Transparency Within Spotify: Unveiling the Constructs of Playlist Cohesion and Algorithmic Continuation

Topic 3, UG Group 42, Sutharsan Athish, Damon Jayasingha, Janghoon Jung, Junhyeok Hwang, Digital Media and Social Networks 2023/24

Abstract

Music streaming platforms like Spotify have revolutionised music consumption with their vast libraries and personalised recommendations. These recommendations' algorithms are unclear, leaving users and artists in the dark about song selection. This study examines Spotify's song popularity as it influences its playlist continuation feature decision-making. We examined song co-occurrence networks and the impact of danceability, energy, and acousticness on their popularity using network analysis and Spotify's API data on a subset of the Million Playlist Dataset. We found a scale-free network with high node role heterogeneity, indicating that a few central hubs dominate popularity while most songs are peripheral. Further analysis of song attributes suggests that certain features consistently associate with higher popularity, suggesting Spotify's recommendation algorithms may favour songs with certain traits. These findings raise the issue of algorithmic transparency and user-centricity in music streaming services. This study reveals digital music curation by examining the complex relationship between song attributes and network structure. Transparency in recommendation systems improves user experience and provides fair artist exposure. Dynamic models and music preference and algorithmic recommendation development could be added to this research.

1. Introduction

Context

The rise of music streaming platforms such as Spotify has transformed the music industry by providing unparalleled global access to extensive music libraries. An essential aspect of engaging users on these platforms involves the proficient utilisation of recommendation algorithms created specifically to curate personalised playlists. These technological advancements have revolutionised the way listeners explore music, establishing new connections between artists and audiences in unprecedented ways.

Research Problems and Motivation

Although these algorithms are effective in assisting music discovery, users often lack transparency regarding the reasoning behind song recommendations. Algorithmic transparency balances user experience and artist visibility by demystifying the recommendation process and ensuring fair exposure for artists. The lack of transparency in these algorithmic choices prompts questions regarding the specifications that control them. This study focuses on Spotify's playlist continuation feature, which is a crucial part of its recommendation system. The goal is to clarify the algorithmic decisions by analysing how certain song attributes and artist characteristics affect the popularity of songs. The primary research inquiry, "Which attributes affect the popularity of songs?" aims to reveal the fundamental mechanisms of Spotify's selection process, facilitating a greater understanding of digital music curation. This study will particularly focus on measurable characteristics such as danceability, energy, and valence, among others, to understand their impact on Spotify's song selection algorithms.

Challenges

The challenge lies in effectively navigating the intricate algorithmic frameworks and extensive datasets that are inherent to music streaming services. This study aims to use network analysis techniques and data from the Million Playlist Dataset Remastered, along with the Spotify API, to identify patterns and characteristics in song co-occurrence networks. These insights are crucial for improving music recommendation practices, emphasising the importance of making algorithms transparent and focusing on the needs of the user. Tackling these challenges is essential, not only for progressing research discussions on AI transparency in music streaming but also for establishing the foundation for fairer and more knowledgeable music recommendation systems.

2. Related Work

The algorithms powering platforms like Spotify play a pivotal role in shaping user experiences and musical discovery. As recommendation systems increasingly influence how music reaches listeners, understanding their mechanisms and biases becomes crucial. By examining these algorithms' impact on music diversity and the potential for algorithmic bias, particularly about user personality traits, we can provide a solid investigation into Spotify's playlist recommendations. We aim to highlight the existing gaps in the literature and to underscore the significance of our project in contributing to a more inclusive and user-centric understanding of music streaming services.

Recommendation Algorithms and Their Impact

The study by Silber comprehensively examines the cultural and economic impacts related to music streaming platforms, specifically focusing on Spotify and SoundCloud (Silber, J. 2019). The paper clarifies that recommendation algorithms prioritise popularity over diversity, therefore operating as filters influencing user perceptions of music. The results of Silber highlight the importance of recommendation systems that give priority to a broader range of music, hence stressing the fundamental driving force behind our project to investigate the complexities of Spotify's playlist recommendations. These biases potentially impact user satisfaction and engagement, which have a direct link to the importance of our work.

Song Popularity by Audio Features

A study by Nijkamp explored the relationship between audio features from Spotify and song popularity, which was defined by stream counts across genres (Nijkamp, R. 2018). After analysing a thousand songs, it was found that while features like key and tempo offer limited predictive power for popularity, there's significant potential to improve prediction models for song success. Despite some limitations, the research established a basis for further study into how song attributes influence popularity, with implications for consumers, record labels, and platforms like Spotify, highlighting the role of audio features in value creation within the music industry.

Predicting Popular Song Hits

In their study on Hit Song Science, they address the task of predicting Billboard hits by utilising Spotify data. Using a dataset consisting of 1.8 million songs, both hit and non-hit, the authors utilised the Spotify Web API to extract audio features and conducted experiments with various models (Middlebrook and Sheik, 2019). Their research uncovered that a random forest model achieved an accuracy rate of 88% in forecasting the success of songs on the charts, providing insight into the predictive potential of audio features for popular songs.

Dynamics of Song Popularity: Mood, Energy, and Atypicality

The study examines the intricate relationships between song characteristics and commercial success, building on audio features and song popularity (North et al., 2018). Their study of over 200,000 songs found that unique songs are more popular, suggesting that the music industry values uniqueness. The study also suggests that songs with higher energy scores are more likely to succeed commercially. Interestingly, the study finds correlations between mood scores and success rates, varying by genre. These findings advance music therapy and mental health and show the commercial potential of mood and energy attributes in music recommendations. The analysis of Spotify's algorithmic preferences shows that diverse song attributes, including those unrelated to musical genres, shape user experiences and recommendation systems.

