

# Automatic playlist continuation using a hybrid recommender system combining features from text and audio

Andres Ferraro  
Music Technology Group -  
Universitat Pompeu Fabra  
Barcelona, Spain  
andres.ferraro@upf.edu

Dmitry Bogdanov  
Music Technology Group -  
Universitat Pompeu Fabra  
Barcelona, Spain  
dmitry.bogdanov@upf.edu

Jisang Yoon  
Kakao Corp.  
Korea  
jason.yoon@kakaocorp.com

KwangSeob Kim  
Kakao Corp.  
Korea  
lucas.kim@kakaocorp.com

Xavier Serra  
Music Technology Group -  
Universitat Pompeu Fabra  
Barcelona, Spain  
xavier.serra@upf.edu

## ABSTRACT

The ACM RecSys Challenge 2018 focuses on music recommendation in the context of automatic playlist continuation. In this paper, we describe our approach to the problem and the final hybrid system that was submitted to the challenge by our team Coplaya. This system consists in combining the recommendations produced by two different models using ranking fusion. The first model is based on Matrix Factorization and it incorporates information from tracks' audio and playlist titles. The second model generates recommendations based on typical track co-occurrences considering their proximity in the playlists. The proposed approach is efficient and achieves a good overall performance, with our model ranked 4th on the creative track of the challenge leaderboard.

## CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Information extraction*; *Music retrieval*;

## KEYWORDS

music recommender systems, automatic playlist continuation, collaborative filtering, content-aware recommendation, challenges

## ACM Reference Format:

Andres Ferraro, Dmitry Bogdanov, Jisang Yoon, KwangSeob Kim, and Xavier Serra. 2018. Automatic playlist continuation using a hybrid recommender system combining features from text and audio. In *Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys Challenge '18)*, October 2, 2018, Vancouver, BC, Canada. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3267471.3267473>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*RecSys Challenge '18, October 2, 2018, Vancouver, BC, Canada*

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6586-4/18/10...\$15.00

<https://doi.org/10.1145/3267471.3267473>

## 1 INTRODUCTION

The ACM Recsys Challenge 2018 [4] consists in building a system for prediction of missing tracks in a test set of playlists. The evaluation is done on different types of test playlists, considering a combination of the following properties:

- presence or absence of a playlist title;
- number of seed tracks in a playlist (between 0 and 100);
- position of seed tracks in a playlist (randomly selected tracks or a sequence of tracks sampled from the beginning of the playlist).

The challenge is organized in collaboration with Spotify music streaming service, who provided a dataset for the challenge, named Million Playlist Dataset (MPD). The dataset contains one million playlists created by the users of this service between January of 2010 and December of 2017. Another set of 10,000 playlists was also released for offline evaluation of the proposed systems. The challenge is divided into two different tracks. For the Main Track, participants can only use the information in the published dataset, while for the Creative Track the participants are allowed to use additional information to improve their systems.

With the transformation of the digital music industry, we now evidence the ever-increasing importance of music streaming services, and music playlists play an important role in music consumption on such platforms [6]. Existing research on music recommender systems has considered a number of related tasks, including Automatic Playlist Generation (APG) and Automatic Playlist Continuation (APC). The former consists in automatic creation of 20-40 tracks with some common characteristic or intention, while the latter considers inference of those properties from the existing playlists for their automatic continuation. Both tasks are very related to a more commonly studied problem of music recommendation lists [9].

As it is described by Schedl in [10], it is very important for the playlist continuation task to accurately identify the purpose or intent of the playlist, but it may be very challenging for many reasons. For example, a playlist can have more than one possible intention or its intention is impossible to identify due to the lack of necessary information. This suggests that using additional meta-data information about a playlist and its tracks may be beneficial.

Bonin and Jannach provide an overview of existing approaches to APG [3]. According to the authors, Collaborative Filtering (CF), a predominant approach in recommender systems, can be applied for playlist generation although it is not specifically designed for this task. In such approaches, one option is to consider playlists as users. For example, Hariri et al. [5] follow this approach using Matrix Factorization (MF) with Bayesian personalized ranking. However, such an approach is problematic for playlists with a small number of seed tracks and a hybrid system combining collaborative filtering with metadata of the tracks can provide better results [3].

