Group Name: helloworld
Group Members: Murong Shen, Yuzhou Zhou, Eddy Liu, Hengde Li, Sabrina Geng

I. Project Summary

The goal of our project is to design and implement computationally efficient and accurate music recommendation algorithms that can be used by streaming platforms. We built a collaborative filtering algorithm using neural networks based on musical features. Our method ensures the successful capture of specific and diverse music preferences of users and can recommend niche songs.

Before constructing the models, we simulated user portrait data that contains a range of users' personal information for later analysis. Then, we performed in-depth Exploratory Data Analysis on the chosen dataset, especially on our objects of interest including music genres, popularity scores, songs, and musical features.

After EDA, we log-transformed our data to make them ready to use. We encoded the musical features of each song into latent space and represented each user's preferences in latent space. Then, we recommended songs to the target audience by measuring the similarity scores with other users. At the end of the project, we undertook detailed evaluations of the recommendation system and assessed the potential limitations of our methods.

II. Introduction

Our group focuses on Music recommendation using machine learning skills. As beginners, there are a lot of questions for us including which algorithms are we going to choose and whether we should recommend a single song or a playlist. Therefore, at the beginning of the project, we decided to learn from the big names: Amazon and Netflix.

The algorithm used by Netflix is called Latent factor Algorithm. Let me explain this: Each user has their own preferences. For example, Ryan likes country music, guitar accompaniment, Taylor Swift, and other elements (in fact, it is the label). If a song (item) has these elements, then the song will be recommended to Ryan. That is, using the elements to connect the user and the songs. Each person has different preferences for different elements, and each song contains different elements. Two matrices are simulated: 1 means the user likes it, and 0 means the user doesn't like it. Then, we multiply the two matrices. This calculation yields a matrix of ratings for different songs by different users. So, as shown in Fig. 1.1, we recommend Ryan Song B with the highest score of the four songs. But we found that creating labels for songs is very difficult. Thus, we chose to simplify the algorithm as follows. First, all songs are categorized into five recommended playlists by clustering. Second, we analyze the users' liked playlist, counting numbers of songs in different groups. We recommend the group that has the

most songs in their list. For example, in Fig. 2.3, the yellow points are categorized into group A. In playlist of sub-sample 1, songs in group A are the most relevant among the 5 groups. Thus, we recommend songs in group A to users in sub-sample 1.

Users Matrix	Pop	Hiphop	Classic	Blues	Taylor Swift
Ryan	0.6	0.8	0.1	0.1	0.7
Bruce	0.1	0	0.9	0.1	0.2
Amy	0.5	0.7	0.9	0.9	0

X

	Pop	Hiphop	Classic	Blues	Taylor Swift
SongA	0.9	0.1	0.2	0.4	0
SongB	0.5	0.6	0.1	0.9	1
SongC	0.1	0.2	0.5	0.1	0
SongD	0	0.6	0.1	0.2	0

T =

	SongA	SongB	SongC	SongD
Ryan	0.68	1.58	0.28	0.51
Bruce	0.31	0.43	0.47	0.11
Amy	1.06	1.57	0.73	0.69

Fig. 1.1 Latent Factor Matrices

The second algorithm is called collaborative filtering. Amazon invented this so-called “people who like this item also like so-and-so” algorithm to find their target customers and do advertisements. In short, this algorithm finds people who are similar to the target users, and then recommend products that are bought by those people but the target users didn’t. Similarly, we can think of the user preferences for different songs as a vector. Then, compute the cosine of two vectors' angles to indicate how similar two vectors are, with the cosine of a 0-degree angle being 1 (indicating that they are identical) and the cosine of a 180-degree angle being -1 (indicating that they are diametrically opposed). The larger the cosine value, the more similar the two users are. For example, in Fig. 1.2, User 3 is more similar to me and the music app should recommend songs that User 3 has listened but I did not.