Technical Methodologies for Playlist Analysis

Centrality measures, notably eigenvector centrality, form the foundation of our investigation framework, providing a detailed understanding of the importance of songs in playlists. The research examines the topic of "Popularity and centrality in Spotify networks" and offers useful insights into the importance of eigenvector centrality in determining the influence of songs and trends in popularity within Spotify networks (South, Rougham, and Mitchell, 2020). Given this information, our analysis seeks to utilise and expand upon these measures of centrality to develop a deeper understanding of the intricate network dynamics that influence playlist recommendations on the Spotify platform.

Critical Analysis and Connection to Our Project

While existing literature lays a comprehensive foundation on the impacts of recommendation algorithms and the role of user personality in music discovery, there is a noticeable lack of study evidence that investigates the relationship between these factors within the Spotify platform. The objective of our project is to address this discrepancy by utilising network analysis methodologies, particularly centrality measures, to analyse the composition of Spotify's playlist suggestions. The examination allows for suggesting well-informed approaches to promote transparency and diversity in the exploration of music.

3. Dataset and Network Presentation

Dataset Description

The dataset used for our analysis derives from The Million Playlist Dataset, released as an additional resource in the 2018 RecSys Challenge (Chen, 2020). A subset of playlists was selected from this dataset, and a song-song co-occurrence network was constructed. Playlists with fewer than 5 songs were excluded. The statistics referenced have been taken from the small_graph dataset, and NetworkX was our chosen tool of analysis.

Network Structure

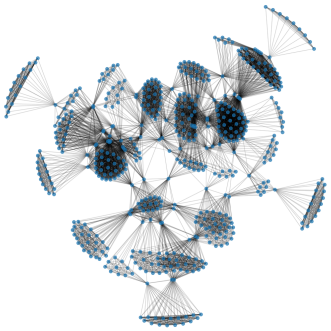


Figure 1: Network

The network comprises 537 nodes and 7715 edges; nodes depict a unique song, while edges symbolise co-occurrences. Two songs may co-occur more than once; therefore, edges possess their own weight and common_playlist property, where the sum of co-occurrences and playlist IDs can be identified.

Edge Weight and Song Pairings

The weight property provides additional information about the significance and strength of the edge. The co-occurrences between “My Way” by Calvin Harris and “Sucker for Pain” by Lil Wayne attained the highest weight, appearing 3 times together in separate playlists. Nevertheless, the prevailing agreement within the network is that most songs only occur together once, with an average weight of 1.01.

Network Analysis Metrics

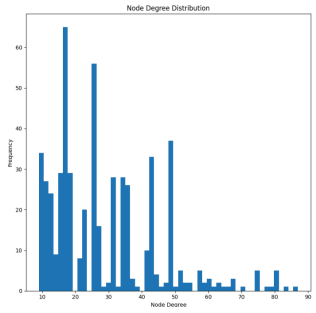


Figure 2: Degree Distribution

Degree distribution was used to identify both versatile and, conversely, niche songs. We observed a wide range of distribution (Figure 2), implying a diverse degree allocation across the network. “Take Your Time” by Sam Hunt ranked highest with 87 unique degree connections, implying high popularity and often appearing with other songs. In contrast, “Walkin’ on the Sun” by Smash Mouth ranked lowest with only 9 unique links, conveying a niche title.

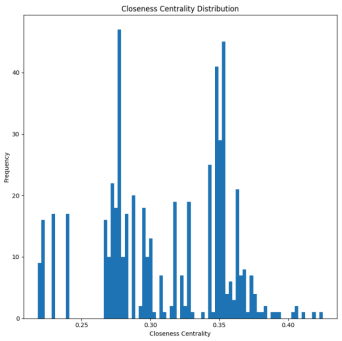


Figure 3: Closeness centrality

Measuring closeness centrality shows a song's proximity to others through playlist appearances. A high score indicates a song's versatility in pairings, but no song achieved this (Figure 3), thus suggesting weak to moderate connections between songs across playlists.

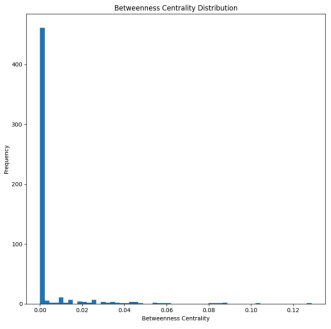


Figure 4: Betweenness centrality

Betweenness centrality highlights songs that connect song clusters, which is crucial for network cohesion. Higher betweenness indicates pivotal roles in linking clusters. However, most songs have low betweenness (Figure 4), suggesting they rarely act as bridges in playlists, meaning that a small number of specific songs may play a key role in connecting different parts.

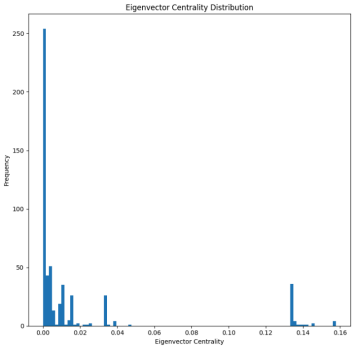


Figure 5: Eigenvector centrality

Eigenvector measures the importance of songs, tied to their connections with other influential nodes, indicating that a song's popularity is linked to how often it appears alongside other songs. However, the diagram (Figure 5) indicates low centrality for many songs, suggesting only a few hold significant influence within the network.