Working on a solution, our intuition was that using a Matrix Factorization model for playlist continuation could give good results with a large dataset of playlists, but it was necessary to integrate metadata of the playlists and content information of the tracks to have better results for playlists suffering a cold-start problem. With this approach we try to identify the purpose of the playlist in order to generate new recommendations, for example, playlists with the same title or playlists sharing common genres identified from the audio are expected to have similar intentions. To improve this approach further, we combined it with another model which recommends the most probable tracks based on co-occurrence and proximity of tracks in playlists in the training data.

## 2 OUR MODELS

In this section we describe each of the models used to generate the recommendations and how they are combined.

### 2.1 Matrix Factorization model (MF)

The first model is a hybrid Matrix Factorization model [7] which generates representations of playlists and tracks based on their interaction and also based on content features describing the playlists and the tracks.

The features used for the playlists are based on their titles. After trying different representations, the best performance was achieved with the one-hot encoding of the normalized titles. The normalization consists in transforming text strings to lower-level chars and removing special characters (. , / # ! \$ % ^ \* ; : { } = \_ ` ~ ( ) @). This is the same normalization that is performed by the challenge organizers in the code provided with the MPD.

For the tracks, the features are computed from audio samples of 30 seconds retrieved from Spotify. To compute the features we use Essentia,<sup>1</sup> an open-source library for audio analysis for music information retrieval applications [2]. Specifically, we used high-level genre annotations generated by the Tagtraum<sup>2</sup> classifier model [1]. These annotations include probability estimates for each of the following 13 genres: Blues, Country, Electronic, Folk, Jazz, Latin, Metal, Pop, Rap, Reggae, RnB, Rock, and World.

The idea behind our hybrid factorization model is to learn interactions between the playlist titles and tracks as well as the relations between the track genres and the playlists. This means that when we need to continue a playlist with a small number of seed tracks (or no seed tracks at all), our model can make more accurate predictions.

We use LightFM<sup>3</sup> [7] with the Weighted Approximate-Rank Pairwise (WARP) loss function for the implementation of this model. LightFM learns representations of the playlist and track features. For a better expressivity of the model, we also include identity matrices for tracks and playlists as their features.<sup>4</sup> Using this model, the predictions are calculated by the dot product of the latent vectors of the playlists and the tracks.

WARP loss has been originally proposed as a memory- and time-efficient solution to train a system for identifying labels of images using very large datasets [3]. In our case, WARP samples tracks for a playlist and updates the representations (using stochastic gradient descent) only when the prediction is wrong, meaning that the sampled track is negative and was predicted higher than the positive tracks. WARP optimizes precision, and we expect this optimization to be correlated with R-precision, one of the metrics used to measure the performance of the systems in the challenge.

We optimized the parameters of the model in our local evaluation environment which is described in the next section. This included the dimensionality of the latent factors for representing tracks and playlists for which we considered a various number of factors between 30 and 300 with the best result being achieved when using 200 dimensions. Another parameter that we optimized is the L2 penalty for the regularization of the playlists features and the tracks features. After searching for the best parameters, the value 1e-6 was used for both cases. Finally, the number of epochs used to train the model was selected by searching between 50 and 200, the best performance was achieved by using 150 epochs.

In order to generate new recommendations for a playlist that we want to continue, we add this incomplete playlist when training the model to get its latent representation. We can then get a recommendation score for each track, multiplying its latent vector to the playlist's vector. Using this model we generate a list of recommended tracks for each playlist, limited to 4000 tracks excluding all the tracks that are already in the playlist.

### 2.2 Track proximity model (TP)

The second model performs recommendations based on the proximity of tracks in the playlists. We assume that the tracks located closer to each other in playlists are more likely to be a good match for recommendations. To this end, for each track in a playlist we count the interactions with all other tracks within a temporal window including 10 previous and 10 posterior tracks. We weight those interactions according to the distance inside the window and store those into a proximity matrix.

That is, for all playlists  $P = \{P_k\}$  and tracks  $T = \{t_i\}$  in the dataset, the track proximity matrix  $S_{i,j}$  is calculated as:

$$S_{i,j} = \sum_{\substack{P_k \in P: \\ t_i \in P_k, t_j \in P_k \\ |pos(t_i, P_k) - pos(t_j, P_k)| < d}} 1 - \frac{|pos(t_i, P_k) - pos(t_j, P_k)|}{d}$$

where  $pos(t_i, P_k)$  is the zero-based position index of a track  $t_i$  inside the playlist  $P_k$ , and  $d = 10$  is the the maximum position difference to consider in the window. Different sizes of windows were

<sup>1</sup><http://essentia.upf.edu>

<sup>2</sup><https://acousticbrainz.org/datasets/61265979-235e-42b9-9a99-243e600275e3>

<sup>3</sup><https://github.com/lyst/lightfm>

<sup>4</sup>Following the documentation online: <http://lyst.github.io/lightfm/docs/lightfm.html>

tested locally, but increasing the size of the windows makes this process much slower and also requires more memory.

