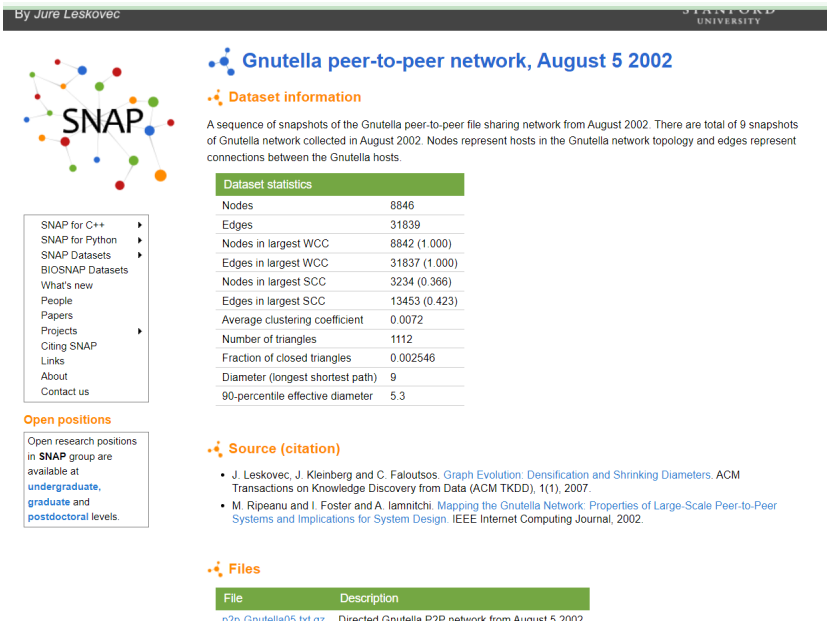
CSE508 Information Retrieval
Assignment 3

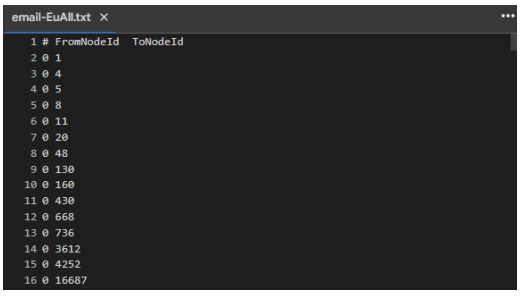
---

1. Link Analysis

Dataset Link: [Link to Dataset]



This is the structure of the dataset which we have used. As we can see in this file, the first three lines are of no use to us as we are creating a dataframe which contains two columns, the first column is **FromNodeID**, and the second column is **ToNodeID**.



After processing the text, and removing the extra lines, we created a pandas dataframe as it is easy to handle.

```
df = pd.read_csv("/content/p2p-Gnutella05.txt", sep="\t")
df.rename(columns = {'# FromNodeId': 'FromNodeId'}, inplace = True)
df.head(10)
```

	FromNodeId	ToNodeId
0	0	1
1	0	2
2	0	3
3	0	4
4	0	5
5	0	6
6	0	7
7	0	8
8	0	9
9	0	10

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

Ans: Adjacency matrix is a matrix that is used to represent a graph. Its shape is of **n*n** dimension where n is the number of nodes in the graph. Initially the matrix is filled with zeros. When there is an edge between node **i** and node **j**, then the cell matrix[i][j] is set to 1. After filling the matrix, our final matrix has the dimension of **(8846,8846).**

Represent the network in terms of its adjacency matrix

```
[78] adj_matrix = np.zeros((len(node_id_map), len(node_id_map)))
     for index, each_row in df.iterrows():
       src = each_row['FromNodeId']
       des = each_row['ToNodeId']
       src_mapping, dest_mapping = node_id_map[src], node_id_map[des]
       adj_matrix[src_mapping][dest_mapping] = 1

     print(adj_matrix)

     [[0. 1. 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.]
      [0. 0. 0. ... 0. 0. 0.]]
```

Now comes the **Edge-List**. The edge-list is a list of pairs of nodes which contain an edge. Let say, I have two nodes n1, and n2 which contain an edge. So the edge-list will look like this [(n1, n2)].

Edge List: *[(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 9), (0, 10), (1, 310), (1, 1091), (1, 1213), (1, 1517), (1, 3082)…]*

The length of the edge list will be the number of edges in the graph.

