Dylan Mohsen

DS 210

5 May 2024

Final Project: Influence in Top Spotify Songs

My project looks into the centrality and influence of songs within Spotify's overall most popular songs. Each song is a node and they are connected on a weighted undirected graph based on the differences in their peak chart positions, stream counts, and days on the charts. Stronger connections between songs are represented by higher weights which correspond to less weighted edges in terms of their impact on connectivity. Lower weighted edges imply a stronger connection between two songs, having less difference in performance. It identifies key songs based on their connectivity and influence over the network, reflecting potential trends and preferences among Spotify users. The analysis is structured into three primary modules: graph.rs, song.rs, and main.rs.

Module Overview

graph.rs

This module defines the Graph struct which models the Spotify song network using adjacency lists:

add_vertex(song) adds a new song to the graph.

add_weighted_edge_by_features(src, dest) adds edges between songs based on shared features, adjusting weights according to differences in peak position, stream counts, and days on chart. This method models the connections between songs, simulating how one song might influence another within the platform's listening patterns.

dijkstra(start_vertex) implements Dijkstra's algorithm to find the shortest paths from a start vertex to all other vertices, which helps in calculating the centrality measures.

closeness_centrality() computes the closeness centrality for each song, identifying how close/similar a song is to all other songs in the network

print_most_central_for_depth() outputs the songs with the highest centrality scores, outputting the most influential songs at different levels of network depth.

song.rs

This module declares the struct for a song with its values, as well as handles the loading and parsing of song data from CSV files:

load_songs_from_csv(file_path) reads the dataset from a specified file and deserializes each row into a Song struct, which includes details like artist name, song name, days on chart, peak position, and total streams.

main.rs

In this module the song data is loaded from the CSV file, it also calls the function to build the graph based on the loaded song data. Finally it calls print_most_central_for_depth to display the most influential songs based on the calculated centrality measures.

Tests

test_graph_construction_with_random_subset() tests the graph's ability to handle and accurately reflect the structure of a subset of data.

test_dijkstra_algorithm() ensures that the shortest path calculations are correct.

test_closeness_centrality() verifies the accuracy of the centrality computation.

test_graph_connectivity() checks if the graph correctly identifies as connected or disconnected based on the input data.

Since this dataset has around 11,000 songs, running it on the entire dataset took me an entire day and I was only actually only able to run it fully twice. Because of this, in my test cases I have made it so it randomly picks 1000 songs and tests it on this smaller subset, as to prevent only picking songs in higher position that would negatively affect the centrality results. Although the results may differ in value based on the songs randomly chosen, after running it multiple times I did consistently observe the same trends (a slight decrease in centrality that eventually plateaus). This can be tweaked as pleased to have more or less vertices and this makes it much more manageable to run, running in less than 30 seconds. If you want to run it with the entire data test, you can simply write cargo run –release in the terminal, however to do it with the test cases cargo test works.

These results can be useful in understanding song dynamics on Spotify, indicating how certain tracks maintain broad appeal and connectivity across various different users. This analysis can aid Spotify in marketing, playlist curation, and user engagement strategies by focusing on songs that effectively pull listeners through their networked relationships.

Output:

```
Depth 1: Song: No Role Modelz, Artist: J. Cole , Closeness: 37.16

Depth 2: Song: goosebumps, Artist: Travis Scott , Closeness: 30.70

Depth 3: Song: XO TOUR Llif3, Artist: Lil Uzi Vert , Closeness: 30.66

Depth 4: Song: Lucid Dreams, Artist: Juice WRLD , Closeness: 29.66

Depth 5: Song: Sunflower  SpiderMan: Into the SpiderVerse, Artist: Post Malone ,
Closeness: 28.10

Depth 6: Song: Jocelyn Flores, Artist: XXXTENTACION , Closeness: 27.09
```

Analysis:

As we go deeper depths into the network, we observe a slight decrease in the closeness centrality of the most central songs, although the differences in centrality scores become progressively smaller at each deeper depth. Starting at Depth 1, "No Role Modelz" by J. Cole stands out with the highest centrality, indicating it has strong direct connections to other songs, showing a significant immediate appeal and popularity. As we explore further into Depths 2 through 4, the centralities of songs like "goosebumps" by Travis Scott and "XO TOUR Llif3" by Lil Uzi Vert decrease slightly, reflecting a slightly less direct but still prominent network connections. By Depths 5 and 6, as seen in "Sunflower" by Post Malone and "Jocelyn Flores" by XXXTENTACION, the centrality scores begin to stabilize and have little variance. This stabilization suggests that these songs maintain consistent connectivity and popularity within the network. Despite not always topping the charts, these songs exhibit long-term influence and a stable presence, underscoring their importance in the network beyond chart success. This pattern highlights their potential as core and iconic songs on Spotify to remain popular and prominent for years to come, and could be analyzed in creating new songs that attempt to recreate their performances, having long term popularity.