Music Playlist Continuation by Learning from Hand-Curated Examples and Song Features

Alleviating the Cold-Start Problem for Rare and Out-of-Set Songs

Andreu Vall
Johannes Kepler University
Linz, Austria
andreu.vall@jku.at

Hamid Eghbal-zadeh
Johannes Kepler University
Linz, Austria
hamid.eghbal-zadeh@jku.at

Matthias Dorfer
Johannes Kepler University
Linz, Austria
matthias.dorfer@jku.at

Markus Schedl
Johannes Kepler University
Linz, Austria
markus.schedl@jku.at

Gerhard Widmer
Johannes Kepler University
Linz, Austria
gerhard.widmer@jku.at

ABSTRACT

Automated music playlist generation is a specific form of music recommendation. Generally stated, the user receives a set of song suggestions defining a *coherent* listening session. We hypothesize that the best way to convey such playlist coherence to new recommendations is by learning it from actual curated examples, in contrast to imposing ad hoc constraints. Collaborative filtering methods can be used to capture underlying patterns in hand-curated playlists. However, the scarcity of thoroughly curated playlists and the bias towards popular songs result in the vast majority of songs occurring in very few playlists and thus being poorly recommended. To overcome this issue, we propose an alternative model based on a song-to-playlist classifier, which learns the underlying structure from actual playlists while leveraging song features derived from audio, social tags and independent listening logs. Experiments on two datasets of hand-curated playlists show competitive performance compared to collaborative filtering when sufficient training data is available and more robust performance when recommending rare and out-of-set songs. For example, both approaches achieve a recall@100 of roughly 35% for songs occurring in 5 or more training playists, whereas the proposed model achieves a recall@100 of roughly 15% for songs occurring in 4 or less training playlists, compared to the 3% achieved by collaborative filtering.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

automated playlist generation, cold-start problem, hybrid recommender systems, music information retrieval, neural networks

DLRS 2017, August 27, 2017, Como, Italy

ACM ISBN 978-1-4503-5353-3/17/08…$15.00
https://doi.org/10.1145/3125486.3125494

ACM Reference Format:
Andreu Vall, Hamid Eghbal-zadeh, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. 2017. Music Playlist Continuation by Learning from Hand-Curated Examples and Song Features. In *Proceedings of DLRS 2017, Como, Italy, August 27, 2017,* 9 pages.
https://doi.org/10.1145/3125486.3125494

1 INTRODUCTION

The ability of recommender systems to suggest coherent sets of items is particularly important in the music domain, where songs are usually grouped into listening sessions. Identifying whether a set of songs fits well together becomes then a crucial and challenging question. The study presented in [8] analyzes interviews with practitioners and postings to a dedicated playlist-sharing web site. They indicate that there is a wide variety of factors (e.g., mood or purpose) that intervene in the process of compiling a playlist and suggest that the quality of a playlist and its potential coherence are not a clearly defined concept.

A common approach to playlist generation relates playlist coherence to homogeneity, meaning that the songs in a playlist should be content-wise similar (see e.g., [14, 20, 22, 23, 29]). Even though this assumption may be reasonable for some playlists, it imposes ad-hoc constraints that need not hold valid in general.

We prefer to adopt a statistical learning approach. We analyze hand-curated music playlists seeking common patterns that capture which songs fit well together. This is achieved by defining a quantitative criterion that needs to be fulfilled, on average, over playlists, thus providing a principled approach to modeling the often ambiguous concept of playlist coherence.

Collaborative Filtering (CF) methods as described in [2, 5] or the collaborative latent Markov embedding presented in [7] are utilized to reveal latent patterns from hand-curated music playlists. However, CF has well-known limitations particularly detrimental for the task of playlist modeling. Most importantly, CF methods are only aware of the songs occurring in the set of training playlists. As a consequence, the songs that never occurred in the training playlists, to which we refer as "out-of-set" songs, can not be recommended. Also, songs that occur rarely in the training playlists are poorly represented by CF models. This problem is likely to arise due to two main reasons: firstly, the amount of carefully curated

playlists is rather scarce (especially compared to the abundant–but not curated–listening logs derived from music streaming services); secondly, music consumption is biased towards popular songs [6] and thus the vast majority of songs occur in very few playlists. Finally, CF is not aware of song characteristics, which can be informative in some cases (e.g., in order to extend playlists with a specific instrumentation or within a genre).

The aforementioned limitations of CF can be mitigated through its hybridization with content-based methods [1]. In this line, we observe that by mining external data sources we can gather a large volume of song-level descriptions (e.g., audio, text descriptions from micro-posts or social-tagging platforms, or play counts from music streaming services). They constitute rich side-information that can be leveraged to make CF robust to rare and out-of-set songs.

In this work, we introduce a novel hybrid playlist continuation model that integrates hand-curated playlists with multi-faceted song features derived from audio, social tags and independent listening logs. Previous hybrid approaches to playlist continuation enhanced their performance through the combination of independently obtained scores by means of weighting heuristics and re-ranking [16, 20]. Instead, we blend the different sources of information into a jointly trained system, where learning is driven by the optimization of a quantitative criterion. The proposed model can then evaluate rare and even out-of-set songs, as long as song features are available, to identify songs fitting a given playlist. An implementation and data are provided for reproducibility.[1]

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents the hybrid playlist continuation model. The datasets of hand-curated music playlists and song features are described in Section 4. Section 5 details the configuration of the proposed model. In Section 6 we elaborate on the experimental results. Finally, Section 7 concludes the paper.