	Love Story	Starboy	Heartless
User 1	3	-1	-1
User 2	5	1	-5
User 3	-5	3	3
Me	1	3	3

cos(U1, Me) = -0.502
cos(U2, Me) = -0.526
cos(U3, Me) = 0.754

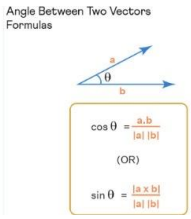

Fig. 1.2 Cosine Similarity Calculation

In practice, we found many limitations of those algorithms:

1. For Collaborative Filtering, it is hard to collect users’ private data: like which song they liked, collected, added to their playlist or just skipped.
2. For Latent Factoring, with high precision, it is hard to label the songs. (Label means different genres or what kind of musical instruments used and so on….)
3. For Clustering (simplified version), although it is easy to perform as we do not need the users’ private data, it doesn’t work well on new songs that hasn’t been clustered (Collaborative Filtering works for new songs).

Thus, we choose to combine the two algorithms we have mentioned above together to create a new method and we name it HelloSpotify. This method uses Autoencoder to do dim-reduction first, and then builds a matrix of 8 columns. Each row of the matrix

represents a user's vector. Finally, a cosine value between the user's vector and song's vector is computed to indicate the similarity between the musical preference of the user and the genres of the song.

This method has all advantages of the methods we learned from the Big Names:

1. Recommend songs rather than only a playlist.
2. Work for new songs and don't need to do clustering first.
3. No need to collect user's private data.
4. No need to label the songs.
5. And it is also easy to practice.

III. Data Analysis & preliminary conception

The dataset includes the top 100 Spotify hits from 2000 to 2010, with non-continuous variables including *artist*, *genre*, and *mode*, and continuous variables including *loudness*, *danceability*, *speed*, and other variables. The total number of observations is 1109.

For a music recommendation algorithm, the user portrait is a very important part. However, the existing data is not sufficient to enable us to obtain a wide range of users' personal information (playlists) and organize it into data sets conducive to analysis. Therefore, we want to sample and simulate different user profiles from this dataset. In this simulation, we use 50 songs per song list as samples to simulate the user portrait.

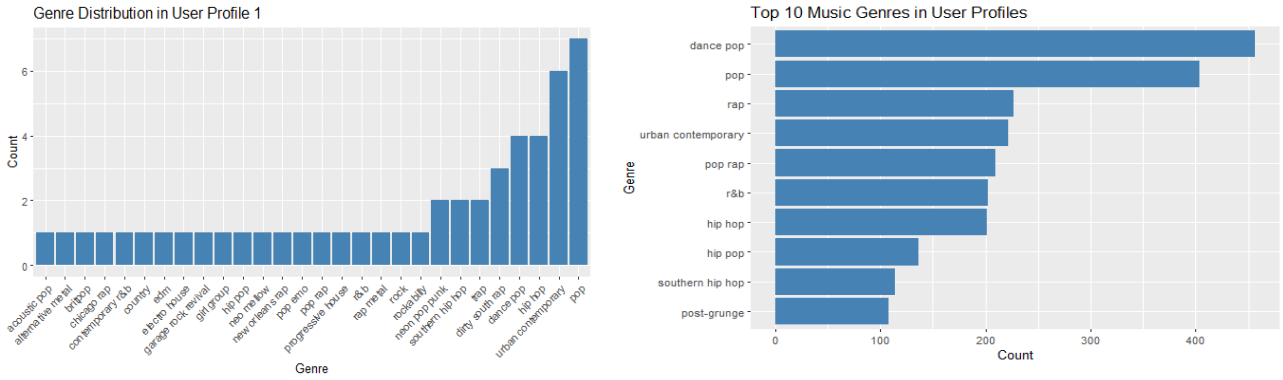

Fig. 2.1 Genre Distribution in User Profile 1 & Top 10 Music Genres in User Profiles

After the user portrait simulation, we conducted preliminary statistics on the music genres included in the user portrait, and the results were as follows. We have noticed that the number of mainstream music songs such as dance-pop, pop, and rap may affect the accuracy of the algorithm. However, due to the popularity of music and the high popularity of these categories of songs, in the preliminary construction, we will retain the desirable

number of these categories to ensure that it aligns with the current trend of the music market.

Given the many similarities between modern music genres, our group believes that identifying music genres with each song's original genre label and promoting it is not actually very effective. When we initially conceived the model, our initial expectation was to classify the numerical variables related to each song to define different types of music, and then construct the model. However, according to the current heat map of the correlation degree between variables, the relationship between different variables is weak.

