CSE-598 Social Media Mining

Project -1

Phase -1

-Kannan Ravindran

Introduction

This Social media mining project focuses on crawling a social media network and analyzing various aspects of the dataset with simulated graphs. In my project, I have used ‘Quora’ as the target social network. This project is based on python and several libraries such as SNAP, networkx, etc. The project is separated into four different parts. Let us discuss them in detail.

Part -1

The first part is crawling the quora for datasets. Starting from a user as root node i crawled all the followers of that user and making them their child. In the second iteration I repeated the process by considering each child as a parent and proceeding with the crawl. The process ends when I have visited 1000 nodes (and their followers). The user list is exported into a single CSV file.

Note: Since the followers for every user i crawled never crossed 1000, The dataset I originally crawled didn't require sampling.

The user list is anonymized in order to protect the privacy of the users.

Deliverables:

Crawl.py: The python file which was used to login, traverse and crawl the nodes

USERLIST.csv: The CSV file containing the crawled Dataset.

ANONUSERS.csv: The CSV file containing the list of anonymized user list

map.csv: The CSV file containing the list of mapped nodes for anonymized list.

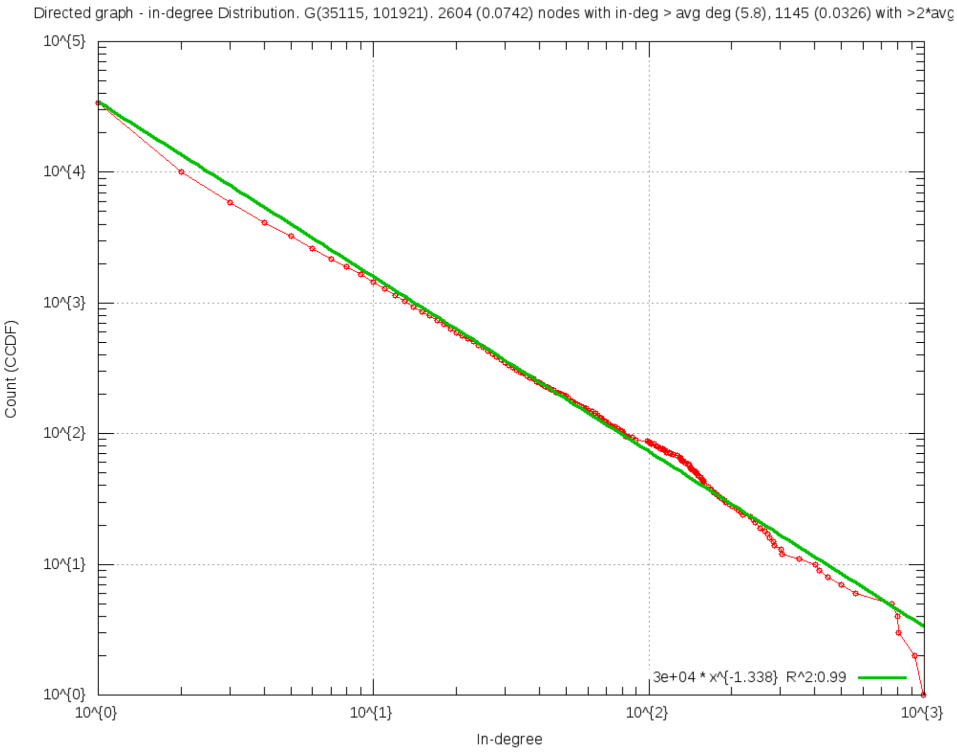
tuple.py: The python file which was used to generate the edgeless

U_tuples.csv: The CSV file containing the edgelist.

