Spotify Artist Co-occurrence across Playlists in the US

Final Report

By: Kyaw Soe Han SJSU ID: 015543671 EMAIL: kyawsoe.han@sjsu.edu
Daniel Chukhlebov SJSU ID: 015379234 EMAIL: daniel.chukhlebov@sjsu.edu

CS 176

Professor Katerina Potika

November 2024

Project Repository: https://github.com/meteor123456/Spotify-Artist-Cooccurence

Table of Contents

I. Introduction

This project aims to analyze the co-occurrences of US-based artists across a sample of Spotify playlists from 2017. Centrality measures and various algorithms were utilized to gain insights into characteristics such as network density, the influence of important artists, community clustering within the network, point out important artists, and more. Visual representations of the data were also generated to help understand how the network is structured [1].

Goals

1. Analyze popularity of artists based on centralities
2. Find communities of similar artists that co-occur across playlists using Louvain
3. Predict possible links using Jaccard Similarity

Data Set

Data Source: https://github.com/rodolfostark/spotify-network-analysis/tree/main/data_CSV [1].
Our data was downloaded from a GitHub repository by user rodolfostark. The repository contained 4 CSV data files containing entries of co-occurrences with artist names attached to a unique playlist ID: artist_name, PID. We decided to choose this data because it is set in an accessible format and allowed us to change it into a graph for later use. In terms of the project, we chose the topic and data of Spotify co-occurrences out of curiosity about the music industry and which artists were influential at the time [2].

Graph Preparation

In order to work with the data, we concatenated the CSV files into a single Pandas dataframe. Then it was reformatted into a Pandas series, where the index was the playlist IDs and the value was a list of artists that appeared in the playlist. In order to increase computation efficiency, we made two sample sets of size 30 and 100. The samples were then reformatted to have the value be a dictionary, in which the key is the name of the artist and the value is the number of their appearances in the playlist. After that, we constructed a co-occurrence graph via NetworkX, where nodes are artists and edges represent co-occurrences of artists in shared playlists. Edge weights represent the number of co-occurrences of songs by the same authors in a playlist. So a heavier edge means artists appear together in a playlist more times. We then checked the graph for connectedness [2]. Because the graph turned out to be unconnected, we saved the unconnected version as well as another where only the giant component was saved. The data was formatted in this manner:

```
<node id="AC/DC"/>
<node id="Billy Joel"/>
<edge source="AC/DC" target="Billy Joel">
  <data key="d0">45</data>
</edge>
```

Sample Comparison

Having made two samples of 30 and 100 playlists, we wanted to see if we could use the smaller sample in place of the larger one to remain computationally efficient. For this, she samples needed to see expected differences in centrality measures, considering the difference in sample size. The 30 playlist sample saw average degree, betweenness, closeness, and eigenvector centralities of 0.0853, 0.0020, 0.4150, and 0.0165 respectively. The 100 playlist sample saw average degree, betweenness, closeness, and eigenvector centralities of 0.0477, 0.0008, 0.4115, and 0.0134 respectively. The change in size between the centrality measures of the smaller and larger sample were -0.0376 (44.8% smaller), -0.0012 (60% smaller), -0.0035 (0.84% smaller), and -0.0031 (18.79% smaller) respectively. The changes in degree and betweenness centralities are expected due to the difference in sample size. The closeness and eigenvector centralities did not change significantly, which is also expected. This means that our samples are robust and inferences made about them can be extrapolated for larger or smaller datasets. However, the 100 playlist sample took about 5 times longer to render visually in NetworkX than the 30 playlist sample, meaning that it was cumbersome. Considering the scalability of our samples, we decided to use the 30 playlist sample for further analysis to avoid heavy computational load. The 30 playlist sample has 743 nodes, 23508 edges, and a diameter of 5, with only the one giant connected component.

In our sample data, the 5 artists with highest betweenness centrality are John Williams (0.135), The Beatles (0.095), Backstreet Boys (0.073, Eminem (0.0714), Ed Sheeran (0.070), Ellie Goulding (0.062), Eagles (0.049), Lynyrd Skynyrd (0.048), Maroon 5 (0.037), and ASAP Rocky (0.034). These artists act as important bridges between various communities within the network and are likely highly influential and popular figures in the music industry.

