Network-Driven Approach to Music Recommendation using Spotify Data

Kristian Kieffer[a], **Bridget Bidwell**[a], **Izzy Zoeller**[a], **and Megan Mitchell**[a]

This manuscript was compiled on August 5, 2025

This project introduces a method of music recommendation, through the use of common network analysis techniques. We use a Spotify data set containing a list of songs from the 1920s to the 2020s, paired with a variety of extracted audio features and descriptors. We construct a network with songs as nodes, connected by edges if they share similar audio feature values. After constructing the network, we calculate and use a variety of network analysis techniques, such as cosine and Jaccard similarity as well as community detection algorithms. By analyzing the structure and connectivity of the graph, we aim to identify songs that are acoustically similar to a given input. Our goal is to explore how simple network calculations can identify song similarity and how they can be used for music recommendation. We find that while community detection does not necessarily recover genre, decreasing the size of the output communities generates clusters of similar songs that can be recommended as playlists. In addition, cosine and Jaccard similarity have the power to suggest songs that are highly similar to a given input.

Song recommendation | Community detection | Node similarity

Significance Statement

Music preference is highly subjective, which makes song recommendation a difficult task. Many algorithms might build on user data to suggest songs based on similar listening history. However, using network science, we have developed a method to recommend songs based on audio features alone, which has the potential to generate songs recommendations that are not influenced by popularity or the listening patterns of other users.

Ever since the advent of music streaming platforms such as Spotify, Apple Music, and Pandora, essentially every song that has ever been released can be accessed and listened to at the click of a button. Spotify alone has millions of tracks that are available to its users. With such an abundance of music, listeners might find it helpful to turn to computational methods to discover new songs, albums, or artists; in fact, many music streaming platforms have their own algorithms for recommending songs to their users. However, this huge database of music has also garnered the attention of network scientists in recent years. Researchers have used networks to answer questions about music listening habits, song similarity, genre, and more.

There are a variety of ways to represent music data in a network. One 2021 study interested in classifying the nature of collaboration between artists modeled this data using a bipartite network, where edges were used to connect artists and the songs to which they contributed (1). The nodes of the one-mode projection of this network each represent an individual artist, and the edges between them represent the number of tracks they collaborated on. By using standard network metrics, such as eigenvector centrality, the authors were able to discover valuable insights about artist popularity and influence. Another strategy is to represent listeners, rather than artists, as nodes; one study in 2022 investigated the structure of sub-genres within metal music by constructing a bipartite graph between listeners and music genres using data from online reviews (2). The one-mode projection onto genres within this network were then used to identify common-interest communities within a larger subset of music. While representing nodes as artists or listeners may be useful to identify major influences or community substructures, if the goal is to recommend songs similar to a certain input, we propose that the most straightforward method is to represent songs as nodes. By collecting data on song features and drawing edges between nodes that share similar features, we can use standard network techniques to identify song suggestions.

This strategy is supported by existing research regarding music networks. Donker (2019) analyzes Spotify's "related artist" feature through the use of centrality metrics such as closeness and betweenness to identify influential artists and genre-spanning connectors (3). The author's work illustrates how artist popularity data can reflect meaningful structural patterns, especially when using network theory to draw conclusions about artist influence and genre connectivity, and to ultimately inform song recommendation. Magalhães Bush (2025) uses an even broader approach, modeling Spotify's collaboration network at a large scale to evaluate its small-world properties (4). By comparing the network's clustering

Author affiliations: [a]UCLA Department of Mathematics, University of California, Los Angeles, CA 90095, USA

coefficient and diameter to those of random and lattice graphs while also applying community detection methods like Louvain modularity, the author demonstrates that artist collaborations lead to the formation of dense genre-based clusters. Together, these findings support our approach by demonstrating that both centrality measures and community structure are valid tools in analyzing song similarity networks.

These prior studies offer valuable insight into how musical relationships can be modeled and analyzed through network science. Research has shown that different network properties—such as centrality, clustering, and modularity—can be used to detect influence, reveal genre structure, or suggest points of connection between artists. Inspired by these approaches, our project aims to construct a song-level similarity network using various audio features, such as tempo, energy, key, and even "danceability." (5) We connect songs based on their similarity across the selected categories and then use similarity metrics and clustering techniques to find songs to recommend based on network structure. Our overall goal is to expand upon insights from existing research on musical networks in order to reveal both structural and perceptual relationships between songs in our dataset, and to use these findings to develop a song recommendation strategy. We find that community detection is a useful method to generate "playlists" of similar songs, while cosine and Jaccard similarity are valuable methods of finding song suggestions based on a specific input.

Background

Once we constructed our network, where each song is a node and similar songs are connected by edges, our goal was to identify songs to recommend given an input node. Network science offers many useful tools for detecting similarity between nodes; we tested several methods, including cosine similarity and Jaccard similarity.

