A Playlist Generation System using Spotify's Million Playlist Dataset

Arushi Chauhan Dhiganth Rao Padamnoor Rajeshwari Swaminathan Taejas Gupta

Abstract

Music recommendation is a key feature of Spotify aimed to improve both the listening experience of users and the popularity of artists and their creations. In this project, we use the Spotify Million Playlist Dataset (MPD) to create a recommendation system for automatic playlist continuation. We experiment with baseline models such as K-NN and Random Forests for identifying relevant features and move on to more sophisticated models including Collaborative Filtering, Co-occurence based recommendation and Markov Models. We identify relevant evaluation metrics to get our results and compare our model performances on various subsets of the dataset. We also train our best performing model on a larger subset of the dataset to obtain results that are comparable with the top-performing teams in the Spotify MPD challenge.

Our code is available on Google Drive.

1. Introduction

Our project aims to create a music recommendation system for automatic playlist continuation for Spotify. We generate personalised playlists for users that resonate with their listening habits and past preferences to cater to their unique taste in music. Our inspiration for this project came from the RecSys Challenge 2018 where the main evaluation task is to predict subsequent tasks in a playlist given an initial seed playlist.

The challenge uses various metrics to evaluate the quality of the predicted playlist such as R-Precision, Normalized Discounted Cumulative Gain (NDCG) and Recommended Song Clicks, as described in (Chen et al., 2018). We have used the inferences from (Chen et al., 2018), (van den Oord et al., 2013), (Hidasi et al., 2016) to strategize different approaches we take for predicting and improving our performances. We have experimented with various baseline models such as Naive Bayes, k-Nearest Neighbours and Random Forests. We subsequently move on to more complex models such as Collaborative Filtering (McFee et al., 2011) and co-occurence based recommendations to achieve improved performance.

Table 1. Basic statistics of the Spotify MPD.

PROPERTY COUNTS	VALUE
PLAYLISTS	1,000,000
TRACKS	66,346,428
UNIQUE TRACKS	2,262,292
UNIQUE ALBUMS	734,684
UNIQUE ARTISTS	295,860
UNIQUE PLAYLIST TITLES	92,944
AVERAGE PLAYLIST LENGTH (TRACKS)	66.35

2. Methods

2.1. Dataset

We used the Spotify Million Playlist Dataset (MPD), introduced by (Chen et al., 2018) as part of the Recommender Systems Challenge held in 2018. The dataset includes, for each playlist, key features such as playlist name, number of unique albums, tracks and artists in the playlist, number of playlist followers, and total play time of the playlist. Each track in the playlist further contains individual track information such as the position of the track in the playlist, the track, album and artist names and their Spotify URIs, and the track duration.

2.1.1. Data Augmentation

In order to make maximal use of the dataset provided, we leveraged the Python Spotify API (SpotiPy) to augment the dataset by scraping artist, track and album information. By augmenting the existing fields in the dataset, it was expected that better recommendations can be made as the models used now have more data to train and base predictions on. However, this was not the case - models trained on augmented data performed worse than models trained on the original subsets, as evidenced by the results shown in Table 2.

2.1.2. Data Pre-Processing

One aspect of the overall pipeline was formatting a subset of the input training data to conform to the challenge dataset format, where each playlist in the dataset had tracks "held out" as the ground truth to compare the predicted tracks for metric calculation. Specifically, since we were not using the challenge dataset provided by the competition (as the ground

Table 2. Baseline model performance on original dataset vs augmented dataset.

MODELS	R-PRECISION	NDCG	SONG CLICKS
TRACK IDS			
K-NN	0.050	0.021	27.786
RANDOM FOREST	**0.119**	**0.049**	**17.283**
ORIGINAL DATASET			
K-NN	0.025	0.011	33.918
RANDOM FOREST	0.082	0.034	22.877
AUGMENTED DATA			
K-NN	0.014	0.006	39.199
RANDOM FOREST	0.092	0.038	22.239

truth to compare our predictions to was not provided), we took a subset of the dataset which was not used for training the model and converted it into a format identical to the challenge dataset format. To achieve this, we performed stratified sampling using the known tracks from our training data, ensuring that each playlist in the curated test set has at least 90% of known tracks, while removing the unknown tracks. We then used the tracks held out from this dataset to compare with our predictions generated from the model.