The 5 artists with the highest clustering coefficients are AC/DC (1.0), Guns N' Roses (1.0), Disturbed (1.0), Van Halen (1.0), Queen (1.0), Beastie Boys (1.0), Aerosmith (1.0), Poison (1.0), Mötley Crüe (1.0), and Twisted Sister (1.0). These artists have maximal clustering coefficients, meaning artists they connect to also connect to one another. This means that they are at the center of a dense community of artists.

With our sample properly formatted, the data was ready for further analysis.

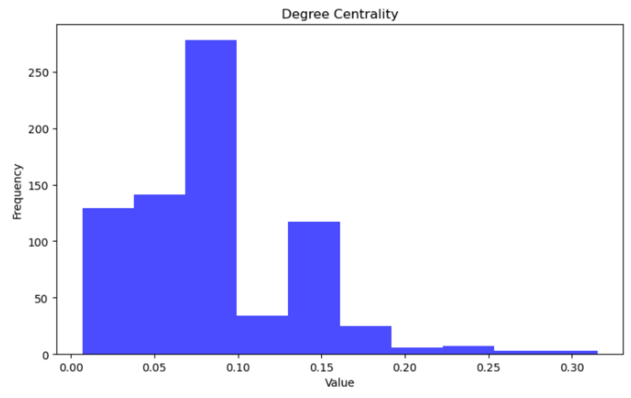
II. Algorithms

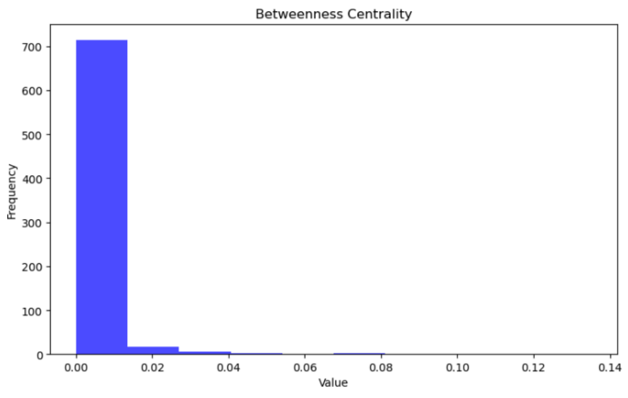
In the analysis section, we will be using Louvain to detect communities within the data set. Furthermore, we will also be using Jaccard to perform link prediction.