Cosine similarity and Jaccard similarity build upon a simple metric: the number of shared neighbors between two nodes. Suppose we have some network with adjacency matrix A. To count the number of common neighbors between nodes i and j, we can use the following formula:

$$n_{ij} = \sum_k A_{ik}A_{kj} = (A^2)_{ij} \quad [1]$$

where n_{ij} is the number of common neighbors between nodes i and j.

Cosine similarity treats each node as a vector, and is computed using the equation

$$\sigma_{ij} = \frac{n_{ij}}{\sqrt{k_i k_j}} \quad [2]$$

where n_{ij} is the number of common neighbors as computed above, and k_i, k_j are the degrees of nodes i and j, respectively. If either node has degree 0, we set $\sigma_{ij} = 0$.

Another approach is to divide the number of shared neighbors by the total number of distinct neighbors of either i or j. This metric is called the Jaccard coefficient, and it can be calculated using

$$J_{ij} = \frac{n_{ij}}{k_i + k_j - n_{ij}} \quad [3]$$

where again n_{ij} is the number of common neighbors between i and j and k_i, k_j are the degrees of nodes i and j.

However, each of these metrics only consider nodes to be similar if they directly share neighbors. But it might be the case that two nodes that are from the same group might have no direct neighbors in common, but have neighbors that are themselves similar. Katz similarity accounts for such indirect connections using the following formula:

$$\sigma_{ij} = \alpha \sum_k A_{ik}\sigma_{kj} + \delta_{ij} \quad [4]$$

where $\alpha > 0$ is some constant and δ_{ij} is added to force nodes to have a high similarity to themselves. In matrix form, this is

$$\boldsymbol{\sigma} = (I - \alpha A)^{-1} \quad [5]$$

where $\boldsymbol{\sigma}$ is the $n \times n$ similarity matrix and I is the identity matrix. To ensure that the matrix is invertible, choose $0 < \alpha < \frac{1}{\lambda_1}$ where λ_1 is the largest eigenvalue of the adjacency matrix.

Another method we used to identify groups of similar songs was Louvain community detection. We used the `louvain_communities` function from NetworkX, an algorithm that sorts nodes into groups to maximize the similarity within each group. Let $g_i \in [1, \ldots, G]$ be the group assigned to node i. We can quantify the assortativity of a network given some grouping of the nodes by computing its modularity, Q, using the following formula:

$$Q = \frac{1}{2m}\sum_{i=1}^{n}\sum_{j=1}^{n}(A_{ij} - \frac{k_i k_j}{2m})\delta_{g_i g_j} \quad [6]$$

where m is the total number of edges in the network, n is the total number of nodes in the network A is the adjacency matrix, g_i and g_j are the groups of nodes i, j, and k_i and k_j are the respective degrees of nodes i and j.

The Louvain community detection algorithm initially sets each node to be in its own group, and iteratively swaps nodes into the community that maximizes the change in modularity that results from the move. The change in modularity can be computed using

$$\Delta Q = \frac{k_{i,in}}{2n} - \gamma\frac{\sum_{tot} \cdot k_i}{2m^2} \quad [7]$$

where m is the number of edges in the graph, k_i is the sum of edge weights incident to node i, $k_{i,in}$ is sum of edge weights connecting node i to other nodes in the given community, and $\sum_{tot}$ is the sum of all edges weights incident to nodes in the given community, and γ is the resolution parameter. Higher values of γ favor smaller communities, and larger values of γ favor larger communities.

After all nodes are initially moved into the community that maximizes modularity, a new network is constructed where each community itself becomes a node, and edges are drawn between communities with weight equal to the sum of the edge weights between nodes in one community to nodes in the other community. The algorithm proceeds in this manner until there is no more positive gain in modularity.

Methods and Models

The overall goal of our study was to generate song recommendations based on the structure of the song-to-song similarity network. We do this in two main ways. The first strategy we tried was to use Louvain community detection to identify similar clusters of songs that could be recommended together as a "playlist." The second was to use network similarity metrics to find a list of song recommendations based on a specific input song. However, our first challenge was to construct the network itself.

Dataset.

Creating the Similarity Measure. To build such a network, we must first determine what it means for two songs to be "similar." Once similarity has been quantified, we can consider how to construct the network itself.

Initially, we thought it would be sufficient to connect two songs if at least a minimum number of their metric features fell within a fixed threshold. For instance, if song A had more than x features within 10 % of the corresponding features of song B, we would draw an edge between them. This approach, however, has several problems, because not all metrics should be treated equally. For example, *liveness* measures how "live" a song sounds, as opposed to being recorded in a studio. Although this is useful, the overwhelming majority of songs in our data were recorded in a studio, so *liveness* is far less informative than a song's *tempo* or *acousticness*. We therefore decided that the procedure must incorporate adjustable weights.

