MUSIC RECOMMENDER SYSTEMS

COMP9727 FINAL PROJECT

YUDEN SAE-UENG: z5359205

JULY 2024

1 Problem

Introduction

The exponential growth of the music industry has resulted in an overwhelming array of options available to users. While contemporary digital products and services provide easy access to extensive collections, this abundance can make it increasingly difficult for users to find tracks that match their preferences. A music recommendation system can help users navigate these extensive collections, surfacing relevant tracks for their evaluation. Despite the breadth of existing research, the psychology of music preference is not clearly understood. Music recommendation systems as a result, are faced with managing the ambiguity of influence from: contextual-factors, the user's subjective view of music quality, and the user's social interactions.

Competitor Analysis

Music recommendation systems have been under development for many years, and have been employed in industry with digital music streaming services such as, Spotify, Pandora and Apple Music. Currently, Spotify stands out as a leader within the music streaming and recommendation domain. Spotify's recommender system consists of individual song recommendations and playlist generation, which has seen success with users expressing interest in recommendations, by adding them into their personal playlists [2].

Although Spotify's recommender has shown significant success in industry, a notable limitation identified is the lack of diversity in music recommendations causing a tendency to generate "Filter Bubbles" [1], a phenomenon where displayed information is restricted around the user's niche.

The Recommendation Problem

Our system aims to deliver personalized track recommendations by leveraging user listening data to recommend a collection of tracks that match the user's previously liked items. In addition to recommending familiar content, we intend to address the diversity of recommendations by

incorporating novel mechanisms that introduce users to new and varied tracks, to circumvent "filter bubbles".

We approach the recommendation problem as a classification task, where we identify songs as either "like" or "dislike". This approach was decided upon to accommodate the fact that users' musical preferences are not static or singular but can vary depending on mood, context, and other factors. The use of a binary label, was to prevent over-commitment to short-term trends, such as if a user repeatedly played a certain new-release and have lost interest over time.

System Deployment and User Interaction

The system is intended to be deployed as a back-end service for a web-based mobile application. The standard use-case of our system does not require explicit user input, and relies on previous user-interactions of the system, through their listening history, to guide recommendations.

The system will not receive direct user-feedback to assess the quality of recommendations, however, user responses through click-through and listening count will contribute in updating the user-profile to guide future recommendations.

2 Dataset(s)

2.1 Spotify-MillionSongs Dataset

The main dataset used for the problem is the Spotify-MillionSongs Dataset, which is a combination of the Million Songs Dataset, with additional metadata from the Spotify database.

The Million Songs Dataset (MSD) is a large-scale collection of audio features and metadata for over a million songs. It was created by Echo Nest to support research in music information retrieval and recommendation systems. The songs are identified by a track id and include songs published from 1922 to 2011. The MSD also consist of user listening information in the form of play-counts which denote the amount of times a user has listened to a certain song.

A major issue in the MSD is mismatch between song ids (used to identify user playcounts) to track ids (used to map a song to its metadata).

The Spotify-MillionSongs Dataset is filtered to remove all songs affected by the mismatch of song and track ids, and replaces the song's metadata with information sourced from spotify, introducing tags and audio-features such as "acoustics" and "insturmentalness". It is delivered as 2 csv files, to include both song data, and user listening history:

File 1: Music Info

Consisting of 50,683 unique songs, identified by a track id. Each song is included with the following meta-data:

Value	Description
track_id	Unique identifier for the track.
name	Title of the song.
artist	Name of the artist or band performing the song.
spotify_preview_url	URL to a 30-second preview of the track on Spotify.
spotify_id	Unique identifier for the track on Spotify.
tags	Descriptive labels or tags associated with the song.
genre	Genre classification of the song.
year	Year the song was released.
duration_ms	Duration of the song in milliseconds.
danceability	A measure of how suitable the track is for dancing, on a scale from 0 to 1.
energy	A measure of the track's energy level, on a scale from 0 to 1.
key	The key of the song, represented as an integer.
loudness	Overall loudness of the track in decibels.
mode	The modality of the track, where 0 represents minor and 1 represents major.
speechiness	A measure of the presence of spoken words, on a scale from 0 to 1.
acousticness	A measure of the track's acoustic quality, on a scale from 0 to 1.
instrumentalness	A measure of the likelihood that the track is instrumental, on a scale from 0 to 1.
liveness	A measure of the presence of an audience, on a scale from 0 to 1.
valence	A measure of the musical positiveness or mood of the track, on a scale from 0 to 1.
tempo	The tempo of the track in beats per minute (BPM).
time_signature	The time signature of the track, represented as a fraction.

