Network Analysis - Report

Olga Pagnotta – DHDK, Martina Pensalfini – DHDK, Marta Soricetti – DHDK
The python code is available at https://github.com/NetworkAnalysisP/NAP.git.

Context

We have identified two subfields. The first is the general context, which deals with Computer Science for the automatic extraction of data; Statistics for the analysis and the measures used in the study and Anthropology for the offline social network/ Studies of human relationships. The second one is the specific context, dealing with Online Social Networks (e.g., Facebook) and Offline Social Network (face-to-face behavioral network).

Specific Application:

We have analyzed three studies:

1. *Organization Mining Using Online Social Networks,* for gaining insights on an organization's structure by clustering its social network and gathering publicly available information on the employees within each cluster.[1]
2. *Learning to Discover Social Circles in Ego Networks*, analysing the data through node clustering algorithms applied on a user's ego network, a network of connections between the given node's friends. [2]
3. *What's in a crowd? Analysis of face-to-face behavioural networks* aiming at mining behavioural networks of face-to-face interactions between individuals, in two widely different events, the INFECTIOUS exhibition and the ACM Hypertext conference.[3]

Problem and Motivation

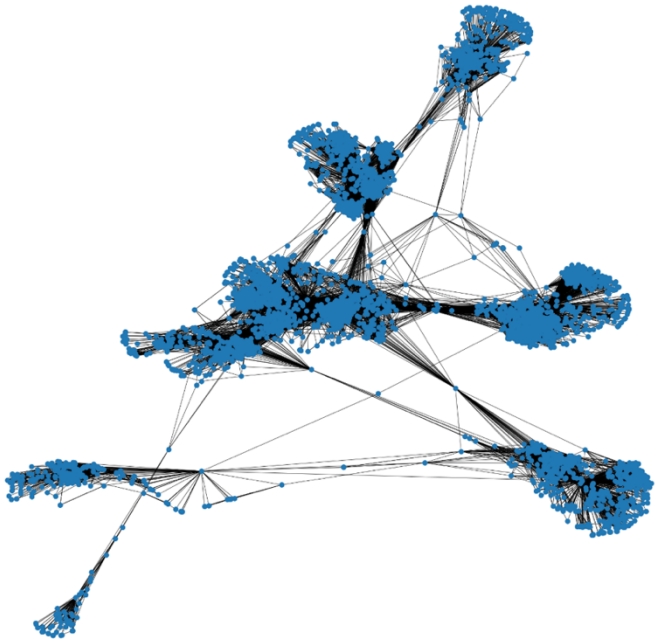
Our goal is to prove that the analysis of Online Social Networks and the measures applied to them are similar (or even equivalent) to those of Offline Social Networks. For this reason, we decided to cross-analyse two Online Social Networks by applying the measures used in the paper *Organization Mining Using Online Social Networks* to the dataset used in the paper *Learning to Discover Social Circles in Ego Networks*, and vice versa. By analysing the results of our application, we have chosen the most effective measures and applied them to the Offline Social Network dataset used in *What's in a crowd? Analysis of face-to-face behavioural networks*.

[1] https://arxiv.org/abs/1303.3741
[2] http://snap.stanford.edu/data/ego-Facebook.html
[3] https://arxiv.org/abs/1006.1260

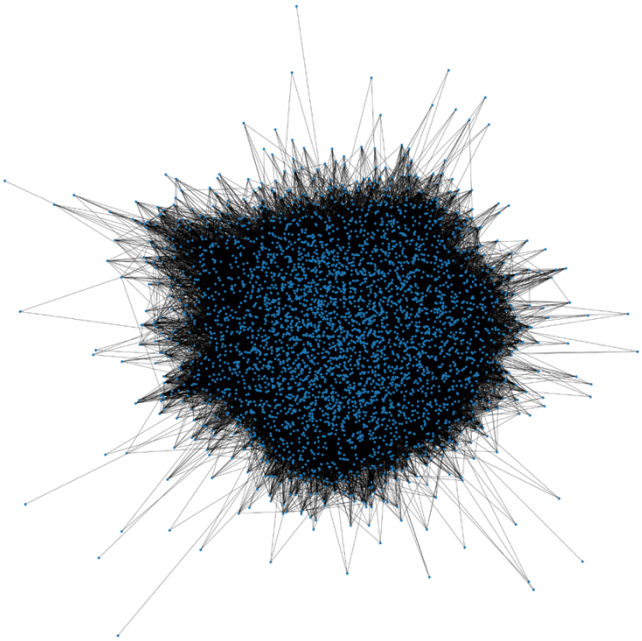
Datasets

Data used in our investigation were taken from public sources, in particular: Stanford Large Network Dataset Collection[4] (*Learning to Discover Social Circles in Ego Networks)*, and Colorado Index for complex Networks[5] (*Organization Mining Using Online Social Networks, What's in a crowd? Analysis of face-to-face behavioural networks)*. For performing the measures, we have used the python libraries NetworkX and Matplotlib.

