Project 1
Social Media Data Analysis

Document submitted towards the completion of Project 1 for the course:

CSE 472: Social Media Mining

to

Prof. Huan Liu

Submitted by:

Tarun Lolla – 1216095647

Anoop Reddy Vutukuri – 1217182122

Contents

IMPLEMENTATION

Setting up a Twitter Developer Account

The social network chosen for the purpose of this project is “Twitter”. Twitter is a micro-blogging website in which a user posts and interacts with his follower via messages termed as “tweets”. Each tweet can consist a maximum of 280 characters in all languages except Chinese, Korean and Japanese.

Twitter provides a user with a developer account for the purpose of using its API. A developer account is granted to a user on need basis after scrutiny on a case-by-case basis. The developer account can be created at http://developer.twitter.com.

Once a developer account is created, in order to establish communication via the twitter API, a user should create a basic application (widely known as the OAuth app) in order to get the following keys:

- Consumer Key
- Consumer Secret Key
- Access token
- Access Secret Token

To perform a few basic tasks with twitter API, the well-organized Twitter API documentation is a good place to start.

Environment Setup

This project is implemented in Python 3. A user is expected to install Python3 along with the libraries Tweepy, NetworkX, Matplotlib and Pandas for executing the scripts in this project. The scripts are tested on both Windows and Linux Environments.

Using the Twitter API on Python – Tweepy

The twitter API can be conveniently accessed from a python script using **"Tweepy”**. Tweepy is a library which is a wrapper around the the twitter API. Tweepy can be installed using the command:

```
pip install tweepy
```

NetworkX and Matplotlib for Graph Visualizations

For the purpose of Graph Visualizations and Histograms for Network measures, the libraries NetworkX and Matplotlib have been used.

Networkx can be installed using:

```
pip install network
```

Matplotlib can be installed using:

```
pip install matplotlib
```

Pandas for data storage

To store the computed Network Measures, data has been stored in Pandas Dataframes.

Pandas can be installed using:

```
pip install pandas
```

Files in the package

The **Code** directory in the submitted zip file consists of all the code and necessary input and sample output files for this project. A brief description of the files in this directory is as below:

- crawlData.py : This file has the code to crawl the users' data. The file takes as input a user's API keys and outputs a file 'data.csv' which has the data obtained from crawling.
- buildGraph.py: On executing this file, the user will be able to visualize data obtained from crawlData.py as a graph and also has the network measures shown as histograms.
- test_connection.py: This file can be used by the user to validate his credentials. On successful validation, the file shows the screen name of the user's twitter account.
- conn_details.txt: This file is used to give input to the script crawlData.py and test_connection.py.
- data.csv: Contains the data obtained from Crawling the user data from crawlData.py.
- network_measure.csv: Contains the network measures for each user.

The **Output** directory in the zip file consists of images which are obtained by running the scripts submitted.

The **Report** directory consists of this file.

API Keys – Input to script

To execute the script crawlData.py, the user has to provide his/her twitter API keys. This can be done in two ways.

1. Provide the keys in the file conn_details.txt
2. Enter the keys when prompted for.

The user can either provide the keys in the file conn_details.txt which will be used by the scripts crawlData.py and test_connection.py. If needed, the user can leave the file blank and enter the keys when prompted by the script.

Sample input in the conn_details.txt file is as follows:

```
consumer_key=abcde

consumer_secret=a1b2c3d4e5

access_token=12345

access_token_secret=1a2b3c4d5e
```

Note: Do not enclose your keys in Single/Double (‘ ’ or “ ”) quotes when giving as an input to the file or when prompted.

Even if the user doesn’t test his/her API keys, crawlData.py does the test to validate the keys before it proceeds to crawl the data.

