A Hybrid Approach to Music Playlist Continuation Based on Playlist-Song Membership

Andreu Vall, Matthias Dorfer, Markus Schedl, Gerhard Widmer
Johannes Kepler University, Department of Computational Perception
andreu.vall@jku.at

ABSTRACT

Automated music playlist continuation is a common task of music recommender systems, that generally consists in providing a fitting extension to a given playlist. Collaborative filtering models, that extract abstract patterns from curated music playlists, tend to provide better playlist continuations than content-based approaches. However, pure collaborative filtering models have at least one of the following limitations: (1) they can only extend playlists profiled at training time; (2) they misrepresent songs that occur in very few playlists. We introduce a novel hybrid playlist continuation model based on what we name "playlist-song membership," that is, whether a given playlist and a given song fit together. The proposed model regards any playlist-song pair exclusively in terms of feature vectors. In light of this information, and after having been trained on a collection of labeled playlist-song pairs, the proposed model decides whether a playlist-song pair fits together or not. Experimental results on two datasets of curated music playlists show that the proposed playlist continuation model compares to a state-of-the-art collaborative filtering model in the ideal situation of extending playlists profiled at training time and where songs occurred frequently in training playlists. In contrast to the collaborative filtering model, and as a result of its general understanding of the playlist-song pairs in terms of feature vectors, the proposed model is additionally able to (1) extend non-profiled playlists and (2) recommend songs that occurred seldom or never in training playlists.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

automated music playlist continuation, hybrid recommender systems, music information retrieval, neural networks

ACM Reference Format:
Andreu Vall, Matthias Dorfer, Markus Schedl, Gerhard Widmer. 2018. A Hybrid Approach to Music Playlist Continuation Based on Playlist-Song Membership. In *SAC 2018: Symposium on Applied Computing, April 9–13, 2018, Pau, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3167132.3167280

1 INTRODUCTION

The automated continuation of music playlists enables music recommendation scenarios such as playing and sequentially extending a music stream (similar to traditional radio broadcasting) or suggesting to the user fitting songs to extend their own music playlists. In both cases, it is crucial to identify candidate songs that fit a given playlist, and this is a particularly challenging question. By analyzing interviews with practitioners and postings to a dedicated playlist-sharing website, Cunningham et al. [9] identified that the playlist curation process is complex and influenced by a variety of factors like mood, theme or purpose. Furthermore, they found that the agreement among practitioners on curation rules was only reduced to rather loose guidelines.

A successful approach to music playlist continuation relies on mining collaborative information through the analysis of curated music playlists. In particular, collaborative approaches based on statistical models explain curated playlists in terms of a defined quantitative criterion, providing a principled approach to modeling the fitness of songs in playlists. In contrast to pure content-based approaches, collaborative approaches tend to reveal more abstract patterns over playlists and songs, but the good performance of collaborative approaches is strongly dependent on the availability of a large volume of training data. This requirement is easily compromised. 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 inescapably biased towards popular songs [7], resulting in a vast majority of songs occurring in very few playlists.

We introduce a novel hybrid playlist continuation model that combines curated music playlists with multimodal song features. The curated music playlists are used to derive training examples of playlist-song pairs that fit and playlist-song pairs that do not fit. The multimodal song features make the proposed model robust to data scarcity problems. The proposed model is designed to flexibly evaluate the fitness of any playlist-song pair, making it possible to extend any playlist by selecting suitable songs among any set of song candidates. In contrast to previous hybrid playlist continuation models, the proposed model is a *feature-combination* hybrid [6], having the advantage that the collaborative information and the song features are implicitly fused into a single enhanced recommender system.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents the hybrid playlist continuation model. The evaluation methodology is described in Section 4. Section 5 describes the datasets of curated playlists and song features used in our experiments. Section 6 elaborates on the results. Finally, conclusions are drawn in Section 7.

2 RELATED WORK

A well-researched approach to automated music playlist continuation relies on the song content. Pairwise song similarities are computed on the basis of features extracted from the audio signal (possibly enriched with social tags and metadata) and used to enforce content-wise smooth transitions [14, 24, 26, 29, 33]. Recommendations based on content similarity are expected to confer coherence to the playlist. However, pure content-based recommendations can not capture complex relations and, in fact, it does not hold in general that the songs in a playlist should all sound similar [25].