2 RELATED WORK

The automated generation of coherent music playlists has often been approached from a content-based perspective. By quantifying aspects of interests in songs, a recommender system can enforce smooth transitions. For example, [23] proposes to exploit timbral similarities to create a playlist of songs most similar to a given seed song. The quality of the resulting playlists is then quantified based on the criterion that the recommended songs are by the same artist, from the same album, or from the same genre as the seed. In [29], playlist generation is treated as a "traveling salesman problem", where distances are defined by timbral similarities. This approach is extended in [22] by incorporating artist similarities computed on the basis of web-based data. The artist similarities are used to prefilter which pairwise song similarities should be calculated, resulting in an accelerated and higher-quality recommendation process. Targeting the playlist generation problem for given start and end songs, [14] propose a multi-stage approach that considers distances between all candidate songs to the start and end songs. These distances are approximated using a single Gaussian to represent the timbral features of each song.

Collaborative filtering methods have been proven powerful to mine underlying structure from, generally, user-item interactions (see e.g., [30]). In particular, CF has been applied to music playlist generation by treating playlists as users, to whom songs should be recommended. In [2], a CF latent-factor model is tailored to mine a collection of Internet radio stations. The model features a specific latent-factor design that accounts for artist, time of the day and song adjacency. Latent variable models based on LDA [4] have also been applied to playlist modeling in [36], where a general music taste model is compared to a specific playlist model. In a similar line, [7] presents the latent Markov embedding for playlist modeling. It is inspired by collaborative methods and represents songs from radio playlists into a Euclidean latent space. The latent Markov embedding puts special attention on the sequential nature of playlists and can be used to generate new playlists.

Hybrid approaches combining CF and song features have also been proposed. Playlists are modeled as random walks on song hypergraphs in [25], where the edges are derived from multimodal song features and the weights are learned from hand-curated music playlists. In [16], the songs in a collection of hand-curated playlists are represented by topic models extracted from song-level social tags. Frequent sequential patterns are mined at the topic level, so that given a playlist, a next topic can be predicted. To extend music playlists, the scores provided by a memory-based CF algorithm are re-ranked using the next topic predicted. Another hybrid approach for the task of playlist continuation is presented in [20]. In a first stage, suitable next songs are preselected based on the combined score of a memory-based CF algorithm, TF-IDF features derived from social tags, metadata and personal preferences. In a second stage, the song candidates are re-ranked to match the recent songs.

Similar to our approach, relating song features and collaborative patterns through neural networks has also been proposed in [35]. A convolutional neural network is trained to predict the CF factors of a song, given the log-compressed mel-spectrogram of the song. However, the two approaches are fundamentally different. Our approach integrates collaborative patterns and song features into an enhanced recommendation model. Instead, the method proposed in [35] emulates CF when usage data is insufficient, and its performance is naturally upper-bounded by the performance of CF.

For a comprehensive survey on music playlist continuation, we point the interested reader to [5] or Chapter 13 in [30].

3 HYBRID MUSIC PLAYLIST CONTINUATION

In this section we introduce the hybrid playlist continuation model. It is based on a song-to-playlist classifier, whose predictions can be used to recommend playlists continuations. We finally describe the evaluation methodology followed to assess its performance.

3.1 Song-to-Playlist Classifier

Assume T is a collection of hand-curated playlists where each playlist $t \in T$ is regarded as a set of songs.[2] Furthermore, for each playlist t a continuation t_r (of possibly several songs) is known and withheld as a ground-truth. Let S be the set of unique songs within the collection of playlists T. Given a song $s \in S$, we denote its D-dimensional feature vector as $\mathbf{x}_s \in \mathbb{R}^D$. We further define a

[1] https://github.com/andreuvall/HybridPlaylistContinuation

[2] As traditional CF models, the proposed model regards playlists as sets in that it does not exploit the song order. See [34] for a treatment where the order is considered.

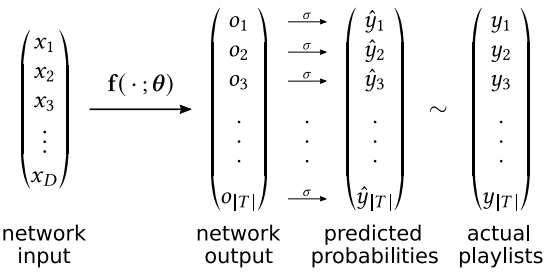

Figure 1: Sketch of the neural network definition. The input is a song feature $\mathbf{x} \in \mathbb{R}^D$. The network output is pointwise passed through a logistic activation function. This yields a vector of predicted probabilities $\hat{\mathbf{y}} \in [0,1]^{|T|}$ approximating the binary target $\mathbf{y} \in \{0,1\}^{|T|}$, i.e., the playlists the song belongs to. The set of parameters $\boldsymbol{\theta}$ is learned from examples and its dimension depends on the network architecture.

binary target vector $\mathbf{y}_s \in \{0,1\}^{|T|}$ indicating the playlists to which the song belongs.

