Spotify Artist Co-occurrence

By:

- Kyaw Soe Han - Daniel Chukhlebov -

Project Overview

This project analyzes the co-occurrence of Spotify artists across playlists to explore the relationships between artists, genres, and their influence within a greater network. We applied algorithms and centrality measures to gain insights into which artists are influential “bridges”, clustering, artists, and important connections.

Data Source

ex:

artist_name,pid

Shawn Mendes,161000

Cheat Codes,161000

Cat Power,161037

The Kooks,161037

DataSet from GitHub: rodolfostark/spotify-network-analysis

- Multiple .csv files consisting of co-occurrence entries
- Each entry connects an artist to a unique playlist…
- …by using two columns: artist_name and playlist ID
- Thereby showing co-occurrences of artists across playlists

Graph Creation

Transforming datasets into something workable...

Sample ex.

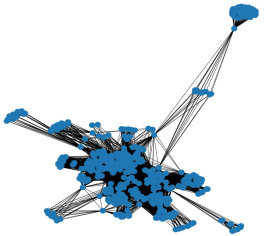
```
<node id="AC/DC"/>
<node id="Billy Joel"/>
<node id="Guns N' Roses"/>
<edge source="AC/DC"
target="Billy Joel">
  <data key="d0">45</data>
</edge>

<edge source="AC/DC"
target="Guns N' Roses">
  <data key="d0">45</data>
</edge>
```

- Initial CSV files were concatenated into a large Pandas dataframe with 269,580 entries
- Dataframe was then reformatted into a Pandas series
- Index is PID and value is a list of the artists that appear in that playlist (4,000 playlist entries)
- To reduce computational load, samples of 30 and 100 playlist files were made
- Samples were formatted to have value be a dictionary where key is artist name and value is the number of appearances in the playlist

Sample Comparison - 30 vs. 100

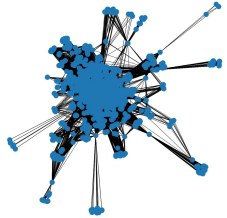
30 playlists - 743 nodes, 23508 edges

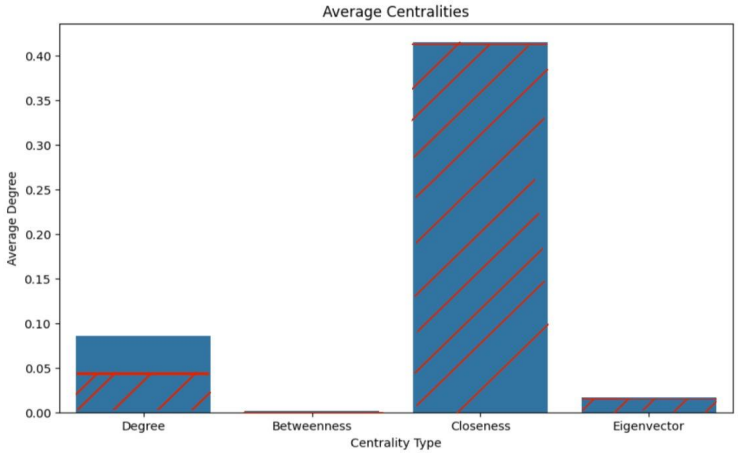
- Has smaller middle cluster
- Average Degree Centrality: 0.0853
- Average Betweenness Centrality: 0.0020
- Average Closeness Centrality: 0.4150
- Average Eigenvector Centrality: 0.0165



100 playlists - 1860 nodes, 82404 edges

- Took 5 times longer to render in NetworkX
- Average Degree Centrality: 0.0477
- Average Betweenness Centrality: 0.0008
- Average Closeness Centrality: 0.4115
- Average Eigenvector Centrality: 0.0134





Centrality Differences from 30 to 100:

Degree Centrality Difference: -0.0376 (44.8% smaller)

Betweenness Centrality Difference: -0.0012 (60% smaller)

