Project 4

Word Adjacencies Network Analysis

Tomi Ahonen

github:
https://github.com/tomiahonen/JSVA20
24

I. Introduction

In this project we are examining “Word Adjacencies”, which is available at https://websites.umich.edu/~mejn/netdata/ in GML file format. The data consists of an undirected network of common noun and adjective adjacencies for the novel “David Copperfield” written in 1850 by Charles Dickens. In this network one node stands for either a noun or an adjective and an edge connects two nodes that appear in adjacent positions.
This project analyzes the relations between word adjacencies using network analysis methods. With the created network, we can explore how different words are connected to each other and what kinds of significance these connections can represent. Using the data provided we create a network where each word is a node and an edge between two words is the connection between them. After the creation of the network, we can use different metrics to analyze the network. These methods include commonly used metrics in network analysis, such as Node degree distribution, betweenness centrality, closeness centrality, community detection and network density. Many of these metrics are created by using a Python package tool “NetworkX” because of its ability to easily manipulate, create and study complex networks.
The results of this project can offer new insights into how the language and meanings are built upon. Also, it can help us name certain themes and events but also reveal possible linguistic or narrative models in the novel. Similar work to this project by Anna Size, “The Words of Dickens: A network visualization” shows the relationship between different data points.

II. Dataset description

In order to do network analysis, it is helpful to create a graph from the given data. To create the graph, we need to name the actors (=nodes) and ties (=edges) that need to be constructed. [1]. By using NetworkX library function “nx.draw” we can display the data as a graph with nouns and adjectives as nodes and their connecting edges.
The file “adjnoun.gml” contains the network of common adjective and noun adjacencies for the novel “David Copperfield” by Charles Dickens. Node values are 0 for adjectives and 1 for nouns. Edges connect any pair of words that occur in adjacent positions in the text of the book. [2]. The file holds 112 nodes and 425 edges connecting these nodes. On top of the node’s value, which is either 1 or 0, there is also the node’s label which is one word, for example “round”, “eye” and “world”. The file also includes an id for each node ranging from 0 to 111. The id makes it easier to identify each node in the graph.

III. General methodology

The general methodology of the project outlines a comprehensive approach to solving the problems.
Firstly, the dataset is read. The length of the dataset makes it easy for a person to riffle through and find the key points in it. After that the data can be prepared for analysis. Preparing the data includes turning it in to an easily readable graph, installing all the necessary tools and libraries like NetworkX, which the analysis heavily relies on, and Matplotlib to help to plot the results.
The analysis using NetworkX includes different network metrics, such as degree distribution, betweenness centrality, closeness centrality, community detection and network density. Visualizing and interpreting the data is done with the help of different diagrams so that the results and measurements are more suitable for viewing. Some numerical values are also printed on the console since they do not require any plotting.
As with all graphs, communities can be detected. Communities in networks are set of easily groupable nodes that have something in common. [3]. In the case of “adjnoun.gml” this common thing could be for example the value of the node, which is either 0 or 1 depending on whether the node is an adjective or a noun. There are a few community detection algorithms suitable for this kind of data such as the Girvan-Newman algorithm.
The results obtained from the project are analyzed thoroughly and conclusions are drawn on word adjacencies.

IV. Detailed methodology

The go through of the project starts with displaying the network of the dataset as a graph using the Python libraries NetworkX and Matplotlib. NetworkX library is a package for the creation, manipulation, and study of the structure, dynamics and functions of complex networks. [4]. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. [5].

Visualizing the dataset

NetworkX library functions used to create the graph are for example “read_gml()” that takes the path to the GML file as parameter. As the dataset is read to the variable G, we can start to make the visualization by using Matplotlib to plot the figure and use the NetworkX function “draw()” to draw the graph in any color, font size and node size we want. For this graph, we chose 150 as the node size and 7 as the font size to support its readability. Otherwise, the nouns and adjectives would be hard to distinguish from the graph. After these steps we can look at the graph (image 1). In the first task we also create the adjacency matrix of the graph. This is done by converting the graph G to a NumPy array using the “to_numpy_array()” from NetworkX.
Adjacency matrix is a matrix containing only elements like ones and zeroes and is used to represent a finite graph. The matrix shows whether a pair of nodes are adjacent in the graph. [6]. Since the adjacency matrix is so large only a part of it is displayed in image 2.

