Spotify Artist Co-Occurrence Demo

Kyaw Soe Han & Daniel Chukhlebov

Link to repository: https://github.com/meteor123456/Spotify-Artist-Cooccurence

Data Collection and Graph Preparation

(Done in the network_init.ipynb file)

Raw data from this repository folder. The initial csv files were concatenated into a large pandas data frame with 269580 entries. Each entry represents a song found on a playlist that has two two columns artist_name and pid, which represents the artist of the song and the playlist it belongs to respectively.

This data frame was then reformatted to a pandas series with the index being the playlist ids and the value being a list of the names of the artist for each song that appears in that playlist. The series has 4000 unique playlist entries.

To reduce the computational load, samples of 30 and 100 were made. The samples then were formatted again to have the value be a dictionary where the key is the artist name and the value to be the number of appearances in the playlist.

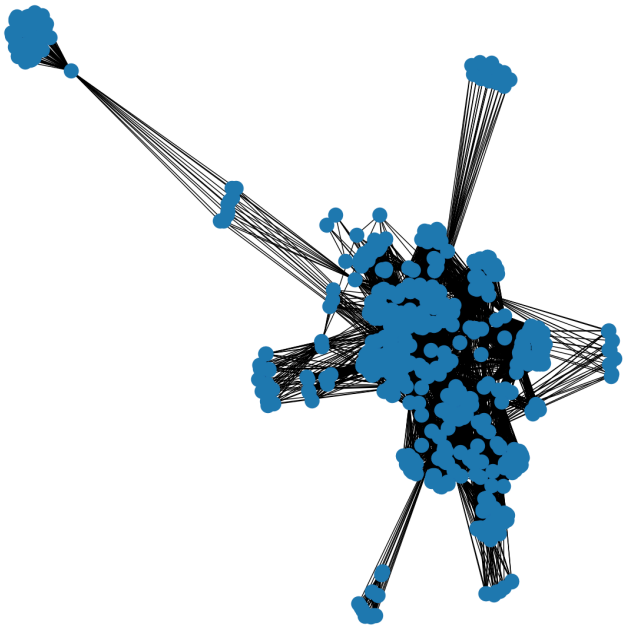
Using the finalized series data, a co-occurrence graph was constructed using NetworkX. A node for each artist is created in the graph and an edge is created if there is a co-occurrence between the two artists on a playlist. The weight of the edge represents the number of co-occurrences of songs by the same artists in a playlists. The larger the weight of the edge, the more times the artists appear together in a playlist.

The graph is then checked for its connectedness. Since the graph was unconnected, an unconnected version was saved while another version where only the giant component exists was saved.

Simple Graph Analysis
(Done in the analysis.ipynb file)

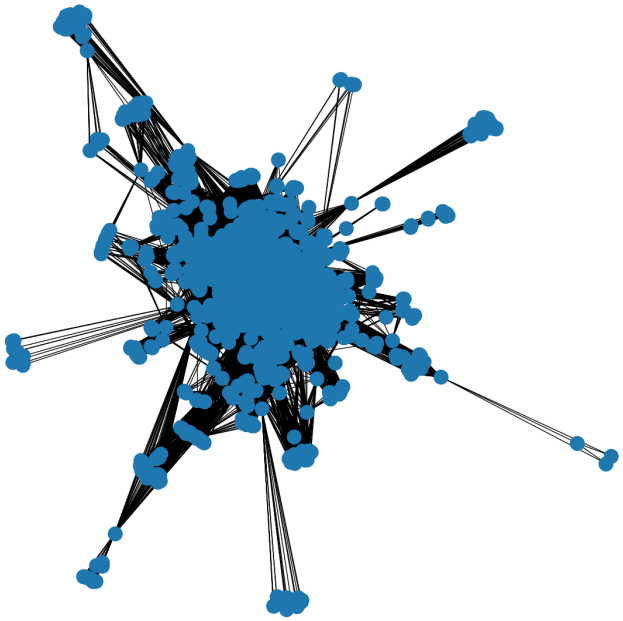
Below are simple NetworkX images of the connected graphs.