Learning to Discover Social Circles in Ego Networks[6] consists of 'circles' (or 'friends lists') from Facebook. Data - anonymized by replacing the Facebook-internal ids for each user with a new value - was collected from surveys, and include circles and ego undirected networks.



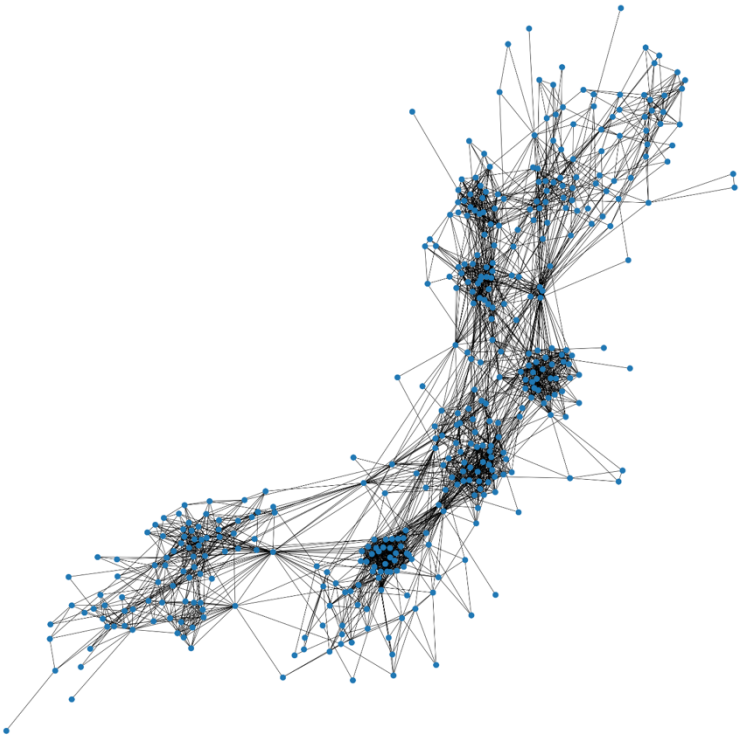
Dataset statistics	
Nodes	4039
Edges	88234
Nodes in largest WCC	4039 (1.000)
Edges in largest WCC	88234 (1.000)
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7



Organization Mining Using Online Social Networks[7] is about six well-known high-tech companies' networks found by collecting information about the company's structure as exposed by the company's employees on Facebook. We have analyzed only the M2 network, representing the structure

[4] http://snap.stanford.edu/data/index.html
[5] https://icon.colorado.edu/#!/networks
[6] http://i.stanford.edu/~julian/pdfs/nips2012.pdf
[7] https://arxiv.org/pdf/1303.3741.pdf

of an international software provider serving a global customer base. Its size ranges between 10,000 to 20,000 employees, and among these 3,862 stated that they work for M2 in their Facebook profiles. These are the nodes of our network with 87,324 undirected links.



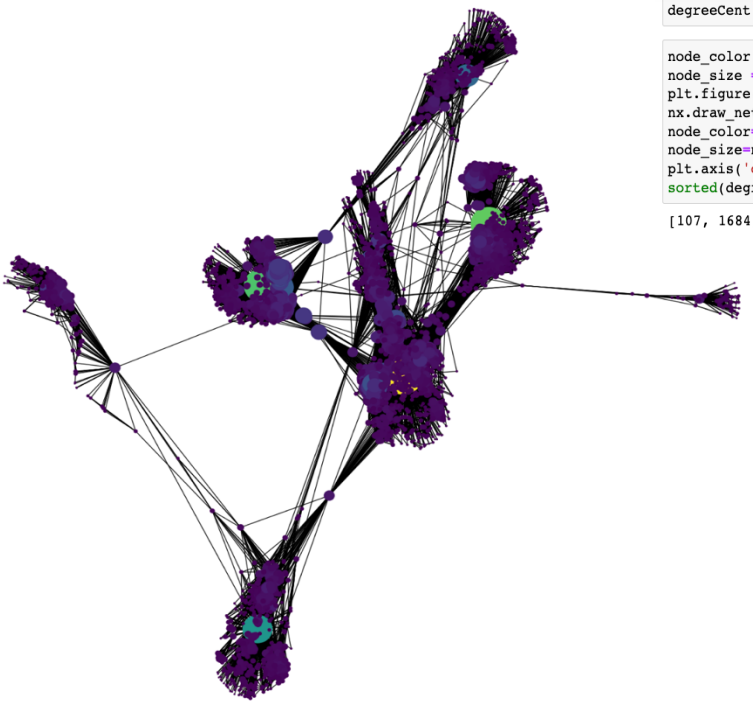
What's in a crowd? Analysis of face-to-face behavioural networks' data are collected from two different events using active Radio-Frequency Identification Devices (RFID) to mine face-to-face proximity relations of participants. We have analysed only the INFECTIOUS: STAY AWAY network in which nodes represent exhibition visitors; undirected edges represent face-to-face contacts that were active for at least 20 seconds during the day with the most interactions.