To make recommendations, for each of the seed tracks in a playlist that we want to continue we combine the values in the proximity matrix and sort them. Finally we remove from this list the tracks that are already present in the playlist.

If  $X$  is the set of seed tracks in the playlist that we want to continue, the recommendation score for each track  $t_i \in T$  is defined by function  $g(t_i)$ :

$$g(t_i) = \sum_{t_j \in X} S_{j,i}$$

In the case when the playlist that we want to continue does not contain any seed track, we use a generic popularity-based recommendation. It is calculated by the same function  $g$ , but in this case the set  $X$  contains all the possible tracks of the dataset, thus  $X = T$ . The generated recommendation list is the same for all such playlists.

### 2.3 Fusion model

Finally, we combine the recommendations produced by the previous models using a rank fusion technique giving a weight to each component model ( $\alpha_{MF}$  and  $\alpha_{TP}$  for matrix factorization and track proximity models, accordingly). To this end, we normalize rank scores produced by our models and use a linear combination of those following [11], but with a different rank normalization as described below.

For a given playlist that we want to continue, if  $S_{MF}$  is the ranked list with the recommendations of the MF model and  $S_{TP}$  is the ranked list with the recommendations of the track proximity model, first we calculate  $M$  as the maximum between the list length of  $S_{MF}$  and  $S_{TP}$ :

$$M = \max(|S_{MF}|, |S_{TP}|)$$

For each track  $t$  in  $S_{MF}$  and  $S_{TP}$ , the function  $w$  gives the score that will be used to combine both lists:

$$w_{MF}(t) = M - r_{MF}(t),$$

$$w_{TP}(t) = M - r_{TP}(t)$$

The values  $r_{MF}(t)$  and  $r_{TP}(t)$  are zero-based index positions of the track  $t$  in  $S_{MF}$  and  $S_{TP}$ , respectively.

The ranking for the final position of a song  $t$  in the recommendations ( $r_f$ ) is calculated using the result of the following equation:

$$r_f(t) = \frac{\alpha_{MF}w_{MF}(t) + \alpha_{TP}w_{TP}(t)}{2}$$

After testing different weight values in our local evaluation environment described in the next section, we decided to use a  $\alpha_{MF} = 0.7$  and  $\alpha_{TP} = 0.3$ .

## 3 EVALUATION

For the evaluation the organizers released a dataset with 10,000 incomplete playlists (Challenge Set) covering 10 different playlist categories, each category containing 1,000 playlists. The categories are:

- playlists with only title;
- playlists with title and only the first track;

- playlists with title and the first 5 tracks;
- playlists with the first 5 tracks but without the title;
- playlists with title and the first 10 tracks;
- playlists with the first 10 tracks but without the title;
- playlists with title and the first 25 tracks;
- playlists with title and the random 25 tracks;
- playlists with title and the first 100 tracks;
- playlists with title and the random 100 tracks.

In order to evaluate the systems, participants were requested to submit a file with lists of 500 tracks recommended for each playlist in the Challenge Set. During the challenge the organizers published a leaderboard with the positions of the participants, based on a Borda Count score combining three different metrics. The organizers only used 50% of the Challenge Set to calculate the scores during the challenge and used the full set for the final evaluation results afterward.

### 3.1 Metrics

The metrics used to evaluate the systems are Normalized Discounted Cumulative Gain (NDCG), R-precision (RPREC) [8] and CLICKS. All the metrics are computed using the top 500 tracks recommended by each system.

NDCG is calculated from Discounted Cumulative Gain (DCG) and ideal DCG (IDCG):

$$NDCG = \frac{DCG}{IDCG}$$

where

$$DCG = rel_1 + \sum_{i=2}^{|R|} \frac{rel_i}{\log_2(i+1)}$$

$$IDCG = 1 + \sum_{i=2}^{|G|} \frac{1}{\log_2(i+1)}$$

Where  $R$  is the list of the playlist's recommended tracks, and  $G$  contains the ground-truth playlist tracks.  $|\cdot|$  denotes the length of the list of tracks and  $rel_i$  value is 1 if the track is the original playlist or 0 otherwise.