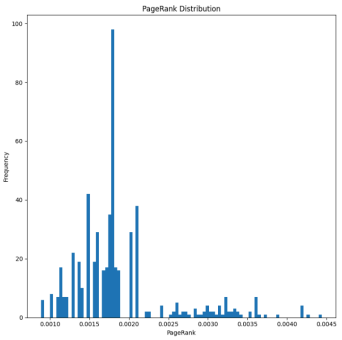


Figure 6: Page rank centrality

Pagerank, similar to the eigenvector, assesses the importance of the song within a network using different principles, factoring in not just the frequency of co-occurrences with other songs but also the importance of those songs. The histogram (Figure 6) shows a weak distribution where most songs possess minimal influence, indicating only a select few songs have high influence within the network.

Overall, the centrality measures indicate that the majority of songs seem to possess a more modest role in the network, neither central nor influential. While a few songs may appear with these traits, it is certainly not the common theme within the network.

Clustering Coefficient and Modularity

In the network, we saw a high clustering coefficient of 0.94, implying the presence of node clusters with strong local connectivity within the network. We can infer from this that meaningful patterns, song associations, and playlist diversity can be found within the network. Conversely, we found a low score for modularity with only 0.36. Modularity within the network reflects the strength of the song division into clusters; with low modularity, few distinguishable song communities can be found within the

network. These two measures point towards tight cluster groups with a tendency for genre cohesion, yet they also show a lack of distinct community segregation.

Implications and Insights

In general, these insights into the network's structure and its connections to songs provide viewpoints on how the Spotify continuation feature works. They subtly promote popular tracks while also catering to the standard range of user preferences. Our uncovering of the network sets the stage for delving into ways to enhance suggestions for a more engaging, diverse listening experience.

4. Network Analysis Methodology

Introduction to the Methodology

Our investigation focuses on the important question: "Which attributes affect the popularity of songs on Spotify?" This methodology aims to analyse the network of Spotify's playlists and assess the relationship between song attributes and their popularity. It provides a comprehensive analysis of music discovery and the user experience on the platform.

4.1 Network Structure Analysis

Objective: The initial goal of the starting point is to gain a comprehensive understanding of the network's structure by examining the connectivity between nodes. This investigation uncovers how the arrangement of connections within the network can impact the prominence and popularity of a song on Spotify.

Methodology:

- **Degree centrality** is used to measure the connectivity of songs by identifying potential indicators of popularity based on high connection counts.
- **Closeness centrality** assesses a song's capacity for interacting with or being interacted with by others, potentially correlating with its widespread acceptance.
- **Betweenness centrality** identifies important bridge songs within the network, potentially emphasising their role in diversifying user experience.
- **Eigenvector centrality** reflects the influence of a song, which can be determined by analysing the quality of its connections. It is suggested that songs that are linked to other popular songs are more likely to be popular themselves.

Justification: These measures provide insights into the structural dynamics of the network, revealing how specific structural attributes may benefit songs in Spotify's recommendations. This analysis is fundamental, establishing the framework for a more in-depth exploration of the attributes that influence the popularity of songs.

4.2 Enhancing Insights with Spotify API Data

Objective: Extending upon the examination of the song's structure, this phase incorporates song attributes obtained from the Spotify API to directly evaluate their influence on the popularity of the song. This comprehensive analysis aims to determine the specific characteristics that exhibit the highest correlation with a song's success on Spotify.

Methodology:

- Data Integration: We integrate Spotify API song attributes like acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, and valence into our network dataset. Each song node has these attributes, increasing the dataset's dimensionality and enabling multifaceted analysis.
- Attribute Analysis: Using the enriched dataset, we compare songs by centrality (degree, closeness, betweenness, and eigenvector) and attributes. Calculating the mean and standard deviation for each attribute across all songs to identify trends and identifying 'well-connected' songs based on centrality measures exceeding network averages and examining attribute profiles.
- Statistical correlation: We examine song attributes and centrality scores using statistical methods. This includes using correlation analysis to find attributes that strongly correlate with high centrality scores, which suggests a link to how popular a song is, and regression analysis to model how well song attributes can predict centrality measures. This shows how some attributes can change where a song is in the network.
- Pattern Identification: This analysis seeks to identify common attributes in well-connected, popular songs. This involves examining attribute similarities among well-connected songs to identify key features that make them popular and attribute distribution variations that may indicate which characteristics influence song recommendations.

Justification: This enhanced analysis using Spotify API data is essential for going beyond network structural insights to directly engage with song popularity attributes. By correlating song attributes with centrality measures, we can better understand Spotify song popularity and develop more nuanced, user-centric recommendation systems. This method links theoretical network analysis to Spotify's recommendation algorithms by grounding our findings in quantifiable attributes.

5. Results and Discussion

5.1 Analysis of Network Structure

Using established algorithms, our analysis produced accurate centrality measures that are crucial for understanding the architecture of the Spotify playlist

network. These metrics offer insightful data about the importance of individual songs in the network, highlighting how their connections impact their popularity and visibility on the platform.

5.1.1 Degree Distribution

The network's degree distribution, which is characterised by its long-tail skewness, indicates that the network follows a scale-free pattern. The figure in Appendix B shows a small number of songs that have a particularly high level of connectivity, which is in contrast to the majority of songs that have fewer connections. This pattern highlights the substantial variation in the popularity of songs and their inclusion in playlists, indicating the existence of a central group of tracks that dominate playlist compositions. This scale-free pattern in our degree distribution matches (Silber, J. 2019) recommendation algorithm dynamics, supporting our investigation into how Spotify's algorithmic preferences for certain songs, possibly due to their high connectivity, reflect broader trends in favouring popularity over diversity.

