CSE508

Report - Information Retrieval Assignment 3

Indraayudh Talukdar - MT22031

Antara Das - MT22014

Arun Sen - MT22018

Dataset information

This is who-trusts-whom network of people who trade using Bitcoin on a platform called Bitcoin OTC. Members of Bitcoin OTC rate other members in a scale of -10 (total distrust) to +10 (total trust) in steps of 1. This is the first explicit weighted signed directed network available for research.

- SOURCE: node id of source, i.e., rater
- TARGET: node id of target, i.e., ratee
- RATING: the source's rating for the target, ranging from -10 to +10 in steps of 1
- TIME: the time of the rating, measured as seconds since Epoch.

	SOURCE	TARGET	RATING	TIME
0	6	2	4	1.289242e+09
1	6	5	2	1.289242e+09
2	1	15	1	1.289243e+09
3	4	3	7	1.289245e+09
4	13	16	8	1.289254e+09
...	...	...	...	...
35587	4499	1810	1	1.453612e+09
35588	2731	3901	5	1.453679e+09
35589	2731	4897	5	1.453679e+09
35590	13	1128	1	1.453680e+09
35591	1128	13	2	1.453684e+09

35592 rows × 4 columns

Link Analysis on Bitcoin OTC trust weighted signed network

Here density of the network is computed the following formula :

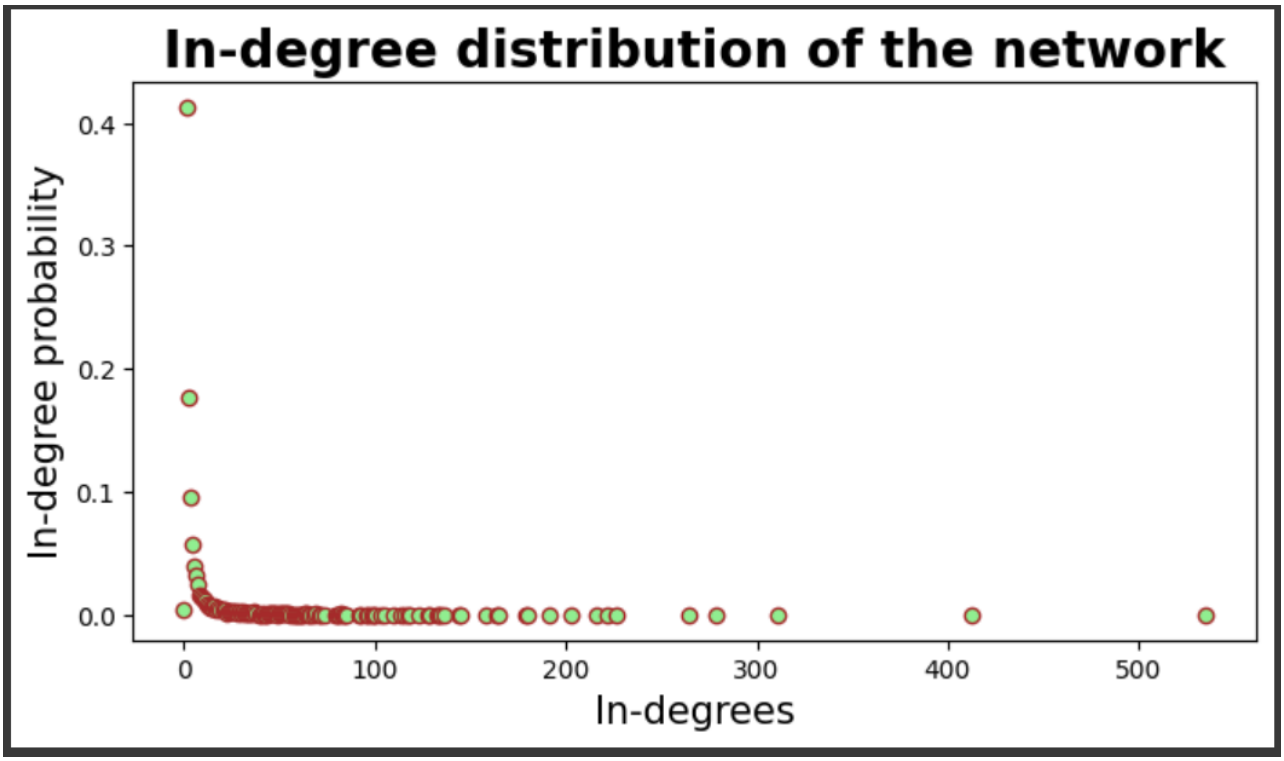
density(G) = E / (V * (V-1))

