Name: Tanushree Chakravorty

ASU ID: 1207664935

CSE 598: Social Media Mining: Project 1 Phase 1

P1:

The social network that I had to crawl was Yelp.com. The graph of the social network is an undirected graph.

I wrote a script to crawl Yelp from scratch and it is attached in Folder P1 titled: 'Scrape_new.py' and instructions to run this code are placed in a readme text file titled: 'Readme_ScrapeCode.txt'

The raw data, that is the unique user IDs given at the end of each URL is crawled and saved in the 'Output.txt' file. This data is non-anonymized as it can be used to directly visit/access the user's yelp homepage.

The file that maps usernames to numbers is in 'Mapper.txt'

The file with the anonymized dataset is in 'AnOutput.txt'

Sampling of the nodes is done at crawling stage itself. Therefore, total number of connected nodes to a friend is the min(actual connections,1000).

Therefore the sampled files anonymized and non anonymized is the same as Output.txt and AnOutput.txt

Instructions to run the program are in the Readme_ScrapeCode.txt file in the P1 folder.

Find the entire data dumps for all the tables and calculations in the respective folders P1, P2, P3, P4

All graphs have been generated in Matlab:

P2: Graph Essentials:

For P2 – P4, I used the following important libraries:

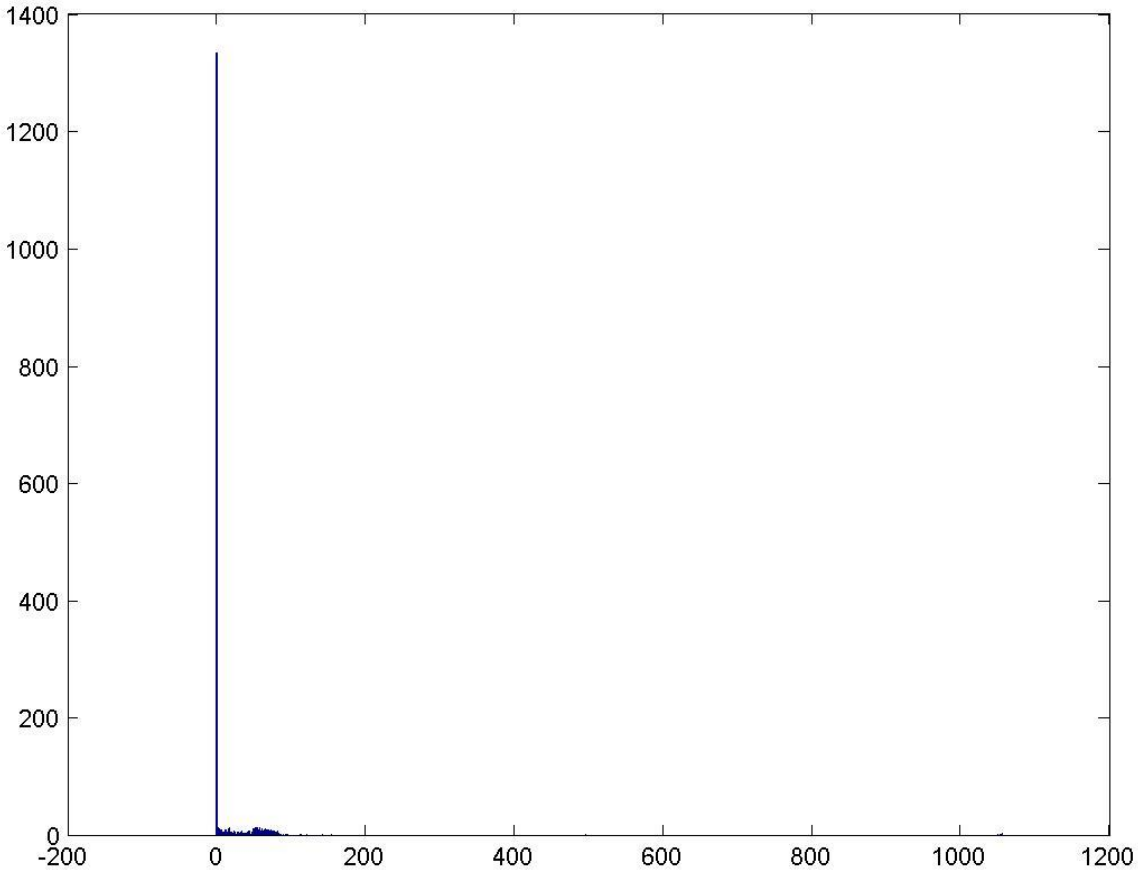
- Snap for Python (http://snap.stanford.edu/snappy/)
- Networkx (https://networkx.github.io/)

This network follows the power law distribution.