Since a simple feature–count threshold is inadequate, we instead compute a separate similarity score for each numerical feature and then weight those scores by their relative importance. Because each feature behaves differently, each requires its own similarity function. Some features lie on the interval $[0, 1]$, whereas others, such as tempo, are unbounded. We began by designing a similarity function for the $[0, 1]$ features—*danceability, energy, valence, acousticness, instrumentalness, liveness*, and *speechiness*. For these metrics we required a function that

- decays smoothly and monotonically,
- passes through $(0, 1)$ and $(1, 0)$, so that a metric difference of 1 yields a similarity of 0.

We discovered that the power function in [8], where m_i is the metric score for song i, satisfies these criteria. As the exponent k increases, differences in the metric are penalized more severely.

$$\mathrm{sim}_m(i, j) = (1 - |m_i - m_j|)^k \qquad [8]$$

Because the typical spacing of values differs from feature to feature, each feature needs its own k. For example, differences in *speechiness* are typically on the order of 10^{-2}, so a difference of 0.1 is very large; by contrast, *acousticness* often approaches 1, so a difference of 0.1 is hardly noticeable. To estimate appropriate k values we performed a simple statistical analysis. We drew a random sample of roughly 3,000 songs and computed all $\binom{3000}{2}$ pairwise differences in the feature of interest. We then found the median of these differences, denoted δ. Solving

$$(1 - \delta)^k = 0.5 \quad \Rightarrow \quad k = \frac{\ln(0.5)}{\ln(1 - \delta)} \qquad [9]$$

gives the exponent that assigns a similarity of 0.5 to the median difference. In effect, a small δ leads to a large k, making the function steeper. This process produced the following k values: `'danceability': 4.116, 'energy': 2.251, 'valence': 2.302, 'acousticness': 1.592, 'instrumentalness': 87.918, 'liveness': 6.377, 'speechiness': 39.488`.

Metrics not confined to $[0, 1]$—namely *tempo, loudness*, and *year*—require a different strategy. Our initial attempt involved a rescaling to $[0, 1]$ via [Eq. x] . However, this fails because the maxima of these variables are unstable and their distribution varies.

$$X = \frac{x - min(x)}{max(x) - min(x)} \qquad [10]$$

Instead, we defined a function that takes the raw difference and maps it to $[0, 1]$. For the *year* difference we needed a function that

- satisfies $f(0) = 1$, so identical release years receive the maximum similarity
- decays non-linearly, so even a 100-year gap still yields a non-negligible score.

The decaying exponential, with i and j being songs, found in [9] (illustrated in [Figure 1]) met these requirements. We found that $\tau_{\mathrm{year}} = 55$ gave the most balanced results. With this value, a 40 year difference corresponds to a 0.5 similarity score.

$$\mathrm{sim}_{\mathrm{year}}(i, j) = \exp\left(-\frac{|\mathrm{year}_i - \mathrm{year}_j|}{\tau_{\mathrm{year}}}\right) \qquad [11]$$

$$\mathrm{sim}_{\mathrm{loudness}}(i, j) = \exp\left(-\frac{|\mathrm{loudness}_i - \mathrm{loudness}_j|}{\tau_{\mathrm{loudness}}}\right) \qquad [12]$$

We applied the same exponential form to *loudness* and *tempo*, estimating their τ values by the same sampling procedure: for each metric we computed all pairwise differences in the 3,000-song sample, took the median δ_{median}, and solved

$$e^{-\delta_{\mathrm{median}}/\tau} = 0.5 \quad \Longrightarrow \quad \tau = \frac{\delta_{\mathrm{median}}}{\ln 2}. \qquad [13]$$

From the data we obtained a median tempo difference of 28.5 BPM, giving

$$\tau_{\mathrm{tempo}} = \frac{28.5}{\ln 2} \approx 41.1,$$

and a median loudness difference of 4.5 dB, giving

$$\tau_{\mathrm{loudness}} = \frac{4.5}{\ln 2} \approx 6.5.$$

As before, a pair whose difference equals the median receives a similarity of 0.5.

For the categorical *genre* feature we used a simple binary match: 1 if the genres are identical, 0 otherwise. Although including this term improved the apparent accuracy of our song-matching tests, we ultimately set its weight to zero when constructing the network, because we evaluate community-detection performance by its ability to recover genres. We

likewise excluded *popularity* because Spotify's raw listener counts do not correct for population growth, platform reach, or technological change.

