ASDSI - Social Network Analysis 2162-F23: Exercise 7 - Graph Theory

2023-03-20 - **Lecture 13**

3.1 Network statistical distributions

Load the two networks from exercise of Lecture 12, namely Game of Thrones and your chosen network, using NetworkX. You will therefore have two graphs, for example named G_{GoT}, and $G_{yournet}$. Make sure to build an undirected network using NetworkX.

3.1.1

Compute the degree of each node and calculate the distribution.

(a) Plot this distribution (you can use a line plot with markers or a histogram) in (i) linear and (ii) log-log scales (use matplotlib `set_yscale(log)` option, default is linear).

(b) Do you see any difference between the plots (i) and (ii)? Which one is easier to read, providing a better understanding of the connectivity of the network? Why do you think it is so?

```
Functions: nx.degree(), collections.Counter(), plt.plot()
```

3.1.2

Comment on the interpretation of the distributions: Explain what a given (x,y) coordinate in this plot means.

3.1.3

Consider the distribution of the G_{GoT} network. Does it have hubs? Does this distribution resemble more a fat-tailed distribution or the distribution of a random network? Did you expect this and why?

3.1.4

(Optional) Compute the distribution for other measures provided by NetworkX, e.g. eigenvector centrality and shortest path length. Compare the networks, and look at the range of values and the shape of the distributions. What is the difference in their shape? Did you expect this difference?

```
Functions: nx.eigenvector_centrality(), shortest_path_length()
```

3.2 Random network model

3.2.1

In a random networks the average degree $< k >$ is, statistically speaking, equal to $< k >= p(n-1)$, where n and p correspond to the number of nodes and the probability for edge creation. Choose an arbitrary number of nodes n, then generate three random networks using NetworkX function `gnp_random_graph(n, p)` such that:

(a) $< k > < 1$

(b) $< k > = 1$

(c) $< k > > log(n)$

```
Functions: nx.degree(), nx.gnp_random_graph(), statistics.mean()
```

3.2.2

See how the network changes as you increase the probability of edge formation p by plotting the networks using the `nx.circular_layout`.

```
Functions: nx.draw(), nx.circular_layout(), plt.show(), np.linspace()
```

3.2.3

(optional) Measure the number of components (that is how many disconnected subnetworks make up the whole network) for each value of p. What kind of behaviour do you find?

```
Functions: nx.connected_components(), nx.spring_layout()
```

4.1 Network mesoscale structure

4.1.1

Study the degree correlations of the networks from the previous classes (namely, the Game of Thrones network, and the network of your choice) in the following way:

(a) Calculate for nodes of degree k, the average degree of their neighbors, $k_{nn}(k)$, as explained in class. To do this calculation, first you have to calculate for each node in the network the average degree of its neighbors, using for example the Networkx function `average_degree_connectivity`. Then you average these values for all the nodes that have the same degree k, obtaining $k_{nn}(k)$. Plot $k_{nn}(k)$ in function of k.

(a) What is the trend of $k_{nn}(k)$? What does it tell us about the degree correlations in the network?

(a) Calculate the degree assortativity coefficient in each network, for example using the Networkx function `nx.degree_assortativity_coefficient`. What does this value tell us about the degree correlations of the two networks?

```
Functions: nx.average_degree_connectivity(), plt.scatter(),
nx.degree_assortativity_coefficient(), nx.read_edgelist()
```

4.1.2

Use either the Girvan-Newman method `girvan_newman` or the modularity maximization `louvain_communities` to detect the community structure of the Zachary karate club dataset (provided with the exercise and described in the reading material of lecture 14).

(a) Draw the network, coloring the communities with different colors. You can create a coloramp in matplotlib with the number of colours equivalent to the number of communities obtained by the algorithm. Then, assign a colour to each partition (set of nodes) and build a dictionary with the color of each node. This dictionary can be used as input for the `node_color` parameter in the draw function.

(a) Do you recover the communities that were formed in the actual splitting of the Karate club?

```
Functions: nx.girvan_newman(), nx.louvain_communities() nx.draw_kamada_kawai(),
nx.draw(), get_cmap(), nx.algorithms.community.quality.modularity()
```