The proposed song-to-playlist classifier is based on a neural network $\mathbf{f}(\,\cdot\,;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a set of parameters common for all songs. The network takes a song feature $\mathbf{x}_s$ as input and its output is pointwise passed through a logistic activation function yielding $\hat{\mathbf{y}}_s = \sigma\left(\mathbf{f}(\mathbf{x}_s;\boldsymbol{\theta})\right) \in [0,1]^{|T|}$, a vector of probabilities approximating the actual binary target $\mathbf{y}_s$. Figure 1 sketches the neural network definition. The song-to-playlist classifier makes as many independent decisions as playlists in the collection.[3] Thus, the set of network parameters $\boldsymbol{\theta}$ is estimated on the training set $\{\mathbf{x}_s, \mathbf{y}_s\}_{s\in S}$ in order to minimize the following binary cross-entropy cost function

$$\mathcal{L}\Big(\boldsymbol{\theta} \mid \{\mathbf{x}_s, \mathbf{y}_s\}_{s\in S}\Big) = \\ -\sum_{s,t} y_{s,t} \log\left(\hat{y}_{s,t}\right) + \left(1 - y_{s,t}\right) \log\left(1 - \hat{y}_{s,t}\right). \quad (1)$$

The terms $y_{s,t}$ and $\hat{y}_{s,t}$ denote the components of $\mathbf{y}_s$ and $\hat{\mathbf{y}}_s$ corresponding to playlist t, respectively. The dimensionality of the set of parameters $\boldsymbol{\theta}$ depends on the network architecture, which is discussed in Section 5. The summation is done over all the possible song-playlist pairs, both occurring ($y_{s,t} = 1$) and non-occurring ($y_{s,t} = 0$) in the training playlists.[4]

Note that the training set $\{\mathbf{x}_s, \mathbf{y}_s\}_{s\in S}$ comprises the song features $\mathbf{x}_s$ and the playlist-belonging targets $\mathbf{y}_s$ (derived from hand-curated playlists). The cost function (1) integrates both sources of information into a jointly learned hybrid model.

3.2 Recommending Playlist Continuations

Given a set of candidate songs, we use the trained song-to-playlist classifier $\mathbf{f}(\,\cdot\,;\boldsymbol{\theta}^*)$ to predict the vector of probabilities $\hat{\mathbf{y}}_s \in [0,1]^{|T|}$ for each candidate song s. The predicted vector is *dense* (not *sparse*), i.e., it contains the probability of song s belonging to each playlist $t \in T$.[5] Then, for each playlist $t \in T$, we rank all the candidate songs according to their predicted probability $\hat{y}_{s,t}$.

The song-to-playlist classifier can evaluate any song for which a feature vector is available, even if the song does not belong to S. As we will see in Section 6, this enables the recommendation of out-of-set songs, and also has a positive impact on the recommendation of songs occurring only in few training playlists. On the other hand, as user- and factorization-based CF models, the proposed model can only extend the playlists for which it has been trained.

3.3 Evaluation

A large-scale on-line evaluation where users could assess the quality of the produced playlist continuations should be the preferred option (see e.g., [30]). However, this would require a complex infrastructure beyond the scope of this work. We instead opt for the standard off-line evaluation performed in [2, 5, 16, 20], where the ability of the model at retrieving withheld playlist continuations is assessed. Precisely, we follow [2], where the length of each withheld continuation depends on the length of the corresponding playlist. This is contrast to [5, 16, 20], where only one song is withheld regardless.

We define S^* by joining S and the set of unique songs in the ground-truth playlist continuations. For every s in S^*, we predict the vector of probabilities $\hat{\mathbf{y}}_s$ indicating its fit to each playlist $t \in T$. We arrange the predicted probability vectors as columns of a dense matrix of probabilities $\hat{\mathbf{Y}} \in [0,1]^{|T|\times|S^*|}$, that has as many rows as playlists in T and as many columns as songs in S^*.

Given a playlist t, we rank all the songs in S^* not already included in t according to the probabilities from the corresponding playlist row in $\hat{\mathbf{Y}}$. This results in a sorted list of song candidates. For each song in the withheld playlist continuation t_r, we compute its rank and average precision within the full list of song candidates, and we further compute its recall within the lists of top 10, top 30 and top 100 song candidates. Having the results for all the songs in all the withheld playlist continuations, we finally report the median rank, the Mean Average Precision (MAP) and the mean recall at 10, 30 and 100. Even though lists of top 30 or top 100 song candidates may seem impractical for actual applications, those and even longer lists are used to assess playlist continuations in [2, 5, 16, 20].

4 DATASETS

We use two datasets of hand-curated playlists. The "AotM-2011" dataset [25] is a collection of playlists derived from the Art of the Mix[6] database. Each playlist is represented by song titles and artist names, linked to the corresponding identifiers of the Million Song Dataset[7] (MSD) [3], where available. As we will describe in Section 4.2, the connection to the MSD is essential to our approach in order to gather additional song descriptions, from which we extract song-level features.

We also use a private playlists dataset from "8tracks",[8] an on-line platform where users can share playlists and listen to playlists other

[3] We also experimented with a softmax activation function yielding a per-song probability distribution over playlists. However, the proposed approach provided more consistent results.

[4] We experimented with different weighting schemes for positive and non-positive observations as suggested in [18], but none showed a consistent improvement.

[5] This property is also found in CF models based on latent factors (see e.g., [18]), where a dense matrix of song-playlist preferences is predicted on the basis of a sparse matrix of interactions.

[6] www.artofthemix.org

[7] https://labrosa.ee.columbia.edu/millionsong

[8] https://8tracks.com

users prepared. Similar to the AotM-2011 dataset, each playlist is represented by song titles and artist names. Mimicking the AotM-2011 dataset, we use fuzzy string matching to resolve the song titles and artist names from the 8tracks dataset against the MSD. Precisely, we adapt the code released in [21] for a very similar task. The matching results in roughly 2,5M song identifiers from the 8tracks dataset (many are spelling duplicates) resolved into 241,123 unique song identifiers from the MSD. The link between the 8tracks dataset and the MSD lets us gather song descriptions and makes the comparison to the AotM-2011 dataset fair.