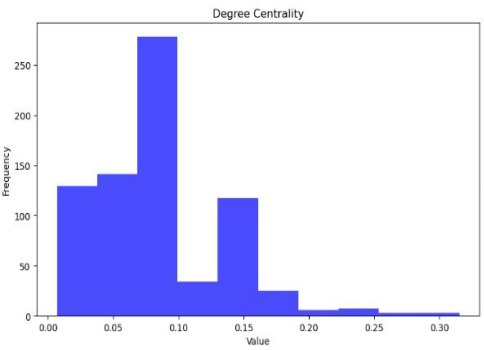
Closeness Centrality Difference: -0.0035 (0.84% smaller)

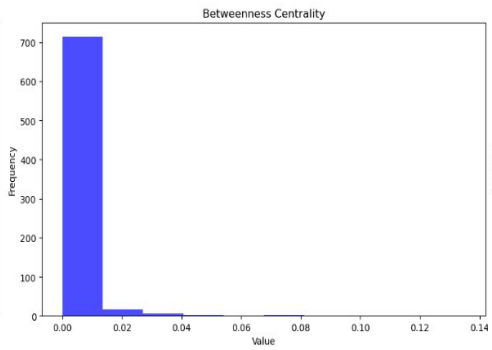
Eigenvector Centrality Difference: -0.0031 (18.79% smaller)

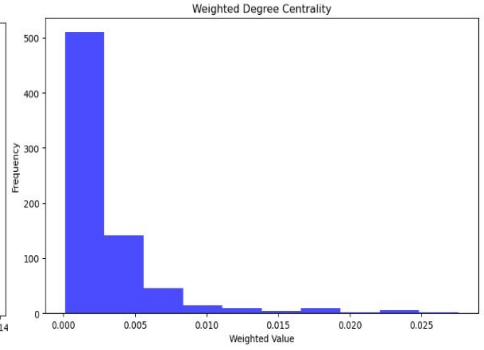
- Decreases in Degree/Betweenness centrality is expected, due to a higher number of nodes
- Closeness and Eigenvector differences are not significant, indicating similar trends between sample files

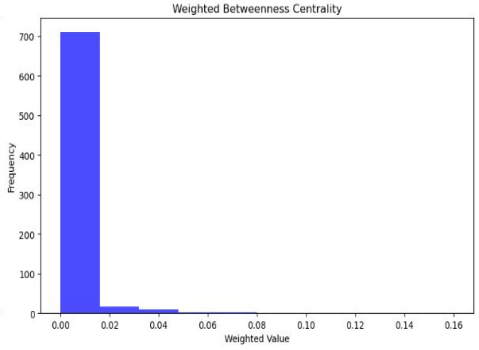
To remain computationally efficient, we used the 30 playlist sample for further analyses

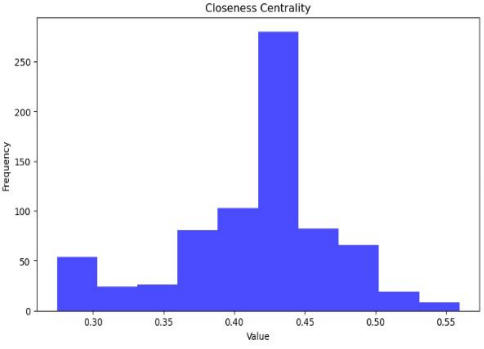
Centrality Analysis (Weighted vs Unweighted)