Collaborative Filtering (CF) has been proven successful to reveal underlying structure from, in general, user-item interactions [1, 34]. In particular, CF has been applied to music playlist continuation by regarding each playlist as a user's listening history, on the basis of which songs should be recommended. Previous research has mostly focused on playlist-neighborhood CF models [5, 16, 20], but Aizenberg et al. [2] also present a latent-factor CF model tailored to mine Internet radio stations, accounting for song artist, time of the day and song adjacency. An important limitation of most latent-factor and playlist-neighborhood CF models is that they need to profile the playlists at training time in order to extend them, by computing their latent factors or finding their nearest neighbors. As a consequence, such models can not extend playlists unseen at training time. To circumvent this issue, Aizenberg et al. [2] replace the latent factors of unseen playlists by the latent factors of their songs, and Jannach and Ludewig [21] show how to efficiently implement a playlist-neighborhood CF model that can extend unseen playlists in reasonable time. Song-neighborhood CF models can in general extend unseen playlists, because they only require pairwise song similarities [35, 39]. A common limitation of all pure CF methods is that they are only aware of the songs occurring in the training playlists. Thus, songs that never occurred in the training playlists, to which we refer as "out-of-set" songs, can not be recommended.

Zheleva et al. [41] propose a latent-variable playlist model based on Latent Dirichlet Allocation (LDA) [4]. They found that a variation of the basic LDA model that takes listening sessions into consideration provides better playlist continuations. In a similar line, Chen et al. [8] present a playlist model named Latent Markov Embedding, that exploits radio playlists to learn an embedding that projects songs into a Euclidean latent space. Both models can extend playlists without precomputing playlist profiles, but they can only recommend songs occurring in training playlists.

Hybrid models combining CF and song features are a common approach to mitigate the difficulties of CF models to represent infrequent songs. Hariri et al. [16] represent the songs in hand-curated playlists by topic models derived from social tags and then mine frequent sequential patterns at the topic level. The scores of a CF model are re-ranked according to the next topics predicted. The approach proposed by Jannach et al. [20] pre-selects suitable next songs based on the weighted combination of scores yielded by a playlist-neighborhood CF model and content-based similarities. The song candidates are then re-ranked to match the recent songs. In both cases, the hybridization follows from the combination of independently obtained scores by means of weighting heuristics or re-ranking. For example, the CF prediction for a song occurring only in a few training playlists would be boosted with content-based information. However, the prediction for an out-of-set song would solely rely on the content-based component.

Similar to our approach, Van den Oord et al. [40] also relate song content with collaborative patterns using a deep neural network. The network is trained to predict the CF factors of a song given its log-compressed mel-spectrogram. However, the two approaches are fundamentally different. Our approach integrates collaborative information and song features into a standalone enhanced recommendation model that can decide if any playlist-song pair matches. In contrast, the model proposed in [40] tries to transform audio features into more abstract collaborative features, which still need to be processed to yield a final recommendation. Also, its performance is naturally upper-bounded by CF.

For a more comprehensive survey on music playlist continuation, we point the interested reader to [5] or Chapter 13 in [34].

3 HYBRID PLAYLIST CONTINUATION

In this section we introduce the proposed hybrid playlist continuation model, including: basic concepts and notation, the data used to train it, the model definition, how to use it to recommend playlist continuations and finally some implementation details.

3.1 Basic Concepts and Notation

Let P be a collection of music playlists. Let S be the universe of songs available, including at least all the unique songs occurring in the playlists of P, but possibly more. A song $s \in S$ is represented by a feature vector $\mathbf{x}_s \in \mathbb{R}^D$, where D is the song-feature dimensionality. A playlist $p \in P$ of length T_p is regarded as a set of songs and it is represented by a feature matrix $\mathbf{X}_p \in \mathbb{R}^{T_p \times D}$ that contains, in each row, the feature vector of each song in the playlist. Different playlists may have different lengths.

We want to remark that, by considering a playlist as a set of songs, we are disregarding its song order. This assumption may seem counterintuitive because the process of listening to a playlist is inherently sequential. However, our preliminary studies on the importance of song order in curated music playlists indicate that the order is actually not crucial to recommend continuations to such playlists. Even though more research is required to fully understand the impact of the song order, we feel confident that disregarding the song order does not harm the contribution of the current work.