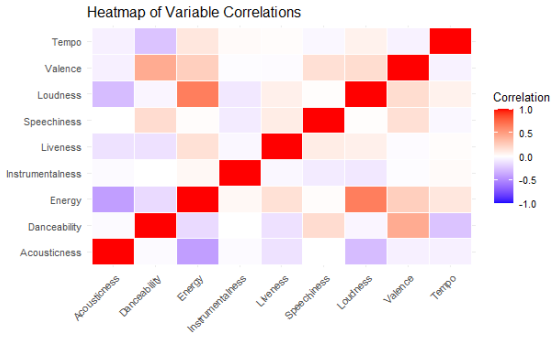

Fig. 2.2 Heatmap of Variable Correlations

Under this idea, we try to conduct different dimensionality reduction with continuous variables related to songs, and cluster analysis to probe the results. We mainly used PCA and LDA for dimensionality reduction and clustering by k-means. The result is shown as follows:

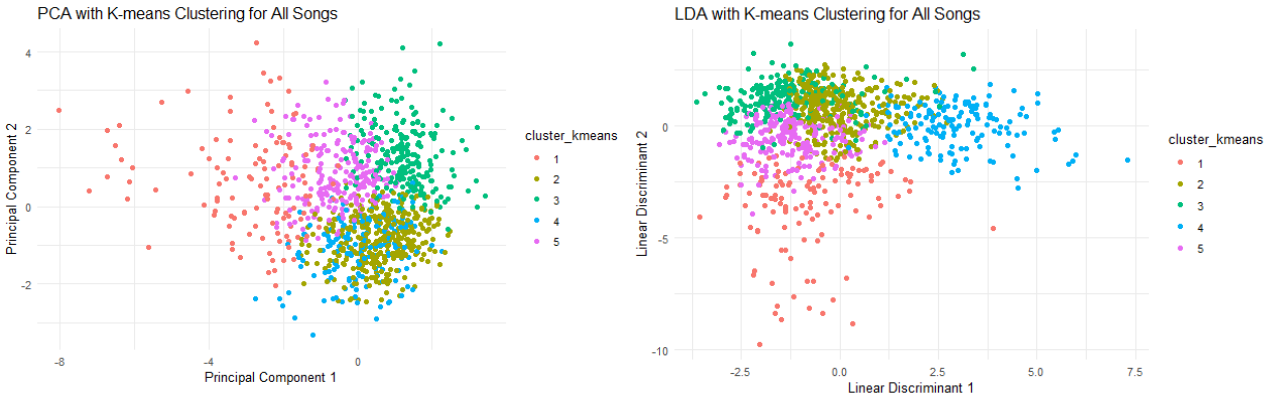

Fig. 2.3 PCA with K-means Clustering for All Songs & LDA with K-means Clustering for All Songs

After clustering the user's playlist as well (as shown in the figure above), we can get the specific type proportion and user preference of the playlist for recommendation.