Table 1: Attributes and Descriptions of the Spotify-Million Songs Dataset

File 2: User Listening History

This file contains the listening data of 962037 unique users. It includes the following values:

Additional Datasets

We supplemented the Spotify-Million Songs Dataset to add additional editorial meta-data and pre-computed audio features.

Value	Description
track_id	Unique identifier for the track.
user_id	Unique identifer for the user.
playcount	An integer denoting the number of times a user has played a song.

Table 2: Attributes and Descriptions of the user listening history dataset

MSD Allmusic Style Dataset(MASD)

Sourced from the Vienna University of Technology (Technische Universität Wien), this dataset contains musical style labels for the Million Songs Dataset. The labels are sourced from AllMusic.com. The labels are matched to songs through the MSD track id.

Last.fm tags

This dataset was provided with the Million Songs Dataset, and includes song level tags sourced from Last.fm. The tags are matched by the MSD track id.

Million Songs MFCC Features

Sourced from the Vienna University of Technology (Technische Universität Wien), this dataset contains pre-computed Mel-frequency cepstral coefficients (MFCCs) of songs in the Million Songs Dataset (MSD). The data was provided in WEKA ARFF format and matched to the song using the MSD track ID. Our team developed a CSV version of the dataset for our project.

MFCCs are widely used in music information retrieval because they effectively capture the timbral texture of audio signals. By representing audio signals in terms of their frequency components, MFCCs can capture nuanced characteristics of music, such as timbre and tonal quality, which are crucial for distinguishing between different genres, styles, and moods.

2.2 Exploratory Data Analysis

TBA

2.3 Known limitations of the dataset

Significant Positively Skewed Distribution of Play-counts

The play-counts associated with each user is postively skewed, with 95.76% users-song play-count values under 10, and over 60.82% of user-song play-count values equal to 1.

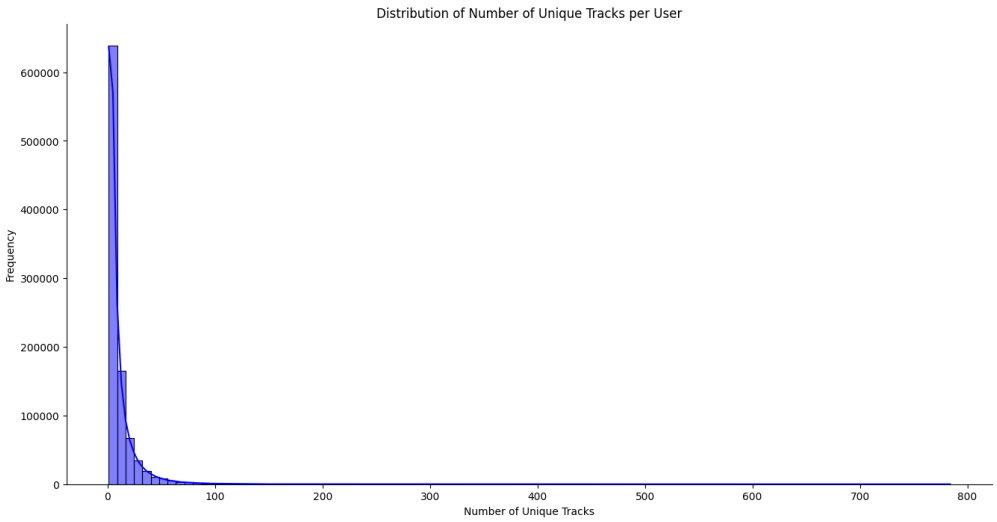

Figure 1: Distribution of unique tracks per user.