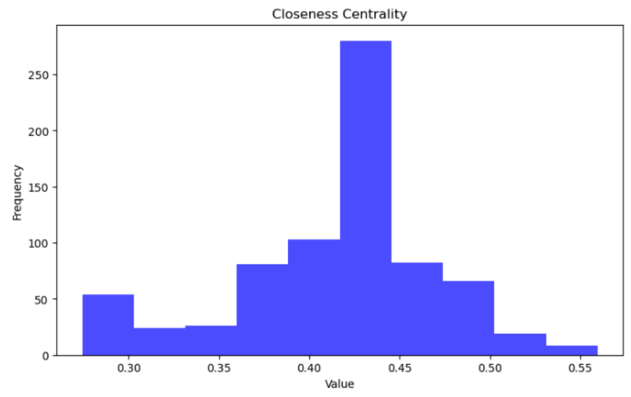
III. Experiments/Analysis

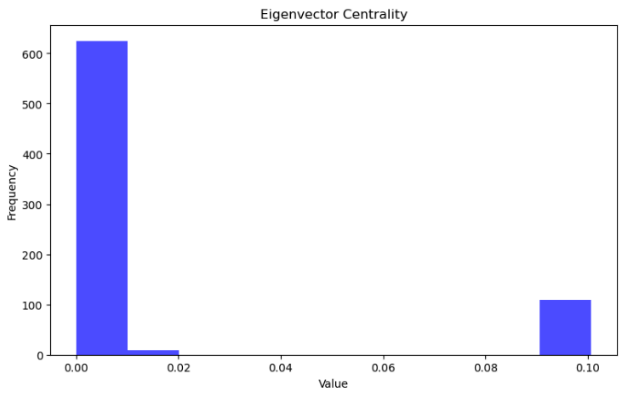
Centrality Analysis

Unweighted

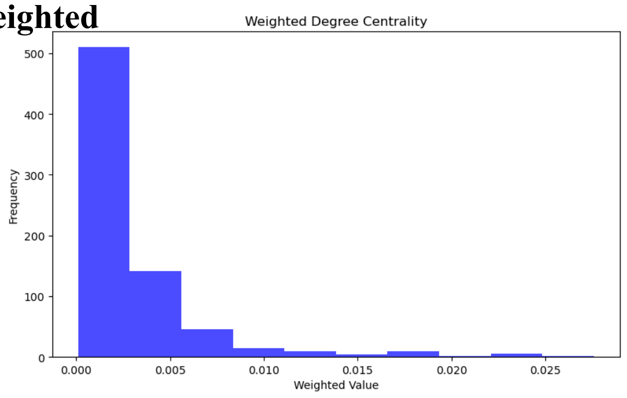


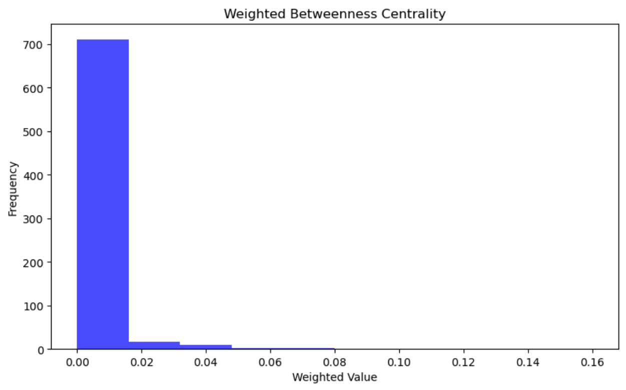


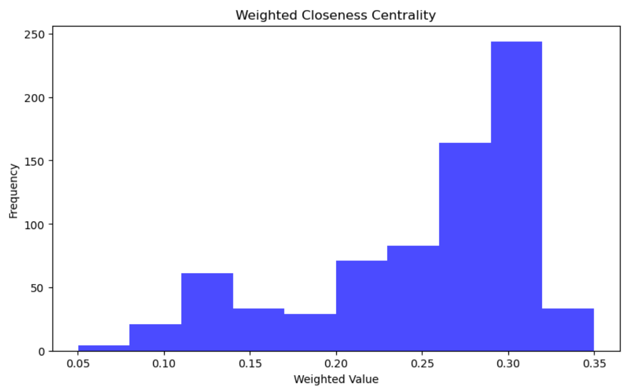


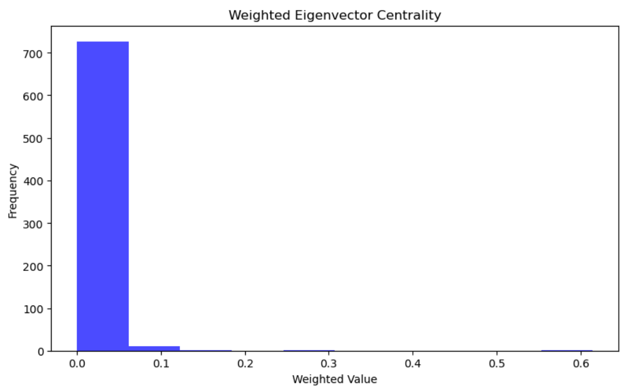


Weighted









Looking at the differences between weighted and unweighted centralities, the unweighted degree centrality is higher than the weighted degree centrality. The unweighted and weighted betweenness centrality is clustered around low values, while the weighted closeness is spread out around higher values. The weighted eigenvector centrality is stronger than unweighted, meaning that nodes with stronger connections are more influential. Generally, the unweighted centralities emphasize nodes with higher closeness and degree centrality, while the weighted centralities emphasize popular, influential artists.

Top centralities

Since the weighted centralities distributions are significantly more skewed compared to the unweighted distributions, we will continue with the unweighted centralities for further analysis. The table below shows the artists with different highest centralities. The same artists that appear across different centralities are highlighted.