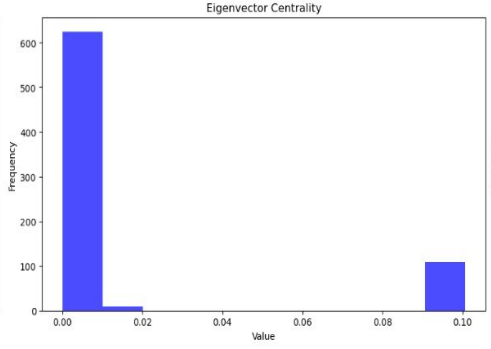


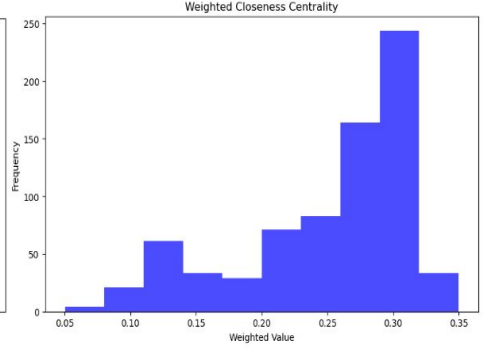


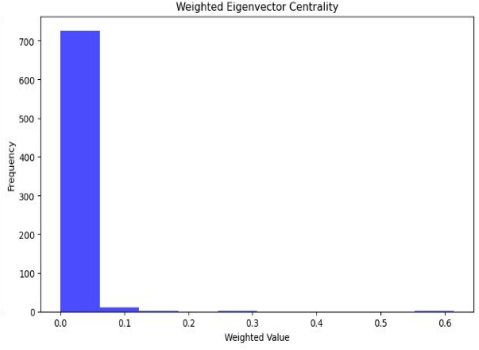






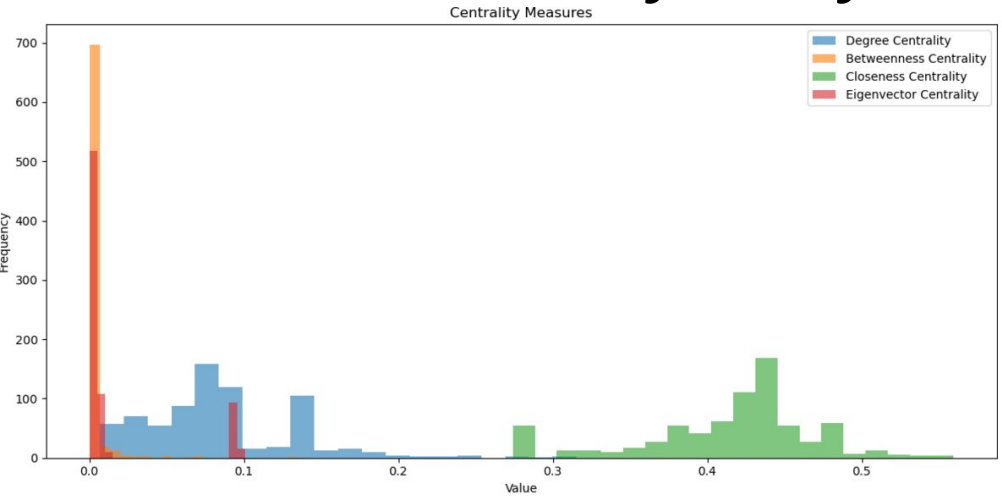


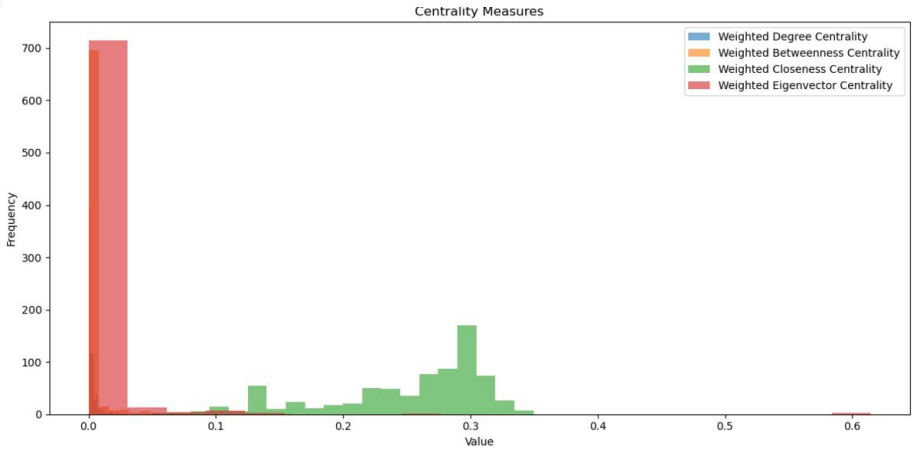




Unweighted degree centrality is relatively higher. Betweenness is concentrated around low values. Weighted closeness is distributed around higher values. Weighted eigenvector reduces domination of influential artists with many weak links.

Centrality Analysis (Weighted vs Unweighted)





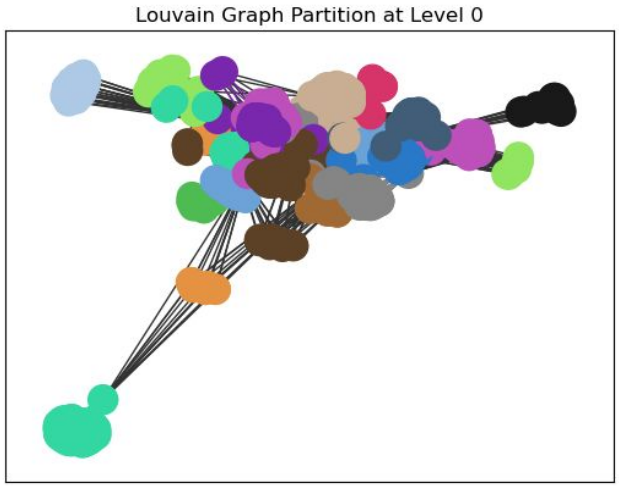
- Weighted Eigenvector centrality emphasizes strong local connections as influential
