CSE508 Information Retrieval Assignment-3 Report

Dataset Description:

A sequence of snapshots of the Gnutella peer-to-peer file sharing network from August 2002. There are total of 9 snapshots of Gnutella network collected in August 2002. Nodes represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts. There are 2 columns and the network characteristics are as follows:

Represent the network in terms of its 'adjacency matrix' as well as 'edge list'

```
[11] num_nodes = len(unique_nodes)
     adj_matrix = np.zeros((num_nodes+3, num_nodes+3))
     print(adj_matrix)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

```
for index, row in df.iterrows():
    fr_row=int(row["FromNodeId"])
    # fr.append(fr_row)
    to_row=int(row["ToNodeId"])
    # to.append(to_row)
    adj_matrix[fr_row][to_row]=1
```

```
[13] #this is the adjacency matrix
     adj_matrix
```

```
array([[0., 1., 1., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
#finding edhge list
edge_list=[]
for index,row in df.iterrows():
    temp_list=[]
    temp_list.append(row["FromNodeId"])
    temp_list.append(row["ToNodeId"])
    edge_list.append(temp_list)
```

```
[18] edge_list
```

```
[['0', '1'],
 ['0', '2'],
 ['0', '3'],
```

Number of Nodes

```
[20] #number of nodes
     print("number of nodes -> ",len(np.unique(unique_nodes)))
```

```
number of nodes ->  10876
```

Number of Edges

```
[21] #length of edge_list
     print("count of edges-> ",len(edge_list))
```

```
count of edges->  39994
```

Average In-degree

```
[22] # code for finding average in degree
     in_degrees = np.sum(adj_matrix, axis=1)
     avg_in_deg = np.mean(in_degrees)
     print("average in degree", avg_in_deg)
```

```
average in degree 3.6762570089162607
```

Average Out-Degree

```
#code for finding average out degree
out_degrees = np.sum(adj_matrix, axis=0)
avg_out_deg = np.mean(out_degrees)
print("Average Out-Degree of Network:", avg_out_deg)
```

```
Average Out-Degree of Network: 3.6762570089162607
```

Node with Max In-degree

```
[24] #code for finding node with maximum out degree
     out_dict={}
     max_degree = 0
     max_node = -1
     for i in range(len(adj_matrix)):
         outdegree = sum(adj_matrix[i])
         out_dict[i]=outdegree
         if outdegree > max_degree:
             max_degree = outdegree
             max_node = i
     print("node with max out degree",max_node,int(max_degree))
```

```
node with max out degree 3109 100
```

Node with Max out-degree

```
#code for finding node with maximum in degree
in_dict={}
max_in_degree=0
max_out_node=-1
for i in range (len(adj_matrix[0])):
  count=0
  for j in range(len(adj_matrix)):
    if (adj_matrix[j][i]==1):
      count+=1
  in_dict[i]=count
  if (count>max_in_degree):
    max_in_degree=count
    max_out_node=i
print("node with max in degree",max_out_node,int(max_in_degree))
```

```
node with max in degree 1054 72
```

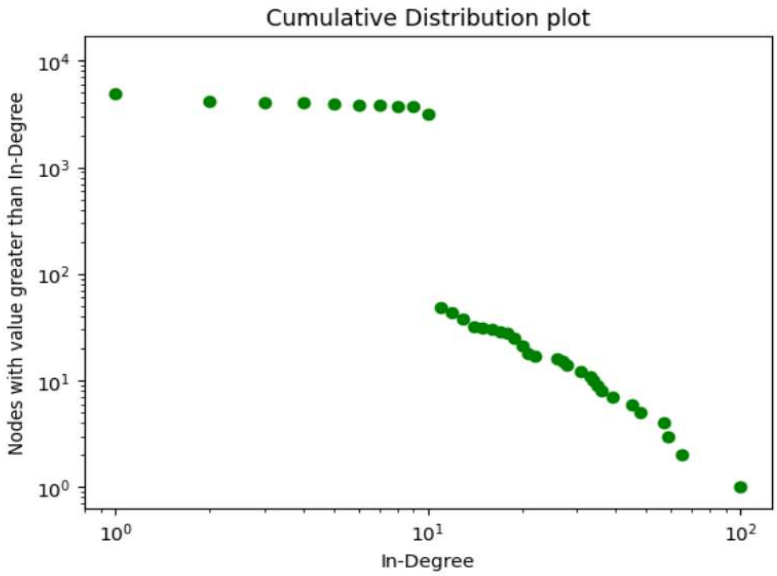
The density of the network