Validity and Reliability

For what concern reliability, given the fact that we are dealing with open-source datasets, thus publicly available, asking ourselves the question "given the same conditions if we repeated our study would we obtain the same results?", the answer is positive. Talking then about validity, we are taking into consideration real events and Facebook user profiles, thus asking ourselves the question "Are we measuring what we intend to measure?", the answer is positive stating that our model closely represent reality.

Measures

We have applied the following centrality measures taken from the paper *Organization Mining Using Online Social Networks* for identifying the central nodes in the *Learning to Discover Social Circles in Ego Networks* network:



```
degreeCent = nx.degree_centrality(G1)
```

```
node_color = [20000.0 * G1.degree(v) for v in G1]
node_size = [v * 10000 for v in degreeCent.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G1, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(degreeCent, key=degreeCent.get, reverse=True)[:5]
```

```
[107, 1684, 1912, 3437, 0]
```

- Degree centrality (DG)[8], the simplest centrality measure. The degree of a node in a network is the number of edges attached to it. From the analysis, we can assume that the nodes with the highest degree play a major role in the network.

- Closeness centrality (CL)[9], which uses the shortest paths in networks, measuring the mean distance from a node to other nodes. It is calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes. The analysis shows that the nodes with the higher values can spread information more efficiently through the graph, even if the difference between the nodes is not so high.

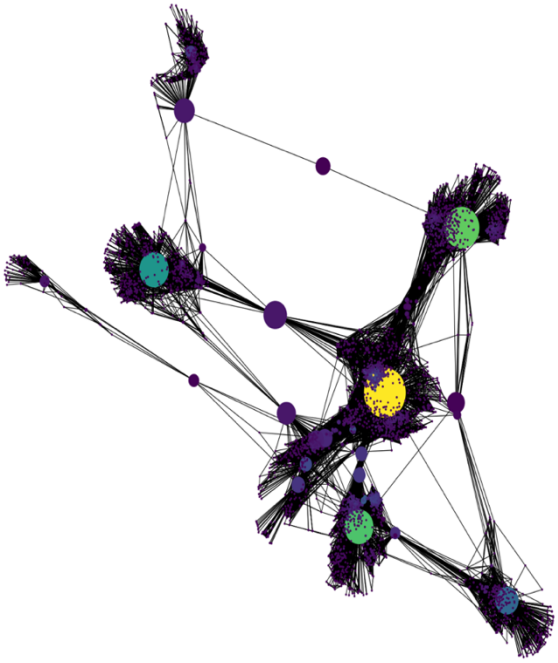
```
closenessCent = nx.closeness_centrality(G1, u=None, distance=None, wf_improved=True)
```

```
node_color = [20000.0 * G1.degree(v) for v in G1]
node_size = [v * 1000 for v in closenessCent.values()]
plt.figure(figsize=(40,40))
nx.draw_networkx(G1, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(closenessCent, key=closenessCent.get, reverse=True)[:5]
```

```
[107, 58, 428, 563, 1684]
```

[8] Degree centrality, NetworkX documentation (https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.degree_centrality.html#networkx.algorithms.centrality.degree_centrality)

[9] Closeness centrality, NetworkX documentation (https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.closeness_centrality.html#networkx.algorithms.centrality.closeness_centrality)



• Betweenness centrality (BC)[10] measures the extent to which a node lies on paths between other nodes: paths lying on "trafficked" shortest paths have a more central role in the network. We can see from the graph 8 clusters built around 8 important nodes – they have the highest betweenness centrality values and thus the highest number of short paths between each other.

```
betCent = nx.betweenness_centrality(G1, normalized=True, endpoints=True)
```

```
node_color = [20000.0 * G1.degree(v) for v in G1]
node_size = [v * 10000 for v in betCent.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G1, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(betCent, key=betCent.get, reverse=True)[:5]
```

```
[107, 1684, 3437, 1912, 1085]
```