4.1 Playlist Continuation Sets

The AotM-2011 dataset contains a considerable number of playlists with songs by one or very few artists. Preliminary experiments conducted on this dataset resulted in a particularly high retrieval performance when utilizing the proposed hybrid model with features derived from social tags. This result could be explained by the fact that some tags inform about the artist name, which would entail data leakage. To prevent that, we discard artist-themed playlists by keeping only the playlists with at least 7 unique artists and with a maximum of 2 songs per artist. The 8tracks dataset does not have this issue because the terms of use of the platform require that no more than 2 songs from the same artist or album may be included in a playlist. Nevertheless, we apply the same filters to both datasets for the sake of consistency. We further filter both datasets by keeping only the playlists with at least 14 songs linked to the MSD. This is to ensure that the song-to-playlist classifier has a sufficient number of songs in each playlist to learn from.

So far we have discarded complete playlists not satisfying the specified requirements. Now, for the sake of comparability, we discard the songs within playlists for which some type of feature is missing (the types of features are explained in Section 4.2). As a result, the playlists are shortened and the exact amount of unique artists may be affected.

Finally, in order to set up the training and the test set, we discard all the playlists that have become shorter than 5 songs after the song filtering. We then split each playlist leaving approximately 80% of the songs as a training example and the rest as a withheld continuation. The training examples form the training set and the withheld continuations form the test set. In this setting, a song may occur in both splits. Pure collaborative filtering approaches need to further ensure that songs occurring in the test set also occur in the training set, otherwise they can not be recommended. Our hybrid approach can deal with out-of-set songs, so this is not necessary.

The filtered AotM-2011 dataset has 2,715 playlists with 12,355 songs by 4,097 artists. The filtered 8tracks dataset has 3,272 playlists with 14,613 songs by 5,119 artists. We name the final datasets "playlist continuation sets". Detailed statistics regarding the distribution of unique songs per playlist, unique artists per playlist and song frequency in the datasets is reported in Table 1.

4.2 Song Features

The MSD, together with the accompanying "Last.fm Dataset"[9] and the "Taste Profile Subset,"[10] provide an heterogeneous collection of data for a million contemporary songs. We use song descriptions based on audio, social tags, and listening logs to extract state-of-the-art song features. For the audio content, the MSD splits songs into segments of variable length (typically under a second) and provides 12-dimensional timbral coefficients (similar to MFCCs) for each segment. Regarding the social tags, the "Last.fm Dataset" provides tagging activity at the song-level and at the artist-level, along with weights describing the relevance of each tag for each song and artist. Finally, the "Taste Profile Subset" provides user-song play counts derived from independent listening logs.[11]

The feature extraction process is described next and is the same for the AotM-2011 and the 8tracks datasets. The extraction of song features from audio and from social tags requires the pre-estimation of models on a set of representative songs (see the details below). We prepare separate sets of songs for the AotM-2011 and for the 8tracks datasets. For each dataset, we select playlists with at least 10 songs linked to the MSD, by at least 5 artists, such that no artist has more than 2 songs in the playlist. The selected playlists are then a superset of the corresponding playlist continuation sets and we assume that the unique songs within them are representative. To prevent leaking ground-truth data we exclude the songs that appear only in the test split of the corresponding playlist continuation set. We refer to the obtained song collections as the "development song sets". For the AotM-2011 dataset we obtain 48,393 songs and for the 8tracks dataset we obtain 47,617 songs.

4.2.1 Average Timbral Features. This feature represents the average of the timbral coefficients over all the segments in a song. Each song is then described by a 12-dimensional vector. We include it as a simple timbre-based reference.

4.2.2 Vector-Quantized Timbral Features. We run the k-means clustering algorithm on the whole pool of segment-level timbral coefficients of the development song set. We set the number of clusters to 200, thus obtaining 200 representative timbral centroids.[12] For each song in the playlist continuation set, we assign each frame to the closest centroid. The vector-quantized (VQ) timbral feature amounts to the count of frames the song has assigned to each centroid and therefore it is 200-dimensional. This approach has been successfully utilized in [17] for music autotagging and further investigated in [31] for music similarity.

4.2.3 I-Vectors from Timbral Features. I-vectors were first introduced in the field of speaker verification [9]. Recently they have been successfully utilized for music similarity and music artist recognition tasks [12, 13]. We build a Gaussian mixture model with 1,024 components on the entire pool of segment-level features of the development song set. Using the songs in the playlist continuation set we train the total variability space yielding 200-dimensional i-vectors. Following the standard i-vector extraction pipeline, we further transform the i-vectors using a linear discriminant analysis model fit on the training split of the playlist continuation set.

[9] https://labrosa.ee.columbia.edu/millionsong/lastfm
[10] https://labrosa.ee.columbia.edu/millionsong/tasteprofile

[11] The MSD also provides high-level features such as *danceability* or *energy*. However, these features are not documented within the MSD nor were they within their original source, the now discontinued Echo Nest API (the.echonest.com).
[12] We extract 200-dimensional vectors for all the feature types (except for the average timbre). Our experiments indicate that 200-dimensional vectors carry rich information and fixing the dimension across features makes the comparison fair.

Table 1: Descriptive statistics for the playlists within the AotM-2011 and the 8tracks playlist continuation sets. We report the distribution of the number of songs per playlist, the number of artists per playlist, and the song frequency in the dataset (i.e., the number of playlists in which each song occurs).