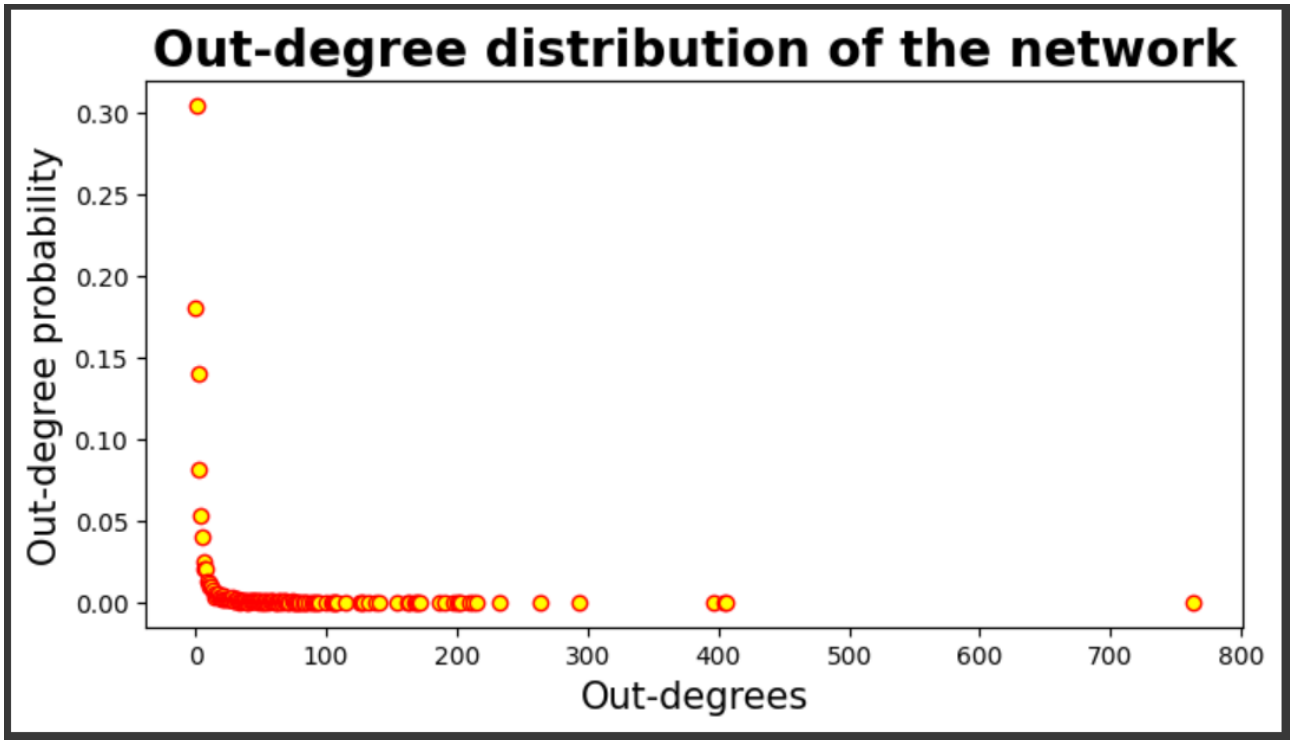
Where E = Total number of edges present in the network
And V = Total number of nodes present in the network.
So basically the denominator of density represents the total number of possible edges in the network.

```
[31] n = len(nodes)
     print("Total count of nodes in the network : ", n)
     print("Total count of edges in the network : ", len(data))

     Total count of nodes in the network :  5881
     Total count of edges in the network :  35592

[33] print("Maximum number annotated to a node : ",max(nodes))
     print("Maximum weight annotated to an edge : ",max(data['RATING']))
     print("Minimum weight annotated to an edge : ",min(data['RATING']))

     Maximum number annotated to a node :  6005
     Maximum weight annotated to an edge :  10
     Minimum weight annotated to an edge :  -10
```

```
] print("Maximum in-degree in this network : ", max(indegree_dict.values()))
  print("Maximum out-degree in this network : ", max(outdegree_dict.values()))

  Maximum in-degree in this network :  535
  Maximum out-degree in this network :  763

] d = len(data) / (n * (n-1))
  print("Density of the network is : ", d)

  Density of the network is :  0.0010292571373048454

] indegs = sorted(list(set(indegree_dict.values())))
  outdegs = sorted(list(set(outdegree_dict.values())))
  print("-------------------All possible in-degrees in the network ------------------------")
  print(indegs)
  print("-------------------All possible out-degrees in the network ------------------------")
  print(outdegs)

  -------------------All possible in-degrees in the network ------------------------
  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
  -------------------All possible out-degrees in the network ------------------------
  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
```