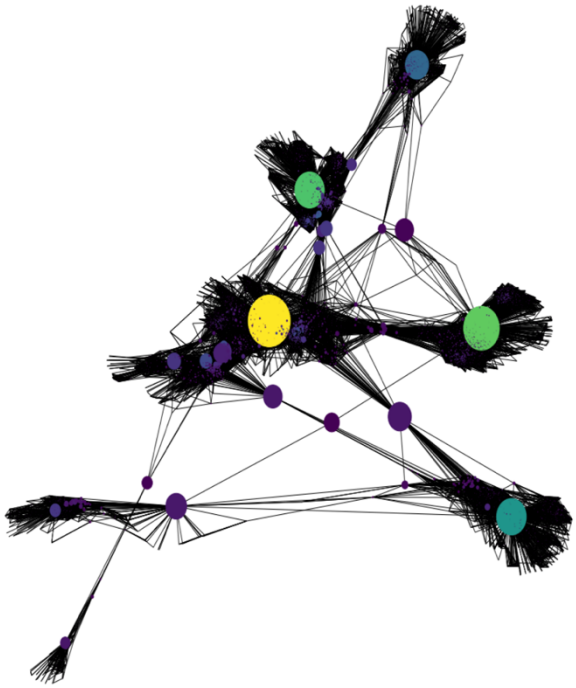
- Eigenvector centrality (EC)[11] deals with a node's importance, which sometimes is increased by having connections to other nodes that are themselves important. Eigenvector centrality is an extension of degree centrality but instead of just awarding one point for every network neighbour a node has, eigenvector centrality awards several points proportional to the centrality scores of the neighbours. Our graph shows that the most important nodes are overlapping, stating that they give importance one to the other.

```
EigCent= eigenvector_centrality(G1, max_iter=100, tol=1e-06, nstart=None, weight=None)
```

```
node_color = [20000.0 * G1.degree(v) for v in G1]
node_size = [v * 100000 for v in EigCent.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G1, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(EigCent, key=EigCent.get, reverse=True)[:5]
```

```
[1912, 2266, 2206, 2233, 2464]
```

[10] Betweenness Centrality, NetworkX documentation (https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.betweenness_centrality.html#networkx.algorithms.centrality.betweenness_centrality)
[11] Eigenvector Centrality, NetworkX documentation (https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.eigenvector_centrality.html#networkx.algorithms.centrality.eigenvector_centrality)



- Load centrality (LC)[12], a variant of BC, is the fraction of all shortest paths that pass through that node. In our graph, we identify 5 nodes with the highest values of LC, which are not condensed into one spot but distributed all over the network.

```
LoadCentr=load_centrality(G1, v=None, cutoff=None, normalized=True, weight=None)
```

```
node_color = [20000.0 * G1.degree(v) for v in G1]
node_size = [v * 10000 for v in LoadCentr.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G1, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(LoadCentr, key=LoadCentr.get, reverse=True)[:5]
```

```
[107, 1684, 3437, 1912, 0]
```

We have then applied the following community detection algorithm – taken from the paper *Learning to Discover Social Circles in Ego Networks,* for analysing the non-overlapping cluster structure of the *Organization Mining Using Online Social Networks* network:

Clique percolation[13] for finding overlapping communities. It is based on the concept that the internal edges of the community are likely to form cliques while intercommunity edges are unlikely to form cliques, which are complete subgraphs, with all their vertices interconnected.

A k-clique is thus a clique with k vertices. In the graph we have underlined the major clique in our network.

```
from networkx.algorithms.community import k_clique_communities
```

```
c = list(k_clique_communities(G, 20))
len(c)
```

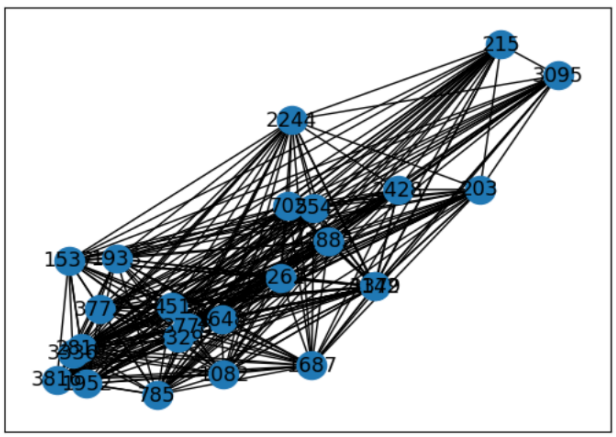
```
1
```

```
c[0]
```

```
frozenset({203,
           215,
           451,
           554,
           702,
           785,
           1082,
           1379,
           1537,
           1687,
           1885,
           1933,
           1952,
           2244,
           2428,
           2648,
           2816,
           3095,
           3142,
           3262,
           3329,
           3336,
           3772,
           3778,
           3816})
```

```
pos = nx.spring_layout(G)
```

```
res= list(c[0])
k = G.subgraph(res)
from matplotlib import pylab as pl
pl.figure()
nx.draw_networkx(k, pos=pos)
```



[12] Load cenrality, NetworkX documentation (https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.load_centrality.html#networkx.algorithms.centrality.load_centrality)

[13] K clique communities, NetworkX documentation (https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.kclique.k_clique_communities.html)

