Zero shot individual song recommendation using deep learning

Colby Parsons
University of Waterloo
200 University Ave W, Waterloo ON
caparson@uwaterloo.ca

Abstract

Automatic song recommendation is an important and difficult problem in the music domain. Streaming services such as Spotify, Apple Music, etc. have vested interest in improving various forms of song recommendations, from individual song recommendation, to automatic playlist continuation. In 2018 Spotify released the Million Playlist Dataset (MPD) as part of the 2018 ACM RecSys Challenge, a challenge focused on automated playlist continuation, with the goal of increasing research interest in song recommendation. This work takes a dataset from the automated playlist continuation domain and uses it to train a ranking model which is then used to recommend new tracks in the user song recommendation domain. We use deep neural networks to rank the tracks and evaluate our model using cross validation and a user recommendation experiment.

1. Introduction

Automatic song recommendation is difficult. In the 2018 ACM RecSys Challenge, the top performing team achieved roughly 0.39 NDCG [5] when evaluated on the MPD. This indicates that there is a lot of room to improve in this area of information retrieval. While the MPD was aimed at automated playlist continuation, this work tries to use the MPD as a training set for the problem of user based individual song recommendation. The assumption we make is that playlists will contain tracks relevant to the user who created it. Thus if we train a model to predict these groupings, we may be able to use it to predict relevant tracks when given a set of tracks deemed relevant to a given user.

We tackle this problem using a deep learning approach and achieve high NDCG in cross validation. Value is added in this research by attempting to apply a model trained using the MPD in a different song recommendation domain than automated playlist continuation. Many attempts at the 2018 ACM RecSys Challenge use similar supervised learning approaches to what we use in this paper [2,3,5].

2. Approach

To learn to rank songs, we trained a deep neural network using playlist data from the Spotify Million Playlist Dataset [1]. This dataset has a million playlists and contains just over 2.2 million songs across the playlists. We generated labeled data from this dataset which was then used for supervised learning. The deep neural network used consisted of multiple dense layers with rectified linear unit activation. The model was optimized during training with a gradient descent optimizer, Adagrad [7]. To create and train the model, python3 with Tensorflow was used. Since we focused solely on ranking we can trade some runtime efficiency for greater model precision. As such, we opted to focus on pairwise ranking and correspondingly, the model output scores for each (playlist, track) pair. These scores were then used to rank the tracks. We used pairwise hinge loss as the training objective and we trained the model over 30 epochs.

2.1. Data

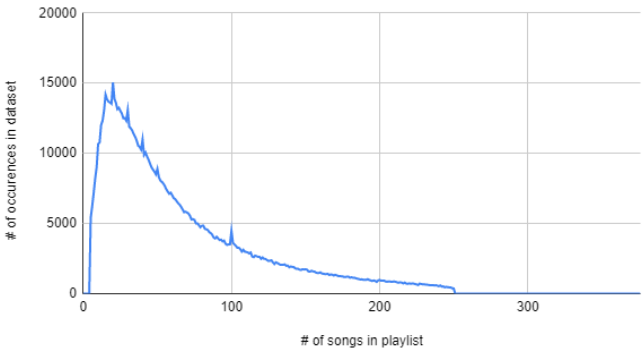
The data from the million playlist dataset included the titular million playlists along with barebones song data for each song in the playlist: song name, album name, artist name and duration. In pursuit of a more feature rich dataset we fleshed out the song data in these million playlists by fetching publicly available data from the Spotify API. The additional features added were: acousticness, danceability, energy, instrumentalness, key, liveness, mode, speechiness, tempo, time signature, and valence. These features are created by Spotify for tracks in their library using audio analysis. Some of these features offer more concrete description of a song such as tempo, time signature and key. The other features such as acousticness and danceability are metrics measured by a float between 0 and 1 that represent a more abstract measure of the track. As an example, danceability is described as “how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity”[8]. Each of the numeric features were normalized before being fed into the model, and each text

based feature was represented using a hash based embedding.

In creating the feature rich dataset, we also filtered out any songs that did not have song features available on the Spotify API. Just under 700 songs did not have features available, and in these cases we removed the song from all playlists it was associated with in the MPD.

2.2. Training

The model was trained using supervised learning. After the features were added to the playlist dataset, supervised training examples were made. We made the queries for our training/testing dataset by taking half of each playlist in the MPD. The half of the playlist taken as a query was selected randomly from the full playlist. The other half of the original playlist was padded with random songs from the full corpus until it was a set of 100 labeled songs and then it was shuffled randomly. This served as a miniature labeled corpus for each query we generated. We ensured that no duplicate tracks, or tracks from the query would be included in the 100 labeled tracks. To maintain a fixed input size, the query was padded to 50 tracks using a bag of words method. Additionally if a playlist in the dataset was larger than 100 tracks it was split into N smaller playlists of roughly equal size where N is the smallest value that results in sub-playlists of length less than 100.