Rank	Degree	Betweenness	Closeness	Eigenvector
1	Backstreet Boys: 0.31536	John Williams: 0.13514	Eminem: 0.55957	Backstreet Boys: 0.10064
2	Eminem: 0.30188	The Beatles: 0.09512	Backstreet Boys: 0.55414	Lynyrd Skynyrd: 0.10029
3	Lynyrd Skynyrd: 0.28706	Backstreet Boys: 0.07329	Ed Sheeran: 0.55044	Justin Timberlake: 0.09979
4	ASAP Rocky: 0.28301	Eminem: 0.07140	Lynyrd Skynyrd: 0.54558	Ed Sheeran: 0.09878
5	Ed Sheeran: 0.28301	Ed Sheeran: 0.07016	Eagle: 0.54121	Jason Mraz: 0.09681

Degree
Artists with high degree imply that they are universally popular, which causes them to appear on multiple playlists with various types of artists. This also implies that their music is varied and can appear in different musical communities.

Betweenness
Artists with high betweenness imply that they are popular artists that act as an intermediary between two separate music communities. They are gateway artists for people from one community to transition to another.

Closeness
Artists with high closeness imply that they are popular artists for musical discovery and exploration. These artists are close to every other playlist, allowing for a wide range of selection for people to browse.

Eigenvector
Artists with high eigenvectors imply that they are similar to other popular artists. These artists tend to have high popularity by appearing together with other top-charting artists in playlists.

We can see that across all the different centralities, Backstreet Boys appear in the top 3, with Ed Sheeran appearing similarly in all top 5's. We can infer that both these artists are popular in every sense of the word and can conclude that they are the most popular artists of the data set.

Louvain Analysis

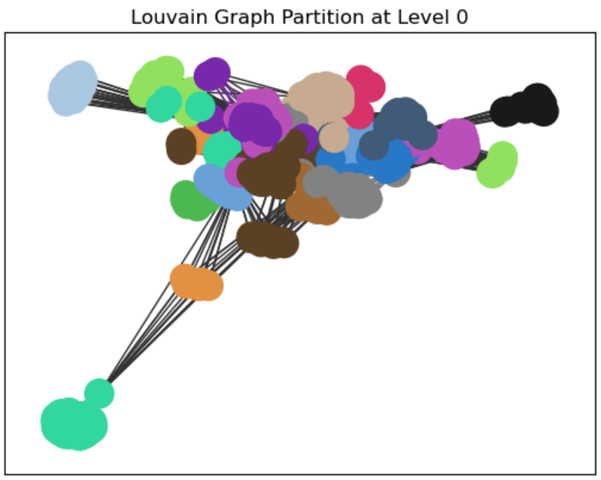
In this project, Louvain is significantly more efficient than the Girvan-Newman algorithm since the data set has a large edge to node ratio. Maximum modularity was achieved after two Louvain phases while one phase of Girvan-Newman was infeasible to calculate on a personal computer. By using Louvain to group artists that co-occur, we can find communities among the playlists and discover musical patterns across different artists.

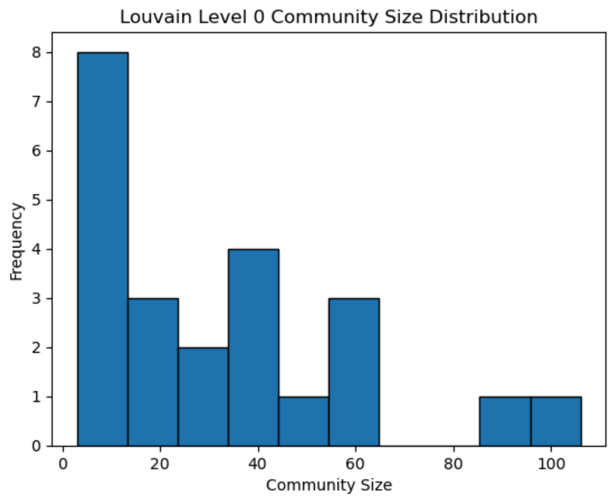
We used the default Louvain algorithm provided by NetworkX with the following parameters:

- `weight = 'weight'`
- `resolution = 1`
- `threshold = 0.0000001`
- `max_level = None`
- `seed = 123`

Iteration 1

The first figure below shows the communities discovered after the first iteration of Louvain. There are 23 communities, and the graph has a modularity score of 0.6868545445520104. The score even after one partition already indicates highly modular communities of artists