Sample of 30 playlists connected graph: 743 nodes and 23508 edges

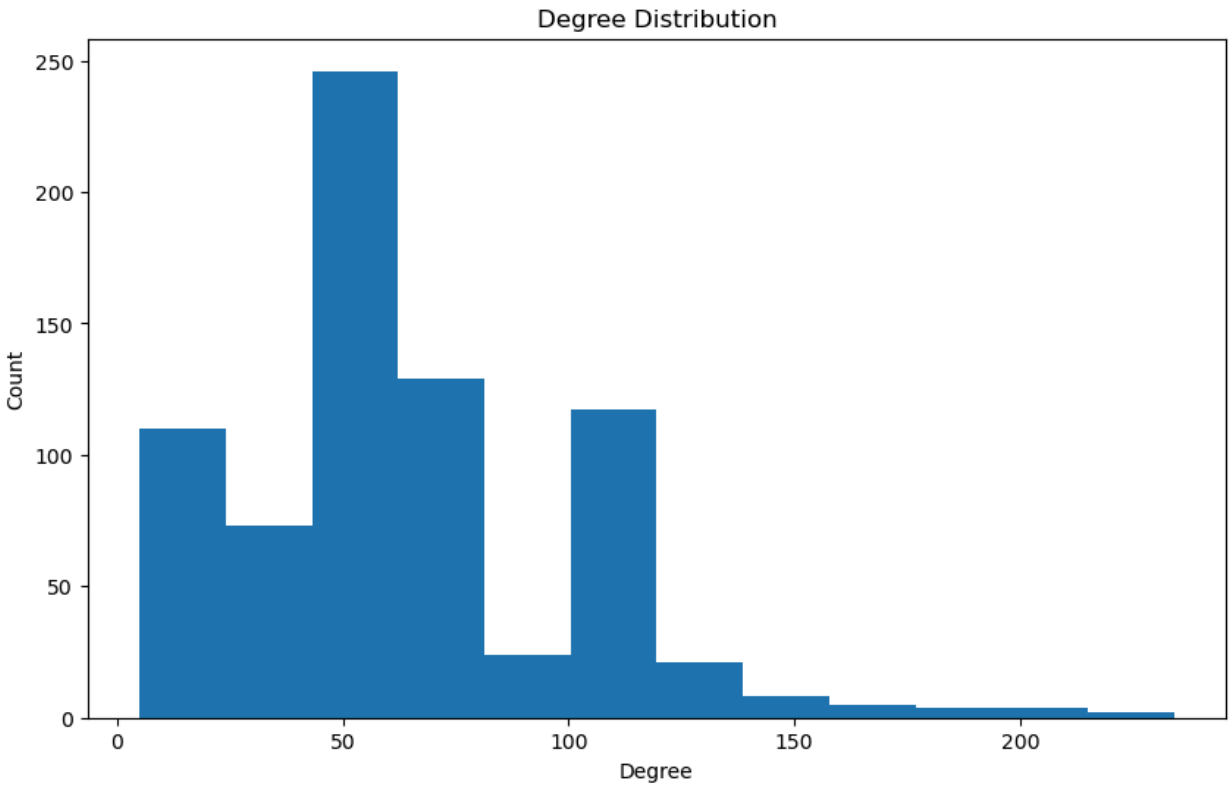


Sample of 100 playlists connected graph: 1860 nodes and 82404 edges

The larger sample took about 5 times longer to render and also has a much more cluttered image output. Therefore, moving forward, the graph of the smaller sample will be used for analysis and calculations.



Degree Distribution (12 bins)



The degree distribution of this sample seems to be skewed to the right and does not closely follow the expected power law distribution. However, this may just be due to the small sample size of the data.

Analysis of Centrality Measures (30 playlists)

`Average Degree Centrality: 0.0853`

The fairly low degree centrality indicates artists are likely to appear together in shared playlists, instead of being present in many different playlists. On average, every artist is connected to only 8.53% of other artists in the network.

`Average Betweenness Centrality: 0.0020`

The very low betweenness centrality shows that artists generally do not lie on the shortest paths between other artists. There is probably a very small number of artists who act as bridges between clusters of artists.