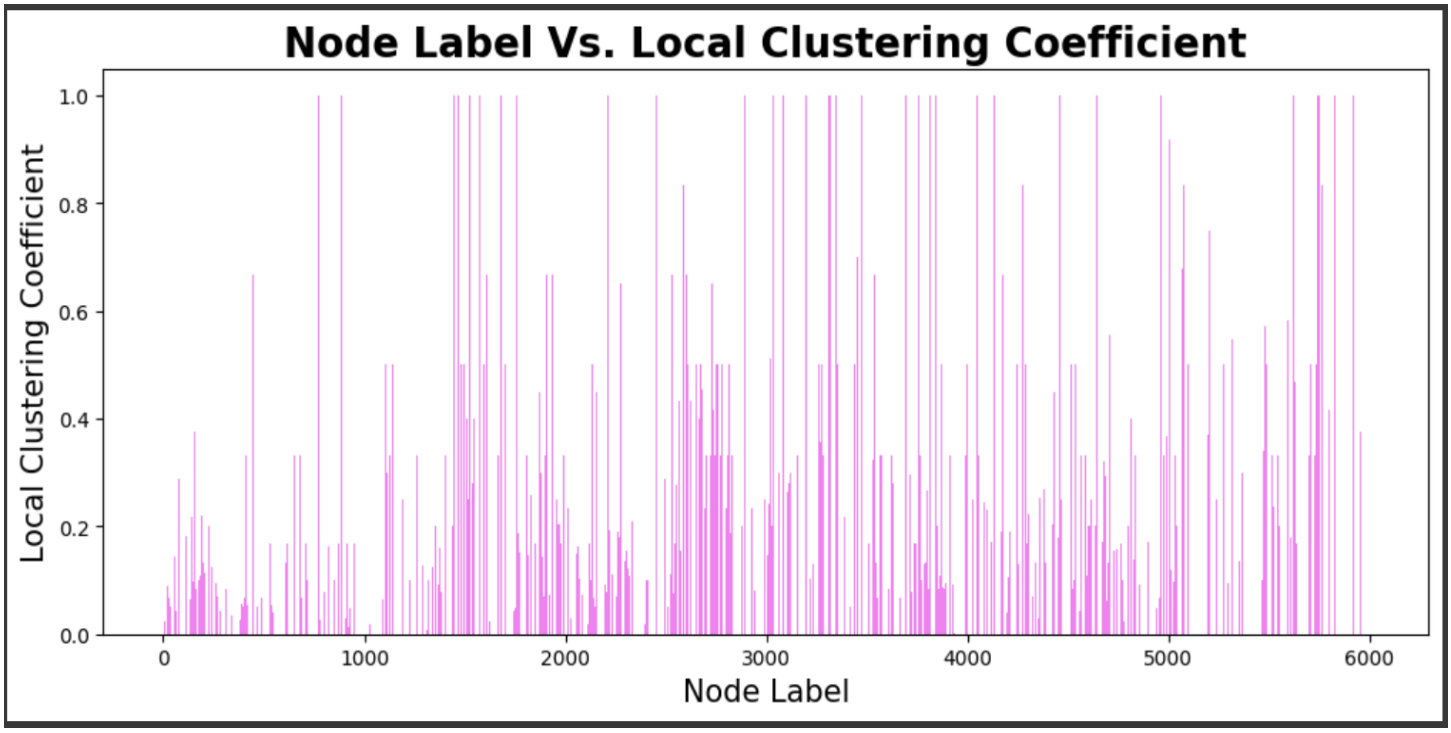
The degree distribution of a network represents the probability distribution of <k, P(k)> where k is a degree of a node in the network and P(k) is the fraction of nodes having degree k.
The followings are the degree distributions of the network and important stats about it :

