CSC111 Project Proposal: Interactive Music Genre and Album Recommendation Tree

David Wu, Kevin Hu

Tuesday, March 16, 2021

Problem Description and Research Question

To many of us, music is a large part of our lives, helping us get through the day. According to the International Federation of the Phonographic Industry (IFPI), the average person spent 18.4 hours a week listening to music in 2021. With the sheer number of music available on modern streaming platforms such as Spotify or Apple Music, it can be extremely difficult and time consuming for the casual listener to find new music that they enjoy, especially since listeners themselves may not know what they like. A listener may know that they typically enjoy Rock music but "Rock" is a broad term that encompasses many distinct subgenres, perhaps they are into the raw, distorted sound of Noise Rock or maybe they prefer the more technical, complicated time signatures of Math Rock. This is not to mention that different albums within a subgenre could also be further categorized by different descriptors such as warm, aggressive or depressing. The casual listener is simply unaware of these distinctions when trying to find something they like on the Rock section of Spotify. We believe that good music recommendations should be fast and easy, **our goal is to provide users with the best possible song recommendations, given the current preferences of the user in either the form of a preferred genre or a favourite album,** letting users spend less time looking for music and more time listening. We would also provide an interactive visualization that shows how our top recommendations fit within the larger tree of music genres and albums, allowing users to explore a specific musical pathway on their own and potentially discover new songs and albums they can enjoy, without having to search through individual songs on music streaming sites.

Computational Plan

Our project will use a tree to represent the albums in our dataset in a hierarchical manner. Starting from the root, the tree will branch into broad genres (pop, rock, hip-hop, etc.) with each genre then branching into more specific subgenres (ex. the genre 'Rock' may branch into subgenres like 'Noise Rock', 'Pop Rock' or 'Psychedelic Rock'). Subgenres may also have their own subgenres (ex. 'Hip-Hop' may branch into 'Trap' which branches into 'Drill' which branches into 'UK Drill'). This tree will be visually displayed on a graphical interface and users will be able to interact and explore the tree on their own by clicking deeper into the tree. This is the tree that is displayed by default even before the user makes any inputs, we will call this the "GenreTree". To prevent our trees from becoming too large in the visualization, we will only display a tree with a maximum depth of 2 on the interface each time, whenever a user clicks on a genre, it will update the interface to display a new tree with that genre being the new root of the tree.

In terms of computations, we will first create an "Album" data class:

```
class Album:
    """
    a class to represent an album.
    """
    name: str
    artist: str
    genres: list[str]
    rank: int
    release: str
    descriptors: list[str]
```

The instance attributes include the name of the album, the name of the artist, a list of the album's associated genres, its position on the music database RateYourMusic's list of most popular albums, it's release date and a list of descriptors for the album. For this information we are using a dataset of RateYourMusic's top 5000 most popular albums (as of 2022) found on Kaggle. We will then create the aforementioned tree using a list of every genre that is listed on RateYourMusic's website. While we cannot find a dataset of the full set of genres, this information is available on a list on it's website and we will create our own dataset of this information by scraping the data off the website with requests and beautifulsoup4 libraries. At the bottom of every branch of the tree would be albums corresponding to the genre at the root of that subtree. We will create a tree plot of this by using the plotly and igraph libraries and displaying it visually using an interface made with plotly dash. We will make the tree plot interactive by using plotly dash's abilities to use callback functions.

Since users can give either the input of a favourite album or their preferred genre of music, there will be two cases for how we update the interface and give our recommendation. If given the input of an album by the user (for users who want a specific recommendation based on something they know they like), we will compute the recommendations for the user by comparing how many matching descriptors a given album has to other albums on the full list of albums. We will store a new list of albums ordered by how many descriptors are matching, the most matched being at the top of the list and vice versa. In the case of a tie, we will decide the order based on how high it placed on the original popularity list using the rank attribute of the Album class. We will then visually display the top recommendations in a ordered list within the graphical interface and give users the options of how many albums to display at once, the default being the top 10 recommendations. The visualization of the tree will be a different type of tree than the default GenreTree called a RecommendationTree. A RecommendationTree will have their original album at the root of the tree with each of it's subtrees being a computed recommendation. If you click on one of the recommendation nodes, it will update the RecommendationTree to have that recommendation be the the root of the tree instead and give recommendations based on the new root album. It will also update the displayed list. To make sure the same albums aren't recommended over and over, albums already existing in the upper levels of the tree will not be in the comparison at lower levels of the tree. If the user first inputs a genre, the top albums of that genre will first be displayed.

If the user gives the input of a preferred genre instead (this is for users who may not be looking for specific recommendations based on an album they like but want to explore the top recommendations of a given genre), our recommendations will simply be the albums with highest popularity ranking within that genre. When given a genre, we will combine all of its subgenres and it's subgenres subgenres together, this is for the case that the user gives a very broad genre such as Pop. If the user clicks on one of the recommended albums in the GenreTree, it will then be updated to display a RecommendationTree instead with that album being the root album just like in the case where the user inputs their own album from the start.

For every RecommendationTree, we will store the previous RecommendationTree/GenreTree so that users can always go back and explore a different recommendation instead of having to start a new session of the program. To do this, we will create a linked list for every instance of the program. The first node of the tree will always be the default GenreTree with each subsequent node being a RecommendationTree generated by the actions of the user.

We will also use the Spotify API to embed a web player on our interaface, giving users the ability to playback a recommended album within the interface itself so that they can quickly decide whether they like the recommendation or not without having to search for the album themselves on a seperate platform.

References

IFPI. (2021). Engaging with Music. https://www.ifpi.org/wp-content/uploads/2021/10/IFPI-Engaging-with-Music-report.pdf

Bryan O. (2022, March). Rate Your Music: The Top 5,000 Most Popular Albums, Version 1, Retrieved March 8, 2023 from https://www.kaggle.com/datasets/tobennao/rym-top-5000

Plotly Dash. (n.d.). Introduction — Dash for Python Documentation. https://dash.plotly.com/introduction

Plotly (n.d.). Tree Plots in Python. https://plotly.com/python/tree-plots/

Richardson, L. (2007). Beautiful soup documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Reitz, K. (n.d.). Requests: HTTP for Humans™. https://requests.readthedocs.io/en/latest/

Spotify (n.d.). Spotify Web API. https://developer.spotify.com/documentation/web-api/reference/#/

Sonemic (n.d.). RateYourMusic All Music Genres. https://rateyourmusic.com/genres/