We intentionally excluded *key, mode, explicit*, and *duration* from the similarity calculation. The feature *key* (tonic and scale) is difficult to compare meaningfully from song to song and is partially captured by composite metrics such as *energy*. *Mode* and *explicit* are binary; the information in *mode* overlaps strongly with *valence*, and *explicit* is largely irrelevant to musical similarity. The same reasoning applies to *duration*.

Having computed per-feature similarities, we formed an overall similarity as a weighted linear combination. The weights, which sum to 1, reflect our judgment of each feature's importance. After dozens of iterations and listening tests, we settled on the weights shown in [Figure 2]. To evaluate "accuracy" informally we

- ran the algorithm on a diverse set of artists whose discographies are considered stylistically consistent by Gemini 2.5 Pro (e.g., Frank Sinatra, Miles Davis, AC/DC, The Ramones, Mozart, Bob Marley);
- checked whether the top-K list for a random song included other songs by the same artist or at least within the same genre. If it consistently did, the weighting was considered satisfactory; if not, we revisited the weights or investigated possible data issues.

The final similarity formula appears in [10], where i and j are the two songs being compared, w is the weight corresponding to the metric, and m corresponds to the metrics on the scale $[0, 1]$. Our program retrieves the k most similar songs to a given input song.

$$\begin{aligned}\mathrm{Sim}(i,j) =& w_{\text{year}} \exp\left(-\frac{|y_i - y_j|}{\tau_{\text{year}}}\right) \\ &+ w_{\text{tempo}} \exp\left(-\frac{|t_i - t_j|}{\tau_{\text{tempo}}}\right) \\ &+ w_{\text{loudness}} \exp\left(-\frac{|l_i - l_j|}{\tau_{\text{loudness}}}\right) \\ &+ \sum_m w_m (1 - |m_i - m_j|)^{k_m}\end{aligned} \tag{14}$$

Network Construction. With the similarity measure in place, our next step was to construct the undirected similarity graph. We drew an edge between two songs whenever their similarity exceeded a threshold x. Because evaluating all $\binom{100\,000}{2}$ pairs is computationally prohibitive, we worked with a stratified random sample of 5,000–7,000 songs.

To obtain the sample, we bucketed songs by genre and decade and selected each song with probability proportional to its bucket's share of the full 98 000-song database. Scaling these probabilities by the desired sample size (here 7,000) yielded the target counts per bucket.

We then computed the weighted-sum similarity for every pair in the sample and retained edges whose score exceeded 0.92. Higher thresholds produced graphs that were too sparse; lower thresholds produced graphs that were too dense and impaired genre recovery. At a threshold of 0.85 the resulting graph contained roughly 229,917 edges; the remaining $\binom{7000}{2} - 229,917$ pairs fell below the cutoff. We saved the edge list as a CSV, and imported it into NetworkX to construct the graph.

Community Detection for Playlist Generation. Once the graph was constructed, we used the `louvain_communities` module from NetworkX to identify clusters of similar songs. After running the algorithm with a resolution parameter of 1, we found that the algorithm identified ten very large groups of songs that shared similar audio features, each with hundreds of nodes. Our initial goal was to use community detection to potentially recover song genre using network structure alone. However, by tuning the resolution parameter, we were able to generate much smaller communities that shared similar features that could be recommended together as a playlist. We tested a wide range of resolution parameter (γ) values, and found that using a resolution of 30 produced the most communities in the ideal range of 10 to 20 songs each.

Similarity Metrics for Song Recommendation. In addition to playlist generation, we used network similarity tools to suggest songs based on an input. We defined functions to compute the cosine and Jaccard similarity between a node and every other node in the network. Given a certain input song, we sorted the similarities from high to low and output the top five most similar songs to the input as our recommendations. Because our network was constructed such that similar songs should be either direct neighbors or share direct neighbors, we opted to use cosine and Jaccard similarity rather than Katz similarity, which considers indirect connections that are not applicable to our particular network.

Adapting the Similarity Measure for Validation. While the results of our community detection and similarity functions were promising, song recommendations are inherently difficult to validate, as their success relies on the preference of the individual listener. Therefore, we explored a non-graph based approach to compare with the recommendations produced by our network-driven algorithm.

Instead of applying a threshold and using the resulting edge list for clustering, we maintained the full continuous similarity scores between songs, using the similarity vector computed between a query track and the remaining songs in the dataset. This further allowed us to directly compare individual songs by bypassing the need to discretize relationships between nodes into edges, instead treating them as feature vectors.

We reused the same similarity formation as previously introduced in our Creating the Similarity Measure subsection: exponential decay for year, tempo, and loudness; power-law similarity for normalized continuous features; and a weighted sum of all per-feature similarities. By eliminating the hard cutoff used in the edge list construction, this method produced a continuous similarity distribution that captured each song's relative affinity to the input track. This distinction lowered the risk of information loss due to arbitrarily determined thresholds, instead capturing fine distinctions in similarity that would have otherwise been overlooked.