5.1.2 Closeness Centrality Distribution

Variability in closeness centrality scores among nodes indicates differences in the efficiency with which songs can spread to the rest of the network. A higher score indicates a song's ability to rapidly spread across the network, reflecting its potential for widespread popularity, as shown in Appendix B. The distribution of songs on Spotify exhibits a combination of highly accessible tracks as well as others with a more limited reach, showcasing the diversity in how songs flow throughout the Spotify platform. Our study shows that Spotify's song accessibility varies by closeness centrality, which is related to (Nijkamp, R. 2018) findings on audio features and song popularity. It emphasises the need for nuanced recommendation systems that account for user preferences.

5.1.3 Betweenness Centrality Distribution

The betweenness centrality scores reveal the core structure of the network, where only a few songs serve as the main connectors, as you can see in Appendix B. These songs have a vital role in connecting different clusters or genres, enabling a more extensive and diverse music discovery experience for users. Our findings on betweenness centrality, where only a few songs connect genres, support (Middlebrook and Sheikh, 2019) audio feature prediction findings. This supports the idea that certain song attributes may help Spotify discover diverse musical tastes.

5.1.4 Eigenvector Centrality Distribution

Eigenvector centrality identifies songs that have significant influence within the network because of their connections with other influential tracks, as you can see in Appendix B. The occurrence of songs with high eigenvector centrality in a context of generally low scores indicates the presence of influential 'hubs' in the network, which play a crucial role in shaping listening trends. Our eigenvector centrality analysis shows that Spotify has influential 'hubs', supporting (South, Roughan, and Mitchell, 2020) centrality-based song popularity theory. Our study of network dynamics that affect playlist recommendations shows the importance of song attributes and network positions.

5.1.5 Illustrations

The centrality distributions offer a visual and quantitative basis for understanding the structure of the network, including both general connectedness and the intricate roles that songs play in connecting the Spotify universe. The analysis conducted in layers highlights the intricate and diverse nature of the network, which mirrors the multifaceted aspects of music popularity. Our centrality measures analysis highlighted Spotify's playlist ecosystem's complex network dynamics, building on (South, Roughan, and Mitchell. 2020) work. Our analysis confirms the centrality's role in song popularity and adds to the discussion by showing how Spotify's algorithmic biases towards certain musical attributes and connectivity patterns may affect user experience and satisfaction, supporting (Silber, J. 2019) concerns about diversity in music recommendation.

5.1.6 Discussion

The analysis of the Spotify playlist network reveals significant variation, as indicated by an average degree of around 165.80, with a wide range ranging from a maximum of 776 to a minimum of 9. This discrepancy emphasises the presence of central hubs, which are songs that have extensive connections, as well as numerous peripheral or niche songs. This suggests that there are different levels of influence within the network. In addition, the clustering coefficient of 0.582 indicates a moderate level of clustering, which implies the existence of potential community structures, potentially established by genres or moods. These observations highlight the intricate nature of Spotify's music ecosystem, demonstrating a sophisticated balance in the recommendation algorithms that simultaneously promote widely popular songs and cater to individual listener preferences.

5.1.7 Limitations

An analysis of Spotify's playlist network using centrality measures has revealed significant variation in the roles of different nodes. These roles range from highly connected central hubs to less-connected peripheral connectors. Nevertheless, it is essential to acknowledge that centrality measures, although informative, may not comprehensively capture the intricacies of the network. If not properly contextualised, their sensitivity to network size and density could result in misinterpretations. In addition, the visualisations provided, while helpful in illustrating the concept, do not fully capture the complex and multifaceted nature of centrality within the network. Recognising these constraints emphasises the need for a more thorough, potentially adaptable investigation that integrates weighted associations to achieve full comprehension. These advancements have the potential to greatly influence strategies for improving networks, targeted dissemination, and overall effectiveness, leading to enhanced and user-focused music recommendation systems on Spotify.

5.2.1 Node Enrichment and Structure

With the use of Spotify's Web API, we can enrich each node with multiple attributes, enhancing the network's dimensionality. This process significantly enhances the depth of analysis that we can conduct on the network. By incorporating attributes such as danceability, energy, and loudness from the API, each node is enriched with a broad spectrum of musical features and metadata.

This enriched node information allows us to uncover patterns and insights not visible through network structure alone, assisting us to explore questions related to the influence of musical features on the connectivity of songs within the playlist.

In terms of fundamental network structure, centrality measures do not change after enriching the nodes in both large and small datasets. This was as expected, as structure and measures examine the nodes and their edges rather than the attributes of the nodes.

5.2.2 Attribute similarity amongst well-connected song nodes

By observing similarities between attributes of well-connected songs, we can identify key influential features that are important to a node's popularity; understand the consistency of attributes, enabling us to find the key driver of a song's appeal in playlists; and correlate attributes to popularity, revealing patterns within the data.

Songs with a high number of connections are considered to have a higher degree of centrality (D, Du, 2016). Utilising the average node degree within the network, we defined that any 'well-connected' song is a song whose degree is above the network average, and any 'well-connected' song is to be treated as a popular track. We chose this route as it was the fairest assessment to determine which nodes should be considered 'well connected', comparing each song against a universal metric, therefore eliminating subjective bias.

From this selected sample of nodes, the mean average of all song features related to its musical composition and their standard deviation (s.d.) were calculated. The standard deviation provides insight into how dispersed the data is in relation to its mean.

Attributes with a low deviation include 'danceability', 'acousticness', 'energy', 'instrumentalness', 'liveness', 'speechiness' and finally the' valence' score. All these features pertained deviations around 0.2, suggesting less variation and high accuracy from findings. Conversely, high deviation attributes such as 'loudness' and 'tempo' suggest that songs possessing an average far from the mean still may appear in popular songs within the network.