CRAWLING THE DATA

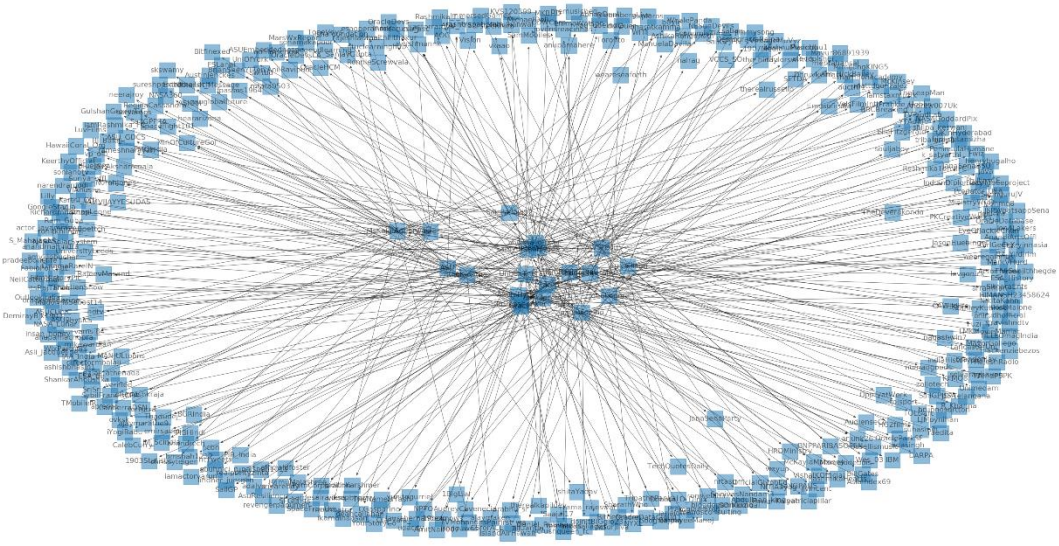
In this document, by friends we mean all the users which the user follows. For example, if A follows B, C, D and E, then we call B, C, D, E friends of A.

Screen name of a user is the name which appears on the user's twitter handle.

Data crawling is achieved by extracting the friends of the user whose API credentials are given and also the friends of his/her friends. The script starts with the API user and for each of his/her friend extracts his/her friends. This is accomplished by the method tweepy.friends() in the library. The method outputs a maximum of 20 users per call and we call it for each of the 20 friends of the API user which gives us a maximum of 400 nodes (in the event of no common friends), which is inline with our project specifications. The data thus obtained is stored in a csv file : 'data.csv' which is saved in the Code directory of the submitted zip file. The file has 2 columns of which the first column is the screen name of the user and the second column is the screen name of the user's friend.

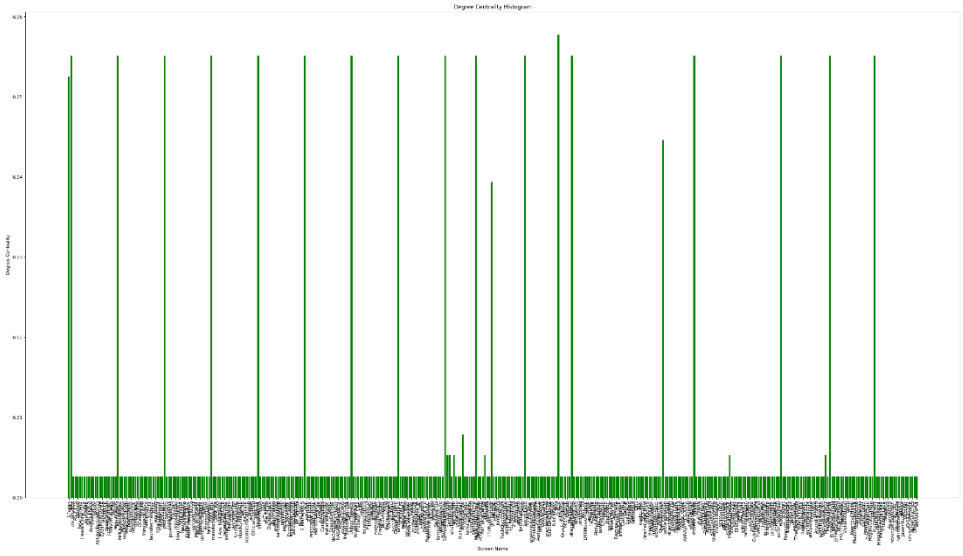
DATA VISUALIZATION

The graph data thus obtained by crawling is read from the csv file and fed to a NetworkX graph. This graph is plotted using the library matplotlib. The graph data appears as below:

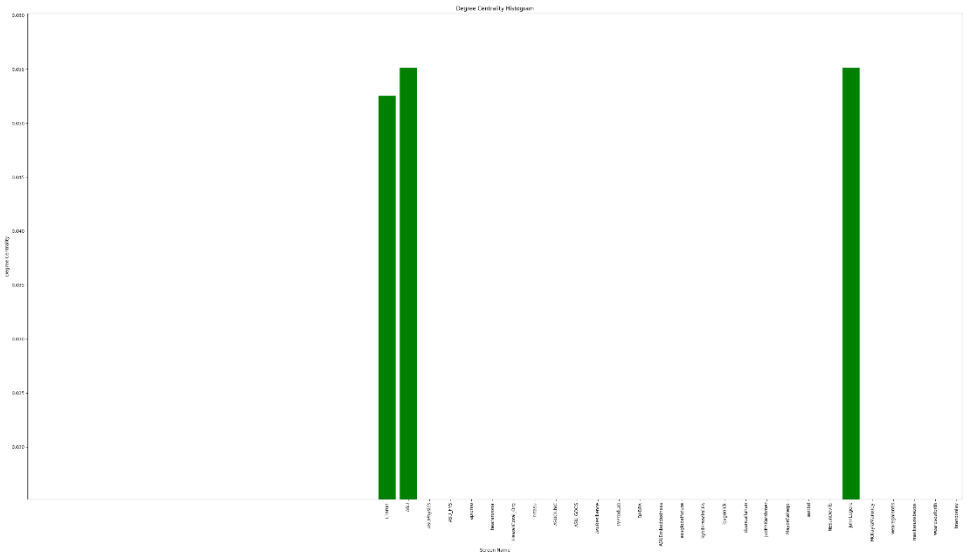


This image is shown to the user when the script buildGraph.py is executed. It is also available in the directory Output in the submission title 'Graph.png'

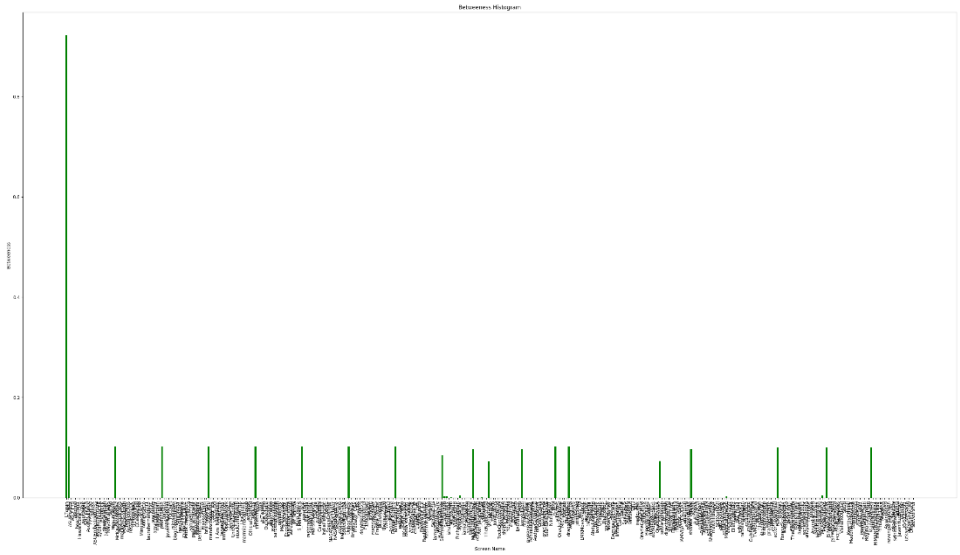
Once the window showing the graph is closed, another window showing the histogram for **Degree Centrality** pops up. The image appears as below:



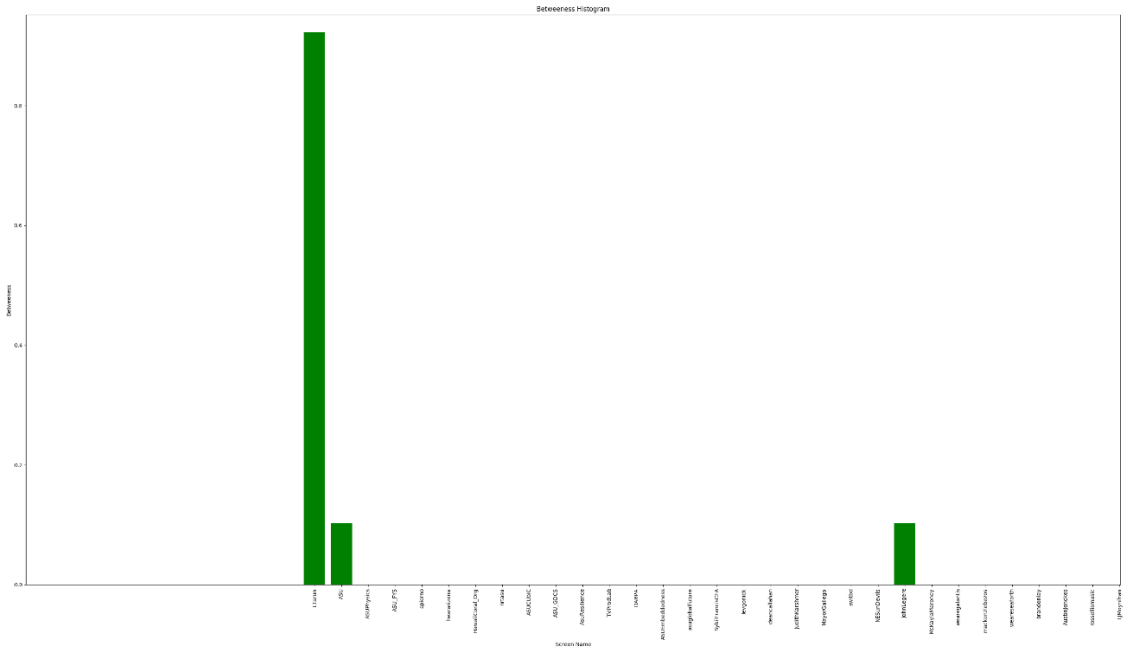
The image can be zoomed in to observe account-wise degree centrality with the zoom option available in the window as below:



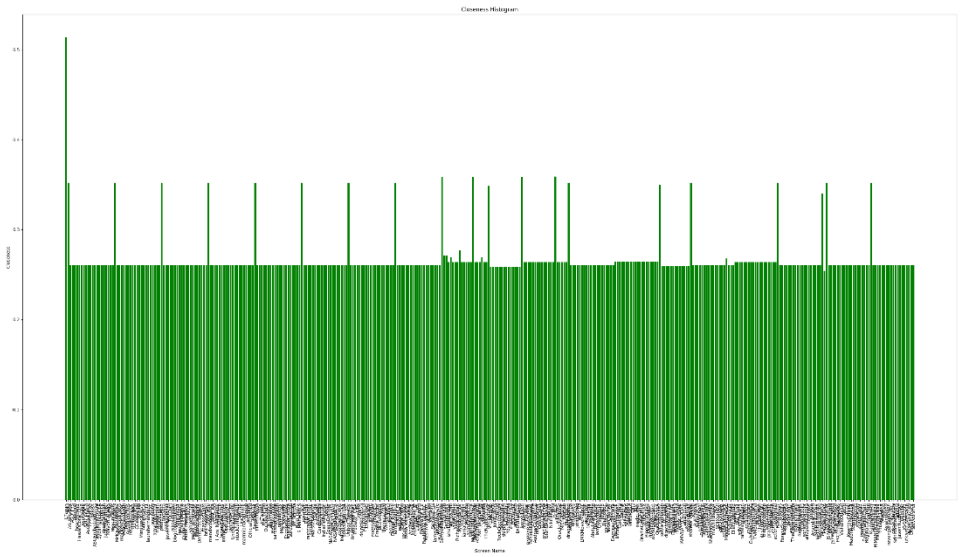
After closing the above window, the histogram showing **Betweenness** appears as below:



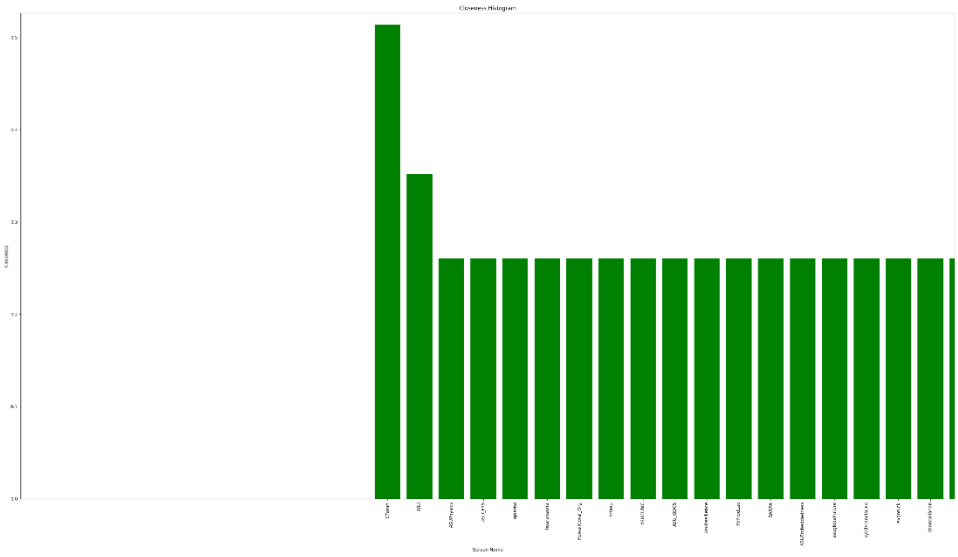
Which can be again zoomed in to observe individual entries:



After closing the above window, the histogram plot for **Closeness** as below:



Which can be again zoomed in as below:



All the above images are available in the directory “Output” in the submitted zip file with appropriate titles.

The numeric values of network measures for all the users are also printed in the console as below:

	Degree Centrality	Betweenness	Closeness
LTarun	0.052493	0.922556	0.513477
ASU	0.055118	0.102362	0.352126
ASUPhysics	0.002625	0.000000	0.260602
…	…	…	…

These network measures are also exported to a csv file ‘network_measure.csv’ to the end of the execution which is saved in the Code directory of the submitted zip file.

References

The following references have helped us achieve the set goals for this project:

- Developer.twitter.com. (2019). Docs. [online] Available at: https://developer.twitter.com/en/docs.
- Tweepy.readthedocs.io. (2019). Tweepy Documentation — tweepy 3.8.0 documentation. [online] Available at: https://tweepy.readthedocs.io/en/latest/index.html.
- Georgiev, P. (2014). NetworkX: Network Analysis with Python, Computer Laboratory, University of Cambridge. Available at: https://www.cl.cam.ac.uk/teaching/1314/L109/tutorial.pdf.
- Networkx.github.io. (2019). Tutorial — NetworkX 2.3 documentation. [online] Available at: https://networkx.github.io/documentation/stable/tutorial.html.
- Matplotlib.org. (2019). Usage Guide — Matplotlib 3.1.1 documentation. [online] Available at: https://matplotlib.org/3.1.1/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py.