Capstone Project Report:

Music Recommendation System

By Dabidin Keshika

Executive Summary

This project proposes to build a music recommendation system for Spotify, which is an international music content provider online, with the use of the Million Song Data Set. Four types of recommendation algorithms are implemented and compared to choose the best recommendation systems: popularity-based recommendation systems, collaborative filtering recommendation systems, content-based recommendation systems and cluster-based recommendation systems. To optimize the systems, the RMSE and F1_scores (both defined in the report) have been calculated to evaluate the overall performance of the model. It is recommended to use a mix of the popularity-based recommendation system, item-item collaborative filtering recommendation system and content-based recommendation system to make recommendations in the platform.

Problem Summary

Internet-based entertainment companies rely significantly on the time spent by the consumers on their platform. It is essential for stakeholders to understand the type of content that is appreciated and enhances the time spent by and the experience of customers. Spotify is an international content provider which regularly releases a large volume and wide array of music online. Spotify users have little time to find and consume content that suits them. The Company has increased its market size considerably because of its ability to efficiently recommend the next song to be played by their consumers. In order to do that, smart recommendation systems are used. These can understand and process the users' musical taste and provide relevant suggestions to the platform.

We propose to implement and build a machine learning pipeline that combines content-based, collaborative recommendation engines as well as popularity-based systems to predict which songs a user will enjoy and recommend a list of 10 relevant songs to the user.

Solution design

The core data is the Taste Profile Subset released by the Echo Nest as part of the Million Song Dataset. There are two files in this dataset. The first file contains the details about the song id, titles, release, artist name, and the year of release. The second file contains the user id, song id, and the play count of users.

There are 563 unique songs ,232 unique artists and 3155 unique users in the dataset. As per the number of unique users and songs, there is a possibility of 3155 * 563 = 1,776,265 play counts in the dataset. But we only have 117,876 play counts in the data frame, i.e., not every user has listened to every song in the dataset. This creates the possibility of building a recommendation system to recommend songs to the users which they have not interacted with.

Overall solution design

Based on the result that we want to achieve; we can explore different recommendation techniques for the system. The following recommendation algorithms were implemented and explored for the following project:

1. *Popularity-Based Recommendation Systems*

Finding the top n songs for a recommendation based on the average play count of songs is possible with this algorithm and can contribute to recommending, for instance, the top songs of the day or the month. We can take the count and sum of play counts of the songs and build the popularity recommendation systems based on the sum of play counts.

2. *User-to-User and Item-to-Item Similarity-Based Collaborative Filtering*

Creating the user-item interaction data enables us to build a model by choosing a similarity-based recommendation system using methods such as cosine similarity

and KNN (K-Nearest Neighbours) to find similar users which are the nearest neighbour to the given user.

Parameters such as precision@k and recall@k, RMSE and F1_Score@k to evaluate the model performance. The Precision@k is the fraction of recommended items that are relevant in top k predictions. Recall@k is the fraction of relevant items that are recommended to the user in top k predictions. The F1-score@k is the harmonic mean of Precision@k and Recall@k and thus, takes into account both of the parameters.

It is interesting to look directly at the F1-score@k to consider both parameters. Moreover, we can change k and text different values to optimize the model. We can also change the threshold value and the test size to improve the model. The RMSE measures the differences between values predicted by the model and the actual values. The less the RMSE value, the more the model is performant. After optimizing the model, by adjusting the hyperparameters, we can propose songs to a given user.

3. *Model-based collaborative filtering (matrix factorization)*

It is a personalized recommendation system whereby suggestions are based on the past behavior of the user and it is not dependent on any additional information. We use latent features to find recommendations for each user. By building a matrix factorization recommendation system, we can make a prediction for user i to song j and make a prediction for the user i who has not listened to the song j. This model optimizes the model using SVD(Singular Value Decomposition).

4. *Cluster Based Recommendation System*

This algorithm works by clustering similar users together and recommending songs to a user based on play counts from other users in the same cluster.

We can adjust the number of nearest neighbors to vary the size of the cluster and optimize our predictions. According to the predictions made on the test set, we can choose one type of recommendation system over another.

5. *Content Based Recommendation Systems*

Another way to look at the problem is to use the song features such as the title, artist name, release year to make recommendations. We can concatenate that information into a text and use packages to analyze text format by tokenizing for example.

Now that we have explored the data, let's apply different algorithms to build recommendation systems.