Attribute	**Mean**	**Standard Deviation**
danceability	0.66	0.15
acousticness	0.16	0.2
energy	0.66	0.17
instrumentalness	0.01	0.06
liveness	0.18	0.14
loudness	-6.53	2.52
speechiness	0.11	0.11
tempo	121.92	27.83
valence	0.52	0.23

Figure 11: Attribute Averages

Observing the well-connected songs on the playlist reveals major patterns; there is preference for rhythmical beat-oriented tracks with an average danceability of 0.66 and also high energy levels, which indicates an inclination for vibrant music. Despite the low mean acousticness, there is considerable variety that allows some acoustic tracks to be highlighted. Because of the lack of instrumentalness, vocal tracks dominate popular titles. What is also evident is low levels of liveness scores, meaning the majority of popular songs are studio recordings, the high loudness levels also reflect this modern production standard. While speechiness shows a broad range, hinting at the diversity in vocal content, the tempo leans towards a mid-tempo like rhythm with a large variance, possibly to accommodate the different genres involved. Valence, a measure of emotional content, indicates that both positive and negative tracks appear to be popular.

These findings are able to replicate some of Al-Beitawi's study results. They observed attributes of the top trending Spotify songs in 2017 using cluster analysis. Their results indicate attributes such as 'danceability' influenced the chart-topping success of songs and how the 'valence' attribute fluctuated across different songs (Al-Beitawi et al., 2020). However, what we found different was that our instrumentalness values were far lower than their findings in the analysis. This could be due to our song data ranging from 2010 to 2017 (Chen, 2020), compared to only just 2017 songs in their study, showing an emphasis on music trends shifting across the years.

This analysis makes it clear that song popularity in playlists is multifaceted. It emphasises that some characteristics, like danceability and energy, makes a song popular, but the mixture of different musical elements is vital for a track to resonate widely among listeners.

5.2.3 Artist Popularity

We determined artist popularity by tallying node degrees together of all songs belonging to each artist. Initially, we thought that this would be the most accurate measure of artist popularity, due to the fairness of including the whole network. Similarly, Spotify determines their artist popularity by also using the popularity of the artists' tracks (Spotify Web API, n.d.).

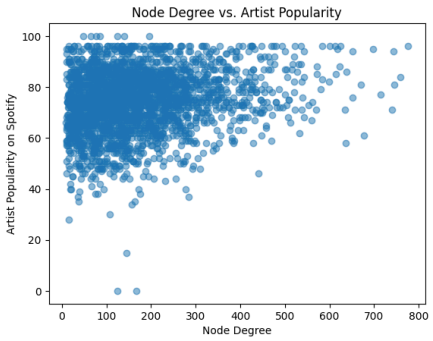


Figure 12: Artist Degree vs Popularity

Using this definition of artist popularity, we plotted this tallied artist degree against their respective 'popularity' attribute taken directly from the Spotify API (Figure 12). The scattergraph shows inconsistency between the distributions; no clear line of best fit can be drawn. Ideally, we were hoping for a positive linear relationship between the two axes; however, most plots appeared to accumulate in the top left region, indicating that low-tallied artist degrees were more popular in rankings.

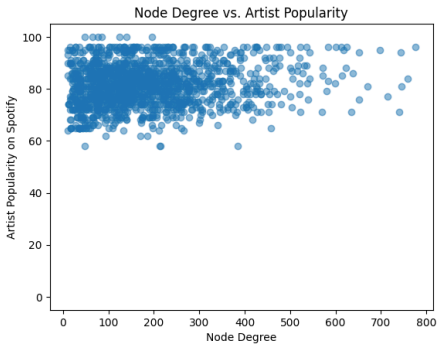


Figure 13: Artist Degree vs Popularity (excl. artists with fewer than 5 songs)

We tried another approach by removing artists that had fewer than 5 songs/nodes in the network. This was done to remove any extreme values that may have altered the appearance of the graph. However, our findings showed that the data still followed the same trend (Figure 13).

Overall, both methods of testing showed with certainty that popularity retrieved from the Spotify API is not reflected in the network. No line of best fit can be formed; one possibility for this is that Spotify may have 'overrated' certain artists, over projecting their influence.

5.2.4 Co-Occurrences of Popular Artists

Our final comparison on popularity is to look into how often popular songs co-occurred with other popular songs in comparison to non-popular songs within the network. By using a similar method that we used to define popularity, we stated that any song that has a node degree above the network average is classified as popular.

We then observed every edge in the network where both connected nodes were either both popular or both unpopular. This provided us with a direct method of comparison. For each popular or unpopular edge, we have calculated the average weight. We used the weight attribute for the measure, as this represents the strength of co-occurrences the best.

Our findings show very little difference in the average weight between unpopular and popular connected nodes. Popular scored an average of 1.3, while unpopular scored 1.05, rounded to 2 decimal points. We can determine from these results that while popular nodes do appear together in playlists more frequently, it is not a significant difference to make a clear claim.

6. Conclusion and Perspectives

Summary of Problem and Methodology

The fundamental question: "Which attributes affect the popularity of songs on Spotify?" prompted this study to investigate the intricate dynamics of Spotify song popularity. Our methodology combined network analysis of Spotify's playlist ecosystem with Spotify API data analysis of song attributes to address this. We used a song-song co-occurrence network and centrality measures to determine the structural underpinnings of song popularity. We also analysed song attributes to understand how they affect songs' network positions and popularity.