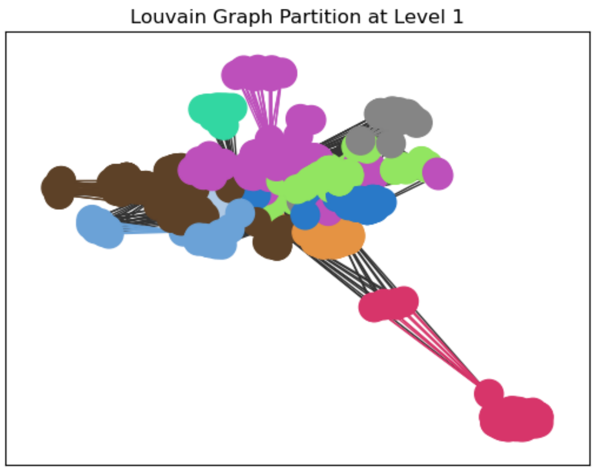


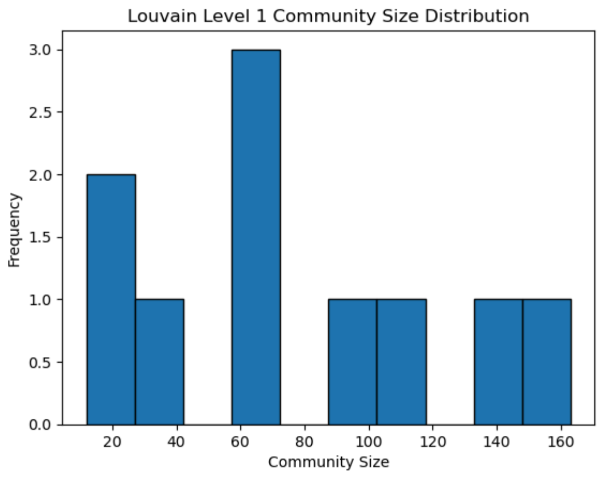


The second figure shows the distribution of community sizes in this first iteration. We can see that most of the communities have low sizes with a few large communities skewed to the right.

Iteration 2

The first figure below shows the communities discovered after the second iteration of Louvain. There are 10 communities with a modularity score of 0.7048695390169439. The score is significantly higher than the first iteration. However, since the algorithm stopped after this, it means that this is the maximum modularity score with an error of 0.0000001 based on the threshold.





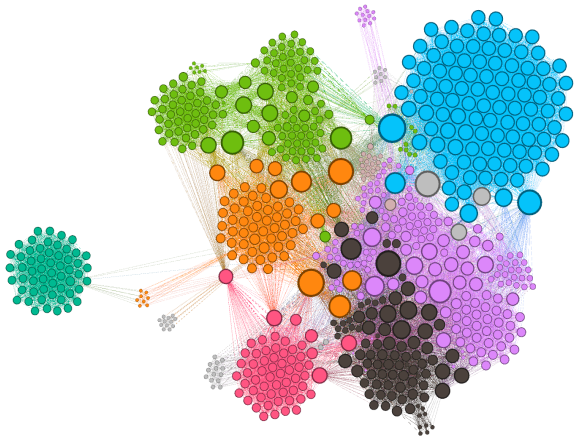
We can see in the second histogram that after reaching the maximum modularity score, the community size distribution also becomes more normalized where most communities have around 60 to 70 members.

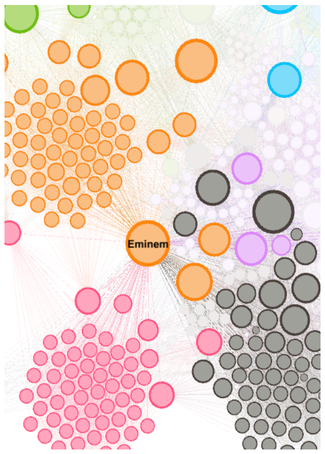
Gephi Visualization

Gephi is a graph software that aids in visualization and evaluation of graphs. It can arrange imported graphs based on Louvain communities, color code them, and display the result effectively. The first image below is one of the layout renderings of the project's data set. It uses Force Atlas 2, which is a graph layout algorithm, to group and represent the nodes. Larger nodes have greater degrees and nodes of the same communities attract each other.