Analysis and Key Insights

1. Popularity-Based Recommendation Systems

It is useful to recommend the top 10 songs on a daily basis. This method can be used to fill in a category of recommendations like 'Top songs of the week' for example. However, we need a more personalized recommended system for the user.

Here is an example of the top 5 songs with their average playing count and playing frequencies in the data set. Here the songs have been sorted out according to their playing frequency.

Song Identity	Average Count	Playing Frequency
8582	1.948069	751
352	2.184492	748
2220	2.220196	713
1118	1.817221	662
4152	1.930982	652

2. Collaborative Filtering Recommendation Systems

To compare different models, we can use the RMSE and the F1_Score@k since it takes into account both the precision and the recall of the model. To optimize the model, we would like to minimize the RMSE.

We will now implement the above in the user-user similarity-based collaborative filtering model, the item-item similarity-based collaborative filtering model and the matrix factorization model and compare those models.

The table below summarizes the RMSE, F_1 score, prediction for a song of actual play count= 2 obtained from the different baseline and optimized models.

The following table summarizes the results obtained:

Model	RMSE	F1_Score	Prediction for the song 1671 for the user 6958 of play count=2	Which one performed better between the optimized model and the baseline model?
User-user similarity-based baseline	1.0878	0.504	1.80	Baseline model
User-user similarity-based optimized	1.0596	0.482	1.53	
Item-item similarity-based baseline	1.0394	0.397	1.36	Optimized
Item-item similarity-based optimized	1.0328	0.489	1.96	
Matrix Factorization baseline	1.0252	0.498	1.27	Optimized
Matrix Factorization optimized	1.0141	0.502	1.34	

All the optimized models have a smaller RMSE than the baseline model. They are supposed to perform better, which is generally the case except for the user-user similarity based collaborative filtering. This might be because the play frequency was not taken into account.

From the above results, we can **conclude that the performance of item-item similarity-based optimized model works better since it provides a very close play count to the actual play count.**

Example: The top 5 recommendations for user_id 6958 with item_item_similarity-based recommendation engine: are the song ids : 2342, 5101, 139, 7519, and 8099.

3. Cluster Based Recommendation System

We will use the Co Clustering algorithm to make clusters of similar users together. The table below summarizes the the RMSE, F_1 score, prediction for a song of **actual play count= 2** obtained from the baseline and optimized model:

Model	RMSE	F1_Score	Prediction for the song 1671 for the user 6958 of play count=2	Which one performed better between the optimized model and the baseline model?
Cluster Based baseline model	1.0487	0.472	1.29	Optimized model
Cluster Based optimized model	1.0654	0.465	1.91	

The optimized Cluster based model gives a good prediction. We will have to choose between this model and the item-item similarity-based optimized model to make a personalized recommendation system. From the corrected ratings, we can see that the cluster-based models give higher ratings than the item-item collaborative filtering models. Also, the RMSE is lower in the item-item similarity-based optimized model than in the cluster-based model. We will thus keep the **item-item similarity optimized model for personal recommendations.**

4. Content Based Recommendation Systems

This recommendation used text information in the song description to recommend other similar items to the user. We had the title of the song, the release and the artist's name in our data. The recommendation system is working well. The songs recommended are of either the same group or of similar genre, for example, rock.

Example: The recommendations for the song with title 'Learn To Fly' are:

'Everlong', 'The Pretender', 'Nothing Better (Album)', 'From Left To Right', 'Lifespan Of A Fly', 'Under The Gun', 'I Need A Dollar', 'Feel The Love', 'All The Pretty Faces', and 'Bones'

Limitations and Recommendations for Further Analysis

For further analysis, we can make use of the original Million Song Dataset as well as other datasets to get other features and use it on the recommendations. Some examples of features to extend the study would be the loudness of the song and the hating score of the song.

We can also use more sophisticated implementation or try other levels of thresholds for optimization as well as trying new libraries.

Conclusion and Further Ideas

For the final solution design, I would propose to keep a mix of the content-based recommendation system and the item-item similarity optimized recommendation system as both models would complement each other. While the content-based recommendation system will find the key words and filter songs easily from the same artist or the same genre, the item-item similarity optimized recommendation system would find new songs for the user to discover and which the user might like. We might also include a top 10 song category using the popularity-based recommendation system so as to make the user also discover the trending songs. We must also explore different thresholds and their influence on the RMSE to find a better optimized model. Finally, we might improve the models if we get some more precise and extended data set on the songs like the timestamp or the loudness of the songs.