		Training set					Test set				
		min	1q	med	3q	max	min	1q	med	3q	max
AotM-2011	Songs per playlist	4	6	7	9	21	1	1	2	2	5
	Artists per playlist	3	6	7	9	21	1	1	2	2	5
	Song frequency	1	1	1	2	35	1	1	1	1	11
8tracks	Songs per playlist	4	6	8	10	30	1	2	2	2	8
	Artists per playlist	3	6	8	10	28	1	2	2	2	8
	Song frequency	1	1	1	2	119	1	1	1	1	27

4.2.4 Semantic Features from Social Tags. The embedding of words into continuous vector spaces [26] allows us to map social tags into a semantic feature space. We gather the song-level social tags corresponding to the development song set and build a music-aware text corpus by fetching the Wikipedia[13] pages of the collected tags. We run the implementation of the continuous bag-of-words algorithm available in *word2vec*[14] on the text corpus to obtain a dictionary of 200-dimensional semantic features for the most relevant words. Then, for each song in the playlist continuation set we look up the words within the song-level social tags. If a tag is compound of several words (e.g., "pop rock") we compute the average feature value. The final song semantic feature is the weighted average of all its tags' features, where the weights are given by the relevance of each tag for the song. Likewise we compute the semantic features for the artist-level social tags, only that given a song we use the social tags related to the song's artist.

4.2.5 Latent Factors from Listening Logs. We factorize the user-song play counts from the "Taste Profile Subset" using the weighted factorization model presented in [18], which is specifically designed for implicit feedback datasets. We use a depth of 200 factors. We discard the user latent factors, which are unrelated to our playlist continuation problem. We keep the song latent factors, that carry rich song information.

5 MODEL CONFIGURATIONS

Both for the AotM-2011 and the 8tracks playlist continuation sets, we split each playlist in the training set leaving approximately 20% of the songs for validation, which we use to determine appropriate model configurations. The ground-truth playlist continuations remain untouched until the final evaluation.

5.1 Proposed Hybrid Model

The proposed model is powered by the song-to-playlist classifier presented in Section 3.1. Precisely, for every playlist continuation set and each type of song features we fit an independent song-to-playlist classifier, with possibly different configurations. The considered neural network architectures, hyperparameters, training strategies and feature preprocessing are detailed in Appendix A.1.

[13] https://en.wikipedia.org
[14] https://code.google.com/p/word2vec

5.2 Collaborative Filtering Baseline

We compare our hybrid model to a CF baseline to assess the advantage of integrating song descriptions and hand-curated music playlists. We choose the state-of-the-art Weighted Matrix Factorization (WMF) model introduced in [18] because it is specifically designed to perform CF on implicit feedback datasets like playlists. The comparison to our model is technically simple. We just need to replace the probabilities predicted by our song-to-playlist classifier with the scores predicted by the WMF model trained on the playlist continuation sets. After that, the evaluation methodology remains valid. Further details on the WMF depth, hyperparameters and training strategy are discussed in Appendix A.2.

6 RESULTS

We evaluate the proposed playlist continuation model using the different types of song features, first as standalone features and then as combined features. Finally, we assess the performance of the proposed model to recommend rare and out-of-set songs.

Remember that the evaluation consists in, given a query playlist, retrieving the songs from its withheld continuation among all the songs in the dataset that did not occur in the query. Note that these continuations have a median length of only 2 songs (Table 1) while the AotM-2011 and the 8tracks datasets have a total of 12,355 and 14,613 songs, respectively. A perfect model would rank the songs from the withheld continuations in the top positions (low ranks). An extremely poor model would rank them in the last positions (high ranks). A random model would, on average, rank them in the middle of the list of song candidates. Thus the actual rank values depend on the number of songs in each dataset.

Table 2 reports the results of the proposed hybrid model with the different types of song features and also the results of the CF baseline, for the AotM-2011 and the 8tracks datasets. The different models have been sorted according to their median rank score.

6.1 Standalone Features

We start by analyzing the results achieved using standalone features in order to provide insights on their retrieval power for the specific task of playlist continuation.

The performance of the different standalone features is consistent in both datasets (Table 2). Latent factors derived from listening logs are the most expressive feature, followed by more than 300

Table 2: Retrieval performance of the proposed model for each dataset and each type of song features, and of the CF baseline. We report the median rank, the MAP and the recall at lists of length 10, 30 and 100. The median rank compares to 12,355 song candidates for the AotM-2011 dataset and to 14,613 song candidates for the 8tracks dataset. Lower is better. For MAP and recall@{10, 30, 100} higher is better. As a reference, we include a random playlist continuation model where the fitness of each song-playlist pair is drawn at random from a uniform distribution in $[0, 1]$.

dataset	feature	med rank	MAP	recall@10	recall@30	recall@100
AotM-2011	i-vectors + song tags + listening logs	860	1.75%	3.28%	8.07%	17.61%
	song tags + listening logs	871	1.69%	3.21%	7.79%	17.43%
	i-vectors + listening logs	952	1.64%	3.27%	7.05%	16.28%
	listening logs	993	1.64%	3.10%	7.37%	16.32%
	i-vectors + song tags	1326	1.22%	1.98%	4.74%	11.73%
	song tags	1372	1.25%	2.34%	5.26%	12.99%
	CF	1444	1.96%	3.99%	7.84%	14.56%
	artist tags	1535	0.68%	1.14%	3.11%	8.68%
	i-vectors	2715	0.53%	0.80%	2.07%	4.85%
	VQ timbres	3425	0.21%	0.26%	0.97%	2.94%
	average timbres	3525	0.23%	0.28%	0.79%	2.54%
	random	6087	0.11%	0.20%	0.28%	0.79%
8tracks	i-vectors + song tags + listening logs	448.5	3.35%	6.59%	13.31%	26.85%
	song tags + listening logs	471	3.23%	6.18%	13.03%	26.37%
	i-vectors + listening logs	544	2.76%	5.38%	12.11%	24.07%
	listening logs	612.5	2.61%	4.83%	10.99%	23.28%
	i-vectors + song tags	778	2.36%	4.91%	10.19%	20.54%
	song tags	935	2.26%	4.15%	8.91%	18.57%
	CF	1000	2.65%	5.06%	10.14%	19.60%
	artist tags	1102.5	1.28%	2.29%	6.07%	14.55%
	i-vectors	1985.5	0.67%	1.07%	2.64%	7.53%
	VQ timbres	2897	0.44%	0.60%	1.47%	4.50%
	mean timbres	3253.5	0.31%	0.33%	1.31%	3.76%
	random	7320	0.09%	0.12%	0.23%	0.66%

