problem-set-networks

August 3, 2018

1 Table of Contents

```
In [1]: import pandas as pd
        import networkx as nx
        import matplotlib.pyplot as plt
        % matplotlib inline
```

2 Import the dataset

First you need to download your network data. In this exercise we will use data about the fictional book series *Game of Thrones* by G. G. Martin.

In this dataset the **vertices** in the data correspond to characters in Game of Thrones, the **edges** are weighted and represent the number of co-occurences of the characters in the book. A co-occurence happens if the names of the characters occured withn within 15 words.

You can download the data from the course homepage, it is a zip that contains networks stored as GML files. Your first tast is to load this data into your current Python session.

To do this, use the function `nx.read_gml`, on which you find plenty of information in the net.

3 Visualize the network

Draw the whole network. This will persuade you that a quantitative analysis is more useful than a visual inspection.

4 Analyze the networks

4.1 Clustering in Game of Thrones

Lets first check to what extend we observe clusterning in the GoT network. To this end, we calculate the *transitivity* of the network.

Calculate the transitivity for the first and the fifth book and compare the values.

What can you say about the magnitude of the clustering, is it high or low?

4.2 Who are the most important characters? Centrality analysis

Now use the centrality measurs we encountered in the lecture to find out who is really important in GoT.

4.2.1 Degree centrality

First, calculate the degree distribution for the first and fifth book. Visualize them via a histogram.

You can use the following code chunk as a starting point:

```
In [ ]: fig, axes = plt.subplots(1,2, figsize=(12,4))

        axes[0].hist(________, color="#3F5D7D", density=True)

        axes[0].set_title("Degree centrality in the first book")
        axes[0].set_xlabel("Normalized degree")
        axes[0].set_ylabel("Nb of vertices")

        axes[1].hist(_________, color="#3F5D7D", density=True)

        axes[1].set_title("Degree centrality in the fifth book")
        axes[1].set_xlabel("Normalized degree")
        axes[1].set_ylabel("Nb of vertices")

        plt.tight_layout(True) # Good to get better alignment

        # plt.savefig("figs/centrality_pagerank_hist.pdf", bbox_inches="tight")
```

Next, have a look at changes on the individual legel. To this end, print the 10 most important characters both for the first and the fifth book. Again, you can use the following code chunk. What you need to insert are the relevant items of the degree dictionary you have created above.

```
In [ ]: sorted(________, key=lambda x:x[1], reverse=True)[0:10] # Look at top 10

In [ ]: sorted(________, key=lambda x:x[1], reverse=True)[0:10]
```

Finally, plot the graph of the fifth book again and color the vertices according to their degree centrality.

What would you say, does it pay off for the Game of Thrones to be well connected?

4.2.2 Eigenvector centrality

Next, compute the eigenvector centrality using the PageRank algorithm (using `nx.pagerank`), again for the first and fifth book.

Then plot the distribution and compare the top 10 characters in the first and fifth book.

Finally, plot the network for the fifth book and color the vertices according to their PageRank scores.

4.2.3 Betweenness centrality

Finally we want to consider the betweenness centrality.

First, compute the betweenness centrality using the function `nx.betweenness_centrality`.

Then visualize the distribution as you did before.

Finally, plot the graph and color vertices according to their betweenness centrality.

4.2.4 Comparison of the centrality measures

Finally, compare the three centrality measures with each other.

To this end, create a figure with three plots.

Each plot should be a scatter plot mapping two different centrality scores against each other. You can use the code chunks below as a starting point.

How would you interpret the results?

```
In [ ]: fig, axes = plt.subplots(1,3, figsize=(12,4))
        for x in range(3):
            axes[x].spines["top"].set_visible(False) # Remove plot frame line on the top
            axes[x].spines["right"].set_visible(False) # Remove plot frame line on the right
            axes[x].get_xaxis().tick_bottom()  # Remove ticks on the bottom
            axes[x].get_yaxis().tick_left()  # Remove the ticks on the left

        axes[0].scatter(______, ______, color="#3F5D7D")
        axes[0].set_title("Degree vs. Eigenvector centrality")


        axes[1].set_title("Degree vs. Betweenness centrality")


        axes[2].set_title("Eigenvector vs. Betweenness centrality")

        plt.tight_layout(True) # Good to get better alignment
```