```
[26] print("density of the network", no_of_edge/(len(unique_nodes)*(len(unique_nodes)-1)))
```

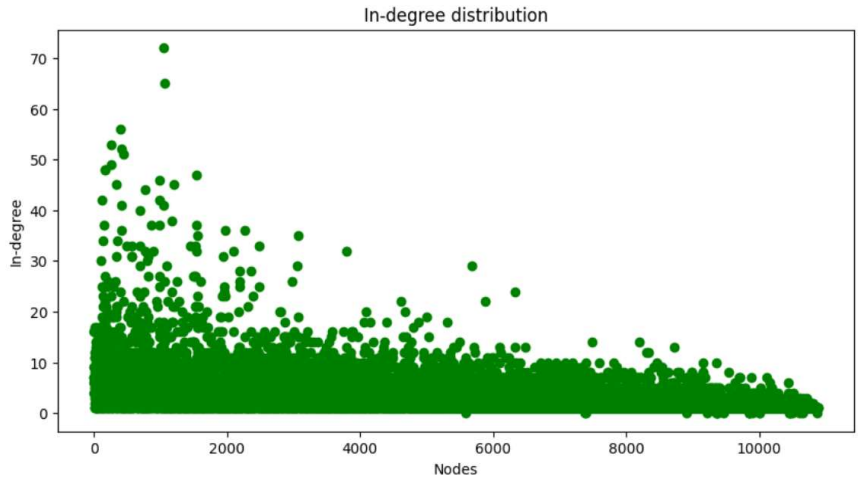
```
density of the network 0.0003381398671756435
```

The cumulative distribution plot shows the in-degree values are distributed across the nodes in the network whereas the scatter plot shows the exact in -degree values for each individual node.

```
sequence_degree_in= sorted(in_degrees, reverse=True)
Count_degree = collections.Counter(sequence_degree_in)
degree, count = zip(*Count_degree.items())
cs = np.cumsum(count)
plt.loglog(degree, cs, 'go')
plt.title("Cumulative Distribution plot")
plt.ylabel("Nodes with value greater than In-Degree")
plt.xlabel("In-Degree")
plt.show()
```

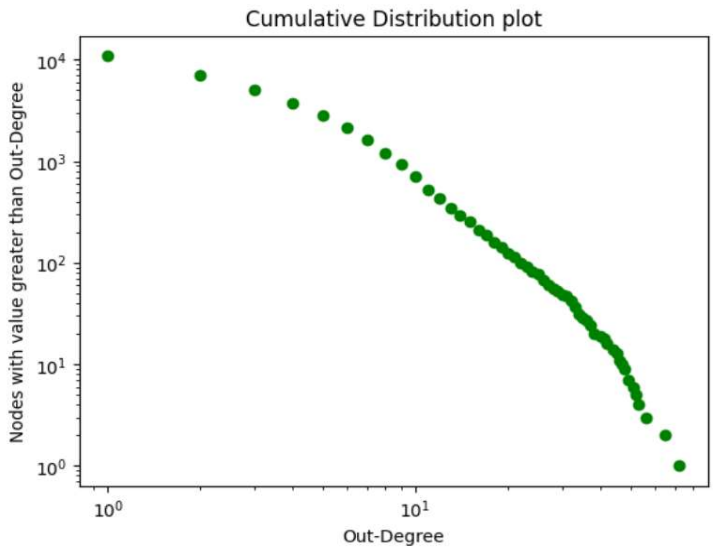


```
labels_in = list(in_dict.keys())
values_in = list(in_dict.values())
# Create scatter plot
fig, ax = plt.subplots(figsize=(15, 8))
ax.scatter(labels_in, values_in, color='green')
ax.set_xlabel('Nodes')
ax.set_ylabel('In-degree')
ax.set_title('In-degree distribution')
plt.show()
```

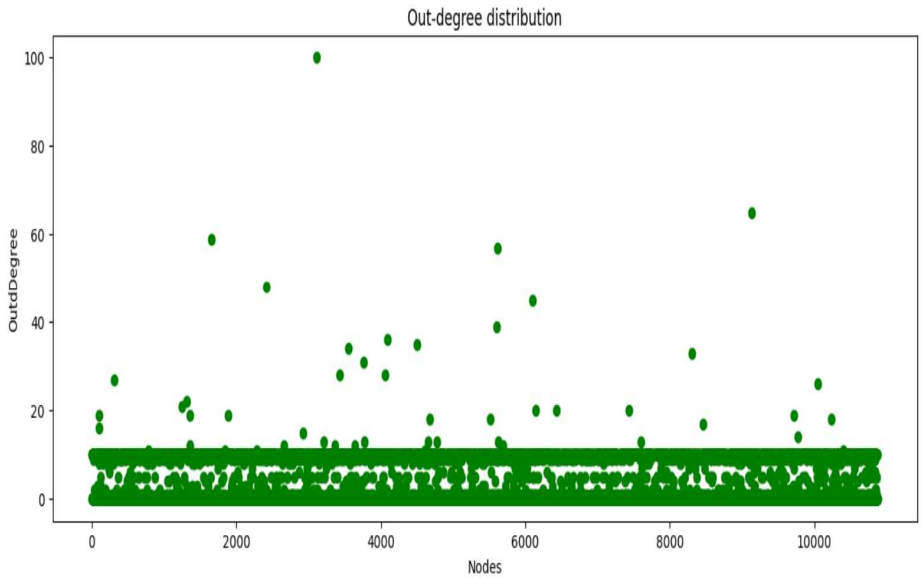



