Music Recommendation System

Sandeep Chowdary Kotapati (1225495817), Vinay Edupuganti (1225210948),
Venkata Bharath Bathini (1225452293), Ajay Kumar Janapareddi (1225475771),
Vishruthi Manne (1225578744)

December 8, 2022

Abstract

One of the great arts that almost every one enjoys is music. It possesses the rare ability to communicate profound emotions, inspire, and motivate change. Music has always been an important part of everyone's lives, and it has become even more so during the pandemic. As the communities fell apart, many people turned to music to help them cope. And, thanks to the Internet's rapid growth and penetration, listening to or discovering new music is now just a few clicks away. With such a vast catalog of music, the listener is always challenged to select and listen to the songs they enjoy the most. In this Project, we attempted to implement a music recommendation system based on specific user preferences, as well as to expand existing playlists with similar new music. In this writing, we would like to discuss the challenges we faced, methods we have implemented. We will also briefly talk about the results. Based on the results and our learnings, we will also discuss potential areas for future work in this domain.

1 Introduction & Motivation

The boom in the usage of Android devices and ubiquitous internet access has made it possible for people to easily access a wide range of music resources especially streaming. However, the sheer number of songs available can be overwhelming, and it can be difficult for individuals to choose from the plethora of options. Over and above this, music service providers like Spotify, Amazon Music also need effective ways to manage their catalogs and enrich user experience to the customers by putting forward new music through high-quality and personalised recommendations. This has created a strong need for effective music recommendation systems.

2 Problem Description

The main of this work is to implement a music recommendation system based on users' preferences deduced implicitly or explicitly from users' previous listening history. Music recommendation systems use different approaches to try to predict what users will like, and each has its own strengths and weaknesses.

In a **popularity-based model**, the system simply recommends the most popular items to all users. This is a simple and intuitive approach, but it may not provide personalized recommendations and may not take into account the individual preferences of users.

Collaborative filtering algorithms, on the other hand, try to predict the preferences of a user based on his interactions with the system i.e. the preferences of the user and similarity with other users. Collaborative filtering algorithms can provide more personalized recommendations.

Content-based models, on the other hand, try to recommend items based on the attributes of the items themselves, such as their genre, artist, or lyrics. These models can be effective in recommending similar items to ones that a user has already liked.

Regarding the dataset we are using for this project, it poses a challenge in the sense that it

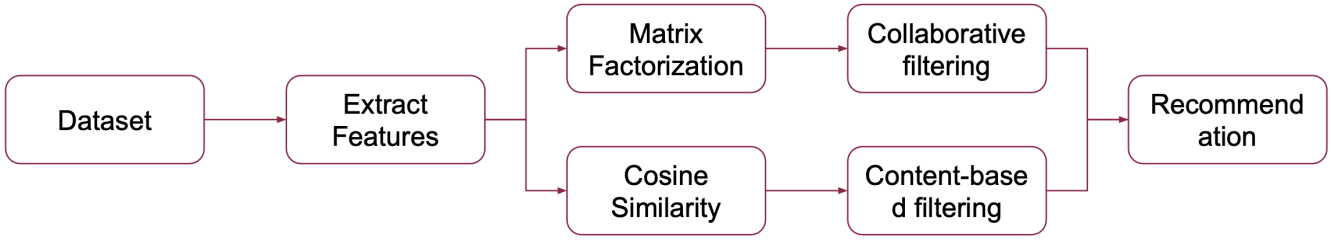

Figure 1: Project outline

doesn't contain explicit user feedback i.e. user preferences which are essential for using collaborative filtering techniques. So, we have to formalize an appropriate implicit feedback metric.

Overall, a music recommender system can be a useful tool for helping users discover new music that they may enjoy. Different algorithms can be used to provide personalized recommendations, but it is important to consider the strengths and weaknesses of each approach to find the one that works best for a particular system. In the following sections, we will discuss the process of Collaborative filtering in music recommendation systems, its advantages and shortcomings, and also how it addresses few of the limitations when combined with a content based method algorithm.

3 Methodology

3.1 Dataset

For the purposes of this project, we used the subset of the **Million Song Data set**[1]. It was created by Columbia University Laboratory for the Recognition and Organization of Speech and Audio and is open to researchers and developers. The dataset includes 48 million (userid, songid, play count) triplets collected from the listening histories of over one million users, as well as metadata for millions of songs (approx. 280 GB). The users on this dataset are anonymous, so their demographics and the timestamps of when they listen to music are not available. There are also no explicit user ratings available, but Instead the user's preference could be calculated implicitly through the play-count.