A fixed query size of 50 was chosen based on the distribution of playlist length in the dataset. It was a good breakpoint so that the queries did not get too large but the majority of playlists also did not get subdivided. 78.3% of playlists were lesser or equal to 100 tracks in length. Thus only 21.7% of tracks had to be subdivided into smaller playlists. The largest playlist contained 376 tracks and the smallest contained 5. The median number of tracks was 50, the mean was 66, and the standard deviation of playlist length was 53.

3. Experiments

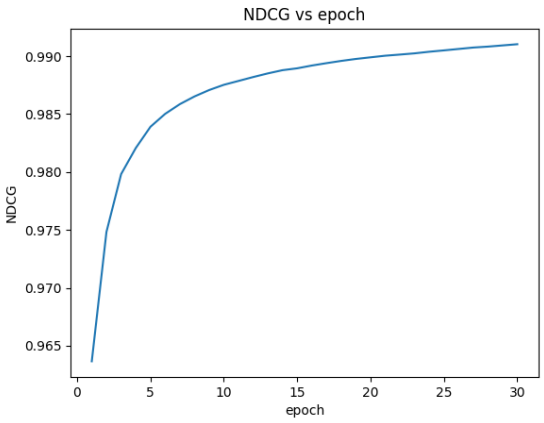
We performed two experiments: cross validation, and a user driven recommendation website experiment. The cross validation allowed us to see if our model performed well on test data and the user driven experiment is our attempt at testing our model in a real world setting.

3.1. Cross Validation

While developing and tuning the model, normalized discounted cumulative gain (NDCG) was used as the sole metric for evaluating this model. As such we performed 5-fold cross validation and measured NDCG. Given that MPD had a million playlists, each validation run was trained on 800,000 playlists and was tested on 200,000 playlists.

5-fold validation run	NDCG
1	0.9876
2	0.9874
3	0.9873
4	0.9875
5	0.9876

The average NDCG across the cross validation runs is 0.98748. These NDCG cross validation results are very high which may indicate that there may have been similar playlists between the training and test sets of the playlist dataset. NDCG was used as the evaluation metric as we are aiming to deliver high precision results with the assumption that users may not look far beyond the first few pages of results returned [6].



NDCG vs. Epoch during training

In this figure we see that we made marginal gains in NDCG throughout all 30 epochs of training and our 5-fold

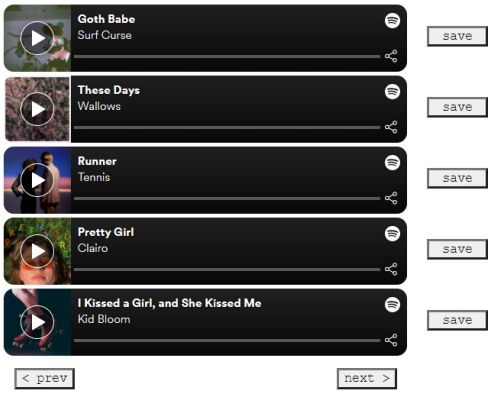
validation seems to indicate that we did not overfit as the NDCG of the cross validation is similar to our test examples.

3.2 User Evaluation

The training of this model was inspired by the RecSys Spotify challenge where given a partial playlist your model is tasked with completing the playlist. They evaluated the output of the model by comparing the recommended songs with the rest of the tracks from the full playlist that they sampled the corresponding partial playlist from. Points were given for exact matching tracks and tracks that had the same artist as tracks in the original playlist. This metric, while quick and easy, does not capture a large part of the essence of song recommendation in our opinion. Whether or not a user likes a song is very subjective. As such we claim that it is hard to effectively evaluate a song recommendation engine without user defined ground truth. They used 3 metrics with this ground truth in the Spotify RecSys Challenge, NDCG, R-Precision, and Click Rate. In this experiment we approximate click rate, but in retrospect it would have been good to track more detailed user data so that we could calculate all 3 metrics.

To evaluate our ranking model, we used Spotify's API to fetch a corpus of 500 songs seeded using tracks that the user had listened to recently. This corpus was then ranked using our model, with the query composed of the top 20 songs the user had listened to in the past 4 weeks, the top 20 songs from the past 6 months, and the latest 10 tracks the user had saved. In future work, one could vary the query per user to aim at improving the query selection for a given user. We chose to only retrieve 500 songs since we were serving the recommendations in real time through a website for this experiment. The average runtime of generating a recommendation was roughly 9 seconds, of which roughly 7.5 was spent requesting the appropriate song data from the Spotify API. As such increasing the number of tracks retrieved greatly increased the response time for generating a recommendation.

The top 50 results returned from the ranker were displayed to users on this website: https://plg.uwaterloo.ca/~caparsons/login.html. (Note: to use the recommender you need a Spotify account). The recommended tracks were displayed in pages of 5 and allowed playback so that users could sample the track. Basic usage metrics were tracked: the number of visitors, the number of songs saved, and the number of pages visited by users. This data gives us a rough idea of whether users liked the songs recommended. While there are no competing models to compare against for this metric, it serves as a small foray into user evaluation of this model.



One page of recommendations displayed to a user

Given more time and a larger user base it could have been beneficial to A/B test this model against a version where we do not rerank the 500 retrieved songs as a control to compare results against [9].