The result of the algorithm is a ranked list of the K most similar tracks to a given query, ordered by total similarity score. Significantly, since no graph structure has been physically created, we were able to treat each song as a singular point in high-dimensional feature space,

with each point having its own set of normalized audio and metadata features. In turn, this enabled us to apply standard vector-space metrics for measuring song proximity, offering a more nuanced perspective on musical affinity. Songs that may not have surpassed the threshold in the graph-based method could now present as near neighbors in the ranked output, highlighting slight similarities that were previously disregarded.

Moreover, retaining the complete similarity distribution, we were able to compare relative distances across multiple query songs, encouraging a more flexible exploration of musical neighborhoods. Comparisons were therefore based not only on absolute similarity to a singular reference point, but also across clusters of similar tracks or broader trends. We found this particularly helpful when validating the recommendations generated by our graph-based approach.

Data and Code. The dataset we used for this study is publicly available on Kaggle. All code used for our data analysis can be found here.

Results and Analysis

As can be seen in Figure 1, the resulting network is approximately power-law distributed. This means that there are many songs with very low degree, and a few songs with high degree.

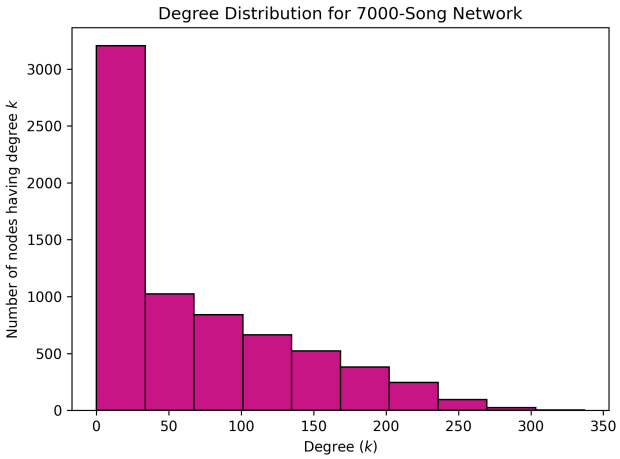

Fig. 1. Degree distribution of network with 7,000 nodes each representing a song. Songs were connected by an edge if the similarity between them exceeded 0.85.

After performing Louvain community detection with a resolution of 1, we identified 10 main communities. The modularity of the network given the partitions predicted by the algorithm was 0.605, which suggests that the network exhibits a high degree of homophily; this is a good sign, because the network was constructed such that similar nodes should be more likely to be connected. There were several smaller communities that contained less than ten nodes, but we excluded them from our analysis because they contain little meaningful information about clustering and network structure. We then computed the average value of each audio feature for each of the largest communities. As can be seen in Figure 2, each community has a very different audio profile. For example, Community G has very high

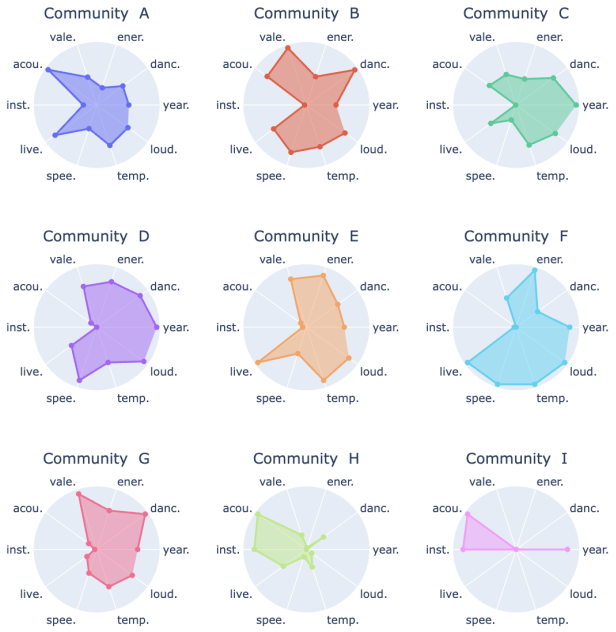

Fig. 2. Average audio features for each of the nine largest communities identified using Louvain community detection.

values of *valence*, *energy*, and *danceability*, while Community I has very low levels of these features but very high values for *acousticness* and *instrumentalness*. From these audio descriptions alone, we might expect Community G to contain happy, upbeat songs, whereas Community I might contain more instrumental or classical music. As can be seen in Figure 3, Community G is primarily composed of songs from the R&B and Pop/Rock genres, including upbeat tracks from Fleetwood Mac, Elton John, Tina Turner, and more. Community I, on the other hand, contains songs from genres including Stage & Screen, Pop/Rock, and Classical genre; however, most of these "Pop" songs in Community I are in fact instrumental movie scores, which suggests that our Louvain community detection algorithm was able to differentiate between songs with different audio features even though they were technically classified in the same genre.