Processing of such a large dataset is highly intensive computationally. Due to the technical constraints, we used a subset of the Million Song Data Set (called MillionSongSubset) which is provided by the creators of the original dataset consisting of 10,000 songs (1%, 1.8 GB) selected at random for our project.

3.2 Exploratory analysis & Preprocessing

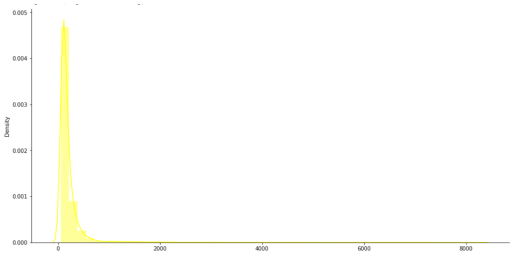

Figure 2: Graph showing the Fraction of songs listened vs No.of users

Doing some exploratory analysis of the dataset, we discovered that a song is listened to by an average of 200 users, with a minimum 48 and a maximum of 8277 users. The above graph from figure 2 shows the fraction of songs listened vs the no.of users. From this graph, it could be deduced that most of the songs in the dataset are listened to by a very less number of users. This becomes one of the critical factors in determining the sparsity of the User-song preference matrix.

As stated in the dataset description, there are no explicit user ratings available. So, we used the

paly-count i.e. no.of times the user listened to a particular song to generate the user's preference implicitly on a scale of 1 to 10 using the binning technique.

Ten categories will be defined. The representative rating of 1 will be applied to the original data values that fall within the range of 0 to 1; 2 will be applied to those that fall within the range of 1 to 2; and so on. When the original numbers are higher than or equal to 9, the final category will be applied.

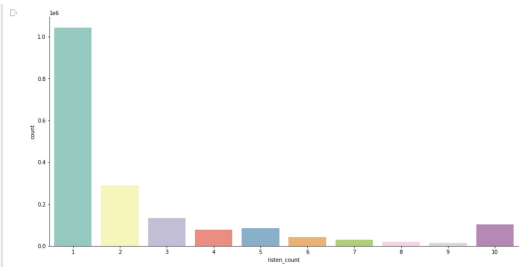

Figure 3: Graph showing no.of songs vs listen frequency. Implicit user ratings are generated using play count i.e. no.of times the user listened to a particular song

3.3 Colloborative Filtering

Using data gathered from a large number of users, collaborative filtering then makes predictions based on measures of similarity between users and songs. Largely, these fall under the categories of user-based and song-based models. The guiding principle of collaborative techniques is that it is sufficient to find comparable people or similar items after processing previous user-item interactions through the system in order to make predictions.

Users with similar listening histories, or those who have previously listened to the same songs, are more likely to share interests and more likely do so in the future, according to the **user-based similarity model**. For example, if there are two users A, B who have comparable musical interests and there is a song that User A enjoys but User B has never listened to, then it is more likely that User B would also enjoy that particular song.

According to the **item-based model**, songs that are frequently listened to together by certain users have a tendency to be similar and are more likely to be listened to together in the future by another user.

In our project, we used Matrix factorization[2] to predict the rating a user would give to a particular song based on which we could put forward the recommendation.

3.4 Matrix factorization

Matrix factorization is a technique to represent users and songs in a lower-dimensional latent space and thus reducing the sparsity of the matrix. We use matrix factorization to decompose the user-song matrix into lower dimensions i.e., it will be written as a product of three matrices. User-Song matrix of dimension (n x d) consists of ratings of songs given by each user, where n is the number of users and d is the number of songs.