For calculating the metric R-precision only the first  $|G|$  recommended tracks are considered. Where for each playlist,  $|G|$  is the number of known relevant tracks. R-precision is calculated by:

$$R - precision = \frac{|G \cap R_{1:|G|}|}{|G|}$$

The metric CLICKS is the number of times a user would have to refresh the recommended list of tracks (of length 10) to get the first relevant track, the range for this metric is between 0 and 51, where 0 is the perfect score.

$$CLICKS = \lfloor \frac{\text{argmin}_i \{R_i : R_i \in G\} - 1}{10} \rfloor$$

To compare all solutions submitted to the challenge, the organizers use a Borda Count score combining system rankings according to each of the three metrics used (RPREC, NDCG and CLICKS). For each of the three rankings of  $p$  submitted systems the top ranked system receives  $p$  points, the second system receives  $p - 1$  points,

and so on. The system with the most total points is considered as the best performing.

### 3.2 Local Evaluation

In order to evaluate our system locally, we divided the MPD in train and test set. We selected 10,000 playlists for the test set following the same distribution of playlist categories as used for evaluation by the organizers.

Table 1 presents the results of our local evaluation. We can see that combining the models using the fusion method increases the RPREC score by 8% (with an absolute increase of +0.010), the NDCG score by 6% (+0.018), and also shows an improvement of CLICKS by 11% (-0.356). The score of all the metrics improved by combining the models for almost all playlist categories (except for the case when a playlist does not have seed tracks) and we can conclude that using fusion approach was beneficial. This simplified our proposed solution, as we did not need to consider applying the fusion method selectively for only some of the categories. Nevertheless, we see that our final model could be improved by using directly the recommendations from the MF model for the case when a playlist does not have seed tracks.

### 3.3 Submission scores

The organizers updated the scores on the leaderboard daily so we could submit multiple solutions to evaluate the performance of the models independently. Table 2 presents our results for the matrix factorization model and the final hybrid model.<sup>5</sup> We see an improvement of 4.6% for RPREC (absolute increase of 0.009), 4.9% for NDCG (0.017) and an improvement of 9.6% according to the CLICKS metric (-0.181). We can see that the absolute values of the differences in the models performance are similar of the values in the local evaluation.

It is important to note that these results are not the same as the final scores published on July 13th, 2018, as they are only calculated using 50% of the Challenge Set. Also note that for the final evaluation results provided by the organizers, the RPREC metric was adapted to give some reward for a partial match when a recommended track is from the same artist. For such a partial match, a weight of 0.25 is added to the numerator of the score.

## 4 CONCLUSIONS AND FUTURE WORK

In this paper, we describe our recommender system for music playlist continuation submitted to the RecSys Challenge 2018. Our system combines two different approaches. One approach is based on Matrix Factorization combining the information about the interactions between playlists and tracks with playlist features extracted from playlist titles and track features extracted from audio. The other approach is based on track proximity in the playlists, recommending the tracks that are most likely to appear together with (and close to) the seed tracks in a playlist.

Our system achieved the 4th position of the Creative Track of the challenge. Comparing our results in the leaderboard with the rest of the submissions we got a very good performance according

to CLICKS metric, finishing in the 2nd position, but a worse performance according to NDCG and RPREC. This suggests that our system may be good at finding the next track to continue a playlist, but not as good at finding all relevant tracks.

One advantage of our solution is that it is possible to incorporate more track and playlist features into the model and we expect that including more relevant information will improve the system. Given the time constraints of the challenge, we were not able to evaluate all combinations of audio features that we planned. We think that other audio features can improve the performance of our hybrid matrix factorization model and, therefore, improve the performance of the final system. We propose to address these ideas in the future work. In particular, we will consider high-level music features available in Essentia audio analysis library, such as acous-ticness, danceability, BPM, key, and moods.

The code of our system is open-source<sup>6</sup> and we encourage other researchers to experiment with our system and combine it with other solutions.