Significant Positively Skewed Distribution of Songs Listened per user

We also observe that the majority of users did not listen to many different songs. With the average amount of songs listened too, per user being 10. We recorded a median of 5 songs, with minimum 1 and maximum 784 songs. The mode value was at 1.

Data Attribute Ambiguity: Energy and Danceability

The attributes Energy and Danceability, is implied to represent the tone of the song, however, we are not given descriptions of how these attributes are sourced.

Lack of Temporal and Contextual Data

The user listening data from the Million Songs Dataset only includes play-counts. The lack of temporal and contextual data, restricts our ability to predict user intent, and thoroughly explore context-based recommendation methods.

Temporal Scope

The Million Songs Dataset only contains song from 1922-2011, and its user listening data also ends around 2011. The temporal scope of the MSD means it cannot be used to observe contemporary music consumption behaviour.

3 Method(s)

To address the proposed recommendation problem, the team developed multiple approaches, employing, content-based, user-based collaborative-filtering, and neural-collaborative filtering methods. This allowed us to employ different paradigms to assess different recommendation strategies for music recommendation.

Each approach was evaluated individually to assess the respective model's performance and re-evaluated against each other to select the model most effective for the recommendation task.

Content-Based Recommender

Strategy

Our content-based recommender system follows the strategy of recommending music based on the user's subjective perception of quality. We expand on the idea of recommending music with acknowledgement of their subjective perceptions of music similarity described by Celma in his review of music recommendation systems [3], by applying the same principle of subjective perception to determine what a user "likes".

The strategy of our content-based approaches extensively utilizes the concept of the *Music Information Plane*, where music is described with a combination of high and low-level editorial, cultural and acoustic metadata. Additionally, the "semantic gap", describes the complex and ambiguous relation between low-level descriptors to high-level attributes, for example, the mapping between audio signals to genre terms like "rock" or "hip-hop".

Our content-based recommendation approach, addresses user subjective perception of quality by categorising users to either, influenced by high-level descriptors, or users who preference is more likely based on intrinsic attributes of a song/track.

Data Pre-processing

We first loaded the Spotify-Million Songs Dataset, and its respective user listening history file. The Spotify-Million Songs Dataset was than augmented with the Mel-Frequency Cepstrum Coefficients dataset, last.fm tags, and AllMusic song styles. These datasets were all combined by joining based on the MSD track id. At this point, all null values were kept.

A new attribute text_content is created by combining all text attributes, separated by a space character. Artist names, first had all space values replace with an underscore and all tags which

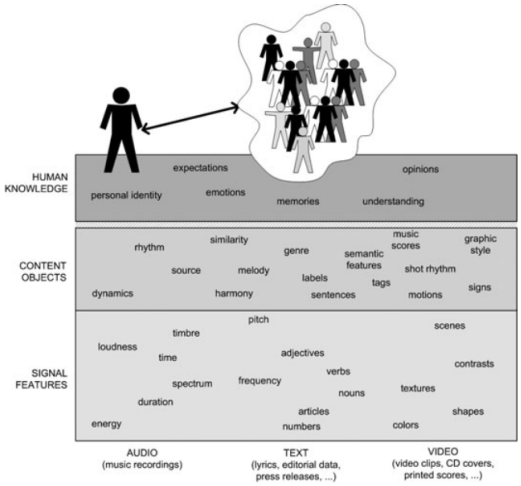

Figure 2: The music information plane

were comma-separated values, had their commas replaced with space characters. This allowed us to ensure, artist names were treated as individual tokens, and tags were processed individually.

The new text_content attribute, was further pre-processed by removing all non-word characters using a regular expression, applying the WordNetLemmatizer to lemmatize words before tokenization.

After, preprocessing the music meta-data, we pre-process the user listening data by assigning a binary label *rating* where 1 denotes any song that the user has listened to, more than once, and 0 being otherwise. This is with the assumption that songs that the user has listened to more than once, are liked by the user.

Creating User Profiles as Text Documents

We first create a user-profile based on text data (corresponding to high-level descriptors in our recommendation strategy). Choosing the top-5 songs ranked by the most playcounts, where the value of playcount is greater than 1, and combining the pre-processed text_content values into a single string document.