In order to seamlessly test the predictions of our various models, we implemented a pipeline that takes in the predictions made by the model on various test sets (generated via the schemes mentioned above), and compares each prediction with the ground truth of the dataset. We made sure to align as closely as possible with the pipeline followed by (Chen et al., 2018), in that the formats of the predictions and the ground truth were of the form *playlistID: list of songURIs*.

2.2. Algorithms Used

2.2.1. BASELINE ALGORITHMS

We utilised three models for our baseline evaluations: Naive Bayes, k-Nearest Neighbors and Random Forest. We chose these models because they are widely used models from different model families (probabilistic, instance based and ensemble learning). In our midterm review, we attempted to train these models to predict the next artist and album in our playlist. These baseline models received the artists/albums of 10 tracks in a playlist as input and the next artist/album appearing in the playlist was the output to be predicted. We used a custom metric to evaluate the model performance which calculates the percentage of correct artist/album predictions i.e. that album/artist is present in the corresponding playlist irrespective of position. Naive Bayes performed the worst out of the three models and Random Forest performed the best, as can be seen in Table 3. Therefore, we excluded Naive Bayes and continued using K-Nearest Neighbors and Random Forest for further analysis.

Table 3. Baseline model performance for artists and albums.

	NAIVE BAYES	K-NN	**RANDOM FOREST**
ARTISTS	0.80	10.45	**18.30**
ALBUM	0.62	3.98	**10.70**

Higher values for artist recommendation validate the hypothesis that it is easier to predict artist than an album. We can extend this to say that predicting an album is easier than predicting a track. The reason for this is that an artist is composed of many albums and album is composed of many tracks. Thus, there are fewer unique artists and albums to predict compared to number of unique tracks which makes the prediction task easier.

Continuing with our study on baseline models, we decided to train them to output top 500 tracks instead of artist or album consistent with the MPD Challenge (Chen et al., 2018) and use metrics established in the challenge to compare their performance. We can see in Table 2 that Random Forest outperforms k-Nearest Neighbors for all dataset variations. Random Forest can capture more complex relationships and it has better generalization capabilities compared to k-Nearest Neighbors, hence Random Forest performs significantly better than k-Nearest Neighbors.

2.2.2. COLLABORATIVE FILTERING

We used collaborative filtering method as a two stage architecture to first filter out the top k playlists out of the training set which have the most similarity with the test playlist. Subsequently, we filtered out the 500 most popular (commonly occurring) tracks to recommend.

In our initial experimentation, we created our test playlists by eliminating 30% of the songs from each test playlist at random. We then find the most similar training playlists corresponding to these test playlists. We have used cosine similarity as a similarity index for collaborative filtering similar to (McFee et al., 2011). We then sorted the tracks based on their occurrences in this filtered data. We observed that for a small test set of 10 playlists we were able to narrow down successfully filter out 97% of the recommendations that were masked from the test playlist for the purpose of this analysis.

We further expanded our filtering to predict songs for 1000 test playlists based on their similarity with 1000 training playlists to get the top 50, 100 and 500 playlists and picked the 500 most popular songs out of these playlists to recommend.

2.2.3. CO-OCCURRENCE BASED RECOMMENDATION

We used a co-occurrence based model to extend our collaborative filtering approach to recommend tracks based on

the number of times two tracks occur together in the same playlist. This approach extends the idea that if a playlist contains a particular track, it is likely to have the tracks contained in other playlists containing this track. Furthermore, a track is more likely to occur in this playlist if it occurs in multiple playlists featuring this particular track.

The model parses the playlists in the training data and creates a sparse matrix containing the co-occurrence values of two tracks, incrementing the value in the matrix by one for every pair of tracks in a playlist. At the time of prediction, the model retrieves the rows in the matrix corresponding to each provided track in the playlist, and sums up the counts across columns to determine the total number of times each track occurs with all provided tracks. The tracks are sorted in descending order by this summed value, and the first 500 tracks that do not occur in the list of provided tracks are returned as recommendations. In case there are not enough recommendations to generate, the remaining recommendations are filled with the top 500 most frequently occurring tracks in the training data in order until 500 recommendations are complete.