Throughout the paper, we take advantage of standard set operations and notation [23, chap.1] to precisely express relations between playlists and songs. We advance here common expressions that we normally use. A playlist p is a set of songs and thus it is a subset of the universe of songs, i.e., $p \subseteq S$. Then, the set difference $S \setminus p$ corresponds to all the songs in the universe S that do not belong to the playlist p. A song s can be regarded as playlist of one song, i.e., as the singleton set $\{s\}$. Given a song in a playlist $s \in p$, the set difference $p \setminus \{s\}$ removes the song s from the playlist p. Despite the use of set operations and notation, for the sake of clarity we continue using the word "length" instead of "cardinality" to refer to the number of songs in a playlist.

3.2 Playlist-Song Training Examples

The proposed hybrid playlist continuation model is based on what we name "playlist-song membership," that is, whether a given playlist and a given song fit together. In Section 2 we have discussed the complexity of deciding which songs fit together and how approaches that exploit collaborative information are able to reveal more abstract relations than content-based approaches. Therefore we use curated playlists as a form of collaborative implicit feedback [18, 31] to derive training examples of playlist-song pairs that fit, as well as of playlist-song pairs that do not fit.

To ease the reading we say that a playlist-song pair that fits is a "match," while a playlist-song pair that does not fit is a "mismatch." In the context of implicit feedback it might be more accurate to use the term "no-match" rather than "mismatch," to stress the fact that missing feedback does not necessarily reflect negative feedback, but we keep the latter for simplicity.

Our basic assumption is that any playlist $p \in P$ implicitly defines matches with its own songs, in the sense that any song $s \in p$ matches the shortened playlist $p_s = p \setminus s$, i.e., the original playlist p where the song s has been removed. We further assume that any song not occurring in the playlist p is a mismatch to the shortened playlist p_s. Thus a mismatch is obtained by randomly drawing a song from the remaining songs $S \setminus p$. In this way we can derive training examples of matching and mismatching playlist-song pairs. Precisely, we follow Algorithm 1, that given a playlists collection P and a universe of songs S yields as many matching as mismatching playlist-song pairs.

Algorithm 1 Derive playlist-song matches and mismatches.

```
Input:
    $P$                                  ▷ playlists collection
    $S$                                  ▷ universe of songs
Output:
    matches                              ▷ list of playlist-song matches
    mismatches                           ▷ list of playlist-song mismatches

 1: matches = []                         ▷ initialize empty lists
 2: mismatches = []
 3: for $p \in P$ do
 4:     for $s \in p$ do
 5:         $p_s = p \setminus \{s\}$                  ▷ remove $s$ from $p$
 6:         $s_+ = s$                        ▷ $s$ is a match to $p_s$
 7:         $s_- =$ sample($S \setminus p$)          ▷ draw a mismatch to $p_s$
 8:         matches.append(($p_s, s_+$))     ▷ store train. examples
 9:         mismatches.append(($p_s, s_-$))
10:     end for
11: end for
12: return matches, mismatches
```

It is important to observe that, in general, considering playlist-song pairs observed in curated playlists as matches, and any playlist-song pairs not observed in curated playlists as mismatches, would yield many more mismatches than matches. This strong class imbalance is well-known in the domain of recommender systems [1, 34] and can be tackled with different approaches such as cost-sensitive learning [18, 31] or sampling techniques [27]. As we have seen in Algorithm 1, we opt for the latter, and derive a balanced number of matching and mismatching training examples.

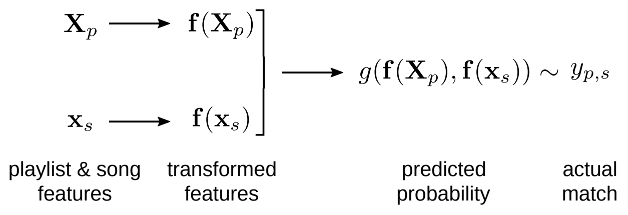
$\mathbf{X}_p \longrightarrow \mathbf{f}(\mathbf{X}_p)$
$\mathbf{x}_s \longrightarrow \mathbf{f}(\mathbf{x}_s)$
$\longrightarrow g(\mathbf{f}(\mathbf{X}_p), \mathbf{f}(\mathbf{x}_s)) \sim y_{p,s}$
playlist & song features
transformed features
predicted probability
actual match

Figure 1: Sketch of the hybrid playlist continuation model. Given any playlist-song pair, its feature matrix and vector are transformed to hidden representations that are then used to decide if the playlist-song pair is a match or a mismatch. The model is trained on labeled playlist-song pairs derived from Algortithm 1.