The top-5 songs were chosen to ensure the there is at least 5 other songs that the user ranked as liked that are not used for building the user profile, which made the system easier to evaluate with top-N metrics.

Classifying User Profiles

The average cosine similarity of the user's liked songs is computed by:

1. Compute the Term Frequency - Inverse Document Frequency (TF-IDF) vector of every string document for each liked song in the user's profile. We used the SciKitLearn tfIdfVectorizer for this purpose.
2. Store results in a $N \times N$ matrix, where N is the number of user liked songs.
3. Calculate of average of all cosine similarity values, ignoring the diagonal elements, which denote the song's compared by themselves (as they have cosine similarity of 1).

We determine a text similarity threshold as any average cosine similarity values above the upper quartile (Q3), which in our dataset was at 0.17.

Given the threshold and the computed average cosine similarity value for all users, we split the users into 2 subsets. Every user with an average cosine similarity above the threshold, is put in group A, denoting users that are predicted to be influenced by high-level descriptors. All other users are put in group B, which we predict to be more strongly influenced by the intrinsic attributes of a track/song.

An additional attribute text_similarity_threshold is added into every user profile, with as a binary label, where a value of *true* denotes the user as in group A (else the user is in group B). When a recommendation request is made for a user, the binary label is used to decide which of the following approaches will be used to generate a recommendation.

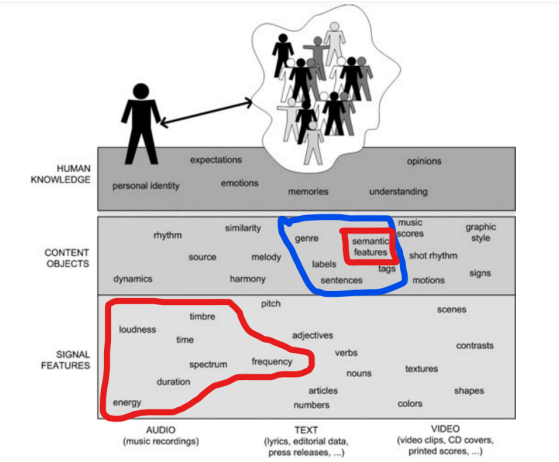

Figure 3: Blue denotes high-level descriptors while red denotes the low-level audio features. Semantic features are included in red with Danceability and Energy

Performing Recommendations for Group A users

Group A users are recommended items based on the similarity of high-level descriptors. For every user, we define a blacklist of songs which are prevented from being recommended to the user, this included all songs that are used to build the user text profile.

The pre-processed text_content of every song in the dataset is vectorized using sciKitLearn's TF-IDF vectorizer to compute a TF-IDF Matrix of all songs' high-level text descriptors in the dataset.

The k-nearest neighbors (KNN) algorithm is employed to create a model of the entire songs dataset using the TF-IDF matrix as features. The KNN model was constructed with a cosine similarity as the distance metric. The value of K is chosen as,

$$K = \sqrt{n}$$

where, $n =$ the total number of songs in the dataset.

The heuristic of choosing K as the square root of n is supported by theoretical results indicating that it provides a good balance between classification accuracy and sample size [4].

To satisfy a recommendation request for a user, we compute the TF-IDF vector of the text document in the user's profile. This is passed to the KNN model to retrieve the nearest neighbors to the user.

We than filter for the top-N recommendations, where N is the desired number of recommendations, that are not included in the user's liked songs (used for the recommendation). The recommendation are surfaced for the user, and are added to the blacklist, so that successive calls do not resurface the same songs.

Performing Recommendations for Group B users

Group B are recommended items based on the similarity of low-level features.

The following features are chosen to fit a KNN model for all songs in the dataset:

- Year
- Duration
- Danceability
- Energy

- Key
- Loudness
- Mode
- Speechiness
- Acousticness
- Instrumentalness
- Liveness
- Valence
- Tempo
- Time Signature
- MFCC Standard Deviations
- MFCC Averages

All values are normalized using Z-score normalization using the SciKitLearn StandardScaler method.