Decomposing the User-song matrix A into a latent feature space that relates users and songs is given as follows

$$A = U.\Sigma.V^T \tag{1}$$

This method is called **singular value decomposition** for Matrix Factorization. Here, A is an $m \times n$ dimensional matrix, U represents user ratings of latent features and is $m \times r$ column orthogonal dimensional matrix, Σ is the weight-age of each latent feature and is a $r \times r$ dimensional diagonal matrix and V represents song ratings of latent features and is $n \times r$ column orthogonal dimensional matrix. We chose r latent features. There are always smaller dimensional matrices U, V, Σ that are a true representation of the original sparse matrix, regardless of how big A the matrix is. [3]

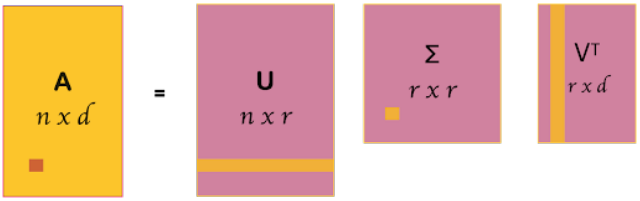

Figure 4: Singular Value Decomposition

We can say that the singular value decomposition is unique if the matrix A is fully defined. But

here A(user-song matrix) is a sparse matrix (with many missing values) as described in section 3.2 of this report. To solve this problem, we randomly initialise the matrices $U, V \& \Sigma$ and update the values until we the product $U.\Sigma.V^T$ converges to A, by using gradient descent.

3.4.1 Example of SVD with gradient descent

To perform the matrix multiplication, we first fill the matrices U, Σ and V^T with random values and then calculate A' by multiplying U, Σ and V^T and update the values as per the calculated gradient by the equation $U'(i) = U(i) + \alpha \times (actual - predicted)V(i)$ and similarly update V and Σ, where α is the learning rate.

$$\begin{pmatrix} 10 & \times & \times \\ 6 & 5 & \times \\ 8 & \times & 6 \\ \times & 4 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 2 \\ -1 & 1 \\ -2 & 1 \end{pmatrix} \times \Sigma \times \begin{pmatrix} 3 & 1 \\ -1 & 2 \\ -6 & 1 \\ 5 & 3 \end{pmatrix}^T \tag{2}$$

$$A \quad = \quad U \quad . \quad \Sigma \quad . \quad V^T$$

The above matrix multiplication *Eq. (2)* shows randomly initialised matrices U, V, Σ and their updated values by back propagation. Please note that Σ is having diagonal values to be 2, 2 in this example. This way we update the weights until convergence.[4]

3.4.2 Limitations

Because of the possibility of the "Cold Start Problem," which affects either songs or users, Matrix factorization alone couldn't be used in recommendation engines. As a result of which we cannot provide suggestions for new users or songs. This is because, for a new user, we don't have any music-listening history and for a completely new song, the listening frequency is zero. The "song-cold-start" problem can be effectively solved by combining Matrix factorization technique with other simpler methods like Rank Based or Content-based Recommended systems which we will discuss in later sections.

3.5 Content-based Filtering

In content-based filtering, we use song characteristics to suggest other songs that are similar to the user's preferences, which we learn via observing their listening habits. The term content alludes to the characteristics of music that a user enjoys. Content-based recommenders[5] can be thought of as user-specific categorizations. To design a content-based recommendation system we must first generate an object representation by extracting characteristics from the song. Second, establish a function that simulates what humans see as an item-item similarity among various song representations. We also use the information given by a user about their music preferences when they first created an account and signed up on the platform.

We build a feature matrix using the song data, where each row corresponds to a song and each column corresponds to a feature. The user profile is likewise shown in the same feature space area. Now, the algorithm will suggest music that is appropriate for the user. The similarity score between the user profile row and the song feature rows is used to determine this. To do this, we compute the cosine similarity statistic. Cosine Similarity is defined mathematically as the cosine of the angle created by two n-dimensional vectors in an n-dimensional feature space. We suggest the songs with the highest scores after computing the dot product of the two vectors. The more the value of the dot product, the greater the similarity and relevance of those songs to the user. These suggestions are particular to this user as this model did not utilize any data about other users.

$$Similarity(A, B) = \frac{\Sigma_{i=1}^{n} A_i \cdot B_i}{\sqrt{\Sigma_{i=1}^{n} A_i^2} \cdot \sqrt{\Sigma_{i=1}^{n} B_i^2}}$$

The concept of recommendations using content-based filtering doesn't require any input from other users. The suggested music is therefore very specific to the user. Another advantage of this approach is that both new and well-known songs are equally likely to be recommended. Thus, the issue of the song cold start is being addressed.

The method's drawback is that it tends to become highly genre specific and restricts the scope for variety of recommendations. The major limitation is that the User cold start problem still exists.