Degree distribution for outdegree v/s frequency of nodes:

The total number of nodes n = 39892

The total number of edges m = 70942



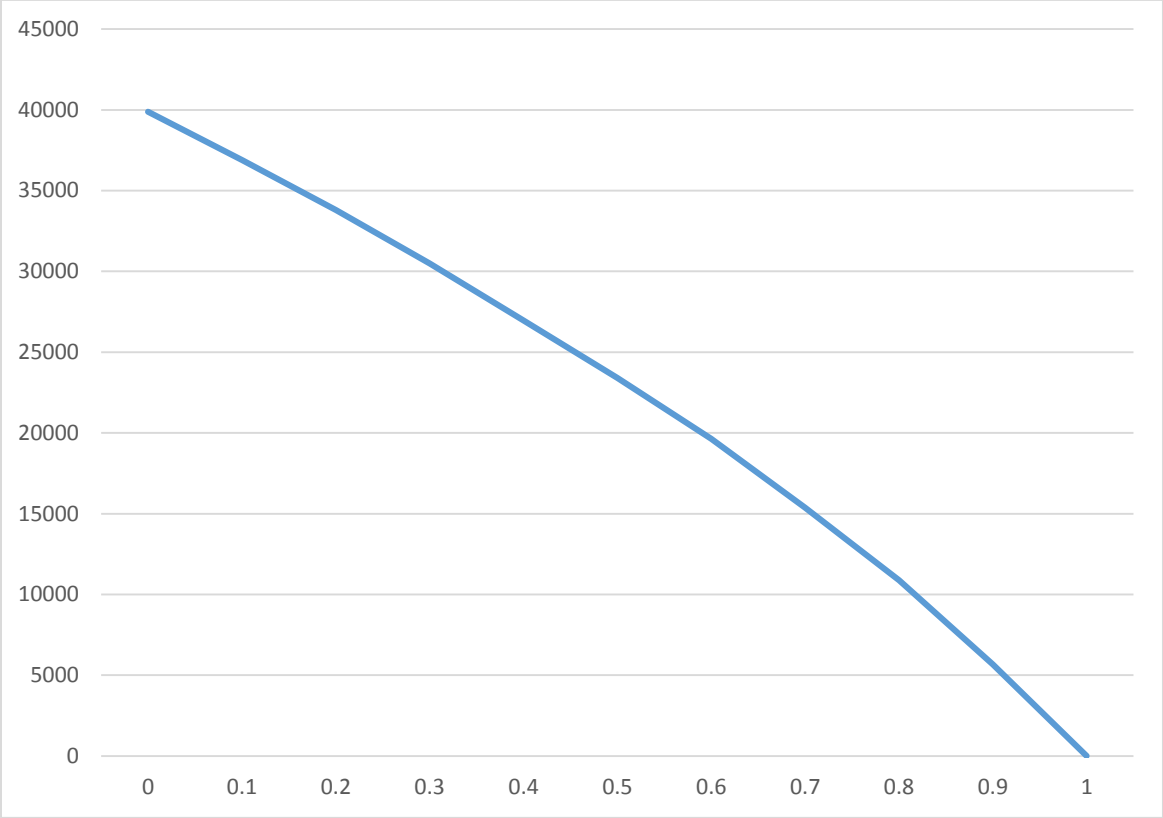
Y: Frequency of nodes; X: Degree distribution

The number of bridges is: 47295

The number of 3-cycles are: 35293

The graph diameters are: 6

The following are the plots at different values of x [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]



X% v/s largest component

0	39892
0.1	36897
0.2	33797
0.3	30485
0.4	26968
0.5	23411
0.6	19637
0.7	15375
0.8	10877
0.9	5660
1	0

----- The data dump this function gave is placed in the P3 folder

P3: Network measures:

Global clustering coefficient = 0.0650025871704

Local clustering coefficient = 0.0735437482521

- Page Rank top 10 nodes:

Node #	Page Rank value	Rank
1951	0.000169008546681268	1
638	5.593909871679229e-05	2
648	5.5042847638319546e-05	3
711	5.499035516799918e-05	4
956	5.443288586920563e-05	5
1451	5.326252996800492e-05	6
8	4.611239842370499e-05	7
9281	4.600025352722133e-05	8
9372	4.600025352722133e-05	9
9632	4.600025352722133e-05	10

- Degree Centrality:

Node #	Degree Centrality value	Rank
1951	0.004462159384322278	1
956	0.001679576846907824	2
11288	0.00152916698002055 6	3
11134	0.0013035521796896544	4
11119	0.0012784838685417764	5
13194	0.0012283472462460204	6
1451	0.0011782106239502646	7
11356	0.0011531423128023867	8
10094	0.0011280740016545085	9
12785	0.0011030056905066306	10

- Eigen value centrality:

Node #	Rank
1951	1
956	2
11288	3
11134	4
11119	5
13194	6
1451	7
11356	8
10094	9
12785	10