```
sequence_degree_out = sorted(out_degrees, reverse=True)
degree_Count_Out = collections.Counter(sequence_degree_out)
degree_out, count_out = zip(*degree_Count_Out.items())
cs_out = np.cumsum(count_out)
plt.loglog(degree_out, cs_out, 'go')
plt.title("Cumulative Distribution plot")
plt.ylabel("Nodes with value greater than Out-Degree")
plt.xlabel("Out-Degree")
plt.show()
```

```
plt.show()
```




```
labels_out = list(out_dict.keys())
values_out = list(out_dict.values())
# Create scatter plot
fig, ax = plt.subplots(figsize=(14, 5))
ax.scatter(labels_out, values_out, color='green')
ax.set_xlabel('Nodes')
ax.set_ylabel('OutdDegree')
ax.set_title('Out-degree distribution')
plt.show()
```

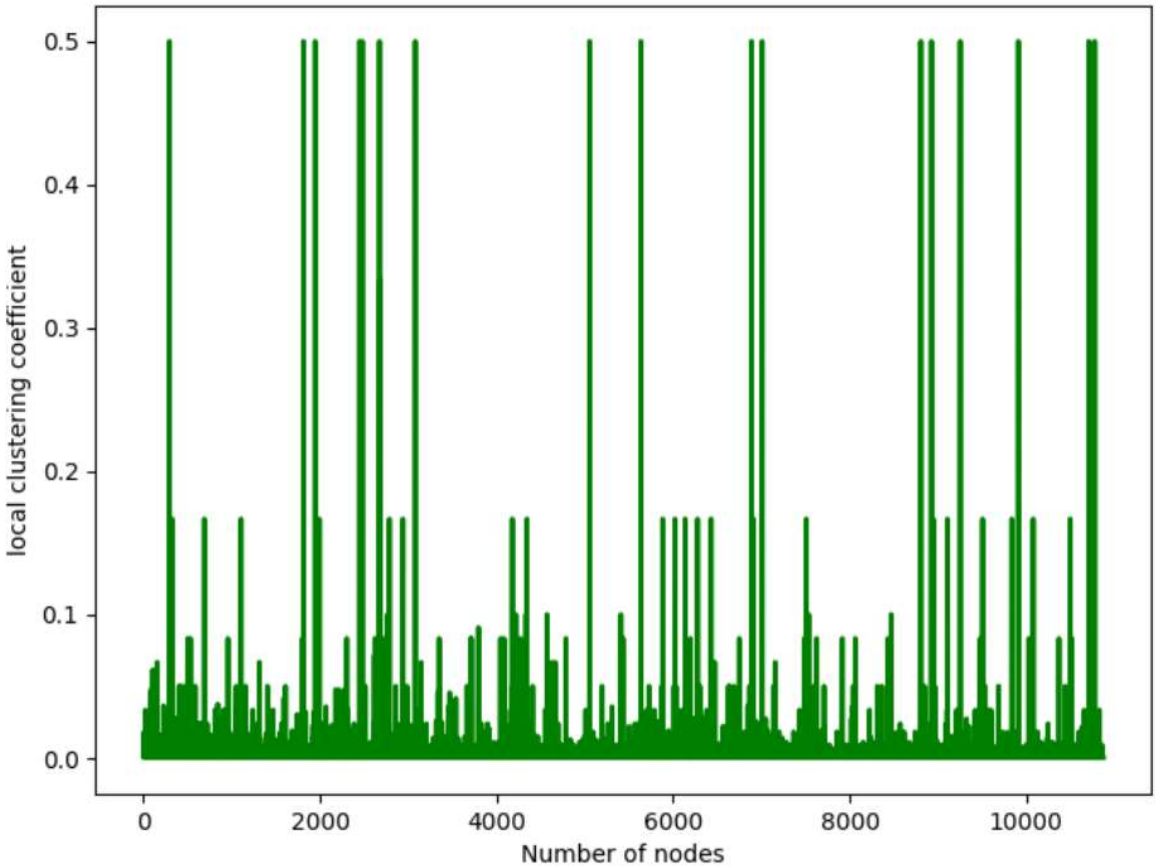


Local clustering coefficient of each node

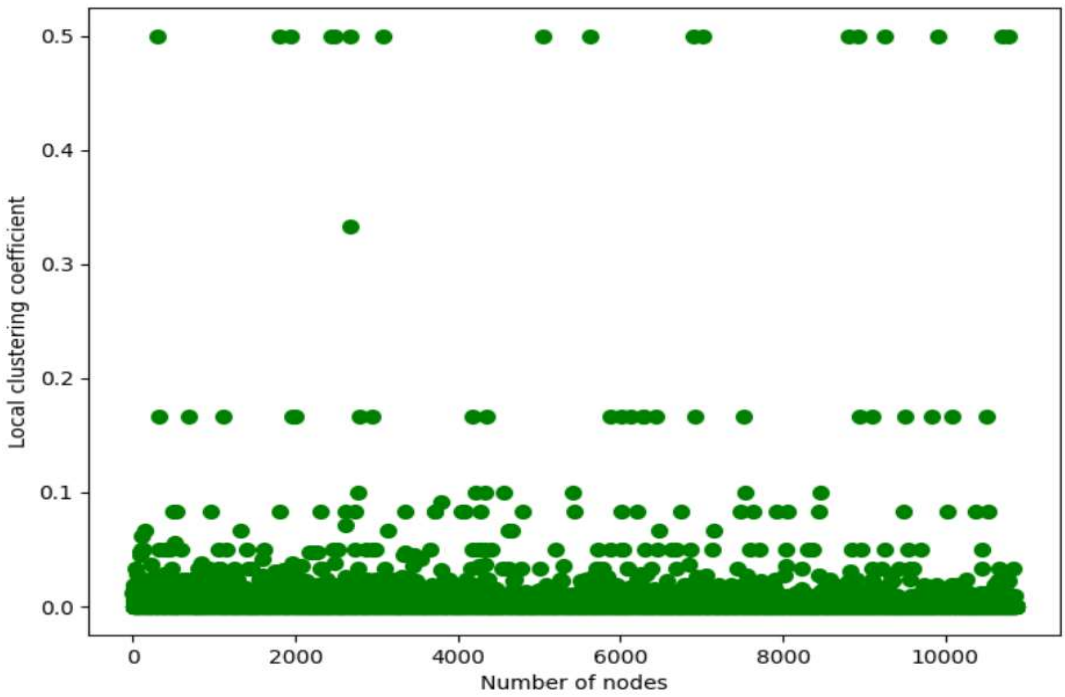
```
lcc = []
for i in range(num_nodes):
    indegree_neighbors = [j for j, row in enumerate(adj_matrix) if row[i] == 1]
    outdegree_neighbors = [j for j, col in enumerate(adj_matrix[i]) if col == 1]