Part - 2

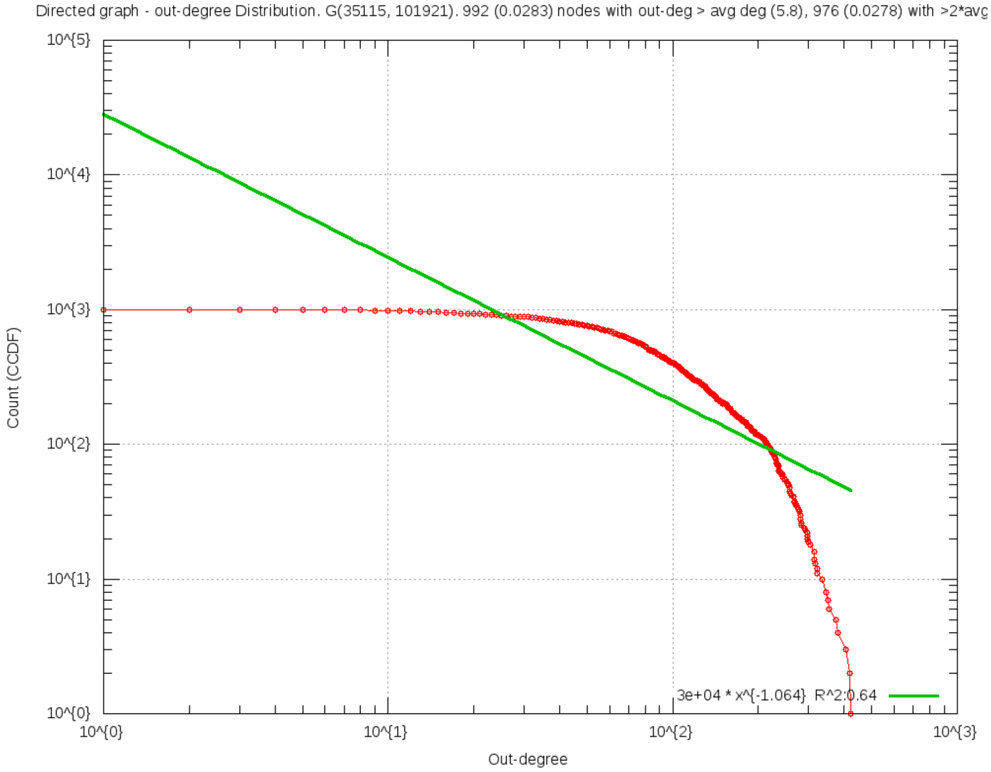
1. IN-DEGREE DISTRIBUTION

The power law distribution is 3e+04*X^{-1.338} R^2:0.99



2. OUT-DEGREE DISTRIBUTION

The power law distribution is 3e+04*X^{-1.064} R^2:0.64



—> The number of 3-cycles in the graph is calculated by using network
The number of 3 cycle graphs are 35668

—> The number of bridges in the graph is calculated using SNAP.
The bridge count in the graph is 24069

—> The Diameter of the whole graph is calculated using the predefined method in networkx
The Diameter of the whole graph is 7.

—> In the graph generated X% of the edges are removed and the largest connected component is calculated. where the X is varied from 1 to 100.

Deliverables:

inDegC.inDegreeDistribution.png: The in degree distribution graph

outDegC.outDegreeDistribution: The out degree distribution graph

degree.py: The python file for plotting the in degree and out degree distribution

bridges.py: The python file for finding the number of bridges in graph

did.py: The python file for calculating the diameter of the graph

triangles.py: The python file for calculating the number of 3 cycle graphs

x_percent.py: The python file for calculating the connected component when x% of edges are removed

diameter.txt: output file of the diameter

triangle.txt: output file of the 3 cycle graphs

x_percent.py: output file of connected components when x% of edges are removed

Part - 3

—> The average local and global clustering coefficients are calculated using python and network libraries.
The Global clustering coefficient was observed to be 0.0 and the local clustering coefficient can be viewed from the output file provided.

—> The Page Rank, Eigen Vector centrality, and degree centrality are found for the graph and the top 10 results are computed and analyzed