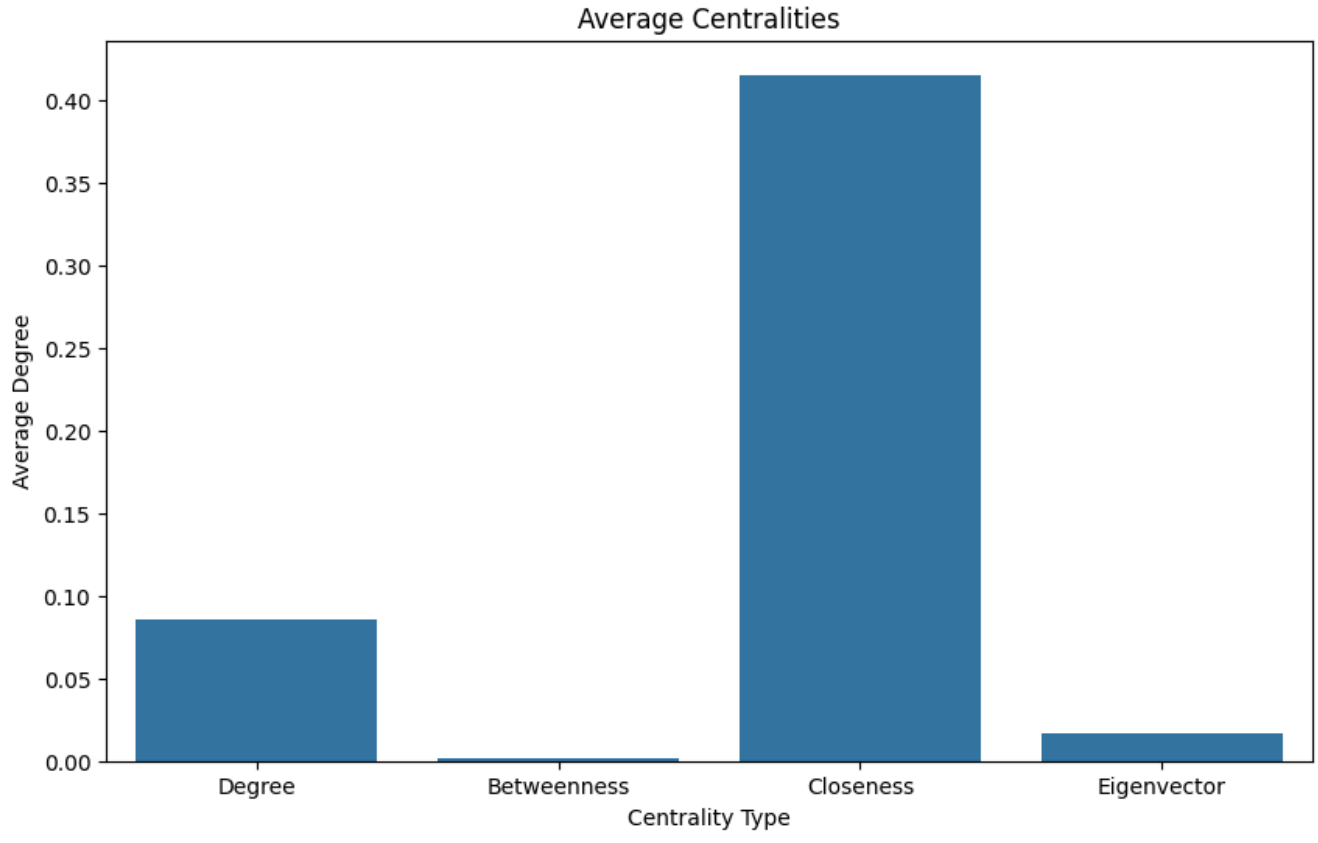
`Average Closeness Centrality: 0.4150`

The closeness centrality is much higher than other centrality measures. This indicates that the network is well connected despite the low degree centrality. We can infer that the graph is easy to traverse, and artists can influence other artists more easily.

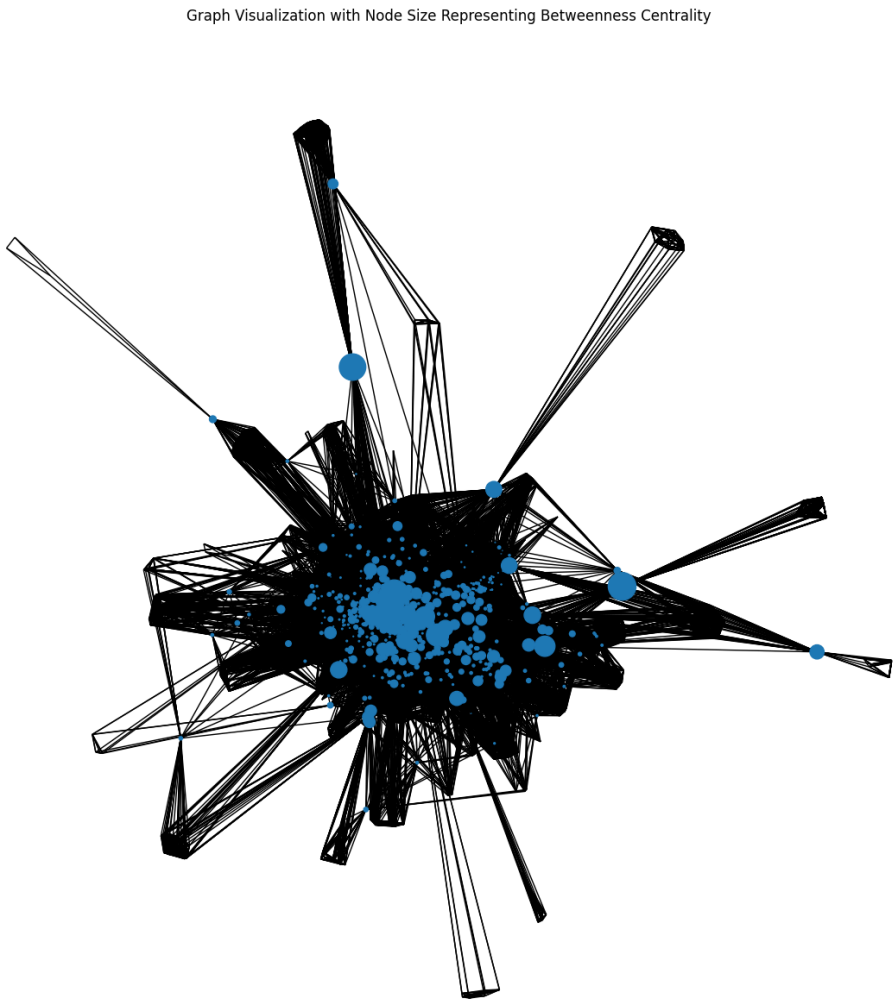
`Average Eigenvector Centrality: 0.0165`