- Weighted Degree Centrality drops nodes with many weak links in favor of nodes with strong connections

Top Centralities (Unweighted)

- Only Unweighted since weighted distribution too skewed

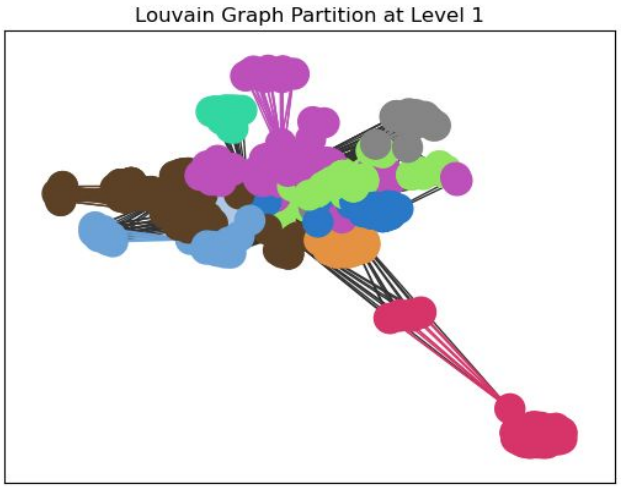
Degree	**Betweenness**	**Closeness**	**Eigenvector**	**Clustering Coefficient**
Backstreet Boys: 0.315 Universally popular, varied music across different communities	John Williams: 0.135 Popular Intermediary, Gateway artist	Eminem: 0.559 Popular artist for musical discovery and exploration	Backstreet Boys: 0.101 Appears together with other top-charting artists	Average: 0.923 Tight-knit community with other artists

Community Detection (Louvain)