Summary of Main Results

We discovered important information about the network's structure and how song attributes affect popularity. The network analysis showed that Spotify's playlists' node roles encompassed everything from highly connected central hubs to peripheral nodes. The enrichment of nodes with Spotify API data revealed patterns and insights into how danceability, energy, and acousticness affect song popularity. Well-connected songs had distinct attribute profiles, suggesting that Spotify's recommendation algorithm may favour certain features. Understanding music recommendation systems and developing strategies that enhance music discovery and user experience demands these insights.

Perspectives

This study proposes further research and applications. Future research could explore dynamic network models and weighted connections to capture evolving music preferences and algorithmic playlist curation. Essentially, our research highlights algorithmic transparency and the need for music streaming services to balance user preferences with musical diversity. This study enables artists, record labels, and platform developers to optimise their Spotify presence and strategy by highlighting song attributes that affect popularity. A more user-centric and inclusive music streaming environment benefits listeners and encourages a diverse musical landscape.

Appendix A - Complete Task Code

https://colab.research.google.com/drive/153Z7uRVFXgCjq1Vl62Fz6xLzDdqGpQiQ?usp=sharing

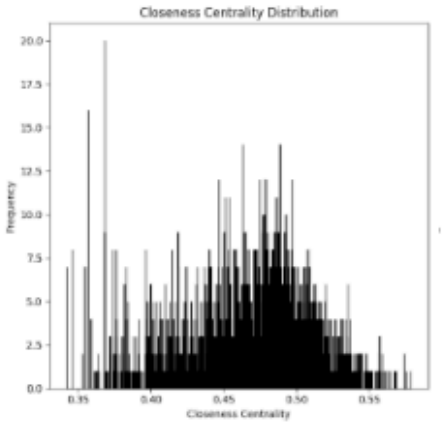
```python
# TASK 1

# load graph
with open('small_graph.pickle', 'rb') as f:
    G = pickle.load(f)
# TASK 1.A

# get nodeedge count
print(f"Number of nodes: {G.number_of_nodes()}")
print(f"Number of edges: {G.number_of_edges()}\n")

# get direction of graph
if G.is_directed():
    print("The graph is directed.")
else:
    print("The graph is undirected.")

# are componenets connected
if nx.is_connected(G):
    print("The graph is connected.\n")
else:
    print(f"The graph has {nx.number_connected_components(G)} connected components.\n")

# weight
edges_weights = nx.get_edge_attributes(G, 'weight')
if edges_weights:

    max_weight_edge = max(edges_weights, key=edges_weights.get)
    max_weight = edges_weights[max_weight_edge]
    print(f"Edge with the highest weight: {max_weight_edge} with weight {max_weight}")


    total_weight = sum(edges_weights.values())
    average_weight = total_weight / len(edges_weights)
    print(f"Average weight of the edges: {average_weight}\n")

else:
    print("The graph does not have weights.")

# clustering coefficient
avg_clustering_coefficient = nx.average_clustering(G)
print(f"Average clustering coefficient: {avg_clustering_coefficient}\n")

# node with highest degree
degree_dict = dict(G.degree())
max_degree_node = max(degree_dict, key=degree_dict.get)
print(f"Node with highest degree: {max_degree_node}, Degree: {degree_dict[max_degree_node]}")

# node with lowest degree
min_degree_node = min(degree_dict, key=degree_dict.get)
print(f"Node with lowest degree: {min_degree_node}, Degree: {degree_dict[min_degree_node]}\n")
# TASK 1.B

# figure size
plt.figure(figsize=(15, 15))

# this specific layout shows the structure the best
pos = nx.kamada_kawai_layout(G)

nx.draw_networkx_nodes(G, pos, node_size=50, alpha=0.7)
nx.draw_networkx_edges(G, pos, alpha=0.2)

plt.axis('off')
plt.show()
# TASK 1.C

# centrality measures
closeness_centrality = nx.closeness_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)
eigenvector_centrality = nx.eigenvector_centrality(G)

# node degree
node_degree = dict(G.degree())

# pageRank
page_rank = nx.pagerank(G)

def plot_centrality_histogram(centrality_values, title, x_label, subplot_index):

    # adjust for 5 graphs
    plt.subplot(1, 5, subplot_index)
    num_bins = len(set(centrality_values))
    # plot
    n, bins, patches = plt.hist(centrality_values, bins=num_bins)

    plt.title(title)
    plt.xlabel(x_label)
    plt.ylabel('Frequency')

plt.figure(figsize=(40, 8))

plot_centrality_histogram(list(node_degree.values()), 'Node Degree Distribution', 'Node Degree', 1)
plot_centrality_histogram(list(closeness_centrality.values()), 'Closeness Centrality Distribution', 'Closeness Centrality', 2)
plot_centrality_histogram(list(betweenness_centrality.values()), 'Betweenness Centrality Distribution', 'Betweenness Centrality', 3)
plot_centrality_histogram(list(eigenvector_centrality.values()), 'Eigenvector Centrality Distribution', 'Eigenvector Centrality', 4)
plot_centrality_histogram(list(page_rank.values()), 'PageRank Distribution', 'PageRank', 5)

plt.tight_layout()
plt.show()
# TASK 1.C MODULARITY
# TAKES LONG TO RUN

# community detection
communities = nx.algorithms.community.girvan_newman(G)

# communities at the highest modularity level
best_partition = next(communities)

# calc modularity based on detected communities
modularity = nx.algorithms.community.modularity(G, best_partition)

print(f"Modularity: {modularity}")
```

