Closeness centrality is a method of identifying node that can spread data productively through a graph. The closeness centrality of a node quantifies its normal farness (backwards separation) to every single other node.
Users with a high closeness score have the briefest separations to every single other user. Here more than 100 users with value tends 0 which means they spread very less data to others and some of these are very far from these users (length of shortest paths are higher). It implies they are less significant.
There are around 30 users with value tends to 1 which means they spread more data to others and most of the users are near to these users (length of shortest paths is less). It implies they are significant.

4. References:

[1] https://github.com/aarnhub/instagram-scraper-python

[2] U. Brandes, S. P. Borgatti, and L. C. Freeman, “Maintaining the duality of closeness and betweenness centrality,” *Soc. Networks*, vol. 44, pp. 153–159, Jan. 2016, doi: 10.1016/j.socnet.2015.08.003

[3] “Closeness Centrality - an overview | ScienceDirect Topics.” [Online]. Available: https://www.sciencedirect.com/topics/computer-science/closeness-centrality. [Accessed: 28-Sep-2020].

[4] “Social network analysis: Centrality measures.” [Online]. Available: https://cambridge-intelligence.com/keylines-faqs-social-network-analysis/. [Accessed: 28-Sep-2020].