- **Briefly describe the dataset chosen and report the following:**
  - Number of Nodes: 8846
  - Number of Edges: 31839
  - Average In-Degree: 3.55925
  - Average Out-Degree: 3.59925
  - Node with Max In-Degree: 842 (Maximum value of In-Degree - 79)
  - Node with Max Out-Degree: 3002 (Maximum value of Out-Degree - 65)
  - The density of the network: 0.0004069

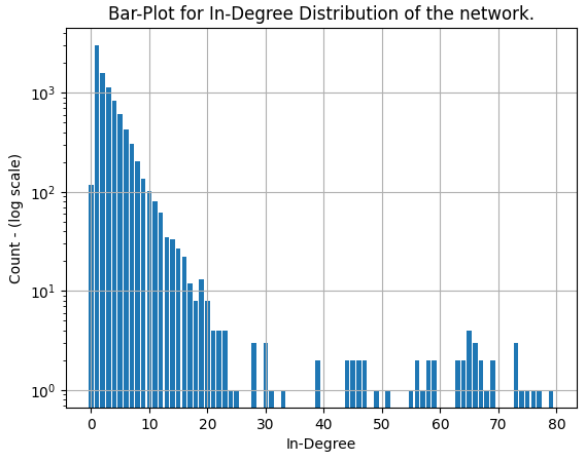
The formula used to calculate the density of the graph network:

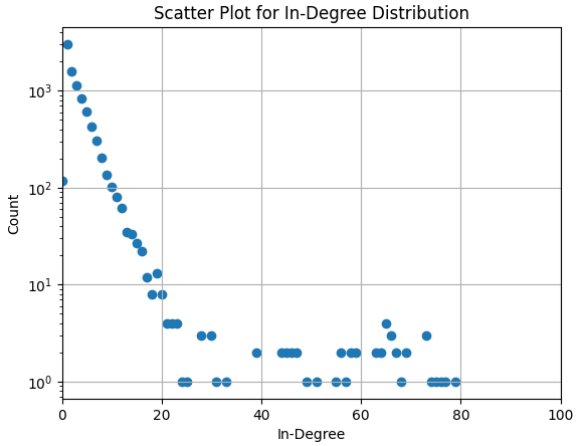
$$\eta = \frac{|E|}{|V|(|V|-1)}$$

Degree Distribution of Network -

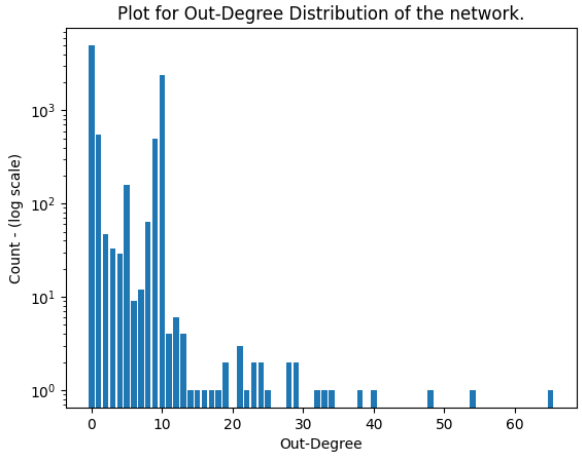
As the values are extremely high and low for the count of values corresponding to a certain value of degree, the scale of Y-Axis is set to Logarithmic for all the plots.