The value of K is similarly chosen as,

$$K = \sqrt{n}$$

where, $n =$ the total number of songs in the dataset.

To perform a recommendation for a user, we take in the list of their top songs by playcount, iterate and for each song we use the KNN model to find the 2 most similar songs that are not already recommended to the user. This is performed iteratively until a satisfactory number of recommendations is generated.

Experimentation: Content-based

To find the optimal parameters for the content-based model, we evaluated the use of different similarity measures, and also changing the user classification threshold to classify users into Group A or B users (recommendation using text-based or audio feature similarity).

We experimented with using different similarity measures, Cosine similarity, Euclidean difference, and Minkowski difference. An evaluation dataset was created as users who have listened to more than 180 songs each. This was chosen due to the sparsity of the dataset which resulted in extreme difficulty to achieve non-zero values, which made it impossible to compare different approaches side by side. We acknowledge that choosing an extremely dense subset results in inaccurate evaluation, as it does not truthfully reflect all users.

The evaluation dataset was split 50:50, by hiding half of each user's liked songs, creating a training and test split.

This was done with the following process:

1. Set the user classfication threshold at the lower-quaritle of average text similarity for all users.
2. Classify all users in the dataset based on the user classification threshold (using the chosen similarity measure)
3. Fit the two KNN models based on the chosen similarity measure.
4. Evaluate top-N recommendations for all users in the training dataset.
5. Calculate TopN metrics: precision, recall and F1-score. True positives denote: a recommended song, matching a song in the test set.

This produced the following results:

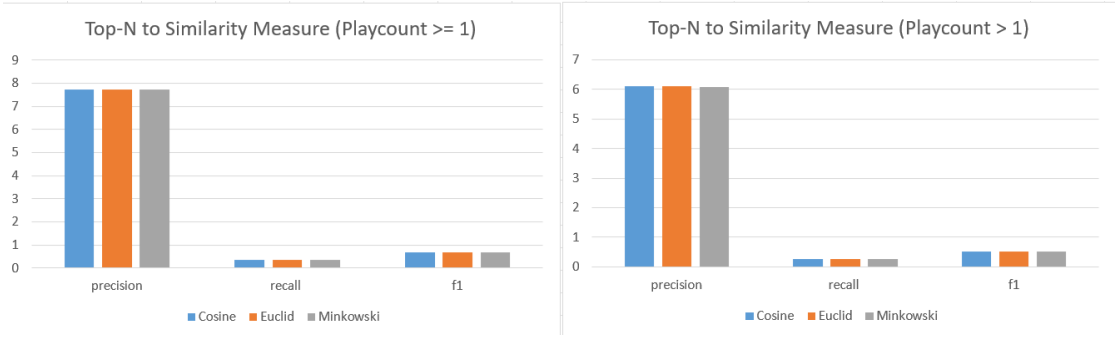

Figure 4: Top-N to similarity measure, Cosine similarity outperformed by a small margin

Cosine similarity was shown to the most effective measure of similarity, however, by a small margin.

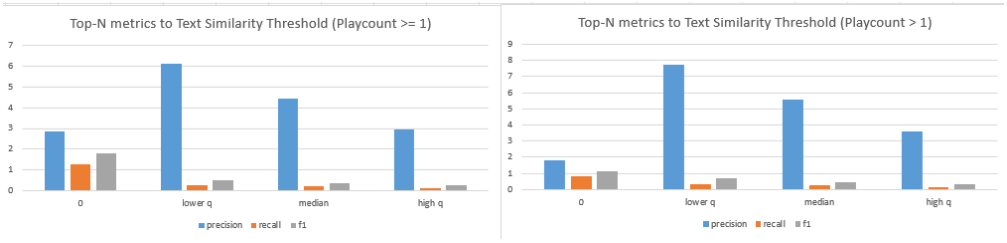

Figure 5: Top-N to similarity threshold for classifying users

Further experimentation was conducted to determine the optimal user classification threshold. This was conducted with the following process.

