

Network Analysis (Project Report)

Sara Vellone, Digital Humanities and Digital Knowledge, 1056935

Git Repository of the project material: <https://github.com/SaraVell1/Network-Analysis>

Context

General Context:

- Scientific discoveries
- Statistics and Computer Science for predictions

Specific Applications:

- Science in Social Networks
- News spreading through Social Networks (Twitter)
- Popularity prediction

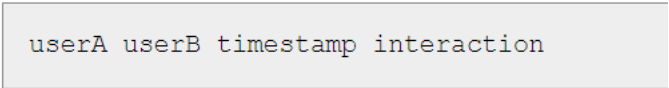
Problem and Motivation

The starting point was a paper, *Predicting the impact of online news articles*¹, which is based on a dataset in which are contained 40+ million tweets related to COVID-2019². The main idea was to reuse the measures applied for explaining if it is possible to predict the popularity of a scientific article on a dataset that is related to the spreading of the Higgs boson-like particles discover on Twitter³, trying to find some similarities. After a in-depth study, it has been noted that unfortunately most of the measures are related to textual information that is not available for the dataset under consideration. For this reason, in this study will be presented other measures that could be interesting for better analysing the “Higgs Twitter dataset”.

Datasets

Data were taken from two public sources: Stanford Large Network Dataset Collection for the paper *The Anatomy of a Scientific Rumor*⁴ and Awesome Public Dataset for *Predicting the impact online of new articles*.

The analysed dataset, the first one, has been built by monitoring the spreading process on Twitter before, during and after the announcement of the discovery of a new particle between the 1st and 7th July 2012. The original data was in the format shown in Figure1.



```
userA userB timestamp interaction
```

Figure 1: Dataset data format

¹ <https://link.springer.com/article/10.1007/s11042-021-11621-5>

² <https://zenodo.org/record/7753101>

³ <http://snap.stanford.edu/data/higgs-twitter.html>

⁴ <https://www.nature.com/articles/srep02980>

For this analysis, the timestamp and the type of interaction has been ignored, keeping only the “source” and the “target” of the interaction itself.

With the mentioned changes, the Dataset statistics presents **304691 nodes** and **457555 edges**. The total number of **triangles** is **101430**.

```
number_of_nodes = nx.number_of_nodes(G)
number_of_edges = nx.number_of_edges(G)
totalTriangles = nx.triangles(G)
number_of_triangles = sum(totalTriangles.values()) / 3
print(number_of_nodes)
print(number_of_edges)
print(number_of_triangles)
```

```
304691
457555
101430.0
```

Figure 2: Numbers of nodes, edges, and triangles

Validity and Reliability

For what concerns reliability and validity, the dataset is available to the public and it has been updated on March 2015. In this analysis, it has been noticed that there is a little difference in the number of nodes recorded by the study. The model presented is based on real users of the online social network Twitter that give it a positive compliance with reality.

Measures

The measures performed on this dataset are:

- The **Degree Centrality**⁵, which measures the number of edges attached to a node. It allows to assume that the nodes with the highest degree are the majors in the networks. In this dataset, the node “88” seems to be the node who plays the major role.

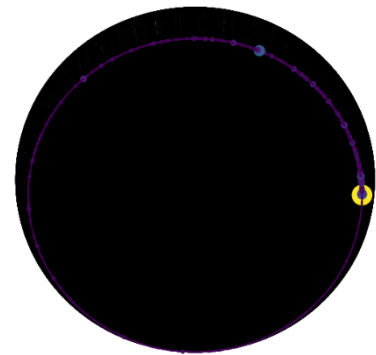


Figure 3: Degree Centrality on network's graph

```
dCen= nx.degree_centrality(G)

node_color = [20000.0 * G.degree(v) for v in G]
node_size = [v * 10000 for v in dCen.values()]
plt.figure(figsize=(10,10))
nx.draw_networkx(G, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(dCen, key=dCen.get, reverse=True)[:5]

[88, 677, 14454, 1988, 349]
```

Figure 4: Degree Centrality's snippet

- The **Closeness Centrality**⁶, which measures the mean distance from a node to other nodes, using the shortest paths. More central a node is, the closer it is to all other nodes. Nodes with higher values spread information more efficiently. The Closeness Centrality was measured on the node “88”.

