SENG474 Final: Hybrid Music Recommendation System

Yule Wang, Weiting (Tony) Ye, Daehwan (David) Kim,
Adithyakrishna Arunkumar, Abhay Cheruthottathil

April 11, 2025

1. Introduction

In the era of digital music overload, users are presented with vast libraries of songs, making personalized recommendation systems essential for enhancing user experience and engagement. This project aims to develop an efficient music recommendation system that fuses the strengths of two machine-learning approaches: collaborative filtering and content-based filtering. By merging two diverse datasets—the Spotify Million Playlist Dataset (providing playlist-level information) and the Spotify 1.2 Million Songs Dataset (providing rich audio features for individual tracks)—our system is designed to generate ranked song lists for a given input (either a single track or an entire playlist). The hybrid approach seeks to capture both the implicit user behavior reflected in playlists and the intrinsic audio characteristics of songs, thereby delivering recommendations that are both contextually relevant and musically similar.

2. Problem Statement

The core objective of this project is to design a recommendation system $\boldsymbol{R}$ that, for a given playlist $\boldsymbol{p}$ or song $\boldsymbol{s}$, returns a ranked list of songs ordered by their relevance. Formally, let S denote the set of all songs. For a given input $\boldsymbol{x} \in \{\boldsymbol{p}, \boldsymbol{s}\}$, the recommendation function is defined as:

$$\boldsymbol{R(x) = \{s_1, s_2, \ldots, s_k\}}$$

$$R(x) = \{s_1, s_2, \ldots, s_K\},$$

Where $\boldsymbol{s_1, s_2, \ldots, s_k} \subset \boldsymbol{S}$ are the top-$\boldsymbol{K}$ recommended songs.

In our initial experiments, for each song in a given playlist, we applied the K-Nearest Neighbors (KNN) algorithm to retrieve the top-1 recommendation; that is, for a playlist containing $\boldsymbol{N}$ songs, this process yielded $\boldsymbol{N}$ recommendation candidates (one per song). The recommendation score for each candidate was determined by the Euclidean distance between the query song's feature vector and that of the candidate. We then selected the candidate with the smallest distance as the final recommended song for the playlist.

The effectiveness of the recommendation is evaluated using the hit rate (HR@K), defined as:

HR@$\boldsymbol{K}$ = Number of Hits/Number of Trials

$$\text{HR@}K = \frac{\text{Number of Hits}}{\text{Number of Trials}}$$

where a "hit" occurs if the hidden song is among the top-***K*** results. In our experiments, the value of ***K*** is carefully selected based on preliminary experiments and validation procedures, and we provide a discussion on its choice in the experimental section.

New Version (check this): where a "hit" occurs if a masked (hidden) song appears within the top-***K*** recommendations. In our experiments, the choice of ***K*** (initially set to 1 for individual song recommendations) and subsequent aggregation of multiple recommendation candidates are determined based on preliminary experiments and validated through cross-validation.

3. Background and Related Work

Recommendation systems have traditionally been categorized into collaborative filtering (CF) and content-based filtering (CBF) approaches. Collaborative filtering leverages user behavior and historical interactions to identify patterns and make recomme\ndations, whereas content-based filtering relies on the intrinsic properties of items—such as audio features in the case of music—to find similarities.

Previous studies have shown that playlist co-occurrence data can effectively capture latent relationships among songs, while audio feature-based similarity measures (e.g., using tempo, energy, and danceability) can enhance the relevance of recommendations. However, each approach has its limitations. Collaborative filtering may struggle with cold-start problems, and content-based methods can be overly simplistic if feature weights are not properly calibrated. Our hybrid model is designed to overcome these limitations by integrating both signals in a mathematically principled manner.

4. Methodology

4.1 Data Integration & Normalization:

We begin by merging the Spotify Million Playlist Dataset and the Spotify 1.2 Million Songs Dataset based on a shared key (track ID). This process retains only the records present in both datasets, ensuring that each song in the merged dataset has both playlist context and detailed audio features (e.g., tempo, danceability, loudness, valence). Our merged dataset contains approximately N unique songs (the exact number is provided after the data cleaning stage), each represented by a feature vector of dimension $\boldsymbol{d}$.

