IR Assignment-3
Report

Solution 1:

We have used the EMail-eu-core dataset.

email-Eu-core network

Dataset information

The network was generated using email data from a large European research institution. We have anonymized information about all incoming and outgoing email between members of the research institution. There is an edge (u, v) in the network if person u sent person v at least one email. The e-mails only represent communication between institution members (the core), and the dataset does not contain incoming messages from or outgoing messages to the rest of the world.

The dataset also contains "ground-truth" community memberships of the nodes. Each individual belongs to exactly one of 42 departments at the research institute.

This network represents the "core" of the email-EuAll network, which also contains links between members of the institution and people outside of the institution (although the node IDs are not the same).

Dataset statistics	
Nodes	1005
Edges	25571
Nodes in largest WCC	986 (0.981)
Edges in largest WCC	25552 (0.999)
Nodes in largest SCC	803 (0.799)
Edges in largest SCC	24729 (0.967)
Average clustering coefficient	0.3994
Number of triangles	105461
Fraction of closed triangles	0.1085
Diameter (longest shortest path)	7
90-percentile effective diameter	2.9

We have make the edge list by the function called

```
def fillEdgeList(myEdgoListo):
  for idx in range(datta_Uk.shape[0]):
    edge_Items = []
    edge_Items.append(datta_Uk.loc[idx][0])
    edge_Items.append(datta_Uk.loc[idx][1])
    myEdgoListo.append(edge_Items)
  return myEdgoListo
```

We have made the adjacency matrix by using the fact that if there is an edge between the source and destination, there should be one at build2dMat[source][destination] in the directed Graph.

1. We have calculated the number of nodes by maintaining the set of source and destination for removing the duplicates, it comes out to be 1005.

2. Similarly the number of edges can be calculated by the length of the edge list.

Number of edges = 25570

3. Now, For calculating the Avg. Indegree,We have made an indegree = [] list which maintains the indegree of each node.

How we have calculated the indegree, we go for the particular column in the adjacency matrix, and traverse on all the rows for that particular column, and if we get a one we keep increasing the indegree value.
Similarly for the out degree, we keep a particular row, and keep traversing on all the columns and we track if matrix[row][col] == 1, if yes, we keep on adding the outdegree for that particular node.

Average indegree and outdegree can be calculated by the formula, sum of all the indegrees/ number of nodes.
Avg Indegree =25.442786069651742

4. Avg outdegree = sum of all the outdegrees/number of nodes.

We have made two lists which keep track of all the indegree and outdegree of the nodes.
Avg Outdegree = 25.442786069651742

5&6. So we have sorted those nodes in decreasing order. That's we got the maximum indegree and maximum outdegree in the Graph.
Once we get the max indegree or max outdegree, we can calculate the respective node,just have to print the index of that highest degree.
Node with the highest indegree is the 160th node with 212 indegrees.
Node with the highest outdegree is 160th node with 334 outdegree.

7. For calculating the density of the network, We have used the formula ,
Density = number of edges in graph/ Max edges in directed graph

Max edges in directed graph can be, if number of nodes is K,
Then the max edges can be K*(K-1) , this is the approach we have used.