Several of the communities, including C, D, E, and F, are comprised mostly of Pop/Rock songs. However, looking at the artist with the most songs in each community reveals that a very different type of Pop/Rock has been identified in each cluster. We see in Table 1 that Jimmy Buffet, a well-known country artist active in the 1970s, is the top artist for Community C, whereas Taylor Swift, who bridges Country and Pop music is the top artist for Community D. The Rolling Stones, a rock band active from the 1960s, are the most prominent in Community E. So although each of these communities are made of mostly 'Pop/Rock' artists, community detection was able to separate this broad category into more specific groups.

While larger communities help us identify broad trends, we also wanted to hone in on more specific groupings. To focus more closely on the effects of community detection, we ran the Louvain algorithm again with a higher resolution of 30. This allowed us to find smaller communities that could be used to create playlists. After performing Louvain

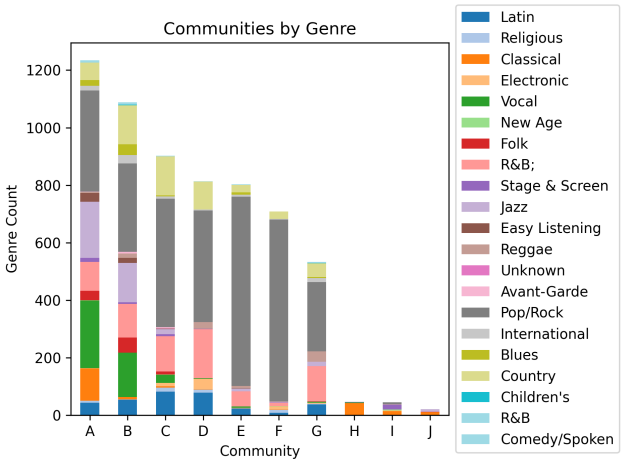

Fig. 3. Genre breakdown of each of the main communities identified by Louvain community detection. Only communities with ten or more nodes are included.

Table 1. Top artist in each community

Community	Top Artist	Number of Songs
A	Wolfgang Amadeus Mozart	31
B	Billie Holiday	29
C	Jimmy Buffett	11
D	Taylor Swift	10
E	The Rolling Stones	18
F	Green Day	13
G	Bob Marley & the Wailers	10
H	Vladimir Horowitz	29
I	John Williams	6
J	Glenn Gould	11

community detection with this higher resolution and filtering for communities that contained between 10 and 20 songs, we extracted 116 viable groups. From these, our goal was to recommend a community as a playlist based on some chosen audio features. For example, to identify the best community for a party playlist, we created a simple party score metric by averaging the normalized values of features associated with upbeat music: *danceability*, *energy*, *valence*, and *loudness*. This score provided a rough measure of how "party-like" each community was. We then selected the community with the highest party score as the most suitable candidate for a party playlist. We repeated this process to create a chill playlist, a sad playlist, and a study playlist, using different audio features. The results can be seen below:

Table 2. Five Songs from the Dance Playlist

Song	Artist	Genre
Hard Times	Paramore	Pop/Rock
Dynamite	Taio Cruz	Pop/Rock
Suavemente	Elvis Crespo	Latin
Troublemaker (feat. Flo Rida)	Olly Murs	Pop/Rock
In The Summertime	Shaggy	Reggae

We also used cosine and Jaccard similarity measures to recommend songs based on a given input. Rather than

Table 3. Five Songs from the Study & Chill Playlist

Song	Artist	Genre
Home From The Forest	Gordon Lightfoot	Folk
Big Deal	Charlie Ventura	Jazz
Prelude and Fughetta in G Major	Glenn Gould	Classical
Lookin' Good	Magic Sam	Blues
Jump, Lester, Jump	Lester Young	Jazz

Table 4. Five Songs from the Sad Playlist

Song	Artist	Genre
That's When Your Heartaches Begin	Elvis Presley	Pop/Rock
Imagination	Ella Fitzgerald	Vocal
Only Trust Your Heart - Remastered	Dean Martin	Vocal
When I Fall In Love - Live	Sam Cooke	R&B
The Nearness of You	The Four Freshmen	Vocal

generate clusters of similar songs, these similarity metrics used underlying network structure to choose nodes that have the smallest so-called "distance" from a given input. For nearly every input song we tested, cosine and Jaccard similarity generated the same recommendations. Many of the recommendations given by these simple similarity measures, which are based on the number of direct neighbors between given nodes, generated recommendations that were relevant to the given input.