Before applying any clustering or similarity measures, we standardize all numerical audio features using z-score normalization via the StandardScaler. This normalization guarantees that features such as tempo, loudness, and valence are on the same scale, thereby preventing any single feature from dominating the subsequent analysis. Specifically, for each feature $\boldsymbol{x}$, the normalized value $\boldsymbol{z}$ is given by:

$$\boldsymbol{z = (x - \mu) / \sigma}$$

$$z = \frac{x - \mu}{\sigma}$$

Where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\sigma}$ is the standard deviation of the feature across all songs.

4.2 Collaborative Filtering via Gaussian Mixture Models

For the collaborative filtering component, we compute an average "sound profile" for each playlist. This profile is defined as the mean of the normalized audio feature vectors of all songs within a playlist. These mean vectors serve as a compact representation of a playlist's overall musical characteristics.

We then apply a Gaussian Mixture Model (GMM) to cluster the playlists. The GMM is a probabilistic model that assumes that the data is generated from a mixture of several Gaussian distributions. Unlike hard clustering methods, GMM assigns a probability to each playlist for belonging to each cluster. Mathematically, the probability that a playlist $\boldsymbol{x}$ belongs to cluster $\boldsymbol{j}$ is given by:

$$P(j|\mathbf{x}) = \frac{\pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)}{\sum_{l=1}^{J} \pi_l \mathcal{N}(\mathbf{x}|\mu_l, \Sigma_l)}$$

where $\boldsymbol{\pi_j}$ is the mixing coefficient, and $\boldsymbol{\mu_j}$ and $\boldsymbol{\Sigma_j}$ are the mean vector and covariance matrix for cluster $\boldsymbol{j}$. This soft clustering approach allows each playlist to exhibit membership in multiple clusters, reflecting the blended nature of musical styles. The CF score for a candidate song is derived from the probability that its associated playlist belongs to the same cluster as the input.

A crucial aspect of fitting a GMM is the Expectation Maximization (EM) algorithm. In our implementation, we use the GMM from scikit-learn, which employs EM as its core optimization routine. The EM algorithm is an iterative method that alternates between two key steps:

- Expectation (E-step) : Given the current parameter estimates ($\boldsymbol{\pi_j}$, $\boldsymbol{\mu_j}$, $\boldsymbol{\Sigma_j}$), compute the responsibilities, i.e., the probability that each playlist belongs to each cluster.
- Maximization (M-step) : Update the parameters by maximizing the expected log-likelihood of the data with respect to the current responsibilities computed in the E-step.

These steps are repeated until convergence, meaning that the improvement in log-likelihood between iterations becomes negligible or a maximum number of iterations is reached.

By applying GMM, each playlist is assigned a soft membership across clusters, which is particularly beneficial in music recommendation because playlists often exhibit blended characteristics. The CF score for a candidate song is then derived from the probability that its associated playlist is similar (belongs to the same cluster) to the input playlist.

4.3 Content-Based Filtering using K-Nearest Neighbors

The content-based filtering component leverages the detailed audio features of individual songs. We used the k-Nearest Neighbors (KNN) algorithm, utilizing a ball tree structure to facilitate efficient similarity searches in high-dimensional feature spaces. The ball tree structure recursively partitions the data set into spheres or "balls" where each node in the ball tree represents a region in the feature space that contains a subset of the data points. For the nearest neighbour queries, , instead of comparing the query point with every single data point, the algorithm quickly eliminates entire regions (or "balls") that are far away from the query point. This is particularly efficient when using Euclidean distance.

We begin the process by extracting and organizing these features into a structured format. In our implementation, song attributes are stored in a dictionary that is converted into a NumPy array of feature vectors. This array serves as the input for our KNN model. We instantiate the model using scikit-learn's NearestNeighbours class with the ball_tree algorithm.