additional rank positions by the semantic features extracted from song-level tags. Their artist-based counterpart performs worse by approximately 200 rank positions and is the first set of song features to perform worse than the CF baseline. They are followed by far (1,200 rank positions in the AotM-2011 dataset and almost 900 rank positions in the 8tracks dataset) by the i-vectors extracted from timbral features. Other audio-based features perform worse, but clearly better than random.

The different types of features are ordered similarly in terms of their achieved MAP or their recall scores, with the exception that the CF baseline achieves a competitive recall@100 in the AotM-2011 dataset. It is also interesting to observe that the MAP values are very low for all the song features. Similar results were already observed in [24, 28] and could be explained by the nature of the playlist continuation problem. That is, a playlist may be continued using different songs, all of them potentially relevant, but the precision score penalizes any continuation that does not exactly match the withheld continuation. Still, we present the MAP scores for reference.

6.2 Combined Features

We extend the analysis by assessing the retrieval performance of combinations of song features. To keep the number of combinations to a moderate amount we pick only the best performing song features derived from audio (i.e., the i-vectors extracted from timbral features) and the best performing song features derived from social tags (i.e., the semantic features extracted from song-level social tags). We also use the latent factors extracted from listening logs. We evaluate all the possible combinations of pairs of song features and also the combination of the three. A combined song feature vector is simply the concatenation of the individual feature vectors. Since all the feature vectors we combine are 200-dimensional, the combinations of two features result in a 400-dimensional feature vector and the combination of the three features result in a 600-dimensional feature vector.

Combining features improves the performance of the playlist continuation model (Table 2). The most interesting case is the combination of all three types of features, which improves the median rank score by more than 100 positions compared to the best standalone feature (the latent factors extracted from listening logs). Furthermore, the recall@100 is improved by almost 1 percentage point (p.p.) in the AotM-2011 dataset and by more than 3 p.p. in the 8tracks dataset. We also observe that the gain of combining features seems to relate to their individual performance. For example, the semantic features extracted from song-level social tags perform better than the i-vectors extracted from timbral features. Then, the

combination of latent factors and semantic features performs better than the combination of latent factors and i-vectors.

The fact that the combined song features provide enhanced performance and the observation that the gain is related to their individual performance suggests that the different types of feature indeed carry different and complementary song information.

6.3 Cold-Starting Rare and Out-of-Set Songs

We now assess the potential advantage of the proposed hybrid model compared to the CF baseline. The proposed hybrid model outperforms the CF baseline when it uses song features derived from song-level tags, listening logs, or any combined feature including them (Table 2). The combination of all the features achieves a recall@100 almost 3 p.p. higher than CF for the AotM-2011 dataset. This difference increases to 8 p.p. in the 8tracks dataset.

The performance gap between the proposed hybrid model and the CF baseline could be explained by the ability of the proposed model to deal with rare and out-of-set songs, together with the fact that in both datasets half of the songs occur only in 1 training playlist and three quarters of the songs occur only in 2 training playlists (Table 1). To investigate this effect, we analyze the performance of the best performing hybrid model and the CF baseline as a function of how often the songs in the withheld playlist continuations occurred in training playlists (Figure 2). The proposed hybrid model performs comparably to the CF baseline for songs with 5 or more occurrences in training playlists. For songs with 4 or less occurrences in training playists, the proposed hybrid model consistently outperforms the CF baseline in all the metrics and its performance is fairly constant regardless of the number of occurrences, even for songs that never occurred in training playists. Thus this result seems to explain the superior performance of the proposed hybrid model.

7 CONCLUSION

In this work we have introduced a hybrid music playlist continuation model that integrates collaborative information encoded in hand-curated playlists with multi-faceted state-of-the-art song-level features. In contrast to previous hybrid approaches, the proposed model fuses the different sources of information into a joint learning procedure driven by the optimization of a quantitative criterion.

We examined the performance of the model using standalone song features, as well as their combinations. Our experiments indicate that features derived from independent listening logs outperform those derived from social tags, which in turn outperform those derived from audio. Combining features improves performance further, suggesting that the different types of song features indeed carry different and complementary song information.

Most importantly, the proposed hybrid model is robust to the cold-start problem for rare and even out-of-set songs. Indeed, our experimental results confirm that the proposed model consistently outperforms the CF baseline when data is scarce. If data is abundant, the proposed model performs comparably to the CF baseline.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (June 2005), 734–749.

[2] Natalie Aizenberg, Yehuda Koren, and Oren Somekh. 2012. Build your own music recommender by modeling internet radio streams. In *Proc. WWW*. 1–10.

[3] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. 2011. The million song dataset. In *Proc. ISMIR*. University of Miami, 591–596.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.

