CSE508 - INFORMATION RETRIEVAL

REPORT - ASSIGNMENT 3

Introduction

The assignment was done together by all three members of the group. The dataset used contains 1400 files in text format. It contains 25570 entries containing two values representing an edge from one node to another.

PROBLEM 1

Dataset Description

The dataset chosen is '**email-Eu-core network**' represents a directed graph. It contains 25571 entries containing two values representing an edge from one node to another. The total number of nodes are 1005 and the total number of edges are 25571, which means that there was no entry that was repeated.

Methodology and Results

The dataset is used to form a list 'nodes', storing each node appearing in the data; an 'edge list', that is a list containing all the edges of the graph, which is then used to form the 'adjacency matrix', which tells whether there exists an edge from node i to node j or not, using 0 or 1.

Two lists 'a' and 'b' were also formed to store the in-degree and out-degree of each node; these were formed using the adjacency matrix.

The results of the required metrics, and the formulas used are given below:

Formulas are defined and given below: ('am' is the adjacency matrix, 'nodes' is list of nodes and 'edges' is the edge list)

```
a=list(np.sum(am,axis=0))
```

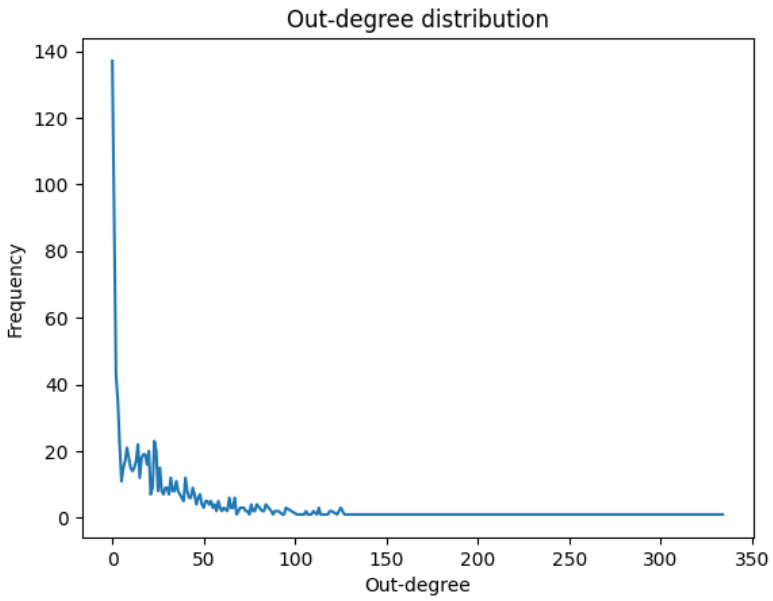
```
b=list(np.sum(am,axis=1))
average_indegree=sum([int(i) for i in a])/len(nodes)
average_outdegree=sum([int(i) for i in b])/len(nodes)
max_indegree,max_in_id=max(a),a.index(int(max(a)))
max_outdegree,max_out_id=max(b),b.index(int(max(b)))
density_of_graph=len(edges)/(1005*1005)
print("Number of nodes: "+str(len(nodes)))
print("Number of edges: "+str(len(edges)))
print("Average in-degree: "+str(average_indegree))
print("Average out-degree: "+str(average_outdegree))
print("Node with max in-degree: "+str(max_in_id))
print("Node with max out-degree: "+str(max_out_id))
print("Network Density: "+str(density_of_graph))
```

The results of these 7 metrics are given below:

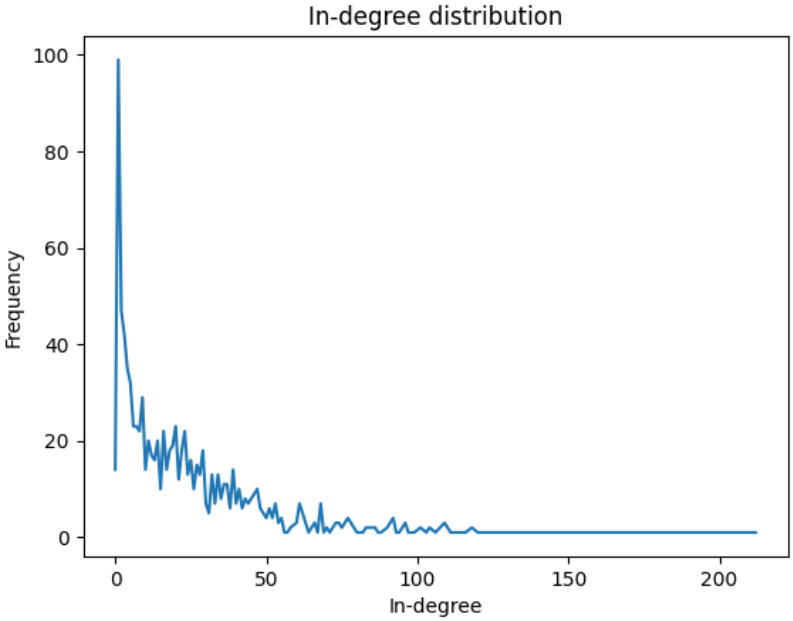
```
Number of nodes: 1005
Number of edges: 25571
Average in-degree: 25.443781094527363
Average out-degree: 25.443781094527363
Node with max in-degree: 160
Node with max out-degree: 160
Network Density: 0.0253171951189327
```

A frequency function was made to store the frequency distribution of values in a list. The degree distribution for both in-degree and out-degree were directly calculated using this function on lists ‘a’ and ‘b’ defined above, respectively. The graphs are given below:

```
d=freq(a)

d = sorted(d.items(), key= lambda x:x[0])

plt.plot([x[0] for x in d], [x[1] for x in d])
```




```
d=freq(b)

d = sorted(d.items(), key= lambda x:x[0])

plt.plot([x[0] for x in d], [x[1] for x in d])
```



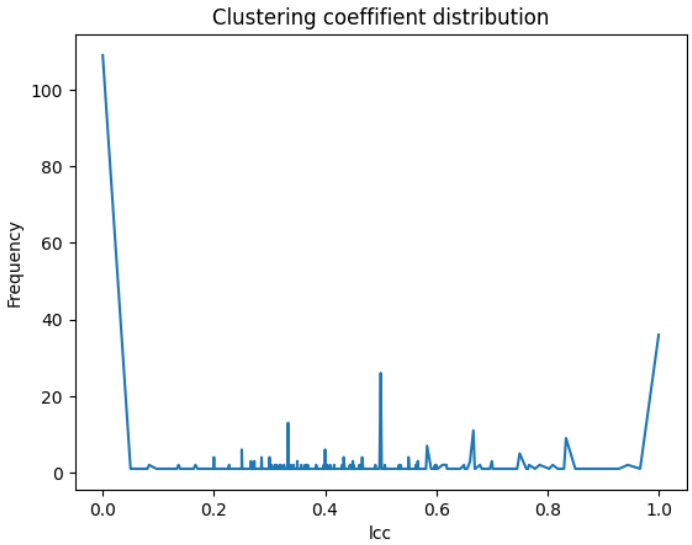
To calculate the local clustering coefficient of each node, a loop was used to iterate over all the nodes. In this loop, another loop over all nodes was first run to get the list of neighbours of the index node, then another loop was run over the neighbour nodes to calculate the number of edges between the neighbours. The term 'num_of_neigh' is used to store the number of neighbours of the index node, and 'count_link' is used to store the number of edges between these neighbours.

The lcc value for the index node 'n' is then calculated using the following command, and stored in the list 'lcc':

```
if num_of_neigh<=1:
    lcc[n]=0
  else:
    lcc[n]=count_link/(num_of_neigh*(num_of_neigh-1))
```

The lcc vs frequency of lcc graph was plotted similarly by using the freq function on the list lcc, the formula/commands and the result are given below:

```
d=freq(list(lcc.values()))
d = sorted(d.items(), key= lambda x:x[0])
plt.plot([x[0] for x in d], [x[1] for x in d])
```



PROBLEM 2

Methodology:

We used networkx library to compute PageRank, hub, and authority scores.

Preprocessing:

Preprocessing is same as previous question.