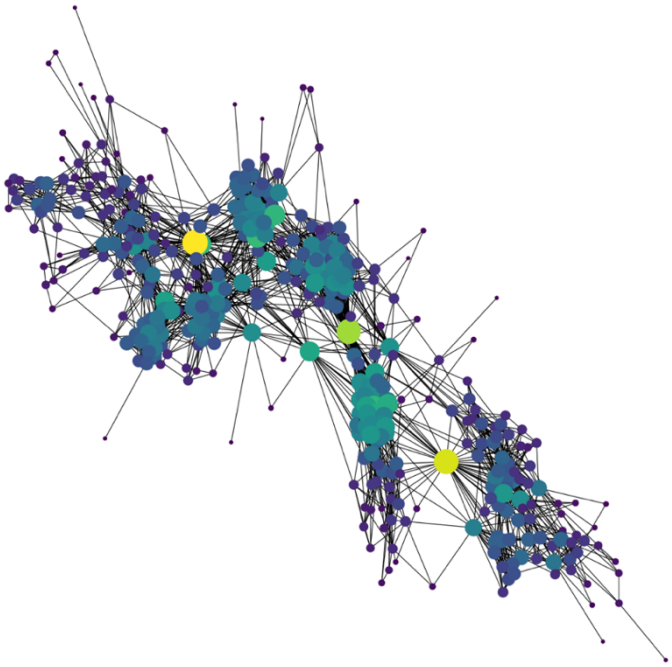
We have decided not to perform four measures which at the beginning seemed suitable for our investigation, but after some testing we discovered that: HITS and PageRank were giving the same results, given the fact that they are suitable for directed graphs, while our datasets are undirected; Communicability centrality (CC), was not giving significant results for our investigation; K-means, because our datasets were lacking of features, fundamental for this kind of analysis.
We have then computed the measures applied to both the datasets above for the Offline Social Network, adding one community detection measure, greedy modularity communities – which maximizes modularity, being the degree to which a system's components may be separated and recombined, often with the benefit of flexibility and variety in use.

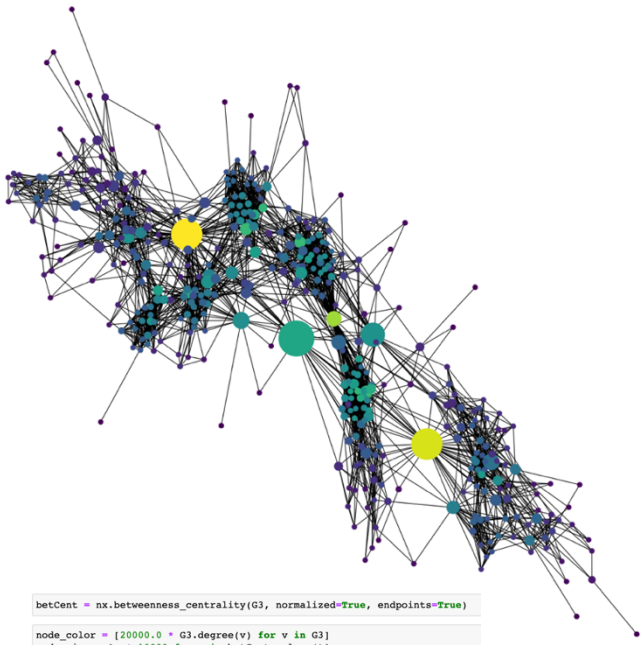
- Degree centrality (DG)– showing that a few nodes play a central role.

```
degreeCent = nx.degree_centrality(G3)
```

```
node_color = [20000.0 * G3.degree(v) for v in G3]
node_size = [v * 10000 for v in degreeCent.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G3, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(degreeCent, key=degreeCent.get, reverse=True)[:5]
```

```
[51, 272, 235, 195, 161]
```





- Betweenness centrality (BC) - We can see from the graph 5 clusters built around 5 important nodes – they have the highest betweenness centrality values and thus the highest number of short paths between each other.

```
betCent = nx.betweenness_centrality(G3, normalized=True, endpoints=True)
```

```
node_color = [20000.0 * G3.degree(v) for v in G3]
node_size = [v * 10000 for v in betCent.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G3, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(betCent, key=betCent.get, reverse=True)[:5]
```

```
[188, 272, 51, 230, 50]
```

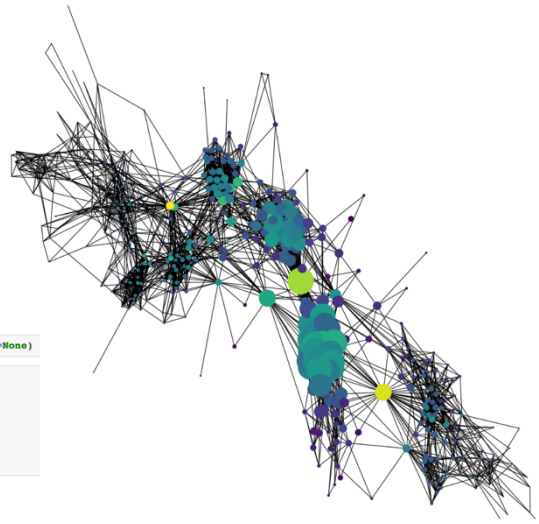
- Closeness centrality (CL)
  Also in this case the values are not so high, meaning that they are relatively all close to each other.

```
closenessCent = nx.closeness_centrality(G3, u=None, distance=None, wf_improved=True)
```

```
node_color = [20000.0 * G3.degree(v) for v in G3]
node_size = [v * 2000 for v in closenessCent.values()]
plt.figure(figsize=(40,40))
nx.draw_networkx(G3, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(closenessCent, key=closenessCent.get, reverse=True)[:5]
```

```
[188, 51, 230, 50, 117]
```

- Eigenvector centrality (EC)
  Our graph shows that the most important nodes are overlapping, stating that they give importance one to the other.