2.2.4. Markov Model

We implemented a Markov Model to model the transitions between tracks in a playlist. Our motivation was to assess the importance of direct song-to-song transitions in a playlist. To model the transitions, we considered each track in a playlist as a state. We particularly worked with a test data where only the first track in the playlist was provided, since we were using a first-order Markov assumption, so any previous tracks would not contribute to the predictions.

We parsed each playlist in the training data, and for each consecutive pair of tracks, we increased the transition count by one, while increasing the track frequency to maintain a total count of states. At the time of prediction, we checked the transitions of the single given track in each test playlist, and predicted the next track as the track having the largest value for the transition frequency divided by the number of times the previous state occurs in the training data. The generated prediction was used as the track for making the next prediction, until 500 recommendations were generated. As with other models, in case enough recommendations were not generated, the top 500 most frequently occurring tracks in the training data were used to complete the list of 500 recommendations.

2.2.5. Neural Collaborative Filtering

We tried to implement a neural collaborative filtering model, which takes the track information along with other metadata associated with each track in the training data as a one-hot encoding and creates a 50-dimensional embedding to feed to a neural network containing two hidden layers with 64 neurons each and a ReLU activation between them. Although we tried training the model using only the first 1000 playlists, the large size of the one-hot encoding and the complexity of the model, even with the reduced number of layers, made the training infeasible, even with a GPU. Therefore, we were unable to generate our results with this model. Nevertheless, we have included the code for our model in the shared Google Drive folder.

2.3. Metrics Used

We used the same metrics as provided by (Chen et al., 2018). A breakdown of the metrics is provided in the subsections below.

2.3.1. R-Precision

The R-precision metric rewards those predictions which have a higher number of tracks in common with the ground truth predictions. Unlike the NDCG (Normalized Discounted Cumulative Gain) metric discussed in Section 2.3.2 or the Song Clicks metric discussed in Section 2.3.3, it does not take the position of the generated tracks into account. A prediction with a high R-precision could also have a very low NDCG or a high number of Song Clicks.

$$R\text{-precision} = \frac{|S_T \cap G_T|}{|G_T|} \tag{1}$$

2.3.2. Normalized Discounted Cumulative Gain (NDCG)

The NDCG metric takes into account the positioning of the recommended tracks and increases when relevant tracks are placed higher in the recommendation list. A track is deemed relevant if the track is also present in the ground truth recommendations. Once it is deemed relevant, it is calculated by the following formulae:

The DCG for each set of ground truth / predicted recommendations is as follows:

$$\text{DCG} = \text{rel}_1 + \sum_{i=2}^{|R|} \frac{\text{rel}_i}{\log_2(i)} \tag{2}$$

where rel_i corresponds to the relevance of the generated track recommendation to the set of ground truth recommendations. It is a binary value; 0 if the track is not present in the ground truth, and 1 if the track is present. R denotes the size of the list of predicted recommendations (which is always 500).

The ideal DCG (IDCG) is in the case where every track in the predicted recommendations is present in the ground

truth recommendations. The normalized DCG is hence,

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}} \tag{3}$$

2.3.3. Recommended Song Clicks

Recommended Song clicks is a measure which is more user-centric; it relates to a feature in Spotify which is Spotify's "Recommended Songs". For any playlist, Spotify can provide 10 recommendations of tracks to add to the playlist. This list can further be refreshed to produced more new tracks. The Recommended Song Clicks measure is the number of hypothetical refreshes needed for the predicted recommendations to produce a relevant track (which is in the ground truth). It is formalized by the following equation:

$$\text{clicks} = \left\lfloor \frac{\text{argmin}\{R_i : R_i \in G\} - 1}{10} \right\rfloor \tag{4}$$

3. Results

3.1. Comparing Model Performances

Table 4. Results of various models trained on 1000 playlists.