Table 5. Top five most similar songs to 'Cornelia Street' by Taylor Swift using Jaccard similarity.

Song	Artist	Genre
Never Seen Anything "Quite Like You"	The Script	Pop/Rock
Beautiful People Beautiful Problems	Lana Del Rey	Pop/Rock
It's Your Love	Tim McGraw	Country
Meet Me in the Hallway	Harry Styles	Pop/Rock
Si Me Tenías	Mijares	Latin

For example, as can be seen in Table 5 the top five recommendations for the input song from the input song 'Cornelia Street' by Taylor Swift using Jaccard similarity included songs by Lana del Rey, Harry Styles, and Tim McGraw. While these might not be the most accurate recommendations, given that the network contains a limited random sample of 7,000 songs and no information about artist or popularity, they seem to capture some degree of relatedness.

We also analyzed Jaccard similarity for "All of You" by Miles Davis, shown in Table 6. The recommendations returned a coherent set of tracks that included jazz and jazz-adjacent artists such as Cannonball Adderley and Ben E. King, alongside vocal and international tracks with compatible acoustic profiles.

Cosine similarity, applied to normalized feature vectors, consistently returned recommendations that aligned closely in both acoustic character and artist style, as well as overall production features such as tempo, energy, etc. In many cases, these recommendations overlapped with those generated by

Table 6. Top five most similar songs to 'All of You' by Miles Davis using Jaccard similarity.

Song	Artist	Genre
Sweet Sue	Miles Davis	Jazz
If You Only Knew	Little Jimmy Scott	Vocal
Carribean Cutie (Alternate)	Cannonball Adderley	Jazz
Tujhse Nar	Lata Mangeshkar	International
Stand by Me	Ben E. King	R&B

the Jaccard similarity measure, suggesting that structurally close nodes also often shared feature-based proximity.

For example, when using cosine similarity to generate recommendations based on a Miles Davis song, the top five hits included another Miles Davis song, as well a song by another saxophonist Cannonball Adderley. The rest of the recommendations, listed in Table 7, included instrumental Jazz and R&B tracks that seem to be relevant to the input song.

Table 7. Top five most similar songs to 'All of You' by Miles Davis using cosine similarity.

Song	Artist	Genre
Sweet Sue, Just You - first version	Miles Davis	Jazz
Stand by Me	Ben E. King	R&B
If You Only Knew	Little Jimmy Scott	Vocal
Carribean Cutie (Alternate)	Cannonball Adderley	Jazz
I Mean You (Take 4)	Thelonious Monk	Jazz

The recommendations for "All of You" by Miles Davis generated by Jaccard similarity in Table 6 and those that were returned by cosine similarity in Table 7 show significant overlap. Both lists include "Sweet Sue" by Miles Davis, "Carribean Cutie (Alternate)" by Cannonball Adderlay, and "If You Only Knew" by Little Jimmy Scott, which further illustrates strong agreement between the two similarity measures. The consistency across these methods suggests that, for well-connected nodes like those associated with these queries, both structural proximity in the network and feature-based similarity emphasize the same or similar neighboring tracks. This agreement supports the idea that cosine and Jaccard similarity capture complementary yet validating notions of relatedness within our graph.

Conclusions and Discussion

We introduced a method of song similarity detection and recommendation that is based solely on audio features instead of popularity, listening patterns, or genre. Our method was able to identify songs that exhibited similar features across genres, which may be useful to help recommend songs to listeners who are looking to explore new music outside of their favorite genre.

In addition, we implemented our own algorithm for network construction using a weighted function of multiple metrics between songs. Previous work in the field constructed networks by drawing an edge between a song or artist that shared a single feature: for example, appearing in the same playlist or having collaborated on the same album. However, by combining multiple audio features and changing the parameters of the similarity detection algorithm, we were able to uncover patterns in network connectivity that were not obvious based on a single audio features alone. While this custom similarity algorithm was fine-tuned for our music dataset, it can be extrapolated beyond song networks to potentially help create networks out of high-dimensional data with no simple method for edge construction.

We also used both community detection and multiple similarity metrics to understand the structure of the network and generate song recommendations. While our community detection did not appear to primarily sort songs by genre, it revealed groups of songs that were similar due to their energy and acoustic features, regardless of artist, popularity, or decade.

However, there were several limitations to our study. The data set we used had a large amount of artists under the Pop/Rock genre – about 2,800. In contrast, other genres such as Classical or Electronic have around 30 and 200 respectively. The lumping of the Pop and Rock genres together is a large oversimplification, as these genres are both large on their own, and distinct. This does not make genre a very strong indicator of similarity, since so many songs fall under this one category.

Future work might consider consulting with experts in the music industry to break up these broad genres into more specific categories to aid the in the validation of our music recommendation system.