⁵ https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms centrality.degree_centrality.html

⁶ https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms centrality.closeness_centrality.html

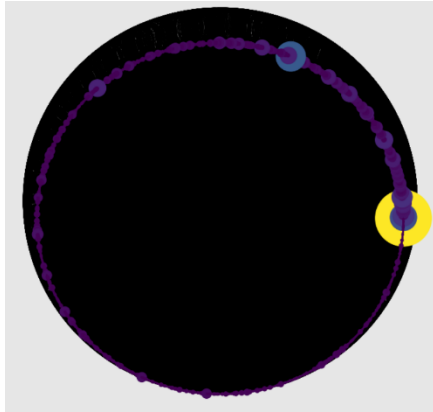


Figure 5: Closeness Centrality on network's graph

```
k = 88
closenessCent = nx.closeness centrality(G, u=k, distance=None, wf_improved=True)
node_color = [20000.0 * G.degree(v) for v in G]
if k == 88: node_size == k*10000
else: node_size = 10
plt.figure(figsize=(10,10))
nx.draw_networkx(G, pos=pos, with_labels=False,
node_color=node_color, node_size=node_size)
plt.axis('off')

(-1.20999999958953505,
1.2099999990756016,
-1.2149999995532044,
1.3149999951891544)
```

Figure 6: Closeness Centrality's snippet (node 88)

- The **Betweenness Centrality**⁷, which measures the extent to which a node lies on paths between other nodes. For this measure, the nodes have been restricted to only 10, due to the high number of them that cause a slow calculation.

```
myK = 10
betCent = nx.betweenness centrality(G, k= myK)

node_color = [20000.0 * G.degree(v) for v in G]
node_size = [v * 10000 for v in betCent.values()]
plt.figure(figsize=(10,10))
nx.draw_networkx(G, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
plt.axis('off')
sorted(betCent, key=betCent.get, reverse=True)[:5]

[88, 1988, 12281, 130884, 677]
```

Figure 7: Betweenness centrality's snippet (k=10)

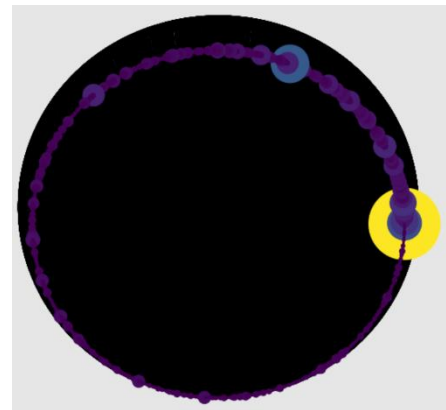


Figure 8: Betweenness Centrality on network's graph

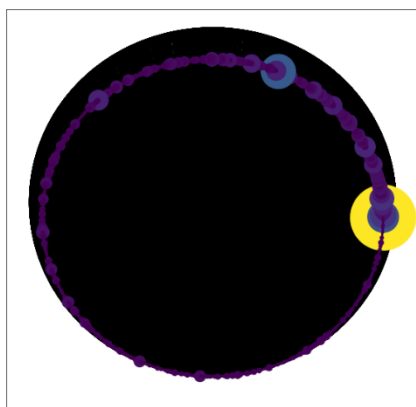


Figure 9: Eigenvector Centrality's on network's graph

- The **Eigenvector Centrality**⁸, that measures the node importance. It is an extension of degree centrality which awards several points proportional to the centrality scores of the neighbours of a node. For this analysis the maximum iterations and the tolerance have been increased.

```
EigCent= eigenvector centrality(G, max_iter=300, tol=1e-03, nstart=None, weight=None)

node_color = [2000.0 * G.degree(v) for v in G]
node_size = [v * 10000 for v in EigCent.values()]
plt.figure(figsize=(10,10))
nx.draw_networkx(G, pos=pos, with_labels=False,
node_color=node_color,
node_size=node_size)
sorted(EigCent, key=EigCent.get, reverse=True)[:5]

[88, 677, 14454, 1988, 349]
```

Figure 10: Eigenvector Centrality's snippet

⁷ <https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms centrality.betweenness centrality.html>

⁸ <https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms centrality.eigenvector centrality.html>

Other analyses have been also applied to the dataset, as:

- The **Average Neighbor Degree**⁹, which

```
myNodes = [88, 677, 14454, 1988, 349]
AvNeDegree = nx.average_neighbor_degree(G, nodes = myNodes)
print(AvNeDegree)
```

```
{88: 7.675695524451356, 677: 10.349765781842516, 14454: 2.734778597785978, 1988: 7.175665399239544, 349: 10.147024160282852}
```

Figure 11: Average Neighbor Degree's snippet

Returns the average degree of the neighbourhood of each node. For this analysis, the nodes under consideration are taken from the previous Eigenvector centrality measure's results.

```
DegreePearsonCoeff = nx.degree_pearson_correlation_coefficient(G)
print(DegreePearsonCoeff)
```

-0.06331247236345368

Figure 12: Degree Pearson Correlation Coefficient's snippet

in a network refers to the tendency of nodes to connect with other 'similar' nodes over 'dissimilar' nodes. In this case we can see a negative results, meaning relationships between nodes of different degrees.

- *Connectivity* measures, like:
 - The **Average Degree Connectivity**¹¹, that is the average nearest neighbour degree of nodes with degree k.

```
edge_connectivity = nx.edge_connectivity(G)
print(edge_connectivity)
```

0

```
node_connectivity = nx.node_connectivity(G)
print(node_connectivity)
```

0

Figure 14: Edge and Node Connectivity's snippets

the minimum number of nodes that must be removed to disconnect G.