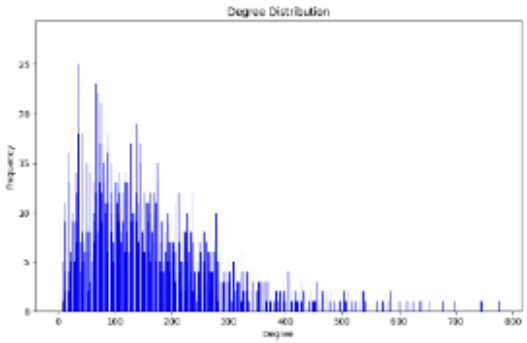
```python
# TASK 2

# Load the big graph from the pickle file
with open('bigger_graph.pickle', 'rb') as f:
    G_big = pickle.load(f)
# TASK 2.A

# node edges
print(f"Number of nodes: {G_big.number_of_nodes()}")
print(f"Number of edges: {G_big.number_of_edges()}\n")

# are componenets connected
if nx.is_connected(G_big):
    print("The graph is connected.")
else:
    print(f"The graph has {nx.number_connected_components(G_big)} connected components.")

# get direction of graphs
if G_big.is_directed():
    print("The graph is directed.")
else:
    print("The graph is undirected.")
# TASK 2.B
# TAKES LONG TO RUN

# centrality measures
closeness_centrality = nx.closeness_centrality(G_big)
betweenness_centrality = nx.betweenness_centrality(G_big)
eigenvector_centrality = nx.eigenvector_centrality(G_big)

# node degree
node_degree = dict(G_big.degree())
# pageRank
page_rank = nx.pagerank(G)

def plot_centrality_histogram(centrality_values, title, x_label, subplot_index):

    plt.subplot(1, 5, subplot_index)
    num_bins = len(set(centrality_values))
    n, bins, patches = plt.hist(centrality_values, bins=num_bins)

    plt.title(title)
    plt.xlabel(x_label)
    plt.ylabel('Frequency')

plt.figure(figsize=(24, 6))

plot_centrality_histogram(list(closeness_centrality.values()), 'Closeness Centrality Distribution', 'Closeness Centrality', 1)
plot_centrality_histogram(list(betweenness_centrality.values()), 'Betweenness Centrality Distribution', 'Betweenness Centrality', 2)
plot_centrality_histogram(list(eigenvector_centrality.values()), 'Eigenvector Centrality Distribution', 'Eigenvector Centrality', 3)
plot_centrality_histogram(list(node_degree.values()), 'Node Degree Distribution', 'Node Degree', 4)
plot_centrality_histogram(list(page_rank.values()), 'PageRank Distribution', 'PageRank', 5)

plt.tight_layout()
plt.show()
```

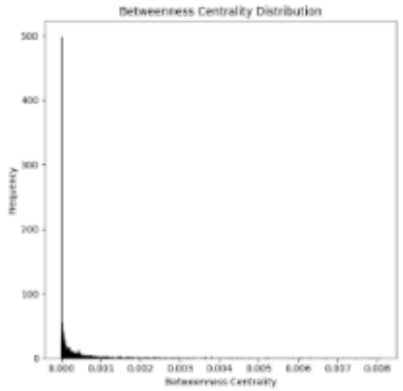
```python
# TASK 3

client_id = '8c0f4330f25e4b6a816f039713cbe823'
client_secret = 'af6ed7e296ef4d798b53ba8db04184af'

# connect to spotify
client_credentials_manager = SpotifyClientCredentials(client_id=client_id, client_secret=client_secret)
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

# TASK 3A
# ENRICHMENT

def get_audio_features_batch(track_ids):
    audio_features = sp.audio_features(track_ids)

    # filter bad responses
    return {feature['id']: feature for feature in audio_features if feature}
    time.sleep(10)

# extract just id
track_ids_to_full_ids = {node.split(":")[-1]: node for node in G_big.nodes()}
track_ids = list(track_ids_to_full_ids.keys())

# split into batches of 100 for api
chunks = [track_ids[x:x+100] for x in range(0, len(track_ids), 100)]

for chunk in chunks:
    audio_features = get_audio_features_batch(chunk)
    for track_id, features in audio_features.items():
        full_node_id = track_ids_to_full_ids[track_id]

        # update node with features
        G_big.nodes[full_node_id]['audio_features'] = features

# get first ndoe to test if working
first_node_full_id = list(G_big.nodes())[0]
print("First node audio features:", G_big.nodes[first_node_full_id].get('audio_features'))
# TASK 3B
# ENRICH FIRST

# calc average degree of the network
average_degree = np.mean([degree for node, degree in G_big.degree()])

# find well connected nodoes higher then average degree
well_connected_nodes = [node for node, degree in G_big.degree() if degree > average_degree]

# attributes to analyze
attributes = ['danceability', 'acousticness', 'energy', 'instrumentalness',
              'liveness', 'loudness', 'speechiness', 'tempo', 'valence']

# dictionary to hold averages and standard deviations for each attribute
attribute_stats = {attribute: {'average': 0, 'std_dev': 0} for attribute in attributes}

# get values for each attribute for well-connected nodes
for attribute in attributes:
    attribute_values = [G_big.nodes[node]['audio_features'][attribute] for node in well_connected_nodes
                        if 'audio_features' in G_big.nodes[node] and attribute in G_big.nodes[node]['audio_features']]

    # calc average and standard deviation for each attribute
    attribute_stats[attribute]['average'] = np.mean(attribute_values)
    attribute_stats[attribute]['std_dev'] = np.std(attribute_values)

for attribute, stats in attribute_stats.items():
    print(f"Average '{attribute}' among well-connected nodes: {stats['average']:.2f}")
    print(f"Standard deviation of '{attribute}': {stats['std_dev']:.2f}")
```