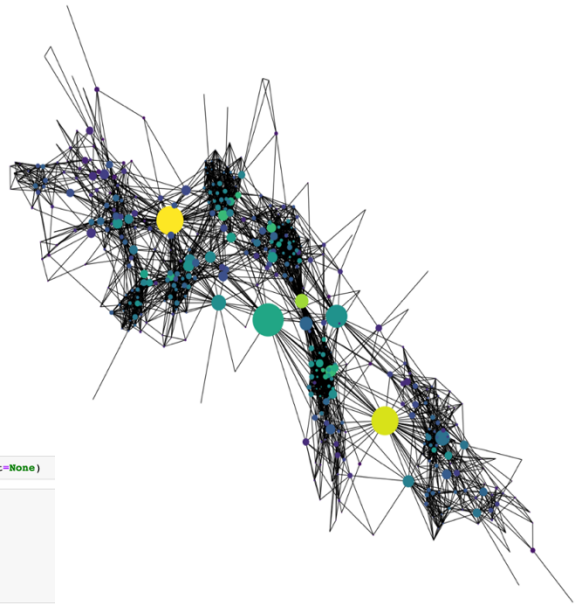


```
EigCent= eigenvector_centrality(G3, max_iter=100, tol=1e-06, nstart=None, weight=None)
```

```
node_color = [20000.0 * G3.degree(v) for v in G3]
node_size = [v * 10000 for v in EigCent.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G3, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(EigCent, key=EigCent.get, reverse=True)[:5]
```

```
[292, 299, 288, 294, 265]
```

- Load Centrality (LC)
  In our graph, we identify 5 nodes with the highest values of LC, which are not condensed into one spot but distributed all over the network

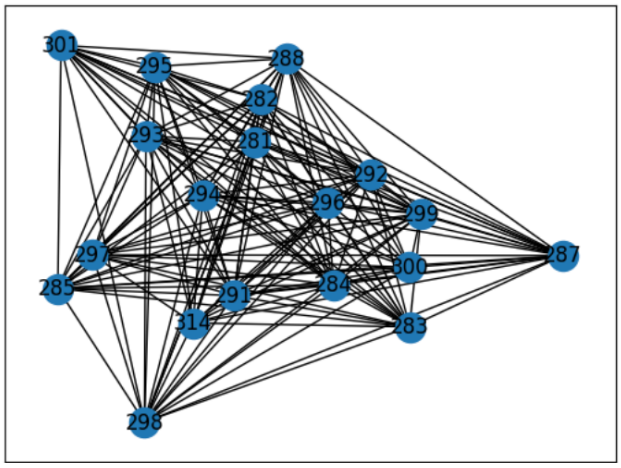


```
LoadCentr=load_centrality(G3, v=None, cutoff=None, normalized=True, weight=None)
```

```
node_color = [20000.0 * G3.degree(v) for v in G3]
node_size = [v * 10000 for v in LoadCentr.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G3, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(LoadCentr, key=LoadCentr.get, reverse=True)[:5]
```

```
[188, 272, 51, 230, 50]
```

- Clique percolation
  In the graph we have underlined the major clique in our network.




```
from networkx.algorithms.community import k_clique_communities
```

```
c = list(k_clique_communities(G3, 13))
```

```
len(c) #it means that in our graph there is just one 13-clique community
```

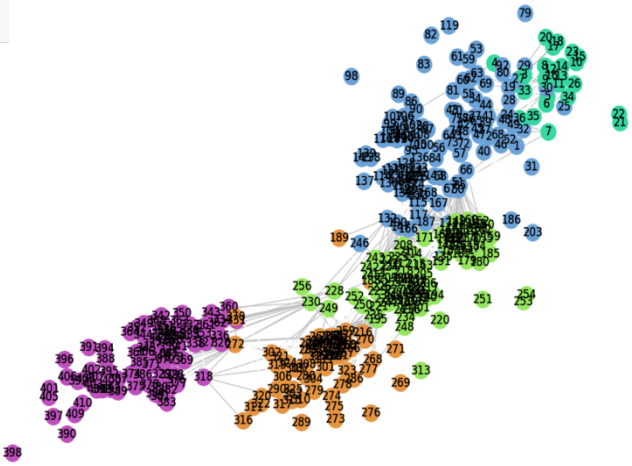
```
1
```

```
sorted(list(c[0]))
```

```
[281,
 282,
 283,
 284,
 285,
 287,
 288,
 291,
 292,
 293,
 294,
 295,
 296,
 297,
 298,
 299,
 300,
 301,
 314]
```

```
res= list(c[0])
k = G3.subgraph(res)
from matplotlib import pylab as pl
pl.figure()
nx.draw_networkx(k, pos=pos)
```