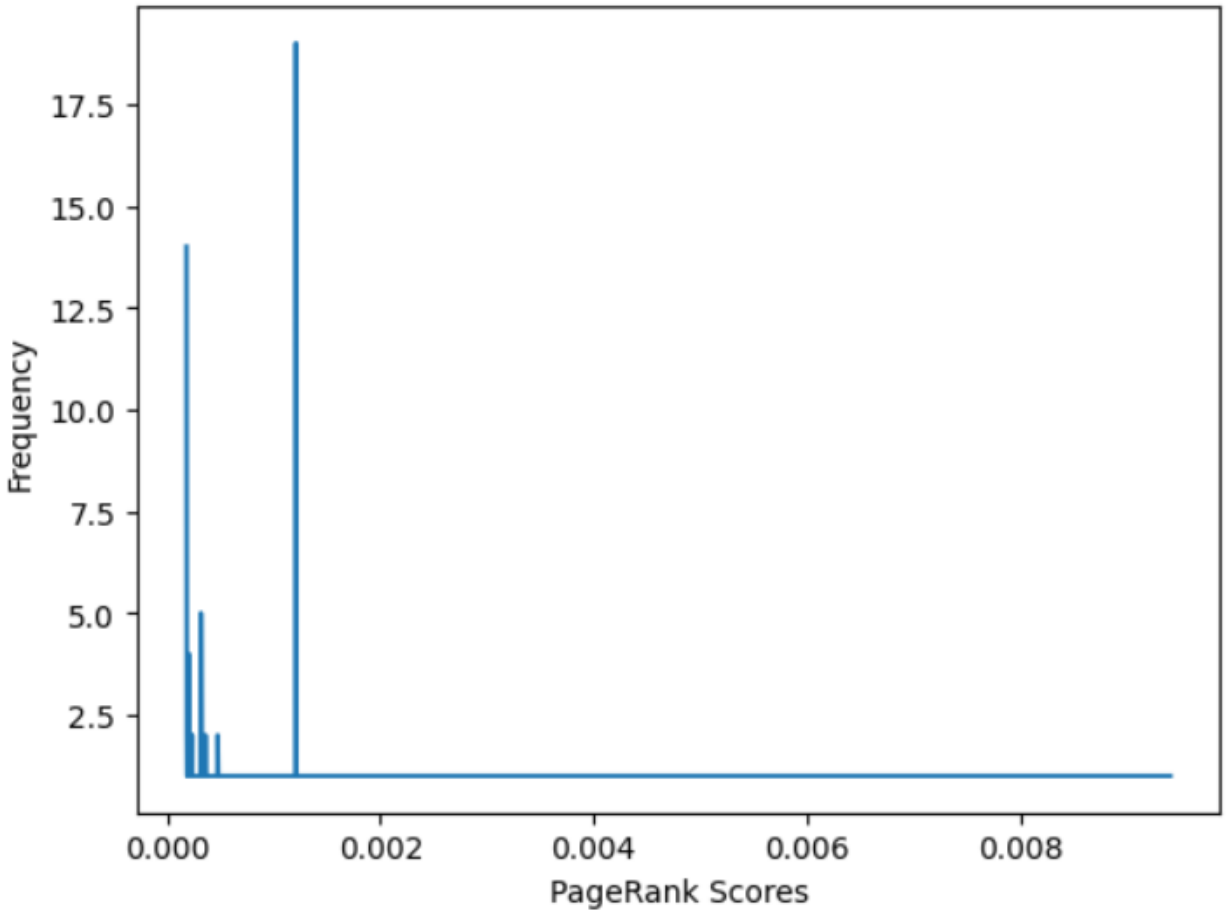
Comparison:

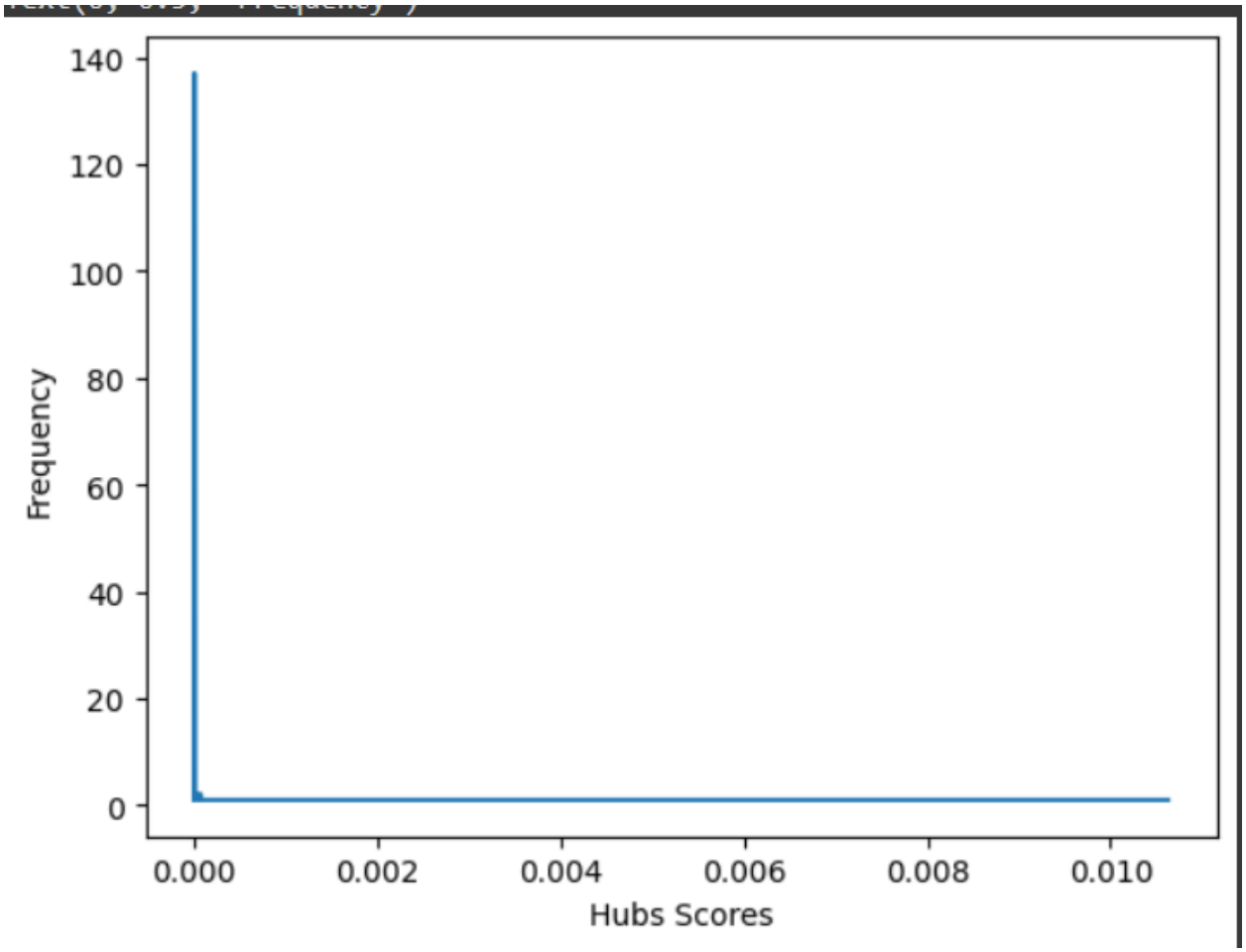
```
Top 20 NodeIDs based on PageRank Score:
[(1, 0.009411560186382712),
 (130, 0.006913890234439256),
 (160, 0.006758893760759583),
 (62, 0.005322217132261051),
 (86, 0.005130048318172175),
 (107, 0.005004366663608102),
 (365, 0.004786689611422023 5),
 (121, 0.004720808207765978),
 (5, 0.004525470848399022),
 (129, 0.004452932454005516),
 (183, 0.004274077086284579),
 (64, 0.004212852305495342)]
```