# of communities: 23

modularity score: 0.6868545445520104

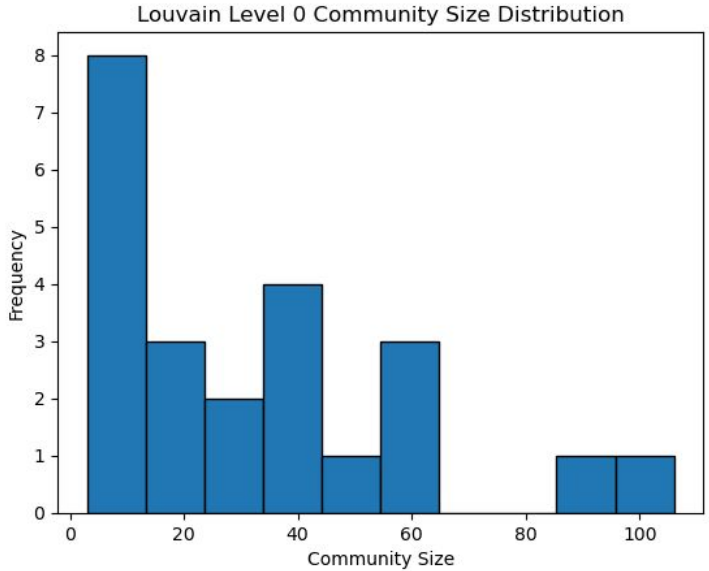


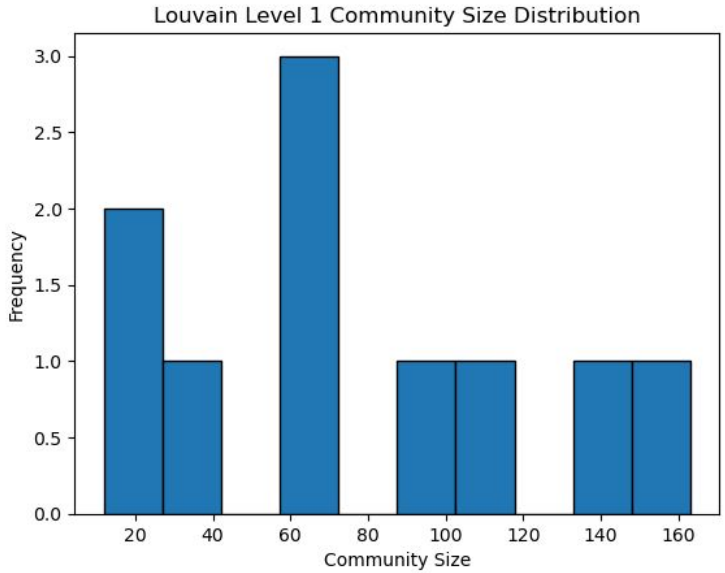
# of communities: 10

modularity score: 0.7048695390169439

- Girvan-Newman too inefficient
- Parameters
  - Resolution: 1
  - Threshold: 0.0000001
  - max_level = None
  - seed = 123
- About 70 nodes per community

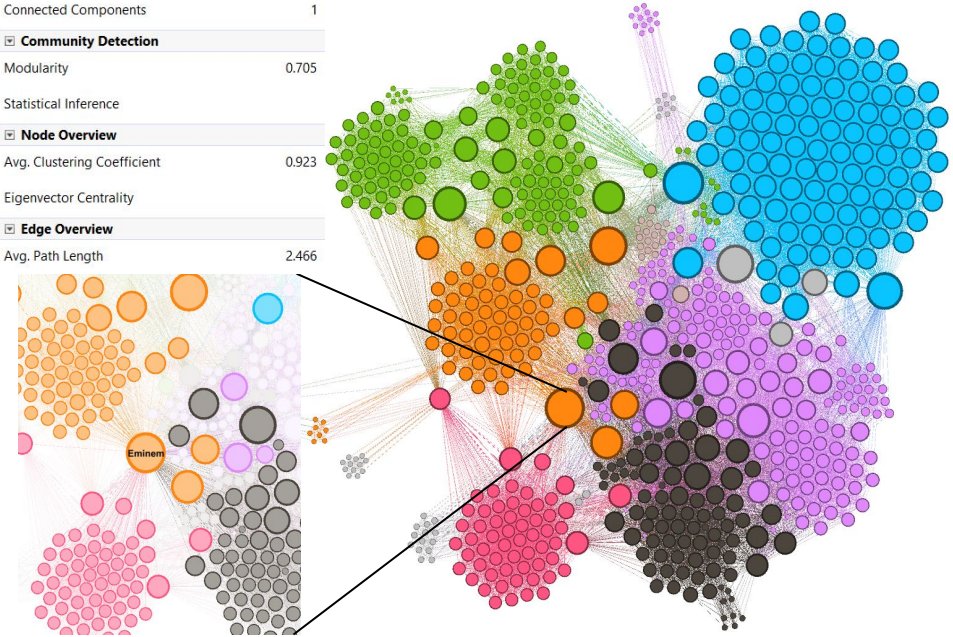
Community Detection (Louvain)