The training examples derived from Algorithm 1 are formatted in order to use them within the proposed playlist continuation model. Each playlist-song pair (p, s) is represented by its feature matrix and vector $(\mathbf{X}_p, \mathbf{x}_s)$, and it is labeled with $y_{p,s} \in \{0, 1\}$, where 1 indicates a matching playlist-song pair and 0 indicates a mismatching playlist-song pair. The final training dataset consists of all the triplets $\{(\mathbf{X}_p, \mathbf{x}_s), y_{p,s}\}$.

3.3 Model Definition and Learning

The proposed hybrid playlist continuation model has to be able to decide if any given playlist-song pair constitutes a match or a mismatch. We generally devise this model as a deep neural network consisting of a "feature-transformation" component $\mathbf{f}$ and a "match-discrimination" component g (Figure 1). The feature-transformation component takes any playlist-song pair (p, s) as input, represented by the corresponding feature matrix and vector $(\mathbf{X}_p, \mathbf{x}_s)$. The song feature vector $\mathbf{x}_s$ is transformed to an H-dimensional hidden representation $\mathbf{f}(\mathbf{x}_s) \in \mathbb{R}^H$. The playlist feature matrix $\mathbf{X}_p$ is song-wise subject to the same transformation, obtaining a hidden representation $\mathbf{f}(\mathbf{X}_p) \in \mathbb{R}^{T_p \times H}$ (where we slightly abuse notation for $\mathbf{f}$). Both hidden representations are passed together through the match-discrimination component that predicts the probability of the playlist-song pair being a match: $g\left(\mathbf{f}(\mathbf{X}_p), \mathbf{f}(\mathbf{x}_s)\right) \in [0, 1]$.

The transformation component $\mathbf{f}$ and the match-discrimination component g depend on sets of learnable weights $\boldsymbol{\theta}_\mathbf{f}$ and $\boldsymbol{\theta}_g$, respectively (omitted so far for the sake of clarity). The weights are adjusted on the basis of the training examples $\{(\mathbf{X}_p, \mathbf{x}_s), y_{p,s}\}$ derived from Algorithm 1, by comparing the model's predicted match probability for a pair (p, s) to the actual match label $y_{p,s} \in \{0, 1\}$. Precisely, the sets of weights $\boldsymbol{\theta}_\mathbf{f}, \boldsymbol{\theta}_g$ are estimated to minimize the following binary cross-entropy cost function

$$\mathcal{L}\Big(\boldsymbol{\theta}_\mathbf{f}, \boldsymbol{\theta}_g \;\Big|\; \big\{(\mathbf{X}_p, \mathbf{x}_s), y_{p,s}\big\}\Big) = \\ -\sum_{p,s} y_{p,s} \log\left(\hat{y}_{p,s}\right) + \left(1 - y_{p,s}\right) \log\left(1 - \hat{y}_{p,s}\right), \tag{1}$$

where $\hat{y}_{p,s} = g\big(\mathbf{f}(\mathbf{X}_p), \mathbf{f}(\mathbf{x}_s)\big)$ is the model's prediction for the playlist-song pair (p, s).

In Section 1 we pointed out that the proposed playlist continuation model is a feature-combination hybrid [6], that is, it implicitly fuses content-based and collaborative information into a joint recommender system. Equation 1 clearly shows how the proposed model relies on matching and mismatching playlist-song pairs derived from curated music playlists, while regarding playlist-song pairs in terms of their feature representations.

3.4 Playlist Continuation

Once it has been trained, the proposed model is used to predict playlists continuations in the following way. Suppose that we recommend a continuation to playlist p using the universe of songs S. To avoid the recommendation of songs already present in the playlist, we restrict the set of candidate songs to $S \setminus p$. We let the model predict the matching probability of the playlist-song pair (p, s), for each candidate song $s \in S \setminus p$. This operation is linear in the number of candidate songs. We then rank the candidate songs in order of preference to extend p. The ranked list of song candidates can be used differently depending on the recommendation scenario. For example, in case of extending a radio stream one could recommend the top result, or assisting a user in identifying relevant songs to extend a playlist, one could show the top K results, or the results with predicted matching probability above a threshold. We detail the evaluation methodology followed for our experiments in Section 4.