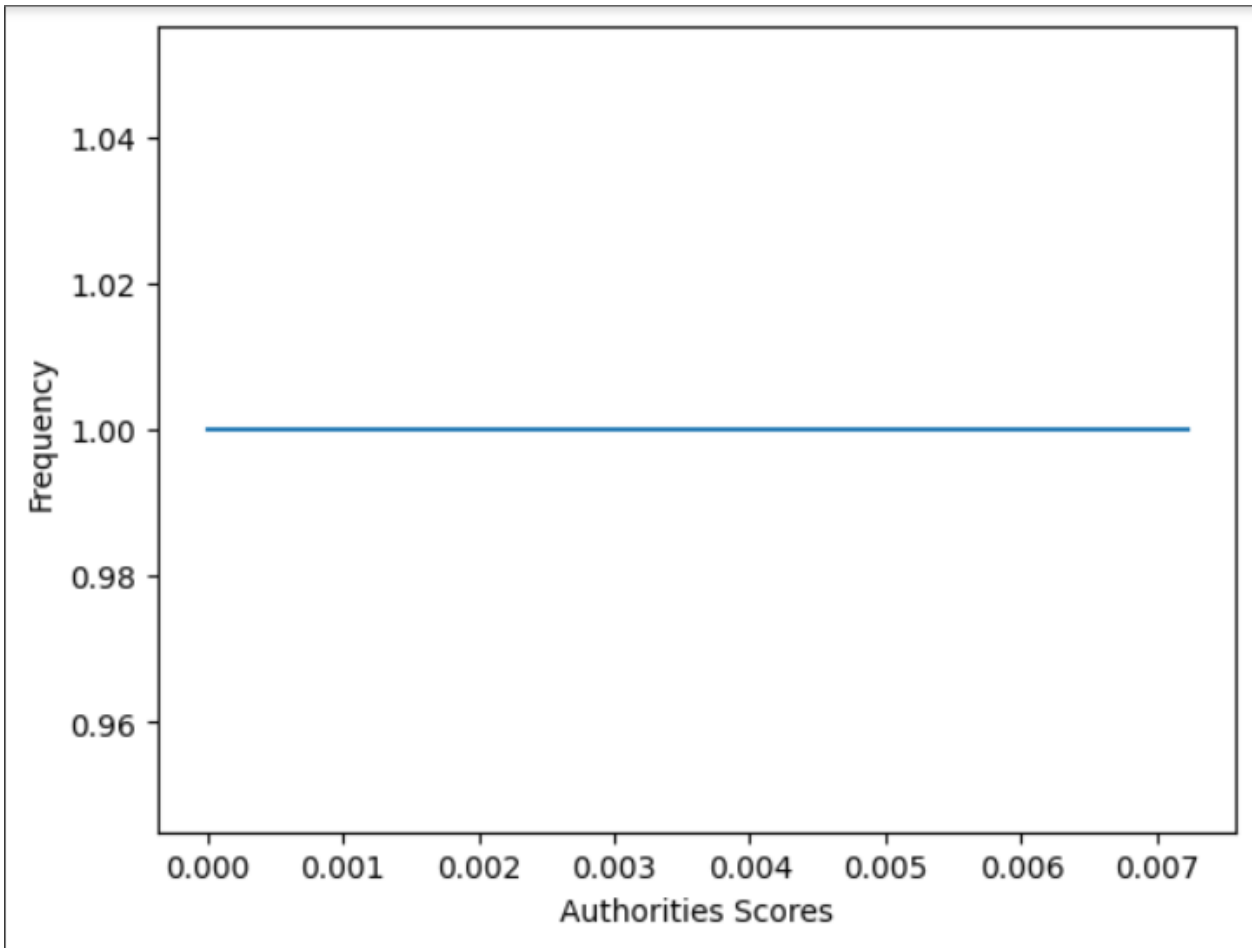
```
Top 12 NodeIDs based on Hub Score:
[(160, 0.010628802611038442),
 (82, 0.009616665861905403),
 (121, 0.009530349046577475),
 (107, 0.008788067113764076),
 (62, 0.008232597715453004),
 (249, 0.008017503203921339),
 (434, 0.007541252050552116),
 (183, 0.0072000203311382936),
 (86, 0.007003074161765551),
 (114, 0.006398316126418316),
 (105, 0.006319911627527695),
 (211, 0.006316843868434014)]
```

```
Top 12 NodeIDs based on Authority Score:
[(160, 0.007220481699191959),
 (107, 0.006898170199864654),
 (62, 0.006695883147202671),
 (434, 0.006485092543979912),
 (121, 0.006471582443168834),
 (183, 0.006040848953683616),
 (128, 0.005947949784933283),
 (249, 0.005729100052968214),
 (256, 0.005703873069050461),
 (129, 0.0056777282828217234),
 (283, 0.00562430245251807),
 (82, 0.00547776809636636)]
```

Below are the distribution plots of each of these scores


17.5
15.0
12.5
10.0
7.5
5.0
2.5
Frequency
0.000
0.002
0.004
0.006
0.008
PageRank Scores


140
120
100
80
60
40
20
0
Frequency
0.000
0.002
0.004
0.006
0.008
0.010
Hubs Scores


1.04
1.02
1.00
0.98
0.96
Frequency
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
Authorities Scores