Another limitation of our research was the small size of our sample. Because network construction required pairwise comparison of similarity between every node, we took a stratified random sample of the whole dataset to make the problem more computationally tractable. However, this excluded a significant number of songs that might have been better matches for a given input. Further research could be done to create a more computationally efficient method of network construction that allows every song in the dataset to be represented.

References

1. T South, M Roughan, L Mitchell, Popularity and centrality in spotify networks: critical transitions in eigenvector centrality. *J. Complex Networks* **8**, cnaa050 (2021).
2. Z Jiang, HN Huynh, Unveiling music genre structure through common-interest communities. *Soc. Netw. Analysis Min.* **12**, 35 (2022).
3. S Donker, Networking data: A network analysis of spotify's socio-technical related-artist network. *Int. J. Music. Bus. Res.* **8**, 67–101 (2019).
4. RA Magalhães Bush, Analysis of a spotify collaboration network for small-world properties (2025) arXiv preprint.
5. V Mavani, Music recommendation system using spotify dataset (2022).
6. A Gopnik, The pure artistry of frank sinatra. *The New Yorker* (2023) Accessed: 2025-06-04.

Supplementary

While the main goal of our project was to recommend songs, we explored other network-based tools to better understand the structure of the network, including betweenness and Katz centrality. We hypothesized that analyzing betweenness centrality might reveal "genre-bending" songs that could be good recommendations for fans of a particular genre looking to branch out in their exploration of music.

As can be seen in Table 8, the song in the network with the highest betweenness centrality is "Back on My Feet Again" by The Babys. The song did not immediately appear to be genre-bending; in fact, upon investigation of its audio

Song	Artist	Genre
Back on My Feet Again	The Babys	Pop/Rock
The Days Gone By	Eddy Arnold	Country
I Gotta Right To Sing The Blues	Julie London	Vocal
The Sound of Silence - Electric Version	Simon & Garfunkel	Pop/Rock
I Saw the Light Remaster	Todd Rundgren	Pop/Rock

Table 8. Top five songs by betweenness centrality.

features, we found that it had almost exactly the mean value for the Pop/Rock genre for nearly every audio feature. This song may have a high betweenness centrality because it is likely to be connected to many other Pop/Rock songs, which make up the majority of the dataset. The other songs with high betweenness centrality show more direct influence of multiple genres; 'The Sound of Silence - Electric Verison' bridges the Pop/Rock and Electronic Genres, and 'I Gotta Right To Sing the Blues' has both Vocal and Jazz influences. Todd Rundgren making the list is no surpise - he is known for his range of styles from rock to pop to blues.

Additionally, we also wanted to analyze the Katz centralities of our network. With our constructed network, we hypothesized that songs with high Katz centrality could be songs that sit in central positions among influential tracks, making them potentially influential or trend-setting in terms of musical features or style.

Song	Artist	Genre
Don'cha Go 'Way Mad	Frank Sinatra	Vocal
How Do You Like Me Now?!	Toby Keith	Country
Luka	Suzanne Vega	Pop/Rock
Need Your Loving Tonight	Queen	Pop/Rock
You're Lost Little Girl	The Doors	Pop/Rock

Table 9. Top five songs by Katz centrality.

As can be seen in Table 9, the song in the network with the highest Katz centrality is "Don'cha Go 'Way Mad" by Frank Sinatra. Frank Sinatra being the artist of the song with the highest Katz centrality makes sense; Sinatra released a substantial body of work and was known for bringing swing and jazz vocals into the pop mainstream (6). His music may share audio characteristics that recur across many other songs in the network, both within and beyond his genre.

Interestingly, "Luka" by Suzanne Vega has a high Katz centrality despite Vega not being as widely recognized as the other artists listed. This suggests that the song may sit in a structurally important position in the network, connecting multiple influential or stylistically similar clusters. Its production style or audio features may align closely with broader trends in the network, meaning it might share many features with songs that are influential without directly being influential itself. While this might be interesting to note, it is not incredibly revealing in the overall picture of our network.

1. Author Contributions

Kristian: Performed necessary work to define the similarity measure, create the network using a minimum threshold connection strategy, and form the Louvain communities for analysis. Wrote the sections 'Creating the Similarity Measure' and 'Network Construction'.

Bridget: Analyzed the network using Louvain community detection, using Louvain to extract and curate playlists. Additionally, performed supplemental analysis using Katz centrality.

Megan: Created figures to visualize the network and communities. Helped perform similarity analysis using cosine and Jaccard similarity and supplemental analysis using betweenness centrality.

Izzy: Helped to conduct similarity analysis using cosine and Jaccard similarity, as well as Katz. Created tables for analysis and helped to write the abstract and introduction sections, as well as some of the analytical and background information.