- Greedy Modularity Communities[14]
  As showcased by the graph, through the employment of the greedy modularity algorithm, five communities were found in the dataset explored.



```
import networkx.algorithms.community as nxcom
from networkx.algorithms.community import greedy_modularity_communities
communities = sorted(nxcom.greedy_modularity_communities(G3), key=len, reverse=True)
```

```
len(communities)
```

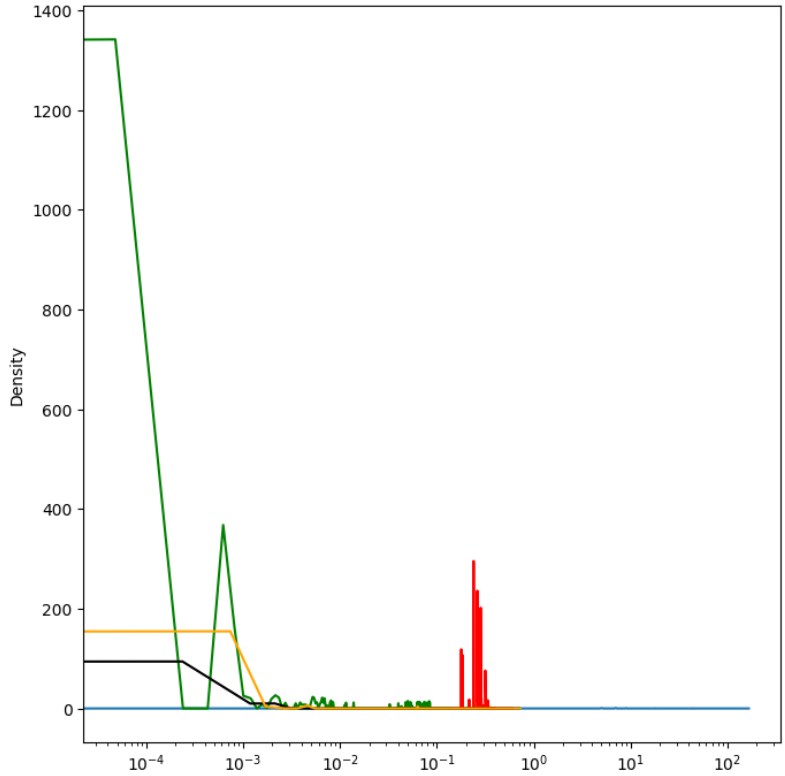
```
5
```

Results

After our investigation we have obtained significant results. The centrality measures taken from the *Organization Mining Using Online Social Networks* paper applied to the *Learning to Discover Social Circles in Ego Networks* data resulted in some interesting graphs: the central nodes of the network, the most important ones, divide the network in clusters and, even if slightly different, the nodes that result are repeated throughout the experiment. For what concerns the community detection measure taken from the

[14] Greedy modularity communities, NetworkX documentation (https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html)

Learning to Discover Social Circles in Ego Networks paper and applied to the *Organization Mining Using Online Social Networks* data has given significant results too: we have focused only on one clique since it was the most important and it underlined the presence of an extended subgraph. Finally, we have chosen to apply all the measures performed above to the offline social network Infectious from the paper *What's in a crowd? Analysis of face-to-face behavioural networks* and we added one community detection method to have a clearer view of the results, which also in this last application are significant.
Some further results were drawn through the help of some graphs drafted on the results of the measures employed in the datasets *Organization Mining Using Online Social Networks* and *What's in a crowd? Analysis of face-to-face behavioural networks.*



This graph shows the probability density functions obtained through the Pandas method 'kde', passing as input all the results obtained for each of the five centrality measures applied in our investigation. pandas.DataFrame.plot.kde[15] generate a Kernel Density Estimate plot using Gaussian kernels. In statistics, kernel density estimation (KDE) is a non-parametric way to estimate the probability density function (PDF) of a random variable (as we can consider each of the executed centrality measures). This function uses Gaussian kernels and includes automatic bandwidth determination. In this case we decided to set the scalar bandwidth equal to 0.0005, which we found it suitable for the very small values we deal with. In particular:

- The blue plot represents the PDF of the degree centrality.

[15] Pandas documentation, https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.kde.html

- The red plot represents the PDF of the closeness centrality.
- The green plot represents the PDF of the eigenvector centrality.
- The orange plot represents the PDF of the betweenness centrality.
- The black plot represents the PDF of the load centrality.