Bipartism of the graph

In task two we were asked to write a script to figure out whether the graph G is bipartite or not. A graph is bipartite if the nodes of the graph can be divided into two disjoint groups so that no two nodes within the same set are adjacent. [7]. To figure out if the graph is bipartite or not, we can use the NetworkX function “is_bipartite()”. This function returns True if the graph is bipartite and False if it is not bipartite. In this case we get the result False, which means that the graph G is not bipartite.

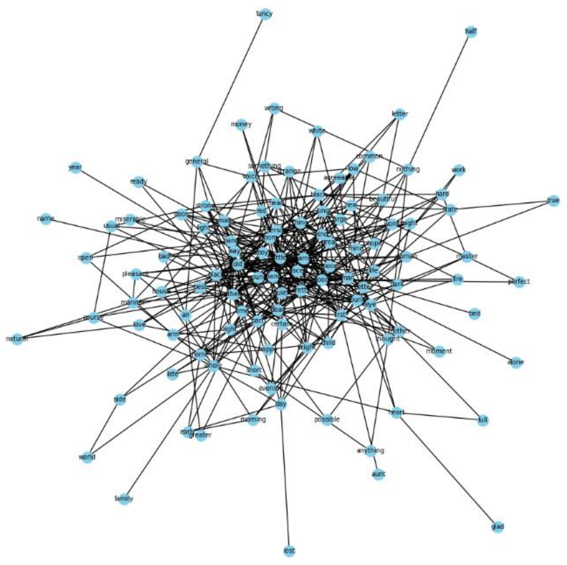
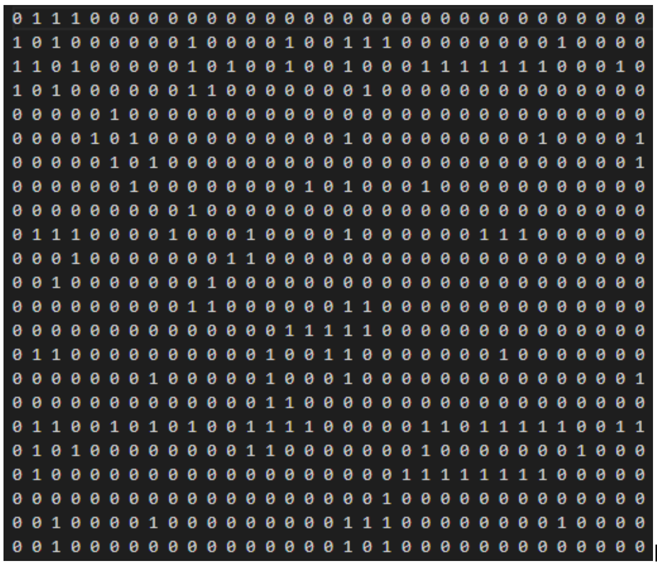