3.5 Model Implementation

The results presented in this work (Section 6) were obtained by using the model architecture detailed in Table 1. The model hyperparameters were selected on a withheld validation set, choosing those that yielded best validation cost. The model was implemented using Lasagne [11] and Theano [37].

4 EVALUATION

A large-scale on-line evaluation where users could assess the quality of the playlist continuations recommended by the proposed model should be the preferred option [34], but would require a complex infrastructure beyond the scope of this work. Instead we design an off-line evaluation experiment, similar to previous approaches in the literature [2, 5, 16, 20], where we assess the ability of the proposed model to recover withheld playlist continuations. Even though off-line experiments can not faithfully assess the quality of playlist continuations as would do it a user, we deem reasonable to assume that if a playlist continuation approach consistently achieves better off-line performance than another, it will likely perform better in practice as well.

4.1 Off-line Experiment and Metrics

As a reminder, let P be a collection of music playlists and let S be the universe of songs available, including at least all the unique songs occurring in the playlists of P, but possibly more. Given a playlist p, we assume that a continuation p_c, proportionally shorter than p, is known and withheld for test. This assumption on the length of the continuation p_c follows the evaluation methodology used by Aizenberg et al. [2], but differs from the approach of Bonnin and Jannach [5], Hariri et al. [16] and Jannach et al. [20], where the withheld continuations have a single song regardless of the length of the playlist p.

Table 1: Architecture of the hybrid playlist continuation model. The input to the model are the features $(\mathrm{X}_p, \mathrm{x}_s)$ of a playlist-song pair (p, s). X_p^t denotes the t-th row of the feature matrix X_p, i.e., the t-th song of the playlist p. The upper part of the table corresponds to the feature transformation component f, and the lower part of the table corresponds to the match-discrimination component g. The boldface layers $\mathrm{DE_k}$, $\mathrm{BN_k}$ in the transformation component f share their weights for all the songs in the playlist p and for song s (Section 3.3). The dimensionality of each layer is annotated in parentheses. DE: Dense layer, RE: Rectify non-linearity, BN: Batch Normalization [19], DR: Dropout [36].

	X_p^1 (D)	$\dots$	$\mathrm{X}_p^{T_p}$ (D)	x_s (D)
f :	$\mathbf{DE_1}$ + RE (512)	$\dots$	$\mathbf{DE_1}$ + RE (512)	$\mathbf{DE_1}$ + RE (512)
	$\mathbf{BN_1}$ + DR (512)	$\dots$	$\mathbf{BN_1}$ + DR (512)	$\mathbf{BN_1}$ + DR (512)
	$\mathbf{DE_2}$ + RE (512)	$\dots$	$\mathbf{DE_2}$ + RE (512)	$\mathbf{DE_2}$ + RE (512)
	$\mathbf{BN_2}$ + DR (512)	$\dots$	$\mathbf{BN_2}$ + DR (512)	$\mathbf{BN_2}$ + DR (512)
	Average (512)			
	Concatenate (1024)			
g :	$\mathrm{DE_3}$ + RE (512)			
	$\mathrm{BN_3}$ + DR (512)			
	$\mathrm{DE_4}$ + RE (512)			
	$\mathrm{BN_4}$ + DR (512)			
	$\mathrm{DE_5}$ + RE (1)			
	Loss			

To prevent the model from recommending songs already present in the playlist p, we only consider the songs in $S \setminus p$, and rank them according to the model's predicted probability that they match p. Since this is an off-line experiment, now we do not really prepare a recommended playlist continuation for a user to evaluate. Instead, we compute different evaluation metrics based on the ability of the playlist model to rank the songs within the playlist continuation p_c in top positions of the list of ranked song candidates. For each song in the withheld continuation p_c, we compute its rank within the whole list of ranked song candidates. For the whole continuation p_c, we compute the average precision given the whole list of ranked song candidates. Finally, for the whole continuation p_c, we compute the recall within lists of top 10, top 30 and top 100 ranked song candidates [28, chap.8]. Even though lists of top 30 or top 100 song candidates may seem impractical for actual recommendation scenarios, these are common list lengths used to evaluate playlist continuations in off-line experiments [2, 5, 16, 20].