    list_neighbour = indegree_neighbors + outdegree_neighbors
    new_list_neighbour = list_neighbour[:]
    if len(new_list_neighbour) < 2:
        lcc.append(0)
        continue
    number_pairs = len(new_list_neighbour) * (len(new_list_neighbour) - 1)
    if number_pairs == 0:
        lcc.append(0)
        continue
    number_connected_pairs = 0
    for u in new_list_neighbour:
        for v in new_list_neighbour:
            if u == v:
                continue
            if adj_matrix[u][v] == 1:
                number_connected_pairs += 1
    cc = number_connected_pairs / number_pairs
    lcc.append(cc)
```

```
#Plotting
local = np.array(lcc)
fig, ax = plt.subplots(figsize =(8, 6))
plt.xlabel("Number of nodes")
plt.ylabel("local clustering coefficient")
ax.plot(local,color='g',linewidth='2')
plt.show()
```




```
] #scatter plot
  local = np.array(lcc)
  fig, ax = plt.subplots(figsize=(8, 6))
  ax.scatter(range(len(local)),local, color='g', linewidth=2)
  plt.xlabel("Number of nodes")
  plt.ylabel("Local clustering coefficient")
  plt.show()
```



References:

Geeks for Geeks, Stack overflow

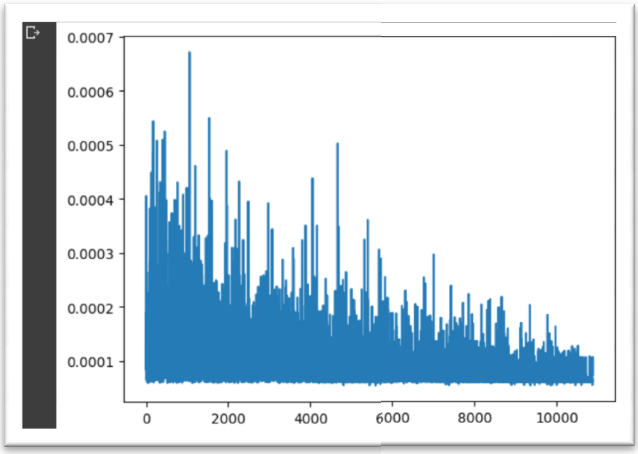
Question 2 - [35 points] PageRank, Hubs, and Authority
For the dataset chosen in the above question, calculate the following:

Methodology:
We first sliced the chosen "p2p-Gnutella04.txt" dataset to remove the unnecessary text and then read it into a data frame. Then used the networkx library for calculating the PageRank, hub, and authority.

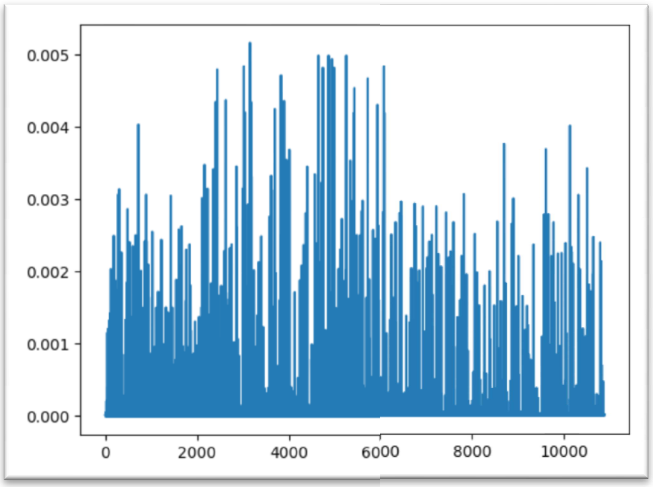
Assumptions:
We are assuming that the graph is directed and contains at least 100 nodes.

Results:

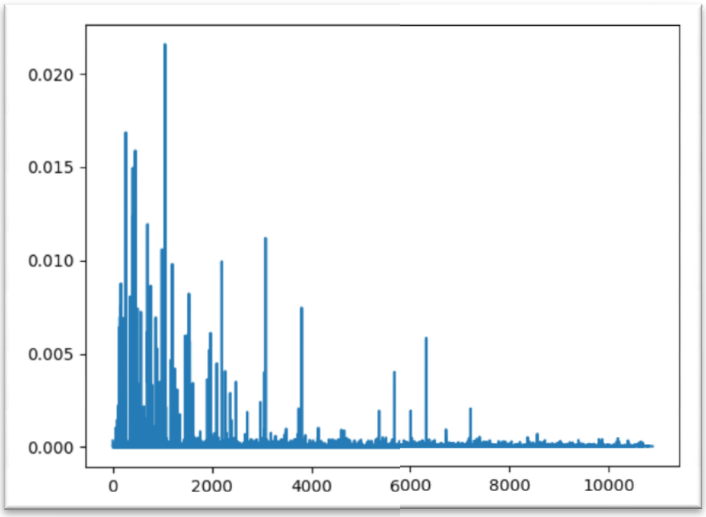
Plot of Page Rank



Plot of hub



Plot of authority



Compare the results obtained from both the algorithms in parts 1 and 2 based on the node scores.

PageRank computes a ranking of nodes in the graph based on the structure of the incoming links. On the other hand, the HITS algorithm computes the authority score for a node based on the incoming links and computes the hub score based on outgoing links.

	Node	PageRank	Hub	Authority
0	0	0.000121	0.000020	0.000332
1	1	0.000085	0.000020	0.000012
2	2	0.000136	-0.000000	0.000035
3	3	0.000122	0.000021	0.000045
4	4	0.000405	-0.000000	0.000121

PageRank: It is used as a metric to determine a website's popularity for better ordering of the search results. If a page has a higher PageRank, which means it is popular.

Hub: Metric to determine how many authoritative pages a page is pointing to. If a page has a higher hub score, it means that it itself is a good page as it is pointing to many authoritative pages.

Authority: Metric to determine whether a page is authoritative or not and how that page will be ranked in the search results.

References:

GeeksForGeeks