Analysing the graph obtained from the centrality measures applied to the Facebook Ego Network dataset, we can infer some information:

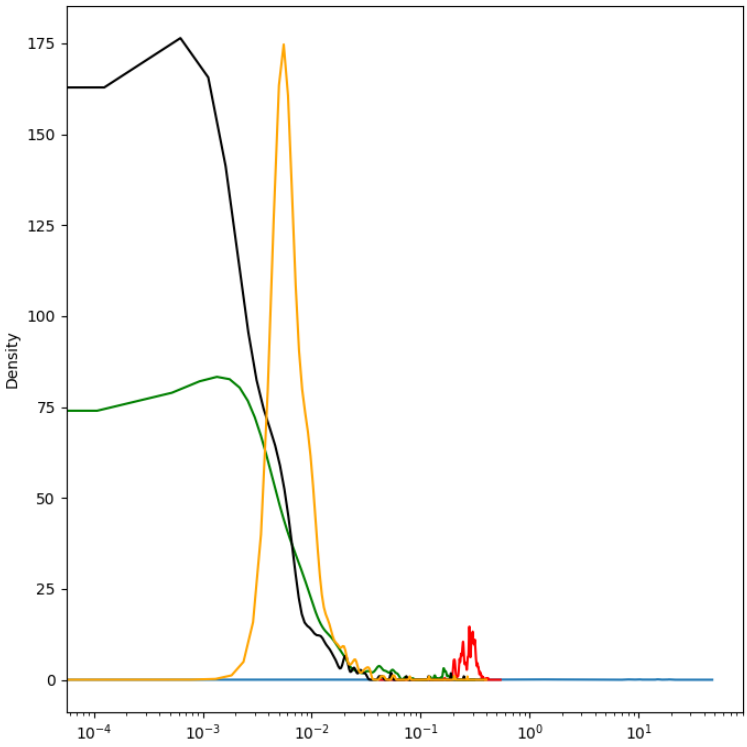
- The degree centrality nodes' values seem to be very different one from the other, the plot is very close to be a line, there isn't a peak in density and the distribution of the values is really spread.
- The red plot, that represents the closeness centrality values' distribution, seems to show the highest concentration of values in the same density area. There are some peaks, but we can say that the closeness centrality nodes' values do not vary a lot one from another (differently from the degree centrality distribution).
- The load centrality and the betweenness centrality density functions have a very similar shape, even if the orange plot decreases more rapidly: this prove that the load centrality measure is just slightly different from the betweenness.
- The green plot has the most uncommon trend: starting from very low eigenvector centrality values, the curve decreases rapidly till 0, then there is a quite high and defined peak, after which the curve tends to go to 0 for higher eigenvector centrality values.

So, we can say that for this network while there is a strong similarity between the betweenness and the load centrality, there is a difference between the closeness centrality and degree centrality values. It means that while there are few nodes with a high number of connections to other nodes, the Facebook Ego Network's nodes have very similar values of closeness centrality, so there are few marginal nodes in the network and it is highly interconnected

The following is the graph showing the probability density functions passing as input all the results obtained for each of the five centrality measures applied to the real-world network INFECTIOUS. A scalar bandwidth equal to 0.05 was set, because also the results we gathered from the execution of the centrality measures are higher than the one found in the Facebook dataset. Consequently, for this graph the choice of a very small bandwidth value would have resulted in over-fitting.
Also in this case:

- The blue plot represents the PDF of the degree centrality.
- The red plot represents the PDF of the closeness centrality.
- The green plot represents the PDF of the eigenvector centrality.
- The orange plot represents the PDF of the betweenness centrality.
- The black plot represents the PDF of the load centrality

We can extrapolate some interesting information by looking at the graph:



- Also, in this case we can notice a strong difference between two distributions, closeness centrality and degree centrality; while the blue plot is very close to be a line, there isn't a peak in density and the distribution of the values is really spread. The red plot shows the highest concentration of values in the same density area. There are some peaks, but we can say that the closeness centrality nodes' values do not vary a lot one from another.
- The load centrality and the eigenvector centrality distributions have a similar tendency: the distributions reach soon their highest peak and then they decrease rapidly.
- The tendency of the betweenness centrality distribution is curious: for very small values the curve tends to 0, then it reaches very soon its highest peak (for values a bit minor of 0,01) and decreases very quickly with some minor peaks for higher betweenness centrality values.

Conclusion

As a conclusion, we can surely affirm that the parallelism between different online networks is possible by applying, as in our investigation, the measures of one dataset to the other and vice versa. We can also confirm our hypothesis that offline and online networks have some similarities for what concerns the measure performed for analysing the data, given the assumption that they need to be of the same kind (e.g., undirected networks).

Critique

We have found some difficulties with the dataset and the analysis itself in the *Learning to Discover Social Circles in Ego Networks* paper. We found it lacking fundamental information to successfully apply the measures to other data. For example, the lack of data features blocked the execution of some community detection measures, such as K-means. Another problem we found was with the offline social network dataset: to perform our measures we have created a sub-dataset with only the information necessary to us, ignoring additional data which would have slowed the overall process.