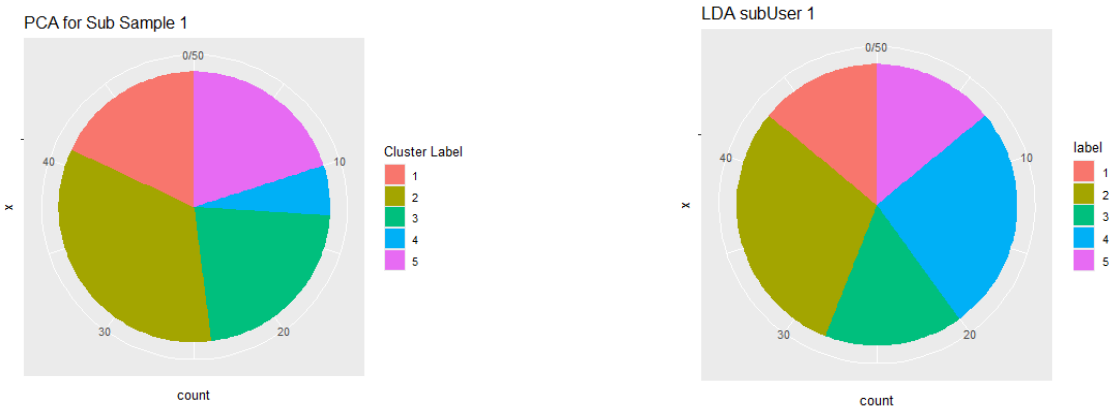

Fig. 2.4 PCA for Sub Sample & LDA for Sub Sample

When continuing the research, we found the defects and limitations of this scheme: First, the clustering results in the above figure were not ideal, and the data did not form different clusters well. Secondly, since only k = 5 is used for clustering in the attempt if the value of k is increased, the performance of clustering will become worse (as shown in the figure, there are quite a lot of overlapping parts), which may be due to the poor correlation between variables, so we have stopped the in-depth exploration of this method. The team members think that an ideal recommendation algorithm should include as many kinds of music as possible, so we chose another method to improve the accuracy of the recommendation.

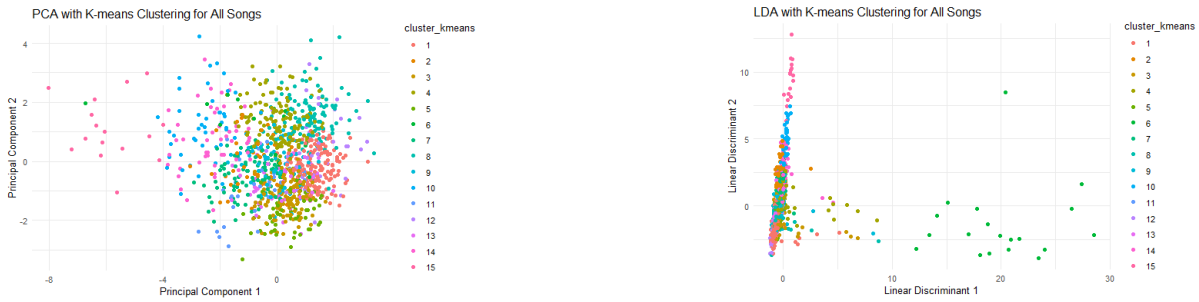

Fig. 2.5 PCA with K-means Clustering for All Songs(k = 15) & LDA with K-means Clustering for All Songs(k = 15)

IV. Methodology

System Overview

The music recommendation pipeline we developed incorporates both the musical features and elements of collaborative filtering recommendation. The main component of the pipeline is an encoder-like neural network that models user preferences based on

their playlist. The essence of this recommendation process involves using the encoder to capture the intricacies of musical tastes and provide personalized song suggestions that sound alike to the user's preferences, instead of solely relying on manually labeled genres or simple collaborative filtering. A detailed description of the components of our project pipeline and the underlying model architecture will be given below.