Image 1. Visualized graph of the dataset.



```
0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0
1 1 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 0
1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1
0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Image 2. Adjacency matrix.

Degree centrality, closeness centrality and betweenness centrality

Task three asks us to explore the three highest degree centrality, closeness centrality and betweenness centrality. Degree centrality is a metric that tells us the number of connections a certain node has to other nodes in the network. In other words, the nodes that have the highest degree centrality have the greatest number of edges. Closeness centrality of a node tells us how important the node is based on its distance to all other nodes. In other words, if a node has a high closeness centrality, the node is close to all other nodes in the network. Lastly the betweenness centrality describes the number of times a node is included in the shortest path from any node in the network to another node in the network. In other words, nodes with the highest betweenness centrality have the greatest number of shortest paths between them.
In task three we find the nodes with three of the highest centralities. Once again, we refer to NetworkX documentation for this. Firstly, these three centralities are calculated by using the functions “degree_centrality()”, “closeness_centrality()” and “betweenness_centrality()”. After those calculations are made it's possible to figure out the three nodes with the highest centralities by using Python function “nlargest()” which takes the number of nodes we want and the calculated centrality values and outputs the result.

The three nodes with the highest degree centrality are “little”, “old” and “other”. What that means is that these nodes have the greatest number of edges out of all nodes in the network. The three nodes with the highest closeness centrality are “little”, “old” and “good”. These nodes are close to all other nodes in the graph, which means that these nodes can quickly interact with other nodes. The three nodes with the highest betweenness centrality are once again “little”, “old” and “other”. These nodes have the most number of shortest paths that pass through them.

Plotting, cumulative degree distribution and clustering coefficient distribution

Task four tells us to plot the distributions mentioned in the previous chapter. This is simply done with the Matplotlib functions “hist()” to visualize the distribution in a histogram and “show()” to show the created histogram.
Cumulative degree distribution tells us the distribution of node degrees in a network graph. What this means in other words is that the cumulative degree distribution shows us the proportion of nodes that have a degree less than or equal to a certain value. Clustering coefficient distribution, however, tells us how many certain nodes in a graph tend to cluster with each other. [8]. The clustering coefficient distribution shows the frequency of nodes that have a certain value. The clustering coefficient of 0 means that none of a node’s neighbors are connected to each other, while 1 means that all of them are connected to each other. The clustering coefficient for this graph was calculated using the NetworkX library function “average_clustering()”.

Distributions of the centralities