The density is coming out to be 0.025341420388099344

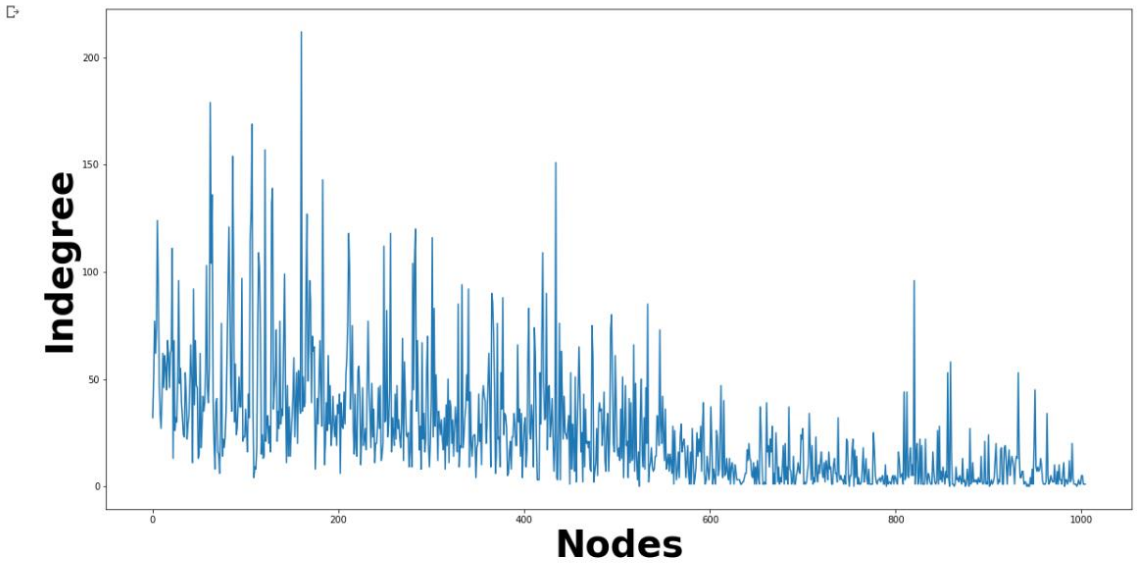
Degree Distribution of the Graph,
We have used the matplotlib,
And we have plotted the nodes on the X-axis and the indegree and outdegree values on the Y-axis.

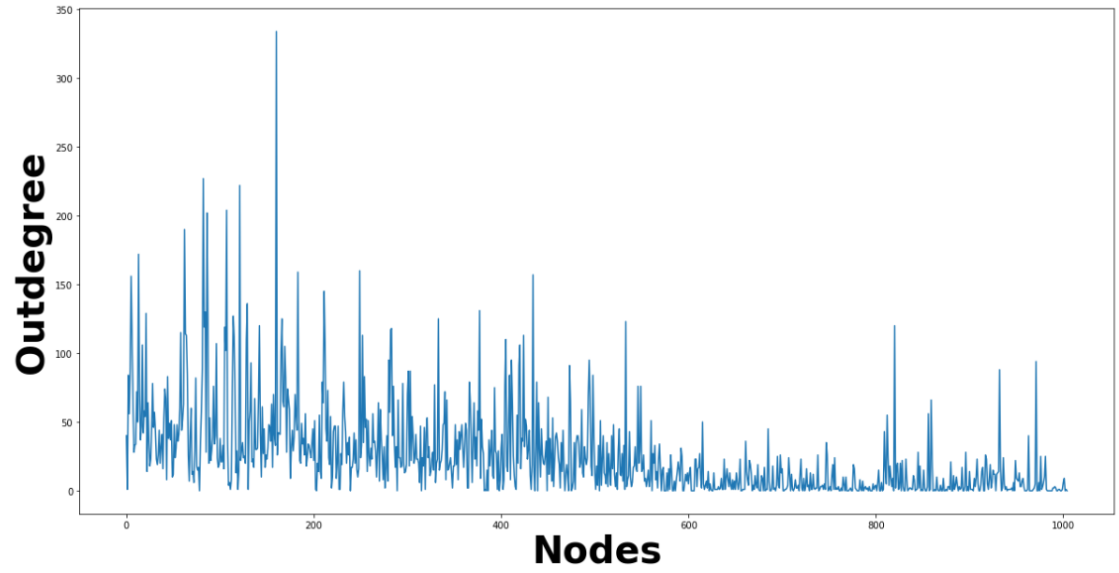
8.Local Clustering Coefficient, We have used the formula to make the LCC = [] list , the formula is given by:
Local clustering coefficient = nv/ kv * (kv-1)
Where nv is the edges between the neighbors of the node, and kv is the degree of the node.