[5] Geoffray Bonnin and Dietmar Jannach. 2014. Automated generation of music playlists: Survey and experiments. *Comput. Surveys* 47, 2 (Nov. 2014), 1–35.

[6] Òscar Celma. 2010. *Music recommendation and discovery*. Springer.

[7] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *Proc. SIGKDD*. 714–722.

[8] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. 2006. "More of an art than a science": Supporting the creation of playlists and mixes. In *Proc. ISMIR*.

[9] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. 2011. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing* 19, 4 (May 2011), 788–798.

[10] Sander Dieleman et al. 2015. Lasagne: First release. (Aug. 2015). http://dx.doi.org/10.5281/zenodo.27878

[11] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12 (2011), 2121–2159.

[12] Hamid Eghbal-zadeh, Bernhard Lehner, Markus Schedl, and Gerhard Widmer. 2015. I-vectors for timbre-based music similarity and music artist classification. In *Proc. ISMIR*.

[13] Hamid Eghbal-zadeh, Markus Schedl, and Gerhard Widmer. 2015. Timbral modeling for music artist recognition using i-vectors. In *Proc. EUSIPCO*. 1286–1290.

[14] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. 2008. Playlist generation using start and end songs. In *Proc. ISMIR*. 173–178.

[15] Ben Frederickson. 2017. Fast python collaborative filtering for implicit datasets. (2017). https://github.com/benfred/implicit

[16] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware music recommendation based on latent topic sequential patterns. In *Proc. RecSys*. 131–138.

[17] Matthew D. Hoffman, David M. Blei, and Perry R. Cook. 2009. Easy as CBA: A simple probabilistic model for tagging music. In *Proc. ISMIR*, Vol. 9. 369–374.

[18] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proc. ICDM*. 263–272.

[19] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167* (2015).

[20] Dietmar Jannach, Lukas Lerche, and Iman Kamehkhosh. 2015. Beyond "hitting the hits" Generating coherent music playlist continuations with the right tracks. In *Proc. RecSys*.

[21] Andreas Jansson, Colin Raffel, and Tillman Weyde. 2015. This is my jam Data dump. In *Proc. ISMIR*. 175.

[22] Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. 2006. Combining audio-based similarity with web-based data to accelerate automatic music playlist generation. In *Proc. International Workshop on Multimedia IR*. 147–154.

[23] Beth Logan. 2002. Content-based playlist generation: Exploratory experiments.. In *Proc. ISMIR*.

[24] Brian McFee and Gert RG Lanckriet. 2011. The natural language of playlists. In *Proc. ISMIR*. 537–542.

[25] Brian McFee and Gert RG Lanckriet. 2012. Hypergraph models of playlist dialects. In *Proc. ISMIR*. 343–348.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[27] Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate O(1/k2). *Soviet Mathematics Doklady* 27, 2 (1983), 372–376.

[28] John C. Platt, Christopher JC Burges, Steven Swenson, Christopher Weare, and Alice Zheng. 2001. Learning a Gaussian process prior for automatically generating music playlists. In *NIPS*. 1425–1432.

[29] Tim Pohle, Elias Pampalk, and Gerhard Widmer. 2005. Generating similarity-based playlists using traveling salesman algorithms. In *Proc. DAFx*. 220–225.

[30] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. *Recommender Systems Handbook* (2nd ed.). Springer US.

[31] Klaus Seyerlehner, Gerhard Widmer, and Peter Knees. 2008. Frame level audio similarity-a codebook approach. In *Proc. DAFx*.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[33] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016).