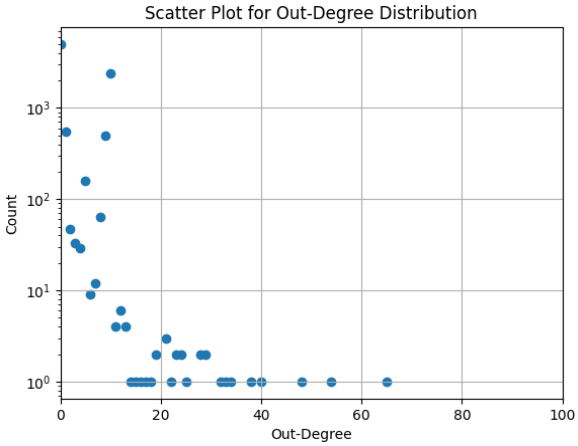
1. In - Degree vs. Count





2. Out-Degree vs. Count





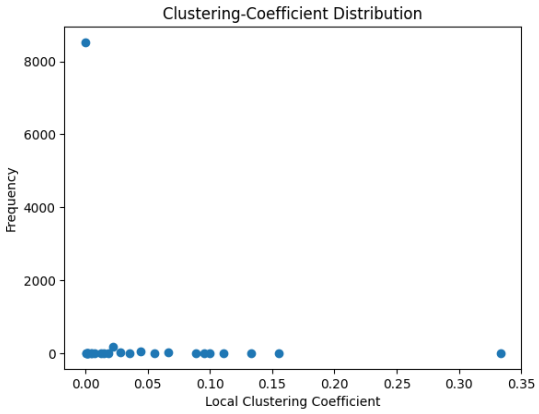
Clustering Coefficient Distribution -

```python
# Calculate the local clustering coefficient of each node
lcc = np.zeros(num_nodes)
for node in range(num_nodes):
  neighbors = np.nonzero(adj_matrix[node])[0]  # Get the neighbors of the node
  num_neighbors = len(neighbors)
  if num_neighbors < 2:
    lcc[node] = 0  # If the node has less than two neighbors, its lcc is 0
  else:
    num_links = 0
    for i in range(num_neighbors):
      for j in range(i+1, num_neighbors):
        if adj_matrix[neighbors[i], neighbors[j]] == 1:
          num_links += 1

    lcc[node] = 2 * num_links / (num_neighbors * (num_neighbors - 1))

lcc_values, lcc_counts = np.unique(lcc, return_counts=True)
```

```python
plt.scatter(lcc_values, lcc_counts)
plt.xlabel("Local Clustering Coefficient")
plt.ylabel("Frequency")
plt.title("Clustering-Coefficient Distribution")
plt.show()
```



Formula used to calculate the local clustering coefficient of each node:

$$lcc = \frac{2 * no\ of\ links}{no\ of\ neighbors\ *\ (neighbors - 1)}$$

2. PageRank, Hubs and Authority

- **PageRank score for each node**

  PageRank is an algorithm used by Google Search to rank web pages in their search engine results. It is a link analysis algorithm that assigns a numerical weight to each node of a web page, with the purpose of measuring its relative importance within the set. We have used the networkX library to implement this algorithm.

```
# Load the dataset from URL
url = "/content/wiki-Vote.txt.gz"
G = nx.read_edgelist(url, comments="#", delimiter="\t", create_using=nx.DiGraph)
```

PageRank score for each node

```
# Compute the PageRank score for each node in the network
pagerank_scores = nx.pagerank(G)
```

```
# Print the PageRank score for each node in the network
for node in pagerank_scores:
    print(f"Node: {node}, Pagerank Score: {pagerank_scores[node]}")
```

- **Authority and Hub score for each node**

- **Authority Score:** Authority score is a measure of the relative importance and trustworthiness of a website or web page. It is usually based on a combination of factors such as the number and quality of links pointing to the website or page, the age of the website, and the overall reputation of the website in its industry.
- **Hub Score:** Hub score is a metric used in the **HITS (Hyperlink-Induced Topic Search) Algorithm**, which is an algorithm that analyzes the links between web pages to identify authoritative pages on a given topic.

Authority and Hub score for each node

```
authority_scores, hub_scores = nx.hits(G)

for node in authority_scores:
    print(f"Node: {node}, Authority Score: {authority_scores[node]}, Hub Score: {hub_scores[node]}")
```

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

To compare the results obtained from both algorithms based on the node scores, we need to analyze the values obtained for each node from the HITS and Pagerank algorithms.

For Node 0, the HITS algorithm has an authority score of 4.415758056445845e-06 and a hub score of 2.7475298528765412e-05, while the Pagerank algorithm has a score of 0.00010661245693315142. This means that Pagerank assigns a much higher importance to Node 0 compared to HITS.

Similarly, for Node 1, the Pagerank algorithm assigns a higher score of 0.0001662986455136566 compared to the HITS algorithm's authority score of 2.7544130314596964e-06 and hub score of 7.477453295145642e-05.

For Node 2, the HITS algorithm assigns a non-zero score, while the Pagerank algorithm assigns a score of 7.900728650495317e-05.

Overall, we can conclude that the Pagerank algorithm assigns higher scores to nodes compared to the HITS algorithm. This may be due to the fact that the Pagerank algorithm considers the global link structure of the entire network, while the HITS algorithm focuses only on the local link structure around each node. Therefore, if the focus is on the importance of nodes in the context of the entire network, Pagerank may be a better algorithm to use.

—----------