The low eigenvector centrality means that most artists generally have little influence in the network. It is possible that a small number of popular artists hold the most of the influence within the network, while the majority of artists are not very influential.

Histogram of Average Centralities



Visualization of 30 Playlist Graph using Betweenness Centrality



The large clump of nodes in the center, (especially large nodes), represents heavily interconnected and popular artists. They are likely the ones that are listed in most playlists and listened to the most often. There are many smaller nodes circling that clump, indicating semi-popular artists who also form important and numerous connections, but are ultimately less influential. It is safe to assume that they belong to popular genres or subgenres of music. There are some large nodes which form important connections between the central cluster and outer clusters. These bridging nodes are critical in the network and have high betweenness centralities. They likely link very niche but still popular communities with the greater central cluster. The small nodes in those distant subcommunities represent less popular artists who belong to that genre of music.

Analysis of Centrality Measures (100 playlists)

`Average Degree Centrality: 0.0477`

The fairly low degree centrality indicates artists are likely to appear together in shared playlists, instead of being present in many different playlists. On average, every artist is connected to only 4.77% of other artists in the network. The degree centrality here is roughly 50% less than in the 30 playlist graph.

`Average Betweenness Centrality: 0.0008`

The very low betweenness centrality shows that artists generally do not lie on the shortest paths between other artists. There is probably a very small number of artists who act as bridges between clusters of artists. The betweenness centrality is much lower than in the 30 playlist graph.