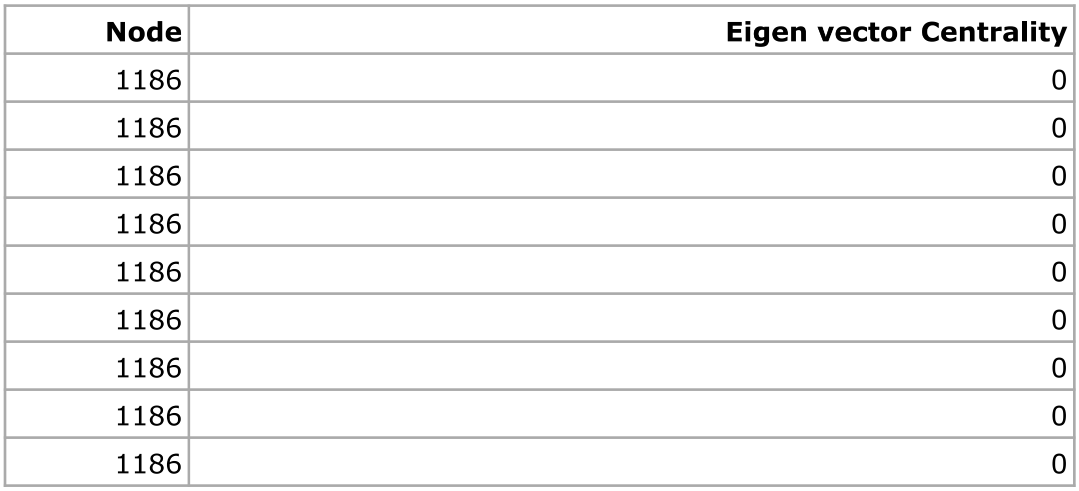
PageRank:

Node	PageRank
1315	0.00019483064547923
1147	0.0003895279483639391
1060	0.00044609014189622893
1186	0.00014929920692289585
1207	0.000304102872232349
1183	0.0002748733803179953
1428	0.00015294659185854082
1320	0.00015085406406843227
1103	0.00027519720475141236
1402	0.00017328193323527482

Degree Centrality:

Node	Degree Centrality
1315	0.01572882496425267
1147	0.02579415145652844
1060	0.027167970392800066
1186	0.01393444921075504
1207	0.022569882524462386
1183	0.021196063588190765
45	0.011775590882328203
765	0.011859702245773403
1320	0.012392407547593012
1103	0.021896991616900775

Eigen Vector Centrality:



Node	Eigen vector Centrality
1186	0
1186	0
1186	0
1186	0
1186	0
1186	0
1186	0
1186	0
1186	0

Note: All the eigen vector centrality is 0 in my case. Thus the top 10 nodes are the same.

—> The jaccaard similarity of all the nodes were calculated and found that the maximum values was 1.

Jaccard Similarity Pair: [(‘ 5980’, ' 31819')]

Jaccard Similarity Maximum: 1.0

Deliverables:

avg_clust.csv: The output file of average local clustering

deg_cent.csv: The output file of degree centrality

eigen_cent.csv: The output file of eigen vector centrality

pagerank.csv: the output file of page rank

clust_global.txt: The output file of global clustering

jaccard_output.txt: the output file of jacquard similarity

clustering.py: The python file for calculating the average local and global coefficient

correlation.py: The python file for calculating the page rank, eigen vector centrality and degree centrality

jaccard.py: The python file for calculating the jaccard similarity

Part - 4

Since we have finished plotting and analyzing the graph we generated. Let us compare it with the simulated graphs.