- The **Degree Pearson Coefficient**¹⁰, which compute degree assortativity of graph. Assortativity

```
AvDeConn = nx.average_degree_connectivity(G)
myKeys = sorted(AvDeConn, reverse=True)[:5]
i=0
for key, value in AvDeConn.items():
    if key in myKeys:
        print(key, ":", value)
```

```
8966 : 10.349765781842516
23148 : 7.675695524451356
5260 : 7.175665399239544
3394 : 10.147024160282852
6504 : 2.734778597785978
```

Figure 13: Average Degree Connectivity's snippet

- **Edge Connectivity**¹², which is equal to the minimum number of edges that must be removed to disconnect G or render it trivial.

- **Node Connectivity**¹³, which is equal to the minimum number of nodes that must be removed to disconnect G.

9

https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.assortativity.average_neighbor_degree.html

https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.assortativity.degree_pearson_correlation_coefficient.html

11

https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.assortativity.average_degree_connectivity.html

12

https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.connectivity.connectivity.edge_connectivity.html

13

https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.connectivity.connectivity.node_connectivity.html

```

Clus = nx.clustering(G)

myK = sorted(Clus, key=Clus.get, reverse=True)[:5]
i=0
for key, value in Clus.items():
    if key in myK:
        print(key, ":", value)

364375 : 1.0
432622 : 1.0
294403 : 1.0
190062 : 1.0
66383 : 1.0

```

Figure 15: Clustering coefficient's snippet

- The **Average Clustering**¹⁵, which is the mean of local clustering, i.e. the fraction of triangles that actually exist over all possible triangles in its neighbourhood. The approximate coefficient is the fraction of triangles found over the number of trials.

```

cliques = nx.find_cliques(G)
max(len(c) for c in cliques)

12

```

Figure 17: Cliques number's snippet

- The **Clustering coefficient**¹⁴, which is a measure of the degree to which nodes in a graph tend to cluster together.

```

AvClus = nx.average_clustering(G, count_zeros=False)

print(AvClus)

0.5710815448684248

```

Figure 16: Average Clustering's snippet

- The **Clique number**¹⁶, that returns the size of the largest maximal clique. A clique is a subset of the nodes such that every two distinct nodes are adjacent. A maximal clique is a clique that cannot be extended by including one more adjacent node.

Results

For what concerns the “Centrality”, the measures applied in the previous section show the node with the highest degree centrality is the “88”, of which the closeness centrality has been calculated. The results of the eigenvector centrality show that this node is also one of the most important of the network.

The analysis of the “Assortativity” pointed out also that the node “88” - even if it is the most important - has an average neighbour degree lower than other nodes, like the “677”. The applying of the Degree Pearson Coefficient Correlation show a weak negative result.

About the “Connectivity” of the network, the Average Degree Connectivity - which is the average nearest neighbour degree of a node of degree k - shows that for five chosen nodes

¹⁴

<https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.clustering.html#networkx.algorithms.cluster.clustering>

¹⁵

https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.approximation.clustering_coefficient.average_clustering.html

¹⁶

https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.clique.find_cliques.html#networkx.algorithms.clique.find_cliques

there are strong dependencies between degree of neighbour nodes. Edge and Node Connectivity both returned 0.

So I thought to check the Clustering coefficient and the Average Clustering, which is about 0.57. The size of the maximal cliques is 12.

Conclusion

The analysis conducted shows that the network's graph is disconnected (due to the results of the node and edge connectivity) which means that there are nodes that are not linked. Given that, it was checked the clustering coefficient and the average clustering that is not strongly significant. In addition, the average shortest path length could not be calculated because the graph is disconnected. The attempt to calculate the coefficients omega and sigma failed due to NetworkX infinite run.

The assortativity measures shows that large-degree nodes tend to attach to low-degree ones, even if weakly. The centrality measures highlight the fact that some nodes have a particular influence than others on the network.

Critique

This work didn't solve the problem presented in the "Problem and Motivation" section. The main reason is that after a deep study of the papers and the datasets, the latter seems to be incompatible with each other for a real comparison. In fact, the "Higgs twitter dataset" does not contain the textual information needed to be analysed with the same measures applied in the Predicting the impact of online news articles. On the other hand, the dataset used in the article is not compatible with the measures of The Anatomy of a Scientific Rumor's one. In general, both the datasets are not easily manageable. They should be compared using different kind of measures, by having – for example – access to the textual information of the tweets of the dataset here discussed. Or, on the other hand, by having the "userA" and "userB" list for the second dataset, that could have allowed to apply the same measures on both the datasets. Finally, probably this type of dataset has to be analysed with different tools because the use of NetworkX's algorithms not always led to results. It was impossible to use other layouts but the circular or the random one and I have also tried to compute other measures not reported here, but -even customizing the inputs – the functions ran without ends.