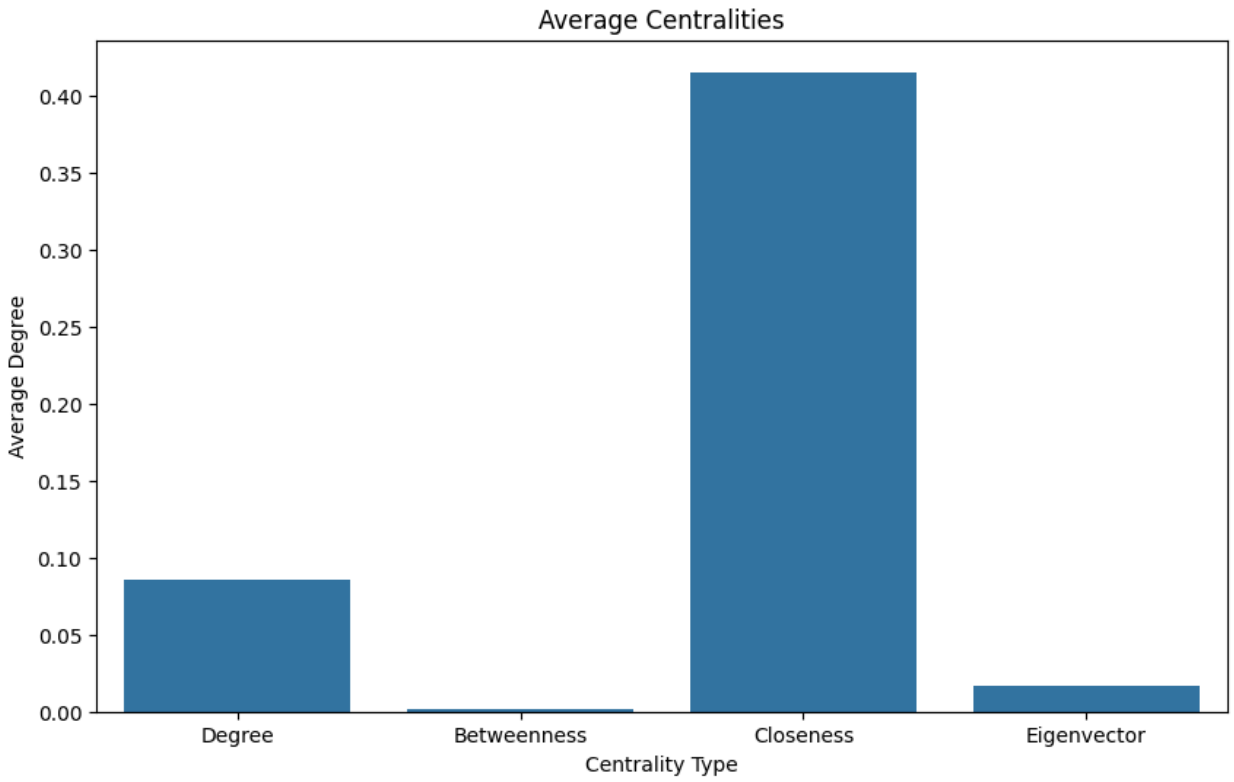
`Average Closeness Centrality: 0.4115`

The closeness centrality is much higher than other centrality measures. This indicates that the network is well connected despite the low degree centrality. We can infer that the graph is easy to traverse, and artists can influence other artists more easily. The closeness centrality here is about the same as in the 30 playlist graph.

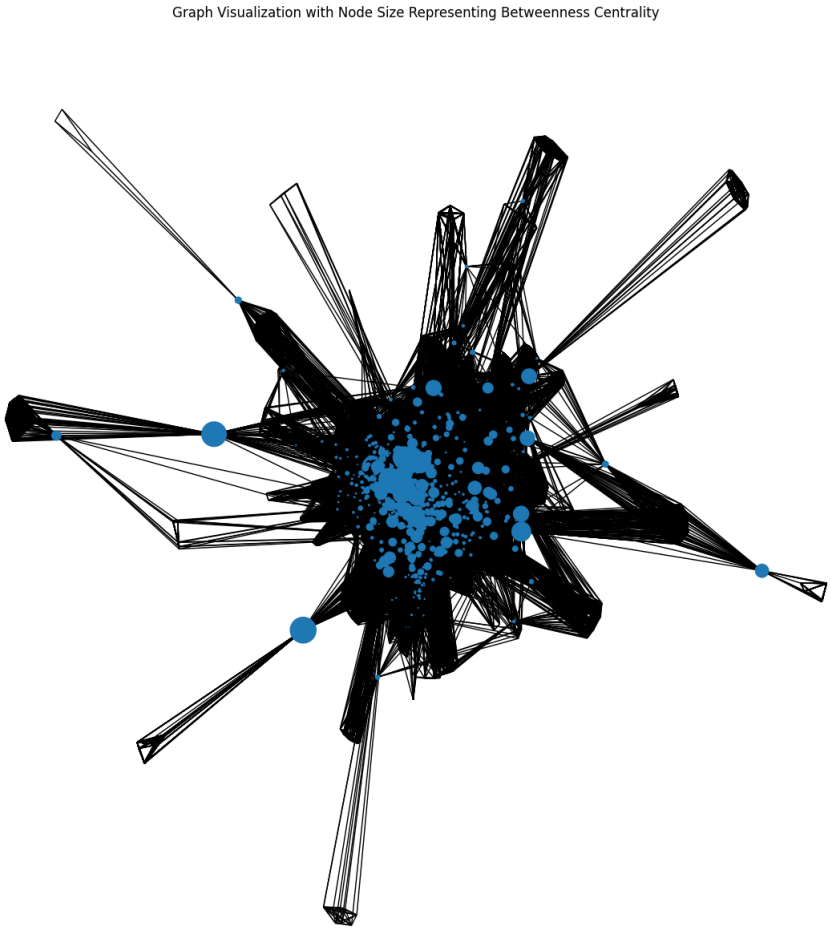
`Average Eigenvector Centrality: 0.0134`

The low eigenvector centrality means that most artists generally have little influence in the network. It is possible that a small number of popular artists hold the most influence within the network, while the majority of artists are not very influential. The eigenvector centrality is a bit less here than in the 30 playlist graph.

Histogram of Average Centralities



Visualization of 100 Playlist Graph using Betweenness Centrality



Despite the increase in graph size, the visualization shows a consistent structure between 30 playlists and 100.