3.6 Hyperparameter Tuning & Model Training

Finding a list of ideal hyperparameter values for a learning algorithm and using this tuned algorithm on any data set is hyperparameter tuning. The model's performance is optimized by using that list of hyperparameters, resulting in better results with minimal errors. For our project, we sued Grid search to get the optimal combination of hyperparameters (no.of latent features, learning rate, regularization parameter, no.of epochs) [6].

As determined by running the Grid Search algorithm[7], the stochastic gradient learning rate was taken as 0.001, 0.1 to be the regularization term and the number of latent factors to be 120 while the number of epochs for training was 100.

To ensure that our recommendations don't get skewed we used cross-validation technique. our training and testing data was split in 70:30 ratio and re-sampling was done in five iterations and finally, the average of the values was taken to get optimal results.

After determining the hyperparameters of the model, we split the dataset into train set and a test set with a test_size of 0.25 and trained the model.

4 Observations & Results

The evaluation metric[8] we have used for collaborative filtering is **RMSE(Root Mean Square Error)** which is calculated by taking the square root of means of errors between the predicted recommendation matrix and the original user-song matrix for each element.

The content-based algorithm matches the user's interests based on cosine similarity. The songs that the user gets recommended mostly match according to the most listeneplaylist1.pngres of the user's playlist. For content-based filtering, we require feedback from the user on how the recommendations match to his/her original playlist. These parameters can be Click-Through rate, listening frequency and average listening duration etc.

```
song = 'I believe in miracles'

new_recommendations = model.make_recommendation(new_song=song, n_recommendations=10)

I believe in miracles
Starting the recommendation process for I believe in miracles ...
... Done

print(f"The recommendations for {song} are:")
print(f"{new_recommendations}")

The recommendations for I believe in miracles are:
Nine Million Bicycles
If You Were A Sailboat
Shy Boy
I Cried For You
Spider's Web
Piece By Piece
On The Road Again
Blues In The Night
Blue Shoes
Thank You Stars
```

Figure 5: Recommended Playlist using SVD collaborative filtering

```
final_algorithm = SVD(n_factors=120, n_epochs=100, lr_all=0.001, reg_all=0.1)
final_algorithm.fit(trainset)
test_predictions = final_algorithm.test(testset)
print(f"The RMSE is {accuracy.rmse(test_predictions, verbose=True)}")

RMSE: 2.1655
The RMSE is 2.165480132330792
```

Figure 6: Error in SVD

The RMSE value for collaborative filtering was **2.165** after training the model with the optimized hyper-parameters of latent factors=120, epochs=100, learning rate= 0.001 and regularization constant = 0.1.

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	live
155235	Rock	Jefferson Airplane	Embryonic Journey	2zbGXr2XCcRR40R7ymEENc	54	0.000132	0.000128	1.0	0.000128	0.000133	D	0.0
154175	Rock	Simon & Garfunkel	A Hazy Shade of Winter	2GB8OypbvrvCee61FKx5dp	58	0.000082	0.000085	1.0	0.000086	0.000081	D	0.0
154365	Rock	The Beatles	Don't Bother Me - Remastered 2009	7y1hMqXNa0dKQLZH7CKbUG	57	0.000070	0.000069	1.0	0.000071	0.000066	D	0.0
226335	Rock	AJR	Call My Dad	5qKTSIIBFkJ2gLJqSIRt1O	51	0.000085	0.000083	1.0	0.000081	0.000079	D	0.0
224464	Rock	alt-J	Dancing in the Moonlight (It's Caught Me in It...	2fpfApt0LZiOWybYlRVXTB	56	0.000126	0.000123	1.0	0.000121	0.000120	D	0.0

Figure 7: Recommended playlist using Content filtering

We observed that **Cold Start problem**[9] is the major challenge that any recommendation system would face. If we dwell deeper, we could realize that this problem could be broadly classified into two types: User Cold Start (When a new user is onboarded) and Item(Song) Cold start(When a new

song is added).

Item Cold start. Whenever a new song is added to the platform, it has zero user interaction and is thus ghosted by the recommendation engine. This potential problem could be addressed by incorporating a Content-based recommendation approach.

User Cold start. Whenever a new user is onboarded onto the platform with no know interests/preferences, a personalised recommendation is not possible. And also, this problem not only occurs with new users but also with existing users in some scenarios. For example, a user's preference might change from time to time in which case his ratings previously obtained may not accurately reflect his current preferences. This problem could only be addressed by considering other factors during recommendations like date, and time of the day and employing other approaches like recommending trending songs.