The specific parameters of Force Atlas 2 used are the following:

- `Threads number = 15`
- `Tolerance = 1`
- `Approximation = 1.2`
- `Scaling = 10.0`
- `Gravity = 1.0`
- `Prevent Overlap : True`
- `Edge Weight Influence 1.0`





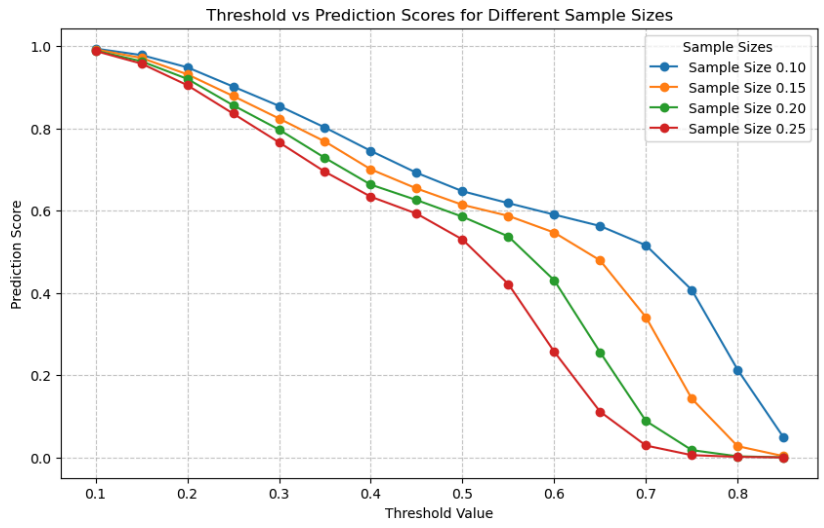
A power feature of Gephi visualization is how interactive and dynamic it is. In addition to layout algorithms, Gephi also allows manual repositioning of nodes. It also shows additional information when hovering over nodes as shown in the second image above. We configured Gephi to show the name of the artist and highlight all its neighbors. We can see how nodes that have high connectivity with other communities are placed further away from the center of their own community.These nodes tend to also be larger, whereas most of the smaller nodes are more surrounded by their own community members.

Jaccard Link Prediction

In the context of this project, we will analyze Jaccard Similarity to see how it performs for predicting whether or not two artists will appear together on a playlist.

The analysis of link prediction using Jaccard Similarity involves testing multiple sample sizes of edge removal and multiple threshold values. From the initial complete co-occurrence graph, we removed a portion of the edges ranging from 0.1 to 0.25. Afterwards, the Jaccard Similarity of each pair of nodes that the edges were removed from were calculated. These similarity values are then compared to a range of threshold values from 0.1 to 0.85. If the similarity value exceeds the threshold, then it is predicted that there will be an edge between the two nodes. The final prediction score for each threshold value is recorded by calculating the portion of how many edges it predicted correctly.

The graph below shows the prediction scores versus threshold values for each sample size. We can see that regardless of the threshold value, the smaller the sample size (the less edges we initially remove), the better the prediction score. This makes sense since there are more edges left over from the prediction model to use, making the results more accurate. Another observation is the convergence of prediction scores at each extreme of threshold values. As the threshold approaches 0.1, every sample sizes' prediction score approaches zero. Similarly, the score approaches 0 as the threshold approaches 1. The final notable observation is that most of the variance between the prediction scores of the sample sizes occur when the threshold between 0.6 and 0.8.



IV. <u>Conclusion</u>

After finding and transforming a dataset from raw Spotify CSV files to GraphML, we performed an exploratory analysis of the data by looking at centralities, top artists, clustering coefficient, and community detection. We found the network's centrality measures, music communities, and tested link prediction in our graph. This project, the results, and the analysis are repeatable and also applicable to other datasets of varying topics. For further improvements and next steps, we would expand our project to implement genres into a feature list, increase the size and variance of our sample, and create a user-friendly, interactive Gephi product.

V. References

[1]
“Spotify Network Analysis,” *GitHub*, Jan. 20, 2022.
https://github.com/rodolfostark/spotify-network-analysis (accessed Nov. 25, 2024).
[2]
J. Santos, “Spotify Network Analysis - Jonatas Santos - Medium,” *Medium*, Jan. 29, 2022.
https://medium.com/@jonatas.santos.700/spotify-network-analysis-e79acb5f8359 (accessed Nov. 26, 2024).