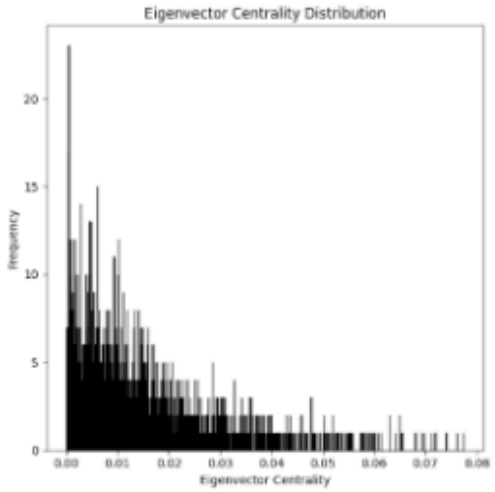
```python
# TASK 3C

# track id to get artist popularity batches of 50 @function
def fetch_artist_popularities_from_tracks(track_ids):
    tracks_info = sp.tracks(track_ids)
    artist_ids = [track['artists'][0]['id'] for track in tracks_info['tracks']]
    time.sleep(30)

    # fetch artist details in batches
    artist_popularities = {}
    for i in range(0, len(artist_ids), 50):
        artists = sp.artists(artist_ids[i:i+50])
        for artist in artists['artists']:
            artist_popularities[artist['id']] = artist['popularity']

    return [artist_popularities.get(artist_id, 0) for artist_id in artist_ids]

# spplit track IDs into batches of 50
track_ids = list(G_big.nodes())
chunks = [track_ids[x:x+50] for x in range(0, len(track_ids), 50)]

# lists to store degrees and artist popularities
degrees = []
artist_popularities = []

# iterate through chunks
for chunk in chunks:
    chunk_artist_popularities = fetch_artist_popularities_from_tracks(chunk)
    artist_popularities.extend(chunk_artist_popularities)

    # append degrees for tracks in the chunk
    degrees.extend([G_big.degree[track_id] for track_id in chunk])

degrees = np.array(degrees)
artist_popularities = np.array(artist_popularities)
correlation = np.corrcoef(degrees, artist_popularities)[0, 1]

plt.scatter(degrees, artist_popularities, alpha=0.5)
plt.title('Node Degree vs. Artist Popularity')
plt.xlabel('Node Degree')
plt.ylabel('Artist Popularity on Spotify')
plt.show()
# TASK 3D

# average degree of nodes
avg_degree = sum(dict(G_big.degree()).values()) / len(G_big)

# split nodes as popular or unpopular
popular_nodes = [node for node, degree in dict(G_big.degree()).items() if degree > avg_degree]
unpopular_nodes = [node for node, degree in dict(G_big.degree()).items() if degree <= avg_degree]

# calc average weight for edges connecting both popular nodes and both unpopular nodes
popular_edge_weights = []
unpopular_edge_weights = []

for u, v, data in G_big.edges(data=True):
    if u in popular_nodes and v in popular_nodes:
        popular_edge_weights.append(data['weight'])
    elif u in unpopular_nodes and v in unpopular_nodes:
        unpopular_edge_weights.append(data['weight'])

# calc average weights
avg_weight_popular = sum(popular_edge_weights) / len(popular_edge_weights) if popular_edge_weights else 0
avg_weight_unpopular = sum(unpopular_edge_weights) / len(unpopular_edge_weights) if unpopular_edge_weights else 0

print("Average weight for edges connecting popular nodes:", avg_weight_popular)
print("Average weight for edges connecting unpopular nodes:", avg_weight_unpopular)
```

Appendix B - Analysis of Network Structure









References

[1] Silber, J. (2019) Music Recommendation Algorithms: Discovering Weekly or Discovering Weakly?. MediArXiv. doi: 10.33767/ http://osf.io/6nqyf.

[2] Nijkamp, R. (2018). Prediction of product success: explaining song popularity by audio features from Spotify data. [online] Available at: https://www.semanticscholar.org/paper/Prediction-of-product-success%3A-explaining-song-by-Nijkamp/ba06fe3e0b65e799fcc7b1434dac387f986da1ed.

[3] Middlebrook, K. and Sheik, K. (2019). SONG HIT PREDICTION: PREDICTING BILLBOARD HITS USING SPOTIFY DATA A PREPRINT. [online] Available at: https://arxiv.org/pdf/1908.08609.pdf.

[4] North, A.C., Krause, A.E., Sheridan, L. and Ritchie, D. (2018). Popularity, Mood, Energy, and Typicality in Music: A Computerized Analysis of 204,506 Pieces. [online] ResearchGate. Available at: https://www.researchgate.net/publication/324488378_Popularity_Mood_Energy_and_Typicality_in_Music_A_Computerized_Analysis_of_204506_Pieces.

[5] South, T., Roughan, M. and Mitchell, L. (2020). Popularity and centrality in Spotify networks: critical transitions in eigenvector centrality. Journal of Complex Networks, [online] 8(6). doi:https://doi.org/10.1093/comnet/cnaa050.

[6] Spotify. (n.d.). Spotify Web API Available at:https://developer.spotify.com/documentation/web-api/reference/get-an-artist.

[7] Al-Beitawi, Z., Salehan, M., Pomona, C. and Zhang, S. (n.d.). Cluster Analysis of Musical Attributes for Top Trending Songs. [online] Available at: https://scholarspace.manoa.hawaii.edu/server/api/core/bitstreams/77da75c1-4993-4b34-a867-ad5ec69bc362/content.

[8] Spotify Engineering and Spotify Engineering (2020). The Million Playlist Dataset... Remastered - Spotify Research. [online] Spotify Research. Available at: https://research.atspotify.com/2020/09/the-million-playlist-dataset-remastered/.

Notes:
Junhyeok was assigned to do section 5 but sustained health problems and had to abandon the group.