In task five we are given the objective of fitting the previously calculated centrality distributions into a power law distribution. Power law distribution means that some nodes are quite common (they appear many times in the data) while others are quite rare, and the ratio of their appearances follows a mathematical relationship. When calculating the goodness of the fit, we can use p-value as an indicator. When p-value is greater than 10%, we can say that the power law is a good fit for the data. As we will discuss in the results, the power law distribution is not a plausible fit for this data, but log-normal distribution provides much better p-values.
To calculate the fit and the p-values we used functions from the Powerlaw library. The function “Fit()” was first used to fit the centrality values and then “distribution_compare()” is used to compare the goodness of fit between the probability

distributions for the data. In this case power law distribution and exponential distribution are being compared. From this, two values are returned. Value R is the loglikelihood ratio between these two distributions and Value p is the p-value for these.

Exponentially truncated power-law

In task six we need to use the exponentially truncated power law instead of the power law distribution. We use the Scipy Python library function to fit the exponentially truncated power law distribution to the centrality distributions. This function is "kstest()" (Kolmogorov-Smirnov test). We gain two values from here. Value D is the maximum distance between the cumulative distribution function (CDF) of the sample and CDF of the reference distribution. The larger the value D is, the worse the fit.

Communities

Community consists of group of nodes that are densely connected to each other but sparsely connected to other dense groups in the network. [9]. There are few ways to detect communities in networks. In the task we are asked to use the label propagation algorithm in NetworkX. However, this provides only one community, so we use the Girvan-Newman and Louvain algorithms to detect communities. Both algorithms are found in NetworkX as functions. Modularities for both algorithms are calculated using the "modularity()" function. Modularity measures the quality of assignment of nodes to communities in a specific network. In other words, it evaluates the results of the algorithms.

Proportions

We are assigned to quantify the density of adjective and noun node topology. For each node in the network, we calculate the proportion of P(X) of neighbors that have the same affiliation as X. First, we create a function to calculate the proportions. We identify whether the word is a noun or an adjective by its given value 1 or 0.

V. Results and discussion

The resulting graph of the data is shown in image 1. The diameter of a graph is the longest path between any pair of nodes. In this particular graph the diameter is 5. Diameters can be used to provide insight on how compact the graph is. In this graph, diameter of 5 tells us that it is quite compact and that means that all the nodes are relatively close to each other. We also gained the same results from the degree, closeness and betweenness centralities. If the diameter of the graph were to be larger, there would be a possibility of bottlenecks and weak points in the network.

We concluded that the graph G is not bipartite. This means a few things; the graph must contain at least one odd cycle length. Non-bipartite graphs have more complex connections meaning that the relationship between nodes is also more complex.

We found the three nodes with the highest degree centrality, closeness centrality and betweenness centrality.

Degree centrality:	Little	Old	Other