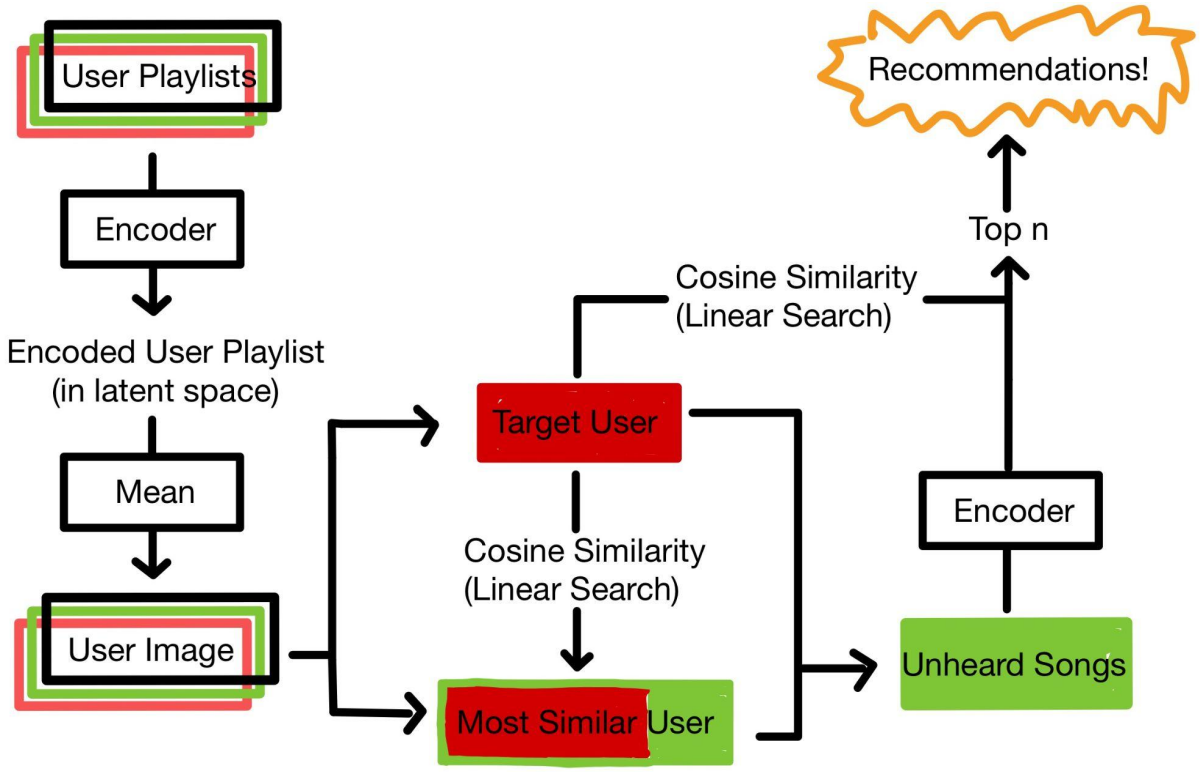

Fig. 3.1 Recommendation Pipeline

Data Preprocessing

The initial step involves normalizing the data with mean = 0 and s.d. = 1 to allow equal treatment of each musical feature when inputting them into the encoder. The features of interest are: *Acousticness*, *Danceability*, *Energy*, *Instrumentalness*, *Liveness*, *Loudness*, *Speechiness*, *Valence*, *Tempo*, *Time*, *Signature*, *Key*, and *Mode*. Selecting these musical features will ensure the embedded vector of each song in latent space is defined by how each song will sound to the user, instead of pre-defined genre names. Then, each song from a user's playlist is encoded into a latent space representation using a neural network-based encoder model to obtain the latent space representation of each user's playlist.

User Representation in Latent Space

After encoding, we average all the latent vectors corresponding to the songs in a user's playlist to derive a single latent vector that represents the user's overall music

preference, termed the "user portrait." This averaged latent vector captures the essence of the user's musical tastes by representing the average song preferred by the user.

Similarity Measurement and Recommendation Generation

For a given target user, the system computes the cosine similarity between the target user's portrait and the portrait of all other users. By comparing the similarity between two users' user portraits, we're analyzing how similar their music tastes are. The most similar users are identified, and the songs unheard by the target user are retrieved into a list. These songs are then ranked based on their cosine similarity to the target user's portrait, and the top-ranked songs are compiled into a recommendation playlist.

Additionally, the system can promote music discovery by randomly sampling songs and ranking them as potential recommendations, thus encouraging exploration of less popular tracks that would've otherwise appear far less in playlists than popular songs.

Model Architecture

Our encoder model is trained and structured similarly to an autoencoder. The first four layers of the model function as the encoder, and the fifth layer serves as the decoder. The input data consists of 12-dimensional vectors representing musical features as described above. The encoder compresses this information into an 8-dimensional latent space, and then the final layer of the model reconstructs the latent vector back into the original 12-dimensional space.

The model is trained using a dataset split into training and validation with a ratio of 8:2 to ensure maximum usage of data in training the model. We employ mean squared error (MSE) as the loss function and use the Adam optimizer for training. The training process focuses on making the encoder effectively condense the song features into the latent space, and then decode the condensed information back to its original form with a simple layer. This ensures that the decoder condenses information in the original vector in the most precise way to capture the hidden relationship so that it can be decoded with a simple dense layer. After training, the decoder layer is removed, and the remaining layers serve as the standalone encoder used in the recommendation pipeline.

```
model <- keras_model_sequential() %>%
  layer_dense(units = 64, activation = 'relu', input_shape = ncol(normalized_train)) %>%
  layer_dense(units = 32, activation = 'relu') %>%
  layer_dense(units = 16, activation = 'relu') %>%
  layer_dense(units = 8, activation = 'linear') %>%
  layer_dense(units = 12, activation = 'linear')
```