Finally, we opine that a **Hybrid recommender system** would be ideal to deal with the above-described cold start scenarios and generate recommendations.

5 Related work

The most effective recommendation algorithms include collaborative filtering, which has been demonstrated by a plethora of research and practical applications.[10]. The early CF systems rely on the opinions of members of a small group, such as a workgroup in an office.

However, the streaming platforms like spotify, Amazon Music, Youtube Music cannot rely on such models. Subsequently, algorithms like Group Lens[11], Video Recommender[12] which identify the groups of similar users by using their preference metrics derived explicitly or implicitly have come up. The advantage of CF-based recommender systems is that they can handle any type of material and recommend any things, even ones that are dissimilar to those previously seen, because they rely on the recommendations (ratings) of other users. The rising use of this method has nevertheless revealed some drawbacks, including sparsity, new user issues, and new item issues. Many approaches like dimensionality reduction, hybrid approaches are proposed.

Recently with the emergence of neural networks and Deep learning approaches, many deep learning approaches are also proposed[13]. Based on our research, by far Netflix's recommendation algorithm is the best recommendation algorithm. It tracks many data points like time, age, gender, browsing & scrolling behaviour to name a few.

6 Conclusion

In this project, we built two models for music recommendation systems. one based on the idea of collaborative filtering and the other based on content-based filtering techniques. We believe that combining both models to realise a hybrid recommender system would be much more efficient. Our initial plan was to build a hybrid recommender system, but couldn't fully integrate both models to generate the weighted result as a playlist. Now, we would like to continue working on building the hybrid recommender and also we would like to explore whether using deep learning techniques as proposed by some of the recent state-of-the-art papers could effectively solve the cold start scenarios described above.

References

[1] Thierry Bertin-Mahieux et al. "The Million Song Dataset". In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.

[2] Yehuda Koren. *Recommender system utilizing collaborative filtering combining explicit and implicit feedback with both neighborhood and latent factor models*. US Patent 8,037,080. Oct. 2011.

[3] Cameron Wolfe. *Building a music recommendation engine with probabilistic matrix factorization in Pytorch*. Mar. 2019. URL: https://towardsdatascience.com/building-a-music-recommendation-engine-with-probabilistic-matrix-factorization-in-pytorch-7d2934067d4a.

[4] Jae Duk Seo. *Step by step backpropagation through singular value decomposition with code in tensorflow*. Nov. 2018. URL: https://towardsdatascience.com/step-by-step-backpropagation-through-singular-value-decomposition-with-code-in-tensorflow-8056f7fbcbf3.

[5] *Building a music recommendation engine. Building a Music Recommendation Engine. - Tanmoy Ghosh Blog. Code is inspired from this blog*. July 2022. URL: https://www.section.io/engineering-education/building-spotify-recommendation-engine/.

[6] Péter Szabó and Béla Genge. "Hybrid hyperparameter optimization for collaborative filtering". In: *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE. 2020, pp. 210–217.

[7] *The keys to creating a collaborative-filtering music recommender system - Inzaugarat Blog. Code is inspired from this blog*. July 2022. URL: https://towardsdatascience.com/the-keys-building-collaborative-filtering-music-recommender-65ec3900d19f.

[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. "Performance of recommender algorithms on top-n recommendation tasks". In: *Proceedings of the fourth ACM conference on Recommender systems*. 2010, pp. 39–46.

[9] *How recommendation systems tackle the Cold Start Problem - YUSP Blog*. July 2022. URL: https://www.yusp.com/blog-posts/cold-start-problem/.

[10] Seok Kee Lee, Yoon Ho Cho, and Soung Hie Kim. "Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations". In: *Information Sciences* 180.11 (2010), pp. 2142–2155. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2010.02.004. URL: https://www.sciencedirect.com/science/article/pii/S0020025510000587.

[11] Joseph A Konstan et al. "Grouplens: Applying collaborative filtering to usenet news". In: *Communications of the ACM* 40.3 (1997), pp. 77–87.

[12] Will Hill et al. "Recommending and evaluating choices in a virtual community of use". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1995, pp. 194–201.

[13] Maxim Naumov et al. "Deep Learning Recommendation Model for Personalization and Recommendation Systems". In: *CoRR* abs/1906.00091 (2019). arXiv: 1906.00091. URL: http://arxiv.org/abs/1906.00091.