- Jaccard similarity:

The two most similar nodes using Jaccard similarity is and Jaccard similarity value is 1.
This is because node 20051 has only one friend, which is node 718, and node 718 has only one friend, which is also node 12027. So Jaccard similarity of node 20051 and 12027 is 1, which is the largest.
There may exist other nodes in the graph which also have the Jaccard similarity = 1

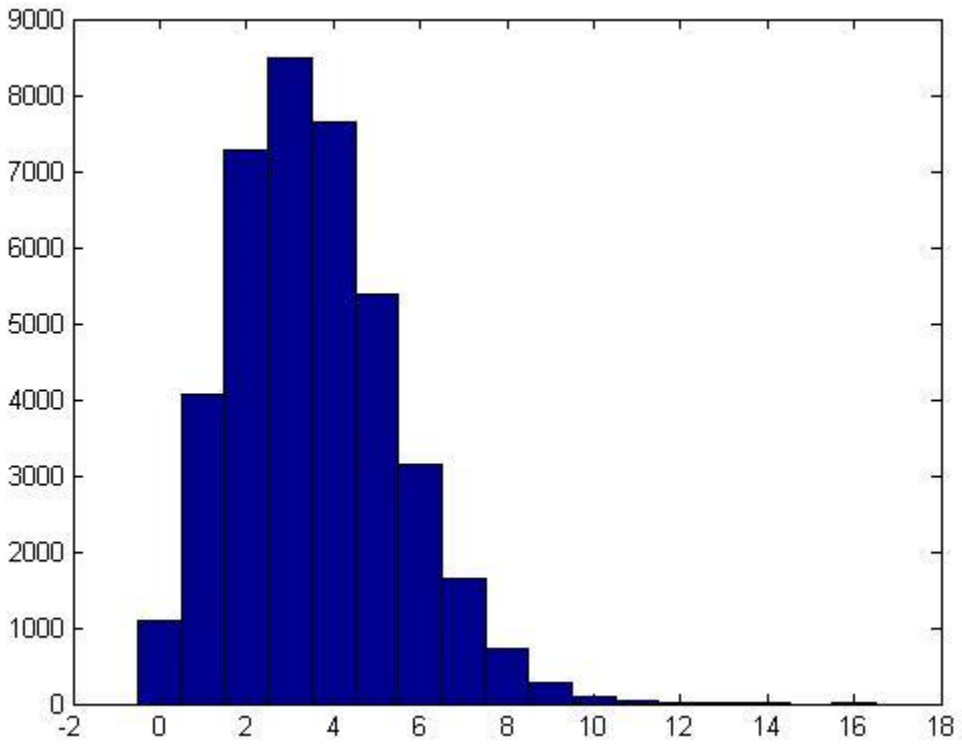
P4: Network models:

1) Random graph:

The degree distribution is shown below. It is an undirected graph.

- This graph was created by using the NetworkX library for Python using the gnm_random_graph method. Networkx (https://networkx.github.io/). It does not mimic the power law distribution.

Average path length = 8.0026136174131
Global clustering coefficient = 6.811403229302
Local clustering coefficient = 3.7182637720102



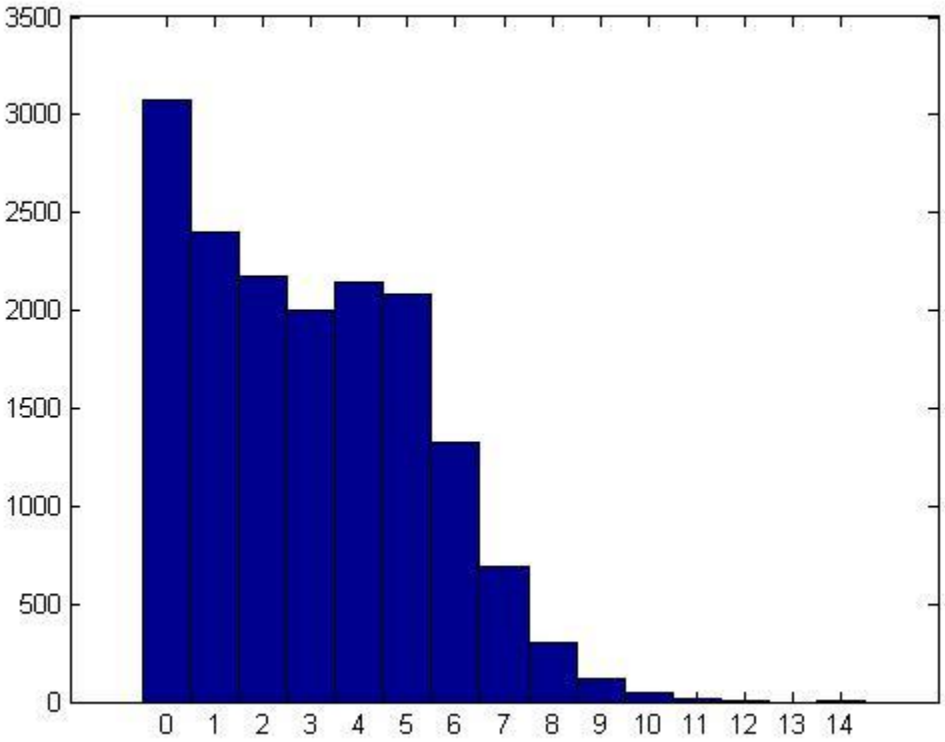
Y: Frequency of nodes
X: Degree distribution