Fig. 3.2 Model Structure

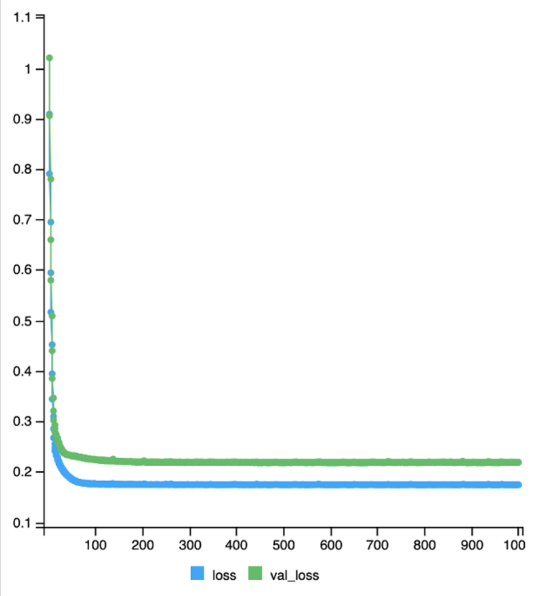

Fig. 3.3 Train Loss vs Epoch and Validation Loss vs Epoch

V. Model Evaluation

In the emerging field of music recommendation systems, the challenge often lies not only in accurately predicting user preferences but also in verifying the consistency of these preferences in the face of changing input data. Our research analyzes how collaborative filtering can be enhanced by focusing on the user's intrinsic preferences rather than just the overlap in the user's playlist.

We grouped users according to their preferred music genres - Group AB (Users A and B) preferred R&B, Alternative Metal, and Soul, and Group CD (Users C and D) preferred Pop Dance and EDM. The initial approach consisted of calculating the degree of overlap between users' playlists to visually assess similarity. Subsequently, the playlists were embedded in the latent space to derive an average vector representing each user's musical tastes. This innovative approach allows for the quantification of similarity through cosine similarity calculation.

	UserA	UserB	UserC	UserD
UserA	30	15	2	3
UserB	15	30	2	3
UserC	2	2	30	11
UserD	3	3	11	30

```
[1]  0.2700381  0.1636970  0.3219614 -0.2550547  0.2165844 -0.4795061 -0.1575767  0.3239458
[1]  0.37041593  0.05966347  0.15883933 -0.24323892  0.28839810 -0.37865247 -0.03959011  0.24732138
[1] -0.30174166  0.29556719  0.28038976 -0.11854850  0.10445561  0.12319179 -0.10524975  0.01434242
[1] -0.26662324  0.40889571  0.13734814  0.08285001  0.23011995  0.05356246 -0.04417573 -0.15518146
```

Fig. 4.1 Number of Overlap Songs And User Image (Mean of Playlist Embedding)

Preliminary findings indicate a high level of within-group similarity, with members of group AB having a similarity index of 93.9%. In contrast, intergroup similarity was significantly lower between members of groups AB and CD, highlighting different genre preferences. This observation highlights the effectiveness of embedding and overlapping techniques in distinguishing between users' different musical tastes.

	UserA	UserB	UserC	UserD
UserA	1.00000000	0.9390138	0.1582186	-0.00233108
UserB	0.93901377	1.0000000	-0.0742335	-0.14993976
UserC	0.15821856	-0.0742335	1.0000000	0.80537854
UserD	-0.00233108	-0.1499398	0.8053785	1.00000000

Fig. 4.4 Cosine Similarity Matrix

The study then moved to the second scenario where direct overlap was intentionally removed from the A and B and C and D playlists. This was done to determine whether users' intrinsic music preferences remained consistent even after common songs were removed from the analysis. The results of this setup confirmed the initial findings, with consistently high similarity scores between users in the same group. This suggests that the users' musical preferences are deeply rooted and not just an overlap of songs, indicating that their musical tastes are essentially the same.