Number of users	165
Number of pages visited	587
Number of saved tracks	254

We can see that on average a user visited 3.56 pages, which in turn means on average a viewer viewed 17.79 songs. The average user saved 1.54 tracks. This supports our use of NDCG in cross validation since users on average did not look far beyond the first few pages of results. If we consider the user saving behavior to be the ground truth, then precision and recall is very low but there are many obscuring factors. It may not have been clear that the save button on the website saves the track to Spotify. Some users prefer to build playlists instead of saving songs to their library on Spotify, and as such may not opt to save songs from the web app. Further work will need to be done to make a more controlled experiment that accounts for these factors. This experiment clearly suffers from the informality of the format. On the flip side, marketing this project as a recommendation engine and keeping it informal allowed us to reach a large test user base of 165 users over the course of just 6 days of testing. This seems to indicate that user interest in song recommendation is significant and could be used to drive user recommendation evaluation in the future.

In retrospect we should have collected more fine grained data so that we could try and emulate one RecSys challenge's metric, click rate. Click rate is the number of refreshes of 10 shown recommendations needed before the first song is chosen. We didn't track the occurrence of the first save so we can't calculate this metric precisely but we

can calculate a decent approximation. Our pages are 5 songs in length so our average number of size-10 page refreshes per user session are 1.77 and given that each user on average saved 1.54 songs we can estimate over the course of their session they save 0.87 songs per size-10 song refresh. Thus we can predict that roughly 1/0.87 = 1.14 pages will need to be seen before a first song is saved, giving us an approximate click rate.

This recommendation engine was shared with users over social media. Overall feedback was positive and many users indicated that it recommended new songs that they found interesting. Additionally some constructive criticism was provided, aimed at improving the recommender. One common request was that the results returned commonly had the same artist showing up frequently and that the users wanted more artist diversity. Other requests included giving users the option to provide songs, a playlist or a mood as a query. Others asked to be able to save all tracks as a playlist instead of saving to liked songs as many users indicated that they preferred making playlists over liking songs. This feedback corroborates earlier work that indicates that to successfully recommend songs with users in mind, a nuanced approach is needed [4].

4. Conclusion

In this work we detailed our approach to individual song recommendation using a deep learning model trained using the Spotify Million Playlist Dataset. We incorporated a rich feature set, created a labeled dataset and trained a deep learning model to generate pairwise (playlist, song) scores. The model was evaluated using NDCG and 5-fold cross validation. We made a case for using users to evaluate song recommendations, and performed an informal foray into user testing of our model. The model was served to users over a website and usage data was collected. Our model achieved high NDCG in cross validation, and general user satisfaction with the recommendations. We approximate click rate and perform better on our user data by this metric, than the RecSys participants do by the same metric in the RecSys challenge.

5. Future Work

Future work improving user evaluation of song recommendations using A/B testing and controls can build upon our approach. Controls could include retrieval without ranking, and random song selection.

Other future work involving the Million Playlist Dataset could include an overview of feature usage and effect on recommendation results. A comprehensive look at how useful each feature is for recommendation (perhaps by omission from the model) could aid in feature selection for future song recommendation approaches. Additionally we discovered a significant bias towards popular music in our model trained by the MPD. Further work could be done to explore this bias and see if it should be mitigated.

Albums and playlists are often made by users with the “flow” of the songs in mind. In our reading of the related literature we did not see many attempts to replicate this in automated playlist continuation which lends way for future work.

Another angle that could be explored is explainable song recommendations. Perhaps adding explainability to song recommendations may increase users' engagement with results if they understand why something is recommended.

All in all song recommendation is still a growing field and there are many avenues to pursue in the area.

6. References

[1] C.W. Chen, P. Lamere, M. Schedl, and H. Zamani. Recsys Challenge 2018: Automatic Music Playlist Continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys ’18), 2018.*

[2] M. Volkovs, H. Rai, Z.e Cheng, Et al. Two-stage Model for Automatic Playlist Continuation at Scale. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys ’18), 2018.*

[3] H. Yang, Y. Jeong, M. Choi, J. Lee. MMCF: Multimodal Collaborative Filtering for Automatic Playlist Continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys ’18), 2018.*

[4] M. Schedl, D. Hauger. Tailoring Music Recommendations to Users by Considering Diversity, Mainstreaminess, and Novelty. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*

[5] H. Zamani, M. Schedl, P. Lamere and C. Chen. An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. In *ACM Transactions on Intelligent Systems and Technology, Vol. 10, No. 5.*

[6] Y. Wang, L. Wang, Y. Li, D. He, Et al. A Theoretical Analysis of NDCG Type Ranking Measures. In *Proceedings of the 26th Annual Conference on Learning Theory*

[7] J. Duchi, E. Hazan, Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In *The Journal of Machine Learning Research, Vol. 12.*

[8] Spotify API Web reference. https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features

[9] R. Kohavi, Y. Xu, D. Tang. Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing. In *Cambridge University Press 2000.*