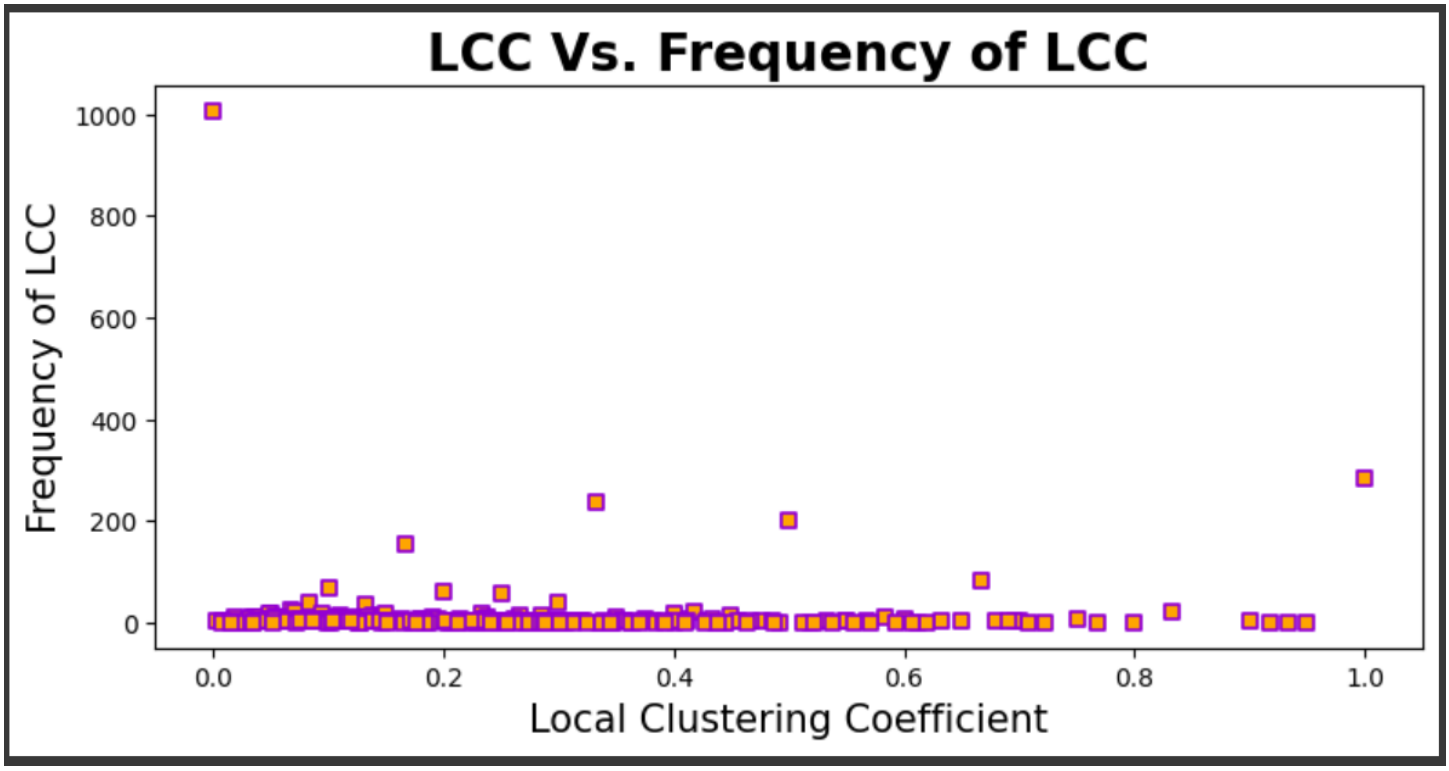




```
Maximum in-degree probability of a node in this network :  0.413

Average in-degree probability of a node in this network :  0.009

Mean in-degree probability of a node in this network :  0.0

Minimum in-degree probability of a node in this network :  0.0

Maximum out-degree probability of a node in this network :  0.305

Average out-degree probability of a node in this network :  0.008

Mean out-degree probability of a node in this network :  0.0

Minimum out-degree probability of a node in this network :  0.0
```