	UserA	UserB	UserC	UserD
UserA	27	0	0	0
UserB	0	28	0	0
UserC	0	0	28	0
UserD	0	0	0	28

```
[1]  0.2402640  0.1824091  0.4021661 -0.2532258  0.2657686 -0.4706125 -0.1832196  0.3749842
[1]  0.04902706  0.06463343  0.05879319 -0.07155430  0.50501340 -0.20701286 -0.09756068  0.10131680
[1] -0.08686953  0.42905070  0.11946022 -0.18405566 -0.17944789  0.17342071 -0.03271212 -0.21077651
[1]  0.06720041  0.16218417  0.06771111  0.19020371 -0.03552739 -0.05995995 -0.07011284 -0.11276742
```

Fig. 4.5 Number of Overlap Songs And User Image (Mean of Playlist Embedding)

	UserA	UserB	UserC	UserD
UserA	1.00000000	0.69110002	-0.09642895	0.05206640
UserB	0.69110002	1.00000000	-0.29702552	-0.03371684
UserC	-0.09642895	-0.29702552	1.00000000	0.32556488
UserD	0.05206640	-0.03371684	0.32556488	1.00000000

Fig. 4.6 Cosine Similarity Matrix

In the application phase of the study, the efficacy of the recommendation system was tested by recommending new songs from User A's playlist to User B, whose musical tastes were closest to User A's. The recommendation system was used for the first time in the application phase of the study. The recommendations were well received despite the removal of overlapping tracks, which further validates the robustness of the similarity measure employed.

Title	Artist	Genre	Score
Ordinary People	John Legend	neo soul, pop, pop soul, urban contemporary	0.9496228
If I Ain't Got You	Alicia Keys	neo soul, pop, r&b	0.9405397
Back To Black	Amy Winehouse	british soul, neo soul	0.7386518
Outta My System (feat. T-Pain & JohntÃ¡ Austin)	Bow Wow	dance pop, dirty south rap, hip pop, pop rap, r&b, rap...	0.7133201
Shadow of the Day	Linkin Park	alternative metal, nu metal, post-grunge, rap metal, r...	0.6662985

Fig. 4.7 UserA/UserB: "R&B", "alternative metal", "soul"

Put Your Records On	Corinne Bailey Rae	british soul, neo soul, pop soul, soul	0.9012613
I Luv Your Girl	The-Dream	dirty south rap, hip pop, r&b, southern hip hop, trap, ...	0.8192038
Diary (feat. Tony! Toni! Tone! & Jermaine Paul)	Alicia Keys	neo soul, pop, r&b	0.7986257
Me, Myself and I	BeyoncÃ©	pop, r&b	0.7795984
In Those Jeans	Ginuwine	contemporary r&b, hip pop, r&b, urban contemporary	0.7595309

Fig. 4.8 Representative Song of User A

The analysis highlights the potential of advanced modeling techniques to capture the essence of user preferences in recommender systems. It shows that even when obvious similarities are removed, underlying preferences (less obvious) continue to drive user satisfaction. These findings are critical to the development of recommender systems that can adapt and accurately reflect individual tastes, even in dynamic environments where user preferences may change or where different content needs to be assimilated into personalized recommendations.

VI. Conclusion & Limitation

According to the training and validation loss plot (Fig. 3.3), we can observe that, at times, validation loss is only about 0.05 higher than the training loss in general. Both validation loss and training loss decrease and stabilize after a certain point. Therefore, we conclude that although there is a slightly overfitting issue, but the model is overall in good fit.

Even though our model achieved a high accuracy on the testing set, the use of the algorithms for music recommendation has a few potential limitations. For instance, we take musical features as input for the collaborative filtering algorithm, but the most common way is to use likes and dislikes data as input for the model. The method works well in our simulated user data, but it’s necessary to test it on real user data.

Another limitation is due to the data security issue. We cannot access private user data including what songs are liked and disliked by the users. To solve the problem, we use simulation to produce groups of user portraits that contain their preferences for different music genres. However, variation might exist between simulated user data and real user data.

References

Hardesty, L. (2019, November 22). The History Of Amazon's Recommendation Algorithm. Amazon Science. https://www.amazon.science/the-history-of-amazons-recommendation-algorithm

Rackaitis, T. (2019, May 31). Introduction to Latent Matrix Factorization Recommender Systems. Medium. https://towardsdatascience.com/introduction-to-latent-matrix-factorization-recommender-systems-8dfc63b94875?gi=25b06a712a9a