Once the data structure is prepared, the model is trained by fitting it to the collection of feature vectors. During the query phase, the KNN model first identifies the nearest neighbors of a given item based on its location within the feature space. It then calculates the Euclidean distances between the query point and these neighbors. The Euclidean distance between two feature vectors, x and y is computed as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

This Measurement captures the absolute differences between each pair of feature components and plays a key role in the identification of similar items. It is important to note that, contrary to some recommendation systems that may use cosine similarity, our methodology relies solely on Euclidean distances. We compute the nearest neighbors first through the efficient ball tree structure before evaluating these distances, ensuring that our recommendations are generated based on precise, numerical proximity within the feature space.

The integration of the ball tree algorithm with the Euclidean distance metric allows our content-based filtering system to operate both efficiently and accurately, providing personalized recommendations even when extensive user interaction data is not available. This approach enables rapid similarity queries and robust item comparisons, forming a critical component of our overall recommendation strategy.

4.4 Hybrid Recommendation Strategy

Our hybrid recommendation strategy integrates the strengths of collaborative and content-based approaches to achieve refined music recommendations. The process initiates by processing the input playlist through our collaborative filtering model, which employs Gaussian Mixture Models (GMM) to determine the playlist's most likely cluster membership based on its aggregated audio features. This clustering step retrieves a set of playlists exhibiting similar musical characteristics.

The songs contained within these similar playlists form the candidate pool for the next stage. In the content-based phase, we apply the k-Nearest Neighbors (KNN) algorithm to this candidate set. Using normalized song features, the KNN algorithm calculates the Euclidean distance between the input playlist's feature vector and the feature vectors of candidate songs.

By selecting the songs with the smallest Euclidean distances, our system refines the candidate pool to identify the top recommended songs. This hybrid approach leverages the broad, latent associations revealed by GMM clustering alongside the precision of content-based similarity metrics, providing a balanced and effective recommendation strategy.

5. Experimental Setup and Results

5.1 Evaluation Metrics

In our evaluation procedure, we adopt a masking strategy to rigorously test the performance of our hybrid recommendation system. For each playlist in our validation set, we mask one song from the complete list—this masked song is treated as an unseen query. The system then generates a list of top-K recommendations using the available songs in the playlist. A hit is recorded if the masked song appears within this recommendation set.

The hit rate is then computed as the ratio of the number of successful recommendations (hits) to the total number of masked trials, as given by

$$\mathrm{HR@}K = \frac{H}{T}$$

where H is the number of hits and T is the total number of trials. In our experiments, we only tested with k =1 resulting in a hit-rate of 0.05%. This metric indicates that in 0.05% of the test cases, the system correctly identified the masked song as the top recommendation. Although the hit rate appears low, it reflects the challenging nature of predicting a single, highly specific song from a vast candidate pool within a complex, high-dimensional audio feature space. The stringent evaluation protocol—where only one recommendation is provided per masked song—combined with the inherent variability in musical characteristics, contributed to this modest performance.

Additionally, we computed the Euclidean distance between each recommended song and the input playlist's tracks and we isolated the pair exhibiting the largest overall distance. By decomposing this distance feature by feature, we observed in **Figure 1** that *tempo* constituted the dominant source of disparity, dwarfing differences in other attributes (e.g., danceability, energy, valence). This imbalance

suggested that tempo's numerical range or inherent variability was overshadowing the contributions of the remaining features.

To mitigate tempo's disproportionate influence, we introduced feature-specific weighting, reducing the weight assigned to tempo. As illustrated in **Figure 2**, lowering tempo's weight led to more balanced feature comparisons, as indicated by the reduced weighted differences. We then recalculated the Euclidean distances under these new weights, resulting in notably smaller overall distances for the same pair. In essence, the recommended songs now aligned more closely with the input tracks in terms of the most musically relevant features.

In **Figure 3**, we further investigated how varying the tempo feature's weight from 0.1 to 0.9 affected the average Euclidean distance across the recommendation set. The plot shows a clear upward trend in distances as the tempo weight increases, reinforcing that a high tempo weight can inadvertently penalize songs that differ even moderately in tempo—even if they match closely on all other attributes. By contrast, lowering this weight yields more musically coherent recommendations.