Closeness centrality:	Little	Old	Good
Betweenness centrality:	Little	Old	Other

Because "Little" and "Old" appear in each centrality, mean that they are very well connected and centrally located inside the network. Because they are the center of the network, they are also critical for the flow of the network because so many paths go through them. The node "Other" is also very well located and centered but does not quite reach the significance "Little" and "Old" do. Even though "Good" is only present once, it still is very well connected in comparison to rest of the nodes outside of the top three.

The distributions for all of the centralities were also plotted and labeled resulting in images 3, 4 and 5.

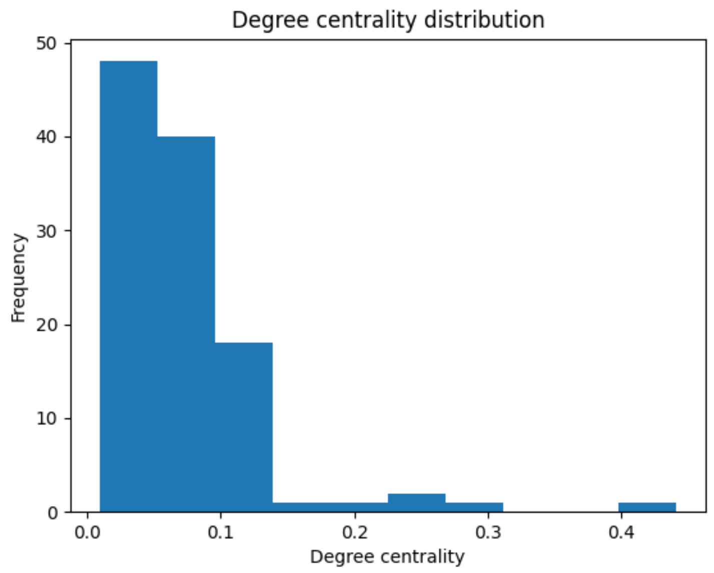


Image 3. Degree centrality distribution.

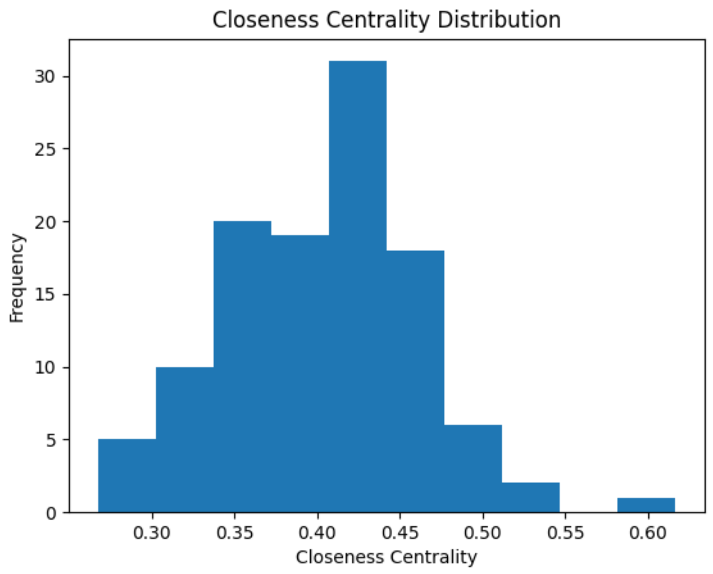


Image 4. Closeness centrality distribution.

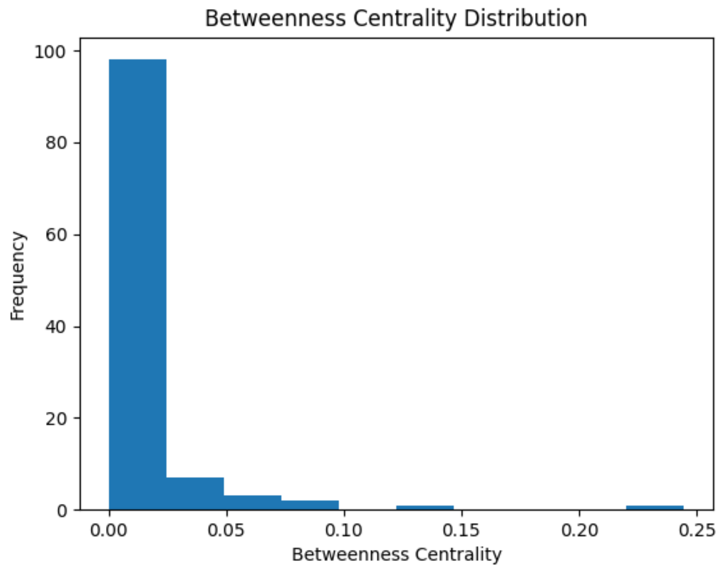


Image 5. Betweenness centrality distribution.

The histogram for degree centrality distribution (image 3) shows that a large portion of the nodes have a low amount of direct connections to other nodes in comparison to what they could have. An assumption could be made that the nodes that have the degree centrality of ~0.4 are the three nodes with highest degree centrality. We can also see that the next nodes with highest degree centrality are quite far from the value of 0.4.

The values from the histogram for closeness centrality distribution (image 4) show that a large number of nodes have the closeness centrality value of ~0.425. Many of the nodes are close to each other in the network and this can be also observed from image 1 where we display the graph. In image 1 there is a large group of nodes in the middle of the graph, and it seems to go hand in hand with the closeness centrality distribution.

Image 5 containing the betweenness centrality distribution looks very different to the previous two histograms. In this histogram we can see that the majority of the nodes have a betweenness centrality close to zero. What this means in terms of the network is that these nodes rarely are found on the shortest path between any two nodes. This further emphasizes the importance of the three nodes with the highest centralities since their betweenness centrality values is far greater than rest of the nodes in the network. We can go as far as to say that these 100 nodes with values around zero are not important for the network since they rarely lie on the shortest paths.

Cumulative degree distribution chart (image 6) shows us that there is a rapid growth in the frequency in the start but after that it hits a plateau when the degree increases. We can immediately see that large amount of the nodes in the network have low degrees, meaning that there is only a small amount of nodes connected to lots of nodes.

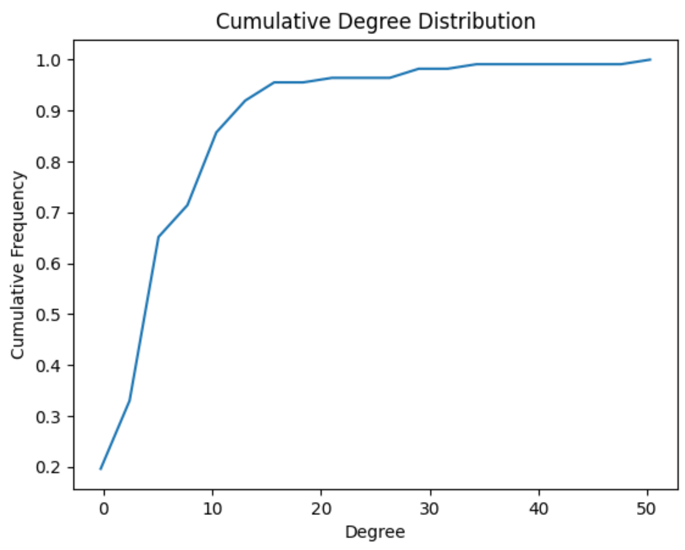


Image 6. Cumulative degree distribution.

Overall, the graph could be described as a major city. Lots of small towns on the outskirts and major hub in the center where all the traffic goes through.

References

[1] M. Oussalah, Introduction to social network analysis, lecture 2 material.

[2] M. E. J. Newman, Phys. Rev. E 74, 036104 (2006).

[3] Wikipedia: Community structure.

[4] NetworkX developers, Networkx.org.

[5] Matplotlib development team, matplotlib.org.

[6] Wikipedia: Adjacency matrix.

[7] Wolfram research, Inc., mathworld.wolfram.com.

[8] Wikipedia: Clustering coefficient.

[9] Mason A. Porter, Jukka-Pekka Onnela, Peter J. Muchas, Communities in networks.