## ACKNOWLEDGMENTS

This research has been supported by Kakao Corp., and partially funded by the European Unions Horizon 2020 research and innovation programme under grant agreement No 688382 (AudioCommons) and the Ministry of Economy and Competitiveness of the Spanish Government (Reference: TIN2015-69935-P).

## REFERENCES

- [1] Dmitry Bogdanov, Alastair Porter, Perfecto Herrera Boyer, and Xavier Serra. 2016. Cross-collection evaluation for music classification tasks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*; (Aug 2016); New York City (NY); p. 379-85. ISMIR.
- [2] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Perfecto Herrera Boyer, Oscar Mayor, Gerard Roma Trepas, Justin Salamon, José Ricardo Zapata González, and Xavier Serra. 2013. Essentia: An audio analysis library for music information retrieval. In *14th Conference of the International Society for Music Information Retrieval*; (Nov 2013); Curitiba, Brazil.; p. 493-8. ISMIR.
- [3] Geoffray Bonnin and Dietmar Jannach. 2015. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys (CSUR)* 47, 2 (2015), 26.
- [4] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. 2018. RecSys Challenge 2018: Automatic Music Playlist Continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. ACM, New York, NY, USA.
- [5] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware music recommendation based on latent topic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 131-138.
- [6] Marc Hogan. 2015. Up Next: How Playlists are Curating the Future of Music. <https://pitchfork.com/features/article/9686-up-next-how-playlists-are-curating-the-future-of-music/>. Accessed on 12.07.2018.
- [7] Maciej Kula. 2015. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439* (2015).
- [8] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2010. *Recommender Systems Handbook* (1st ed.). Springer-Verlag, Berlin, Heidelberg.
- [9] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskis. 2015. Music recommender systems. In *Recommender systems handbook*. Springer, 453-492.
- [10] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. 2018. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval* 7, 2 (2018), 95-116.
- [11] Keshu Zhang and Haifeng Li. 2010. Fusion-based recommender system. In *13th Conference on Information Fusion (FUSION)*. IEEE, 1-7.

<sup>5</sup>We have not evaluated the track proximity model on its own during the challenge due to limitation on the number of allowed submissions per day.

<sup>6</sup><https://github.com/andrebola/creative-recsys-cocoplaya>

**Table 1: Local evaluation results. The best obtained results for each playlist category are marked in bold.**

Playlist category	Matrix Factorization Model			Track Proximity Model			Fusion Model		
	RPREC	NDCG	CLICKS	RPREC	NDCG	CLICKS	RPREC	NDCG	CLICKS
First 100 songs	0.111	0.268	1.633	0.098	0.235	2.33	<b>0.116</b>	<b>0.277</b>	<b>1.487</b>
Random 100 songs	0.201	0.411	0.411	0.176	0.363	0.692	<b>0.213</b>	<b>0.431</b>	<b>0.393</b>
First 25 songs	0.133	0.319	1.769	0.123	0.288	2.146	<b>0.141</b>	<b>0.334</b>	<b>1.607</b>
Random 25 songs	0.194	0.414	0.972	0.189	0.388	1.025	<b>0.212</b>	<b>0.441</b>	<b>0.621</b>
First 10 songs - with title	0.125	0.308	1.925	0.123	0.299	2.922	<b>0.140</b>	<b>0.333</b>	<b>1.675</b>
First 10 songs - without title	0.134	0.308	1.555	0.134	0.304	2.072	<b>0.147</b>	<b>0.329</b>	<b>1.195</b>
First 5 songs - with title	0.095	0.261	4.797	0.102	0.265	5.565	<b>0.110</b>	<b>0.285</b>	<b>4.182</b>
First 5 songs - without title	0.104	0.268	3.778	0.121	0.288	3.888	<b>0.123</b>	<b>0.298</b>	<b>3.113</b>
First song	0.109	0.252	5.120	0.123	0.268	5.515	<b>0.123</b>	<b>0.278</b>	<b>4.043</b>
No seed songs	<b>0.076</b>	<b>0.184</b>	<b>12.636</b>	0.015	0.065	24.689	0.053	0.159	12.800
All playlists combined	0.128	0.299	3.467	0.120	0.276	5.084	<b>0.138</b>	<b>0.317</b>	<b>3.111</b>

**Table 2: Scores of the submitted models during the challenge.**

Model	RPREC	NCDG	CLICKS
Hybrid-MF Model	0.193	0.350	2.065
Fusion Model	0.203	0.367	1.884