These findings highlight the importance of carefully tuning feature weights in a recommendation system. Even a single feature with wide variability can distort similarity metrics if unadjusted. By refining such features' relative importance, one can achieve more balanced distance calculations and, consequently, more accurate recommendations.

5.2 Results

Our experiments reveal that the collaborative filtering component based on Gaussian Mixture Models (GMM) effectively captures latent musical relationships through soft cluster memberships. This approach yielded a preliminary hit rate of approximately 0.05% on our masked validation set, underscoring the inherent challenge in predicting a single, highly specific song from a vast and diverse candidate pool.

In parallel, the content-based filtering component initially relied on the k-Nearest Neighbors (KNN) algorithm with Euclidean distance as the similarity metric applied on standardized audio features. However, early analyses revealed that these recommendations were overly similar, suggesting that certain features were exerting undue influence on the distance calculations. For instance, the Euclidean distances computed before applying any feature weighting were as follows:

```
[1.2343416230932567, 1.096706483065513, 1.1472488831943612,
2.921755937788218, 1.0098685602102448, 0.3025167755792601]
```

After implementing a feature weighting strategy—most notably reducing the weight of the tempo feature—we observed a notable reduction in these distances:

```
[0.7364530416123353, 0.3929645149376182, 0.4190513095078556,
0.813794568671985, 0.5580938531285993, 0.3021595596768714]
```

This reduction—from values such as 1.23 to below 0.8 in some dimensions—indicates that our weighting scheme successfully achieved a more balanced representation of the audio feature space.

Consequently, the content-based filtering is better aligned with musical relevance, resulting in improved recommendation accuracy.

Overall, these findings demonstrate that while the collaborative filtering component lays a solid groundwork through robust latent associations, the refinement introduced by adjusting feature weights in the content-based filtering component is critical to achieving more precise and musically coherent recommendations. The observed improvements in Euclidean distances provide empirical support for the effectiveness of the weighting approach, and they will guide further enhancements to our hybrid recommendation system.

6. Discussion

The experimental results highlight the strength of the collaborative filtering component in capturing the complex relationships among playlists via the GMM. The soft clustering approach enables each playlist to belong to multiple musical genres or styles, which aligns well with the nuanced nature of musical preferences. While the content-based filtering method using KNN also shows promise—especially after introducing weighted features, its contribution to the final recommendation score appears secondary in our current dataset configuration.

While our current experimental setup demonstrates promising results with a hit rate computed at k=1, several avenues exist for further improvement and investigation. First, exploring higher values of k could provide a more comprehensive understanding of the recommendation system's robustness. Although we initially focused on the top recommendation, extending the analysis to consider the top 3 or top 5 recommendations may reveal additional insights into the balance between precision and recall in our model.

Future work could also involve more in-depth hyperparameter optimization. This includes exploring alternative methods for feature weighting, as well as tuning other parameters such as the number of clusters in the Gaussian Mixture Models and the number of neighbors considered during the K-Nearest Neighbors search. Additional experiments could investigate the sensitivity of the system to the range and distribution of individual audio features.

Lastly, integrating more sophisticated fusion techniques to combine the collaborative filtering and content-based signals might enhance overall performance. For example, blending the scores using non-linear methods or employing ensemble models could potentially capture complex interactions that are not fully exploited by the current linear combination strategy.

In summary, while our initial experiments with k = 1 provide a solid baseline, further testing—including varying k, refining feature weights, and integrating user feedback—will be key to evolving our recommendation system into a more robust and user-responsive tool.

7. Conclusion

This project presents a hybrid music recommendation system that integrates collaborative filtering via Gaussian Mixture Models with content-based filtering using K-Nearest Neighbors. By merging two comprehensive datasets and carefully normalizing audio features, our system is able to capture both user behavior and intrinsic musical characteristics. Experimental results demonstrate that while the collaborative filtering component is dominant in our current setup, the integration of content-based signals through weighted feature similarity can further refine the recommendations. Future improvements will focus on optimizing feature weights and exploring alternative clustering strategies to enhance system performance and user satisfaction.