Twitch Friendship Network

by Grace Arnold (A20415197) and Gladys Toledo-Rodriguez (A20419684)

Introduction

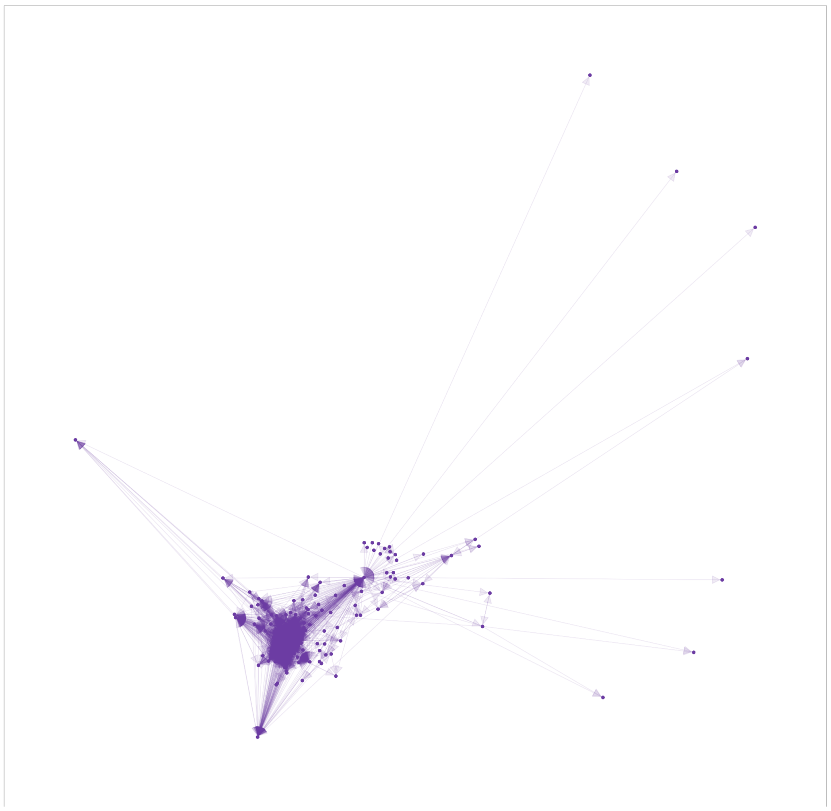
For this project, we decided to graph a friendship network for Twitch users. The nodes would include those following and followed by Twitch user SereneGrachay, who gave permission to use the account id.

Web Scraping

After logging into the Twitch developer portal and creating an application, we retrieved our API keys and authentication tokens. With these and the TwitchAPI Python library, we were able to create the lists of nodes we would be graphing. Because the Twitch API uses pagination when returning follower/following lists, we had to write an algorithm to aggregate them. Once we had these lists of nodes, we could loop through them and call the Twitch API to determine the relationship between two nodes, whether it be follower, followed, or both.

Graph Generation

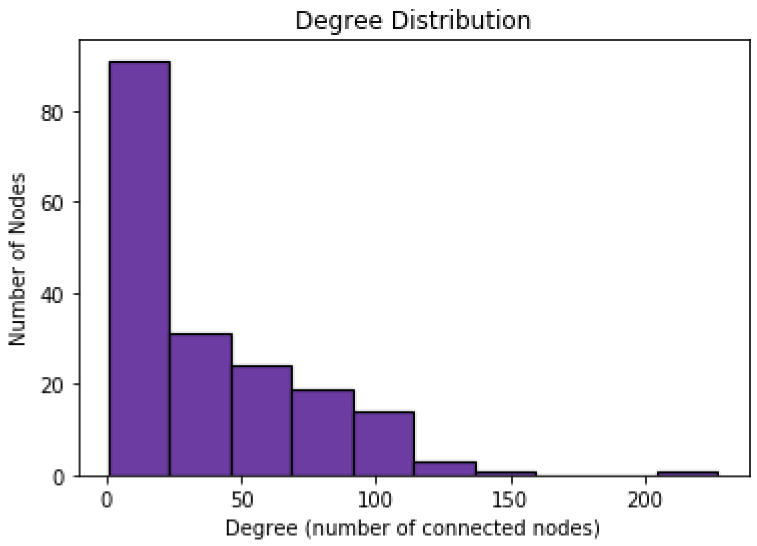
While looping through the nodes, we added the edges to a directed graph using the NetworkX Python library. We also added pairs of nodes, one following the other, to a csv file so that we didn't have to call the API again when we wanted to create the graph. After filling out the graph with all nodes and edges, we plotted it:



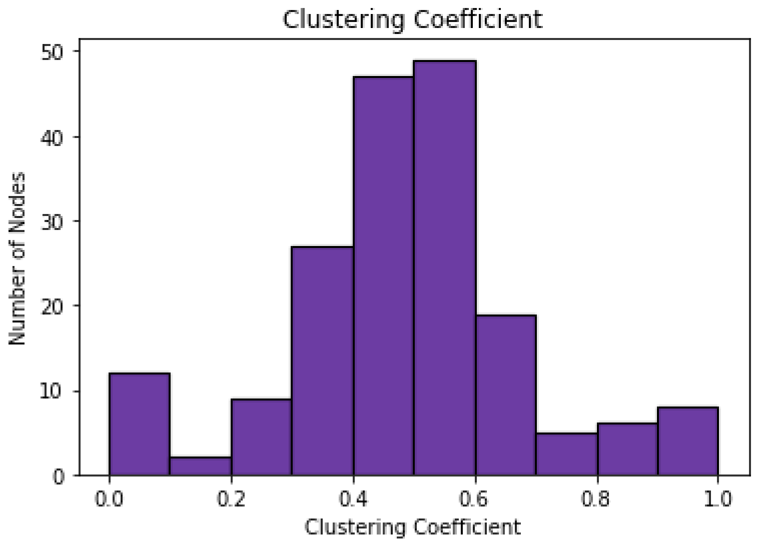
The user that we based this graph around can clearly be seen in the middle of the graph. However, we can see a very tight cluster to the bottom left as well. There are a few outliers that are not a part of this cluster on the outskirts of the graph.

Network Measures

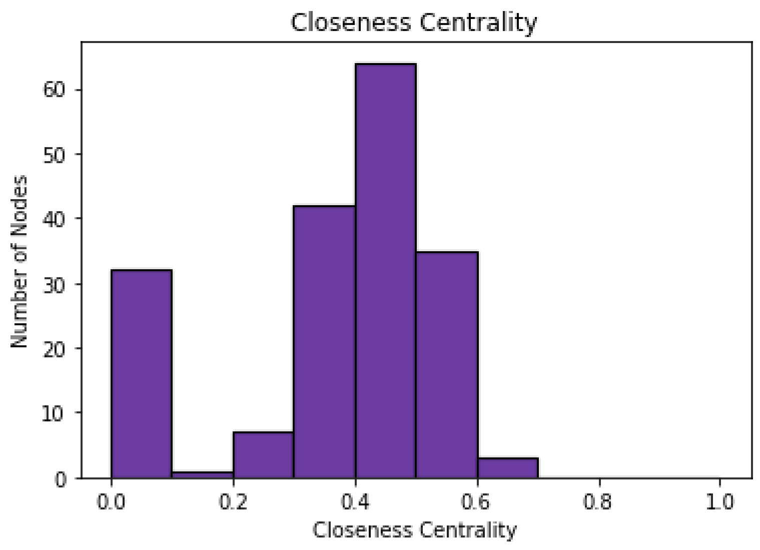
NetworkX has built in functions for determining the degree distribution, clustering coefficient, and closeness centrality, so we used those to determine the values for each node. We then plotted the histograms for the whole graph:



As we can see with this histogram, a large number of nodes have low degree distribution. This means that they are not connected to many other nodes. Only a small number of nodes are highly connected. We can assume that those with high degree distribution are the streamers, while those with low degree distribution are the viewers.



Many nodes have a high clustering coefficient, which means that there are many triangles or clusters in this friendship network. This makes sense because there are close networks of people that follow each other on Twitch.



The closeness centrality is in the middle for many nodes, and there are only some nodes that have a longer average path to the other nodes. This once again makes sense as close networks of people who follow each other can develop on Twitch.

Conclusion

In conclusion, our graph and network measures made sense for the nodes that we decided to use. It was interesting to see how the data clustered, and to analyze the network measures afterwards as well. It may be worthwhile in the future to create friendship networks for streaming “teams” (groups of streamers who affiliate with each other and stream similar content, often sharing most of their followers) to see what those graphs would look like.

References

- Twitch API Documentation: https://dev.twitch.tv/docs/api/
- TwitchAPI (Python library): https://pytwitchapi.readthedocs.io/en/latest/modules/twitchAPI.twitch.html
- NetworkX: https://networkx.org/documentation/stable/tutorial.html
- Matplotlib histogram documentation: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html
- Class notes about Network Measures