Model Name	R-Precision	NDCG	Song Clicks
Random	0.014	0.006	39.180
kNN	0.050	0.021	27.786
Random Forest	0.119	0.049	17.283
Collaborative Filtering	0.338	0.170	8.420
Top-500	0.338	0.171	8.387
Co-Occurence	**0.647**	**0.402**	**1.732**

To perform a comparison of various models and their performances, we work with a reduced training set consisting of the first 1000 playlists of the MPD, and tested it against 1000 playlists with 25 seed tracks provided. Apart from k-NN and Random Forest, we use two additional strategies to obtain a measure of baseline performance – Random and Top-500. For the Random strategy, we recommend 500 random tracks for each test playlist from the known tracks in our training data. For Top-500, we recommend the 500 most frequently occurring tracks in our training data in order for each of the test playlist, irrespective of the seed tracks in the playlists.

Table 4 shows the baseline performances along with the performance of various models. We observe that the random strategy has significantly worse results compared to all other strategies. While k-NN and Random Forest do better, they are still far from the results obtained by using the Top-500 heuristic.

Collaborative Filtering performs comparably to the Top-500 baseline, probably because the most frequent tracks obtained after identifying the top 100 similar playlists would be dominated by higher frequency tracks. From Figure 1, we can see that 94% of the tracks in the first 1000 playlists of the MPD occur in 5 or fewer playlists. Thus, they are unlikely to be recommended by a collaborative filtering model even if all such playlists make it to the list of most similar playlists.

Co-occurrence based recommendation does not suffer from this limitation as it models the direct co-occurrence of tracks, and we can see that it performs significantly better than all other approaches.

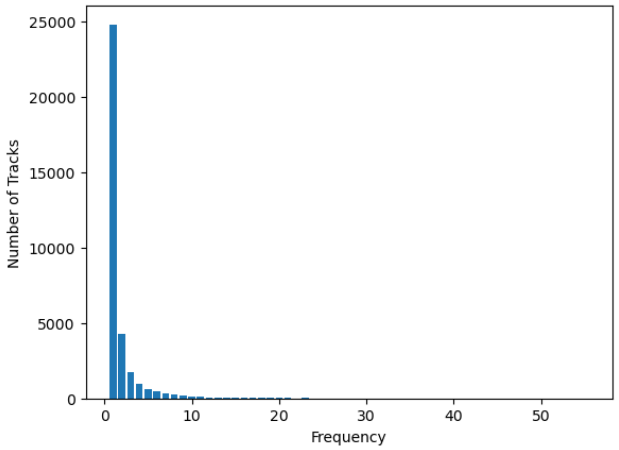

Figure 1. Number of tracks having different frequencies in the first 1000 playlists of the Million Playlist Dataset. 94% of the unique tracks occur with a frequency of 5 or less.

3.2. Modeling Transitions between Tracks

For modeling the direct song-to-song transitions, we experimented with Markov Models. We increased the training data to 10000 playlists as 1000 playlists would have insufficient transition information. For the training data, we only used the first track in the playlist as the seed to generate the subsequent tracks sequentially.

Table 5 shows the performance of the Markov Model along with some baseline measures and that of the Co-occurrence based recommendation model on the same test data. We see that while the Markov Model does much better than random prediction, it still performs significantly worse than the Top-500 baseline. This might be due to the sequential nature of the generation process – if at any stage during the generation a non-overlapping track is predicted, the subsequent generations are also likely to be non-overlapping with the ground truth.

The co-occurrence based recommendation model is able to perform surprisingly well, despite being given only a single

seed track per playlist. This hints at strong correlations between song preferences, indicated by their co-occurrence in multiple playlists.

Table 5. Results of models trained on 10000 playlists, given only the first seed track.

MODEL NAME	R-PRECISION	NDCG	SONG CLICKS
RANDOM	0.003	0.001	46.295
MARKOV	0.141	0.111	12.381
TOP-500	0.212	0.122	10.494
CO-OCCURENCE	**0.409**	**0.293**	**3.205**

3.3. Collaborative Filtering

We observed that the performance upon filtering the top 100 playlists and using them for recommendation is slightly better than using 500 playlists but mostly similar to using 50 playlists. This might be because filtering beyond 100 playlists leads to more generic recommendations and picking just 50 playlists might restrict the model from making valuable recommendations. Our results with this approach are shown in table 6.