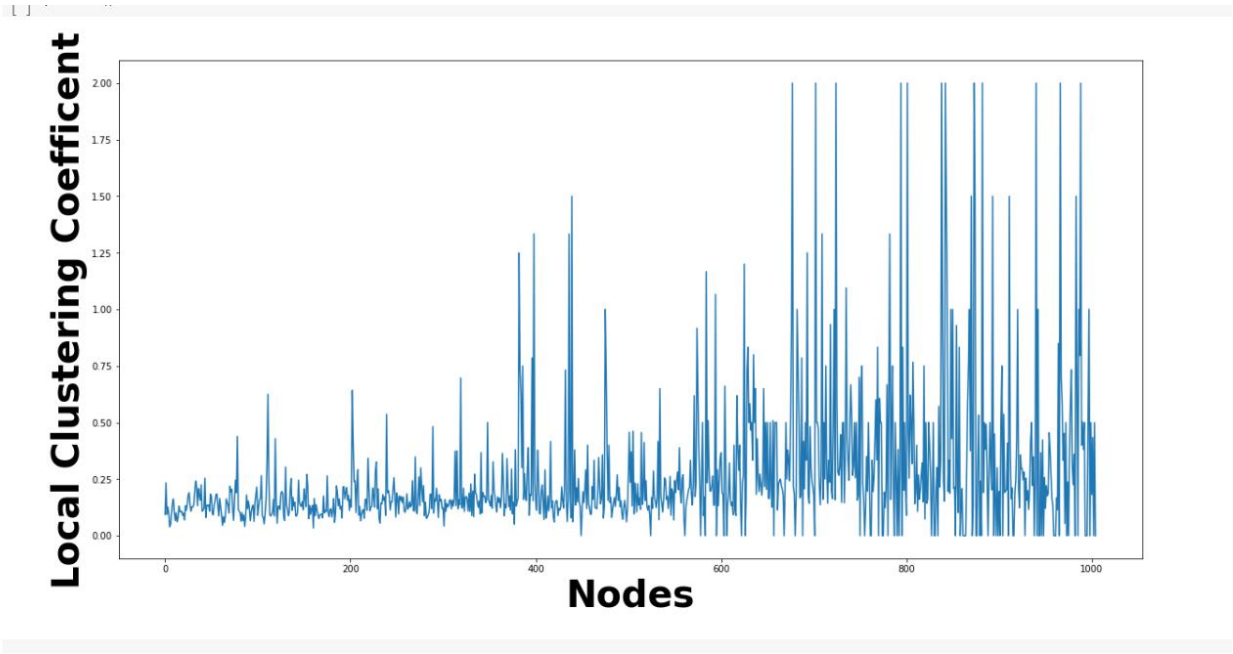
```
# to calculate the degree and neighbors of a node
def make_k_neigh(num,datta):
  neigh = []
  k=0
  for col in range(len(datta)):
      if(datta[num][col] == 1):
        neigh.append(col)
        k+=1
      if(datta[col][num] == 1):
        if (datta[num][col] != 1):
          neigh.append(col)
        k+=1
  return k,neigh

def fillLCC():
  fill = []
  for idx in range(len(nv_Val)):
    if (nv_Val[idx] ==0 or k_Deg[idx] == 1):
      val = 0
    else :
      val = nv_Val[idx] / ((k_Deg[idx] )* (k_Deg[idx]-1))
    fill.append(val)
  return fill
```

This is how we have made the list of local clustering coefficient (LCC)
Then by using the matplotlib, we have plotted the graph nodes(x-axis) vs LCC (y-axis).

Result of the distribution,


Indegree
Nodes
0
50
100
150
200
0
200
400
600
800
1000


Outdegree
Nodes
0
50
100
150
200
250
300
350
0
200
400
600
800
1000



Solution 2:

Part 1):

In this question we have to arrange or rank the nodes according to pagerank , hub scores and authority scores.
And I have implemented the algorithm from scratch
The algorithm for pagerank is as follows;

1) Make a n*n matrix where rows and columns represents number of nodes in the graph. If node 1 is connected to node 2 then entry in cell [1][2] is 1 else it is 0.
2) Then here no of 1's in a row represent the outdegree of the node and number of 1's in the column are indegree of the node.
3) The pagerank for a node is given as follows:
   If A has outgoing edge to b,c,d then pagerank of A is pagerank of b divided by outdegree of b plus pagerank of c divided by outdegree of b plus pagerank of c divided by outdegree of c
4) In this way we have iterate over the pagerank list untill we get very less difference between the old list and the new list.
5) The initial value for a node is calculated as follows:
   1/sum of all the old values

   The final top 10 page rank scores are as follows:

```
[(160, 0.006707217439546308),
 (62, 0.005383786904024949),
 (1, 0.005360025102807274),
 (86, 0.005224613716840153),
 (107, 0.005196796366955661),
 (121, 0.0048568373361498034),
 (365, 0.00477086350580227),
 (5, 0.004496026309813474),
 (129, 0.004416863223996245),
 (183, 0.004384408351107048),
```

Part 2):

Now before calculating the hub score and authority score we have to do some initilizations.