1. Set the user classification threshold at the lower-quaritle of average text similarity for all users.
2. Classify all users in the dataset based on cosine similarity with the chosen user classification threshold
3. Fit the two KNN models based on the cosine similarity measure.
4. Evaluate top-N recommendations for all users in the training dataset.
5. Calculate TopN metrics: precision, recall and F1-score. True positives denote: a recommended song, matching a song in the test set.

This generated the following results:

From this we determined the optimal value for the user classification threshold was at the lower-quartile.

Collaborative Filter Recommender

Strategy

This approach analyzes the similarity between different tracks based on user interaction patterns using play counts. It then recommends tracks to users by identifying those that are similar to the ones they have already expressed interest.

prepossessing

The user listening history dataset was preprocessed to remove all users who have listened to under 20 songs in total, this was to ensure that were using only active users. The playcount values were

than rescaled to be between 0 and 1, and the yeo-johnson transformation was employed to make the playcounts less right-skewed.

Additional data preprocessing included: removing any song that was listened less than 20 users in total, and SciKitlearn's LabelEncoder method was applied track and user ids to turn them into integers.

Methodology

The following methodology was used to perform create ratings and similarity matrix for collaborative filtering:

1. we select the features: danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo and time signature, and normalize these values in a range from 0-1.
2. Create a ratings matrix with userID as rows, SongID as Columns and ratins as values.
3. Calculate a similarity matrix using cosine similarity on the normalised song features.

We than predicted unrated user-song pairs by computing the weighted sum of the K most similar songs that the user has already rated. The value of K is set as 40.

To serve recommendations, the user is recommended the top-N songs with the highest predicted ratings.

Experimentation

To develop our final collaborative filtering model, we performed a series of experiments to evaluate different models and parameters.

Experimentation with user-based and item-based models

We evaluated User-based and Item-based models using basic, mean-centered and z-score normalised approaches to prediction. We also experimented using different similarity measures, cosine, MSD and Pearson, as well as different values of K.

Our user based approach was found to perform the best using cosine similarity, with either mean-centered or Z-score normalised approaches:

The following results were determined for testing different values of K:

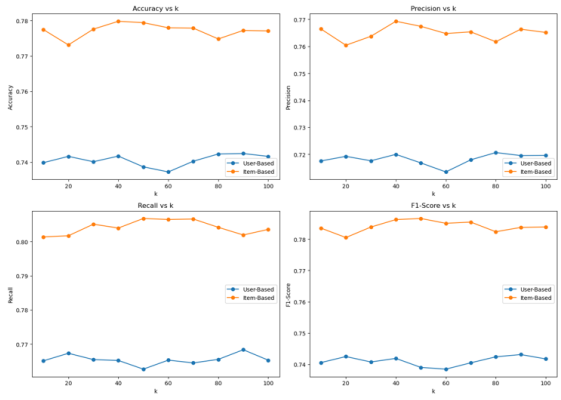

Figure 6: Performance with different value of K

4 Evaluation

5 Reflection

References

[1] Ashton Anderson et al. “Algorithmic Effects on the Diversity of Consumption on Spotify”. In: *Proceedings of the ACM on Human-Computer Interaction* 4.CSCW3 (2020). Ashton Anderson, University of Toronto & Spotify, ashton@cs.toronto.edu; Lucas Maystre, Spotify, lucasm@spotify.com; Rishabh Mehrotra, Spotify, rishabhm@spotify.com; Ian Anderson, Spotify, iananderson@spotify.com; Mounia Lalmas, Spotify, mounia@acm.org, pp. 1–28. DOI: 10.1145/3415206. URL: https://doi.org/10.1145/3415206.

[2] Greta Björklund et al. “An Exploratory Study on the Spotify Recommender System”. In: *Information Systems and Technologies*. Vol. 469. Springer, 2022. ISBN: 978-3-031-04818-0.

[3] Òscar Celma. “Music Recommendation”. In: Springer, June 2010, pp. 43–85. ISBN: 978-3-642-13286-5. DOI: 10.1007/978-3-642-13287-2_3.

[4] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Vol. 31. Springer, 1996. ISBN: 978-1-4612-6877-2.