I have simulated

1. Random Graph
2. Small-World Graph
3. Preferential Graph

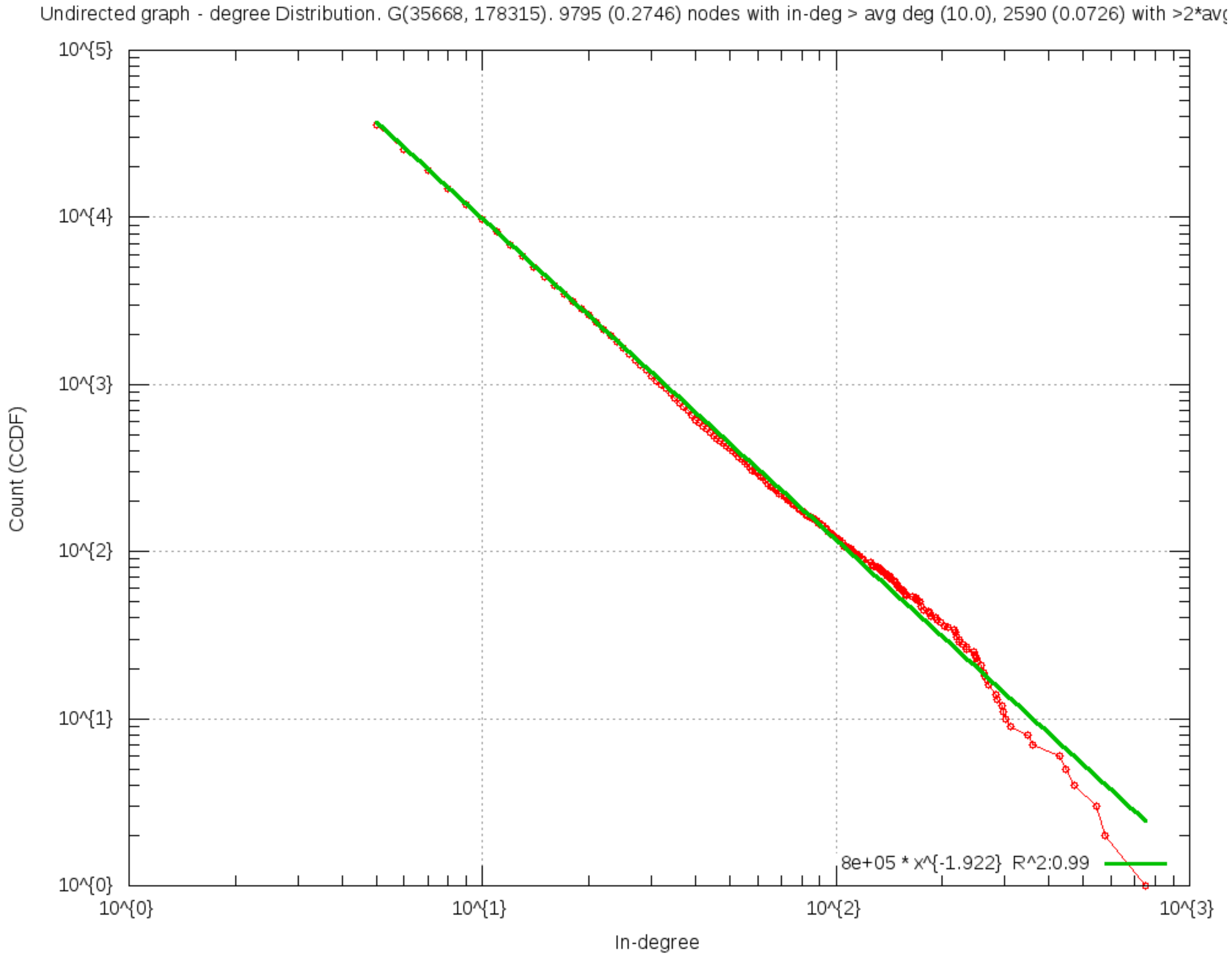
Let us compare the the simulated graphs with the real time graph using the measures