First we assume that initial hub scores are all 1's then we take the transpose of the original matrix and multiply it with the hub matrix to get the first authority matrix.
After that we multiply the original 2D matrix by authority matrix to get the 1st hub matrix.

Then we have to iterate here again on the hub and the authority matrix to get the new values using the old values .

The first new value for a single node is calculated as follows:
Old_value/(square root of the sum of the squares of all the values)

The top 10 best hub scores are as follows:

```
[(160, 0.1899507854296232),
 (82, 0.16126707420938557),
 (121, 0.16024265595151996),
 (107, 0.14674134865554728),
 (62, 0.13924208166527452),
 (249, 0.1370487759080494),
 (434, 0.1290635669236609),
 (86, 0.12776334375021606),
 (183, 0.12492649318997276),
 (211, 0.11145145302881716),
```

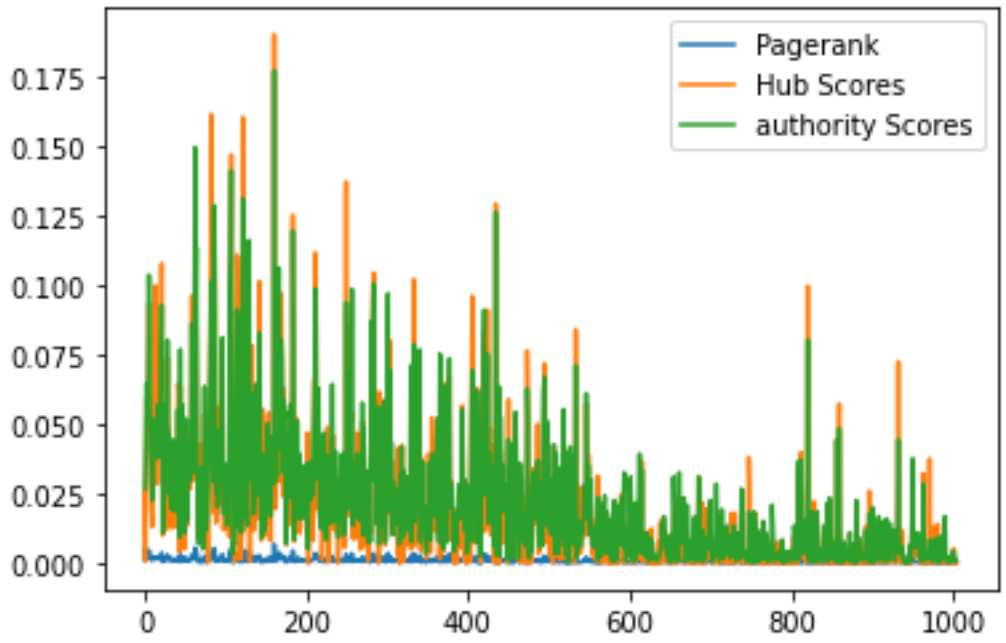
The top 10 authority scores are as follows:

```
[(160, 0.17691144063853392),
 (62, 0.14937333902970554),
 (107, 0.14102845975430298),
 (121, 0.13101460462381995),
 (86, 0.1285111408411992),
 (434, 0.1260076770585784),
 (183, 0.11933177363825637),
 (129, 0.11599382192809536),
 (64, 0.1134903581454746),
 (128, 0.11015240643531357),
```

Part 3) The analysis and comparision shows that

The node with highest pagerank score, hub score and authority score is 160. And there are other very important nodes are 62,107,121,86,434,183.

And when we plot the graph for pagerank ,hub score and authority score the plot is like that:



Here we can see that for many nodes hub scores are more than authority scores and authority scores are more than pageranks.