Gephi Visualization

Network Overview	
Average Degree	63.279
Avg. Weighted Degree	220.6
Network Diameter	5
Graph Density	0.085
HITS	
PageRank	
Connected Components	1
Community Detection	
Modularity	0.705
Statistical Inference	
Node Overview	
Avg. Clustering Coefficient	0.923
Eigenvector Centrality	
Edge Overview	
Avg. Path Length	2.466



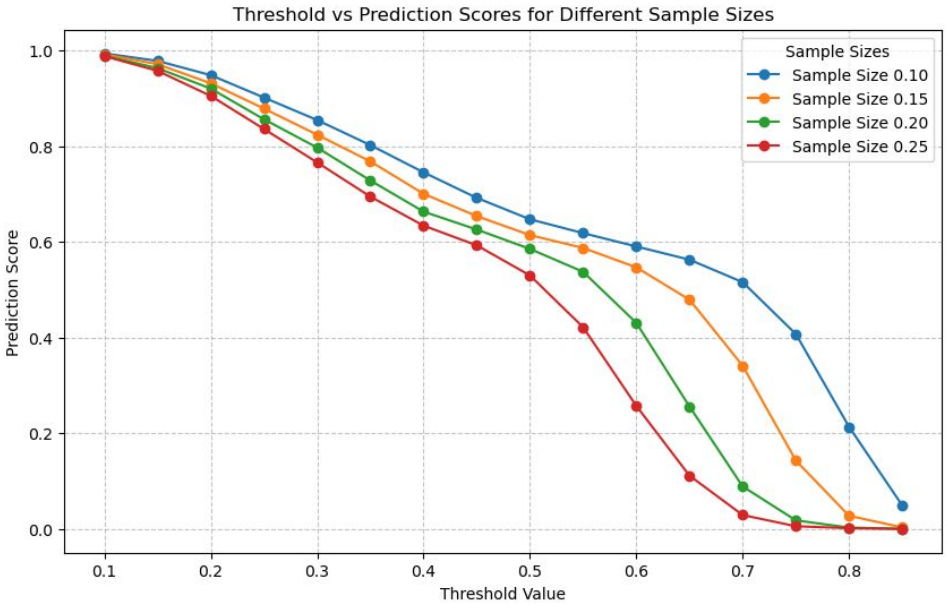
Force Atlas 2

- Each entity with gravity
- Size of node = # degrees
- Color = community

Modularity

- Louvain-based
- Same parameters
- Hover shows adjacent nodes

Link Prediction (Jaccard)



- Will they appear on a playlist?
- Sample size = ratio of edges removed
- Decline at 0.6/0.7
- Low sample size, higher scores
- Converging at ends

Conclusions

Summary

- Transformed data from raw Spotify csv to GraphML
- Performed Exploratory data analysis
- Found Communities
- Tested Link Prediction

Next Steps and Improvements

- Implement genres into feature list
- Increase sample size and variance
- Create interactive Gephi product

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).