The local Clustering coefficient of a node is defined by the number of edges present in the network among the neighbours of a node divided by the total number of nodes possible among those neighbour nodes.
It is implemented as follows: We can get the number of edges present among the neighbour nodes by a set intersection operation between all edges present in the network and all possible edges among the neighbours.






```
Maximum LCC in this network :  1.0

Average LCC in this network :  0.149

Mean LCC in this network :  0.456

Minimum LCC in this network :  0.0
```

PageRank, Hubs and Authority scores of nodes of this network

Methodology:
The networkx library is used to find the PageRank values, hub scores and authority scores.

The scores obtained indicate the following:
The node with the highest page rank score will have quality nodes as its neighbours and a high in-degree.
The node having the highest hub score will have a high out-degree.
The node having the highest authority score will have a high in-degree.

```
    print("--------page rank values--------")
    print(page_rank_values)
```

Python

```
--------page rank values--------
{1: 0.005029048679852529, 2: 0.0009774710321327727, 3: 0.0003827789520076624, 4: 0.001289835811003076, 5: 9.29861627294045e-05, 6: 0.000774108591722850
```

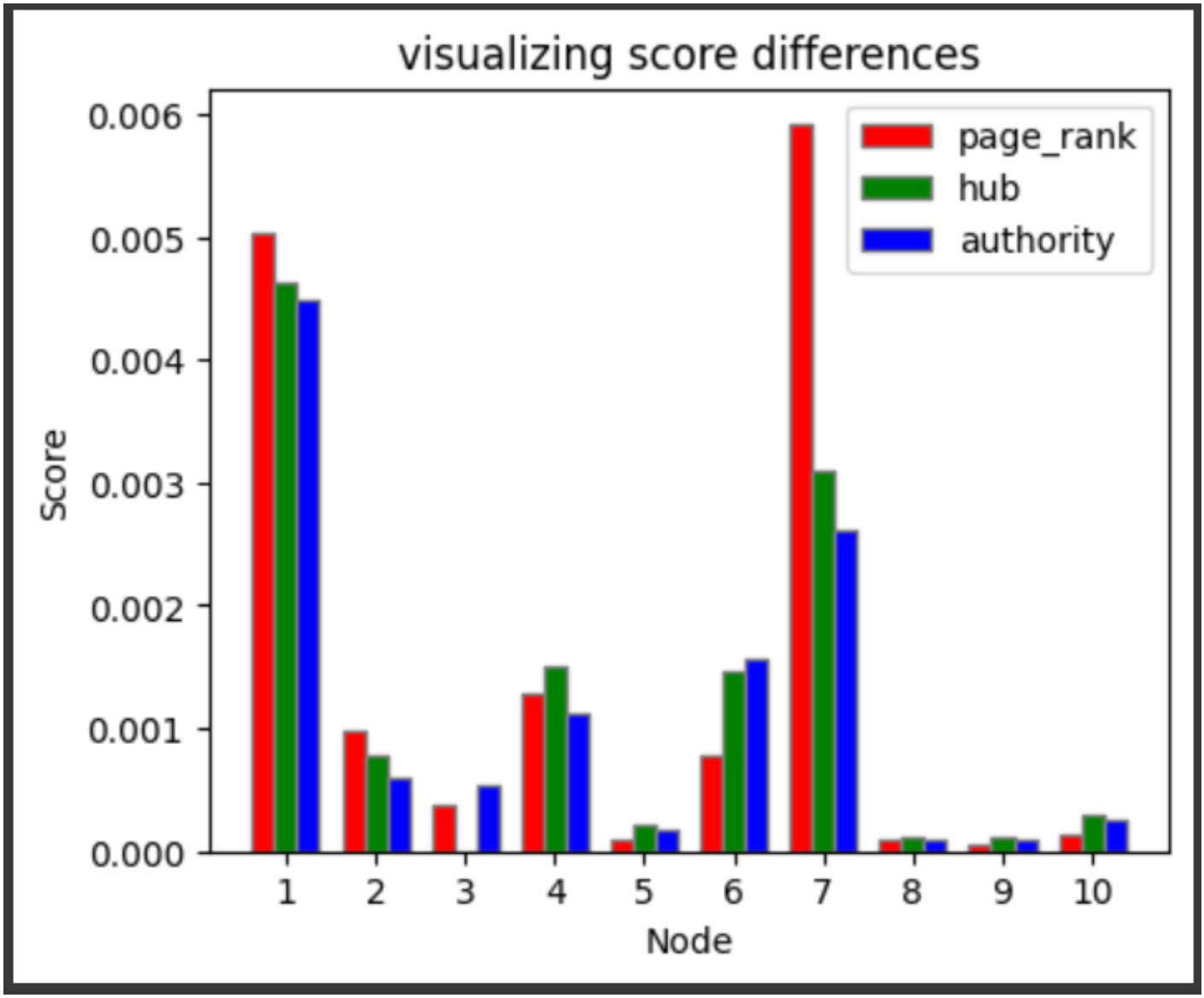
```
    hubs,authority=nx.hits(G)#getting hubs and authority score for each node
    print("--------hub values--------")
    print(hubs)
    print("--------authority values--------")
    print(authority)
```

Python

```
--------hub values--------
{1: 0.004636831266948586, 2: 0.0007758275426393979, 3: -0.0, 4: 0.0015073564036747502, 5: 0.0002087994818831646, 6: 0.0014629173309573145, 7: 0.0030944
--------authority values--------
{1: 0.004496189948700381, 2: 0.0005890168397930885, 3: 0.0005475613411447138, 4: 0.0011197026235601776, 5: 0.0001697030181176477 3, 6: 0.001571882128227
```

	nodes	page_rank_score	hub_score	authority_score
0	1	0.005029	4.636831e-03	4.496190e-03
1	2	0.000977	7.758275e-04	5.890168e-04
2	3	0.000383	-0.000000e+00	5.475613e-04
3	4	0.001290	1.507356e-03	1.119703e-03
4	5	0.000093	2.087995e-04	1.697030e-04
...	...	...	...	...
5876	6000	0.000035	6.646890e-22	-1.381860e-20
5877	6002	0.000065	-0.000000e+00	2.763719e-20
5878	6003	0.000047	-0.000000e+00	2.131752e-06
5879	6004	0.000052	-0.000000e+00	1.130527e-04
5880	6005	0.000052	-0.000000e+00	1.130527e-04

[5881 rows x 4 columns]



Contributions :

Part 1 - Antara Das

Part 2 - Indraayudh Talukdar

[Github repository]