1. Average Path Length
2. Clustering Coefficient.

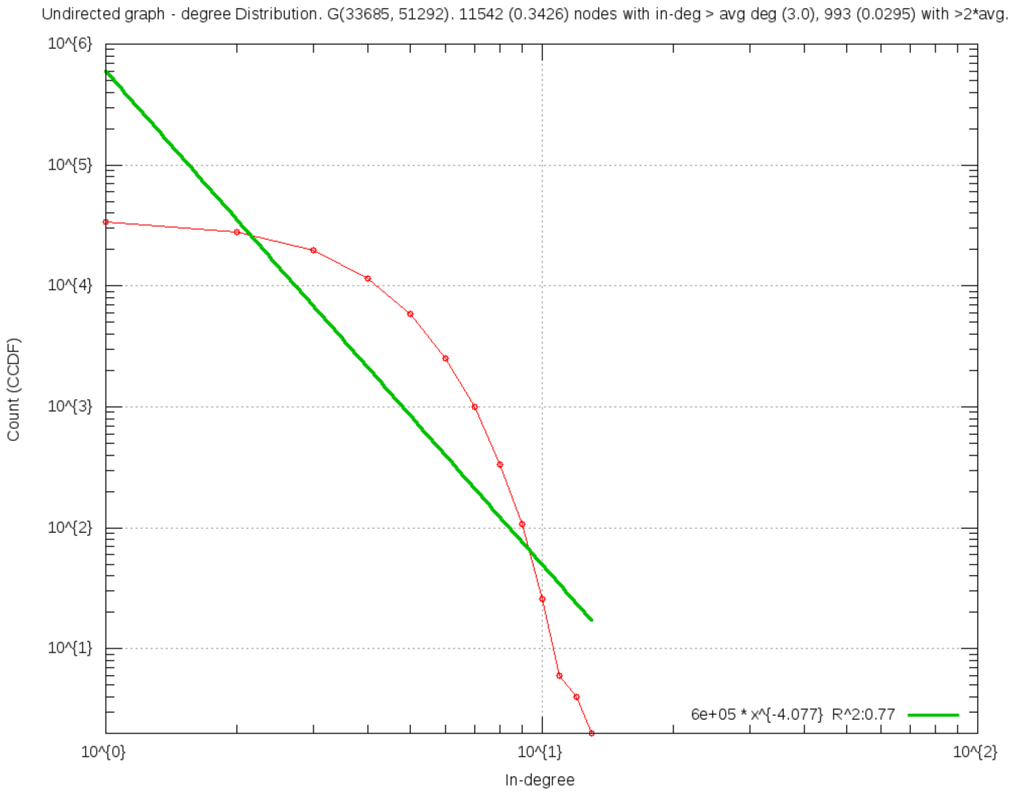
	Average Path Length	**Clustering Coefficient**
Real World Graph	2.96675435456	0.0034567452834
Small World Graph	2.41176470588	0.0000000403564
Preferential Graph	4.01881356701	0.0023722732492
Random Graph	0	0

I wasn't able to calculate the values for the random graph since the random graph generated was not connected and I wasn't able to generate a connected graph using the networkx.

Preferential Graph



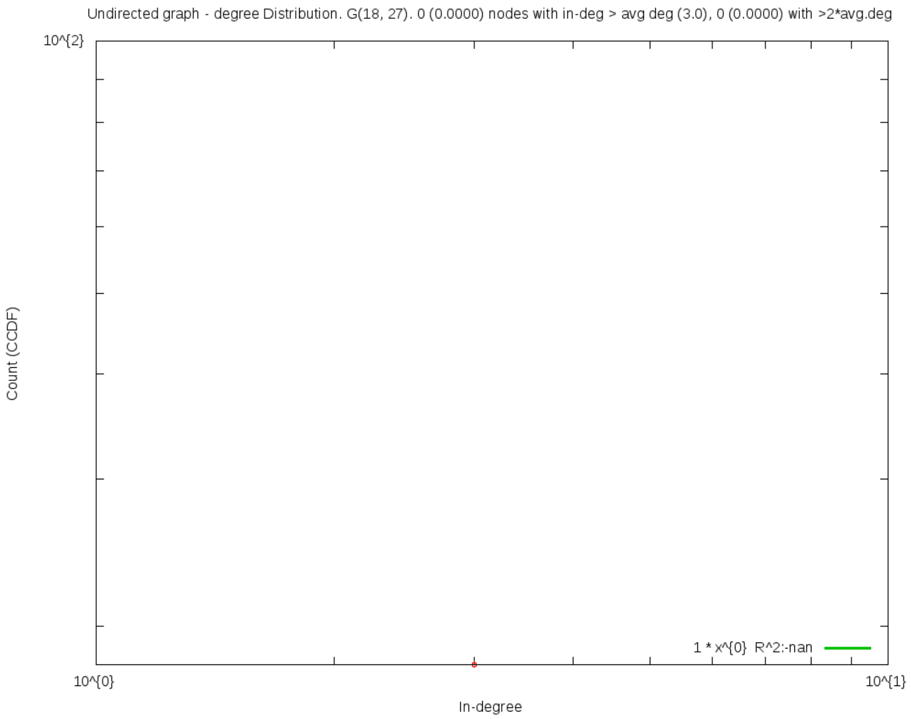
Random Graph



Small Graph

Since the number of nodes is very small in small graph it is barely seen

(look at the middle of x axis.)


Undirected graph - degree Distribution. G(18, 27). 0 (0.0000) nodes with in-deg > avg deg (3.0), 0 (0.0000) with >2*avg.deg
10^{2}
Count (CCDF)
1 * x^{0} R^2:-nan
10^{0}
10^{1}
In-degree