Table 6. Performance of Collaborative Filtering models with different filtered playlist sizes.

MODELS	R-PRECISION	NDCG	SONG CLICKS
TOP 500 PLAYLISTS	0.331	0.167	8.467
TOP 100 PLAYLISTS	**0.338**	**0.170**	**8.42**
TOP 50 PLAYLISTS	0.335	0.170	8.302

3.4. AIcrowd Submissions

We used our best model, the co-occurrence based recommendation, to make our submission to the AIcrowd website hosting the Spotify Million Playlist Dataset Challenge. The challenge set contains 10000 incomplete playlists covering 10 scenarios (1000 playlists for each): (1) title only, no tracks, (2) title and first track, (3) title and first 5 tracks, (4) no title and first 5 tracks, (5) title and first 10 tracks, (6) no title and first 10 tracks, (7) title and first 25 tracks, (8) title and 25 random tracks, (9) title and first 100 tracks, and (10) title and 100 random tracks. The results obtained are the average performance across all 10 scenarios.

There were 3 submissions that we made: (1) Using the first 100k playlists with weighted co-occurrence scores using all non-augmented features, (2) Using the first 100k playlists with binary track information, (3) Using the first 250k playlists with binary track information. The results are shown in table 7, with the performance of the highest performing team of the ACM Recommender Systems Challenge 2018, vl6 (Volkovs et al., 2018), in the last row of the table.

Table 7. Challenge submission results with co-occurrence model along with original challenge winning model.

TRAINING DATA	R-PRECISION	NDCG	SONG CLICKS
100K (ALL FEATURES)	0.119	0.252	4.785
100K (TRACK INFO)	0.166	0.303	3.471
250K (TRACK INFO)	0.167	0.306	3.325
WINNING MODEL (VL6)	0.224	0.395	1.784

We see that using all the provided features hampers the recommendation capabilities of our model as it is unable to learn a good co-occurrence weighing scheme with the large variety of features, with many features involving track and artist information already being correlated with the track information. A binary weighted co-occurrence scheme is sufficient to generate recommendations that give high evaluation scores, approaching the territory of the winning model. Further, increasing the number of playlists used from training from 100k to 250k has negligible effect on the performance, even though the number of unique tracks encountered in the training data increases by 64%. This is likely because the newly encountered tracks have a frequency that is low enough for them to not make it to the predictions, and the relative co-occurrence values largely remain the same.

4. Process

4.1. Data Ablation

Data ablation helps us to understand the relative contribution of different components in machine learning pipeline. For our project, we utilised feature ablation with K-Nearest Neighbors and Random Forest models. By doing this, we expected to gain better insights into relative importance of features in our playlist and track data. We grouped features into playlist-related and album-related features. Album-related and track-related features were obtained from our data augmentation efforts and playlist-related features were already present in the dataset. We trained the models separately on different subsets of the augmented data with a missing feature group. From Table 8, we can see that Random Forest outperforms K-Nearest Neighbors for all dataset variations. The performance of baseline models on ablation datasets without playlist or album info are broadly the same and the models perform best with only track info. We can also see that models perform better on original dataset than on augmented and ablation datasets. This is consistent with our earlier observations and the observations cited in the Spotify challenge (Chen et al., 2018) paper. The organizers mentioned in the paper that the models posted in Creative Track (used augmented data) had worse results than models posted in main track (used original data). We can there-

fore conclude that playlist and album information can be excluded from our recommendation system and train our models on original dataset or only on track information.

Table 8. Baseline model performance with data ablation.