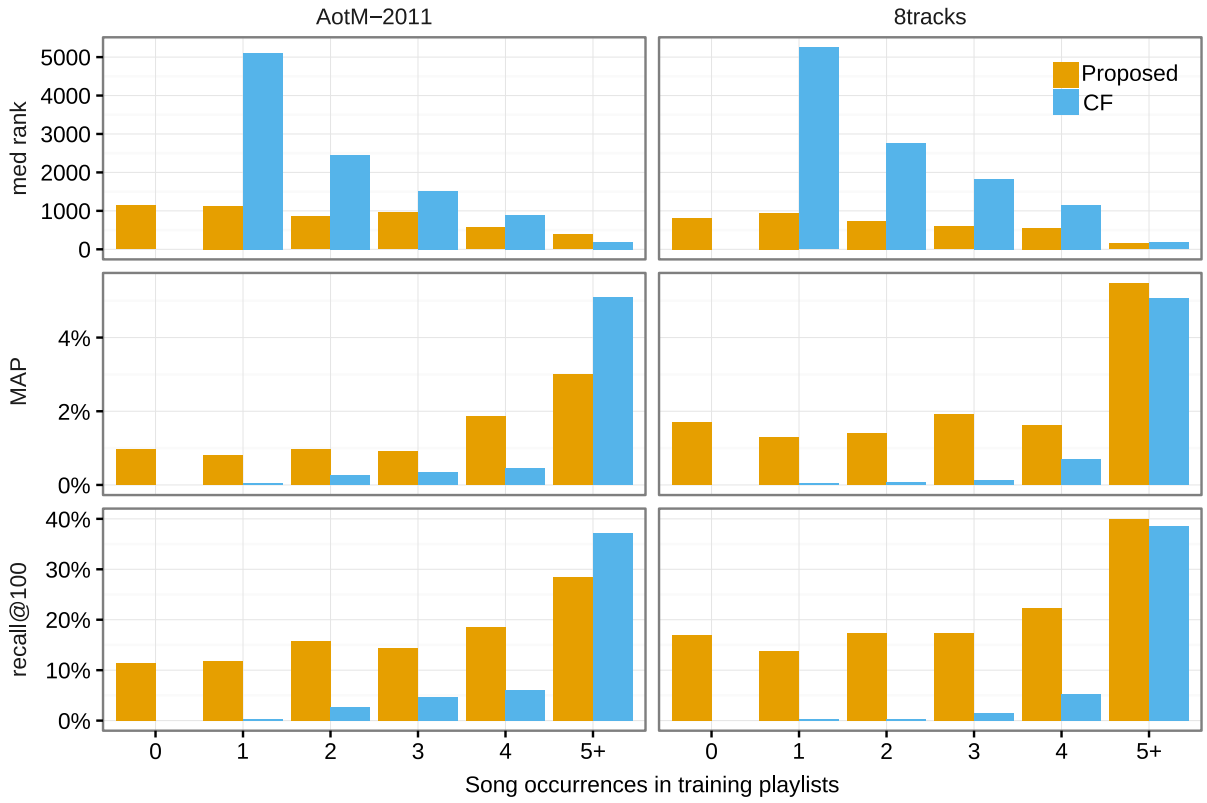

Figure 2: Retrieval performance of the best performing proposed model (all features combined) and the CF baseline. For each dataset, we report the median rank, MAP and recall@100 as a function of how often the songs in the withheld continuations occurred in training playlists. For the median rank lower is better. For MAP and recall@100 higher is better.

[34] Andreu Vall, Markus Schedl, Gerhard Widmer, Massimo Quadrana, and Paolo Cremonesi. 2017. The importance of song context in music playlists. In *RecSys 2017 Poster Proceedings*.

[35] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.

[36] Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. 2010. Statistical models of music-listening sessions in social media. In *Proc. WWW*. 1019–1028.

A MODEL CONFIGURATIONS

A.1 Proposed Hybrid Model

We conducted an initial exploration of architectures by evaluating networks with $\{2, 3, 4\}$ hidden layers, $\{50, 100, 200, 500\}$ hidden units, learning rate values in $\{0.1, 0.5, 1.0\}$ and batch sizes of $\{10, 50, 100, 200\}$ songs. We also experimented with the hyperbolic tangent, the logistic function and the rectifier as activation functions for the hidden layers. We did not perform all the combinations of the aforementioned parameter values, but used the preliminary results to narrow down the actual parameter search space.

Given the results of the preliminary analysis, we systematically explored all the combinations of networks with $\{2, 3\}$ hidden layers and with $\{50, 100, 200\}$ hidden units. We fixed the learning rate to 0.5, the batch size to 50 songs and the hyperbolic tangent as the activation function for hidden layers. Recall that the output layer of the network is passed through logistic functions (see Section 3.1).

We use batch normalization [19]. We also experimented with different dropout probabilities [32] and with $L1$ and $L2$ regularization to prevent overfitting. We finally decided to use dropout with probabilities 0.1 and 0.5 at the input layer and the hidden layers, respectively.

The features were preprocessed. Namely, the average timbral features, the i-vectors from timbral features, the semantic features from social tags, the latent factors from listening logs and the combined features were standardized and $L2$-normalized. The vector-quantized timbral features were only $L1$-normalized according to their histogram-like nature.

The networks were optimized to minimize the cost function (1) using AdaGrad[15] [11] with Nesterov momentum [27]. We trained for a maximum of 1,000 epochs but stopped before if the cost function was not significantly minimized during 100 epochs. The cost function drove the optimizer, but the best model was chosen on the basis of the highest recall achieved on the validation set. We also used the recall on the validation set to decide an appropriate number of epochs for the final training on the entire training set.

[15] Since AdaGrad re-scales the learning rate at every update, setting the learning rate to 0.5 actually refers to setting its initial value.

Table 3 reports the final configuration of each network. We implemented the networks using Lasagne [10], which is built on top of Theano [33].

Table 3: Final network configuration for each playlist continuation set and each set of features. The learning rate is set to 0.5, the batch size to 50 songs and the hyperbolic tangent is the activation function for all the hidden layers. We use batch normalization and dropout with probabilities 0.1 and 0.5 in the input layer and the hidden layers, respectively.

dataset	feature	layers	units	epochs
AotM-2011	average timbres	2	50	100
	VQ timbres	3	50	300
	i-vectors	3	50	200
	song tags	3	100	250
	artist tags	3	50	600
	listening logs	3	100	310
	i-vectors + song tags	3	50	500
	i-vectors + logs	3	100	230
	song tags + logs	3	100	200
	i-vectors + song tags + logs	3	100	150
8tracks	average timbres	3	50	200
	VQ timbres	3	50	450
	i-vectors	3	50	500
	song tags	2	100	408
	artist tags	2	100	500
	listening logs	2	50	540
	i-vectors + song tags	3	100	300
	i-vectors + logs	3	100	360
	song tags + logs	3	100	520
	i-vectors + song tags + logs	3	100	360

A.2 Collaborative Filtering Baseline

The Weighted Matrix Factorization (WMF) model introduced in [18] generally mines user-item interactions while leveraging the intensity of the interactions (e.g., the number of clicks on websites, or the play counts on on-line streaming services). Modeling playlists is a slightly different problem because the examples consist of binary values without intensity information. Thus, the default weighting scheme proposed in [18] is not suited for our task. We used the validation sets to experiment with different weights for the observed playlist-song interactions and found that assigning them a weight of 2 yielded best results. We also experimented with different weights for the $L2$-regularization term and decided to use a factor of 10. We use the implementation provided in [15].