2) Small world model:

- This graph was created by using the NetworkX library for Python Networkx (https://networkx.github.io/). It does not mimic the power law distribution.
- The fast_gnp_random_graph(*n*, *p*, *seed=None*, *directed=False*). Here p is calculated using c: average degree of the original graph. p = 0.40

Average path length = 5.1739267382903
Local clustering coefficient = 0.0750925311707

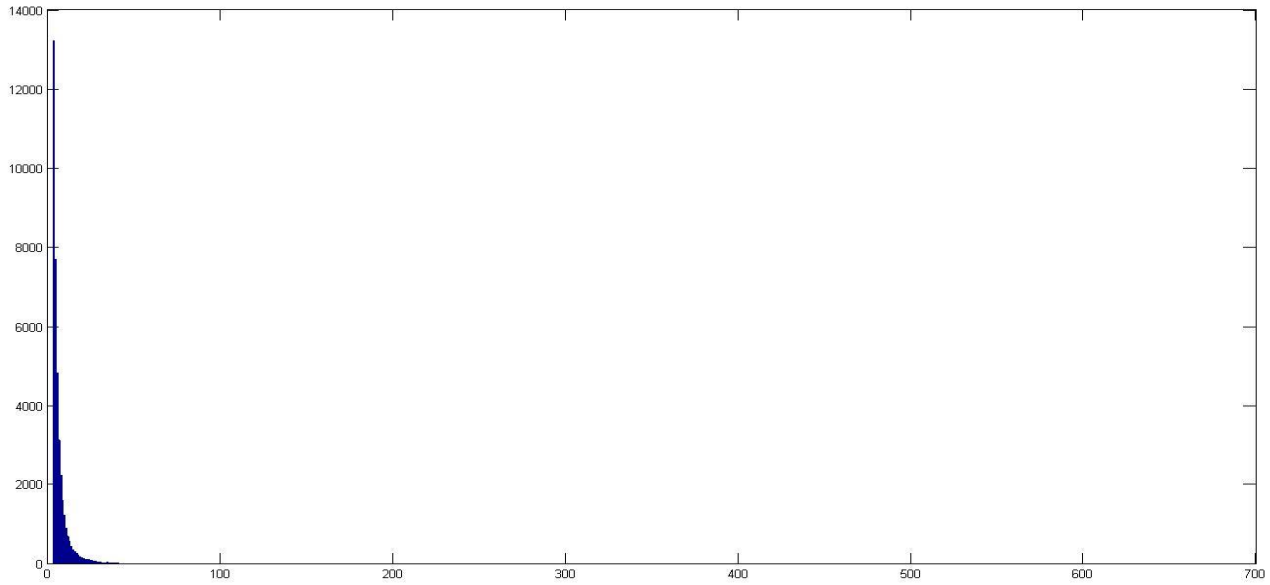
Global clustering coefficient = 0.083293644101



3) Preferential attachment model

- This graph is similar to the original graph in terms of degree distribution.
- barabasi_albert_graph(*n, m, seed=None) method is used to simulate this graph.*
- It takes as input n: number of nodes and d: average degree = 4
- This graph was created by using the NetworkX library for Python Networkx (https://networkx.github.io/). It mimics the power law distribution.

  Average path length = 5.7202182642915
  Local clustering coefficient = 0.0003723162922
  Global clustering coefficient = 0.000262824185



Y: Frequency of nodes
X: Degree distribution

Comparison between the different graph models:

	Average path length	Local clustering coefficient	Global clustering coefficient	Average degree
Original graph	4.1720183291770	0.0650025871704	0.0735437482521	3.4
Random graph	5.1026136174131	3.7182637720102	6.811403229302	3.4
Small world graph	5.1739267382903	0.0750925311707	0.083293644101	4.0
Preferential attachment	5.7202182642915	0.0003723162922	0.000262824185	4.0

It is observed that only the preferential attachment model and the original graph share the power law distribution.

From this comparison, we realize that the average degree of a random graph is same as our original graph.

The clustering coefficients of the original graph are best represented by the preferential model.

The average path lengths are not similar to any of the simulated models.