MODELS	R-PRECISION	NDCG	SONG CLICKS
ORIGINAL DATASET			
KNN	0.025	0.011	33.918
RANDOM FOREST	0.082	0.034	22.877
AUGMENTED DATA			
KNN	0.014	0.006	39.199
RANDOM FOREST	0.092	0.038	22.239
ONLY TRACK INFO			
KNN	0.050	0.021	27.786
RANDOM FOREST	**0.119**	**0.049**	**17.283**
WITHOUT PLAYLIST INFO			
KNN	0.013	0.006	39.773
RANDOM FOREST	0.085	0.034	24.306
WITHOUT ALBUM INFO			
KNN	0.013	0.006	39.773
RANDOM FOREST	0.081	0.032	23.858

4.2. Interpretation of Metric Scores

Looking at R-Precision, NDCG and Song Clicks as measures that describe the performance of our recommender systems, we feel that the recommendations produced by our systems, while not producing metric scores that are as high as that of the winning submissions, are more user-centric. Our recommendations are a mix of the most popular songs in the datasets we train the models on (popular meaning that these tracks occur the most frequently across different playlists) which are relevant to the playlist, and niche tracks that are specifically tailored to the types of songs already present in the playlist. These recommendations will ensure a satisfied consumer experience, which users exposed to new, niche songs which they might not have listened to otherwise, as well as mainstream songs which are popular in today's music world.

4.3. First Tracks vs Random Tracks as Seed

To check if the performance of models can vary when using the first tracks vs random tracks as the seed for predicting the remaining tracks, we used the co-occurrence based recommendation model trained on 1000 playlists as a proxy to compare the results obtained when making predictions using the first 25 tracks as the seed vs using 25 random tracks as the seed from the test playlists. We see that the model performs noticeably better when making predictions using random tracks in the test playlist. This could be because adjacent tracks in playlists are likely to share similar information such as the artist and album. Thus using random tracks as the seed captures more information about the variety of tracks in the playlist, leading to a wider representation of relevant recommendations.

Table 9. Performance of the co-occurence model with first 25 vs random 25 tracks as seed.

MODELS	R-PRECISION	NDCG	SONG CLICKS
FIRST TRACKS	0.586	0.342	2.560
RANDOM TRACKS	**0.647**	**0.402**	**1.732**

5. Contributions

5.1. Dataset Partitioning

It was found that working on the entirety of the MPD was challenging, due to computational constraints. Hence, analyses was restricted to smaller subsets of the dataset, considering subsets of 1000, 10000, 100000 and 250000 playlists at a time. While working on these subsets, we wanted to make sure that we were working with an appropriate amount of unique songs in order to generate predictions; as subsets with a smaller number of unique songs may not be suitable for training, owing to the smaller distribution of songs. The number of unique songs corresponding to the smaller subsets is shown in Table 10.

Table 10. Unique song statistics of MPD subsets.

NUMBER OF PLAYLISTS	NUMBER OF UNIQUE TRACKS
1,000	34,443
10,000	170,089
100,000	681,805
250,000	1,119,310
1,000,000	2,262,292

It is observed that the number of unique songs in the subset of the first 100000 playlists is a significant percentage of the number of unique songs in the entire playlist. However, due to computational constraints, we chose to work with the subsets containing 10000 and 1000 playlists to determine which models worked best with the data. Once the best model was determined, we trained the model on the 100000 playlist subset and made a submission to AIcrowd as well, the details of which are present in Section 3.4.

5.2. Data Augmentation

The data scraping process was exacerbated by rate limiting issues on Spotify's end. We managed to escape this by scraping limited batches of data at a time, and saving the information to use later. However, even this process took considerable time; in order to scrape album as well as artist information, each team member had to run the data scraping process in parallel, taking up around 7 hours of time (5 hours to scrape album information and 2 hours to scrape track information).

References

Chen, C., Lamere, P., Schedl, M., and Zamani, H. Recsys challenge 2018: Automatic music playlist continuation. *12th ACM Conference on Recommender Systems (RecSys '18)*, 2018.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. Session-based recommendations with recurrent neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

McFee, B., Barrington, L., and Lanckriet, G. Learning content similarity for music recommendation. *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 2207–2218, 2011.

van den Oord, A., Dieleman, S., and Schrauwen, B. Deep content-based music recommendation. *26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*, pp. 2643–2651, 2013.

Volkovs, M., Rai, H., Cheng, Z., Wu, G., Lu, Y., and Sanner, S. Two-stage model for automatic playlist continuation at scale. 2018.