This process is repeated for all the playlists we set to extend, and a summary of the described evaluation metrics over all the playlist continuations is reported, namely the median rank, the mean average precision (MAP) and the mean recall at 10, 30 and 100.

4.2 Weak and Strong Generalization

We consider two different evaluation settings (Figure 2), that were also proposed by Aizenberg et al. [2]. The first setting, or "weak generalization" setting, considers a single set of playlists with their corresponding withheld continuations. The playlists are used to train the proposed playlist continuation model. Once trained, the model predicts lists of ranked song candidates to extend the very

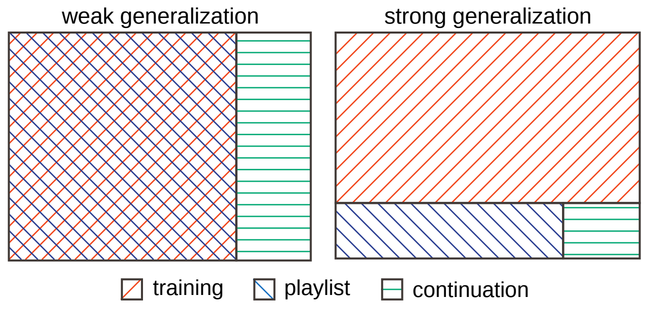

Figure 2: Illustration of the considered evaluation settings. We figuratively picture the datasets as matrices where playlists are stored in rows, and each element in a playlist is a song. The red stripes indicate the playlists used to train the model. The blue stripes indicate the playlists that the model has to extend. The green stripes indicate the withheld continuations used for evaluation. The weak generalization setting considers a set of playlists and their withheld continuations. The model is trained on the playlists. Once trained, we assess the ability of the model to extend the very same training playlists. The strong generalization setting considers two independent sets of playlists. The first set of playlists is used to train the model. Once trained, we assess the ability of the model to extend the playlists from the second set of playlists.

same training playlists. The model is then evaluated using the withheld continuations as describe above. We refer to this evaluation setting as "weak," because the playlist continuation model extends playlists that it has seen before. In Section 2 we discussed that some playlist continuation models (e.g., a latent-factor CF model), need to compute a playlist profile at training time in order to extend a playlist. Such models can operate in the weak generalization setting. The second setting, or "strong generalization" setting, considers two independent sets of playlists. The first set of playlists does not need known withheld continuations because it is only used to train the model, while the second set of playlists does require withheld continuations for evaluation. The playlist continuation model is trained on the playlists from the first set. Once trained, the model is shown the playlists from the second set and it predicts lists of ranked song candidates to extend them. The model is then evaluated using the withheld continuations as described in Section 4.1. We refer to this evaluation setting as "strong," because the playlist continuation model extends playlists that it has never seen before. Playlist continuation models that require a precomputed playlist profile to extend a playlist can no operate in the strong generalization setting.

4.3 Collaborative Filtering Baseline

We compare the proposed playlist continuation model to a latent-factor CF baseline. We choose the state-of-the-art Weighted Matrix Factorization (WMF) model introduced by Hu et al. [18] because it is specifically designed to perform CF on implicit feedback datasets. We apply WMF to the task of playlist continuation by regarding playlists as listening histories. We set a matrix with as many rows as playlists and as many songs as unique songs in the playlists. In other words, the general CF user-item matrix becomes a playlist-song matrix. If a given playlist contains a given song, then the corresponding cell in the matrix is set to 1. Otherwise, the cells are set to 0. The original model by Hu et al. [18] expects frequency information for each observation, e.g., the number of times a user viewed a website or the play counts of a user for a specific song. This frequency information is leveraged to assign appropriate confidence levels to each observation in the matrix. Curated playlists lack such frequency information because they are *static* lists compiled by practitioners, shared on-line and finally encoded as binary values that only indicate which songs belong to the playlists. Thus, we assign the same weight to all the observations, but we tune this weight to yield best validation cost on a withheld validation set. We also experiment with different weights for the L2-regularization term. We use the publicly available implementation of WMF by Frederickson [15].

WMF can only operate on the weak generalization setting because it needs to precompute playlist profiles, in this case playlists factors, in order to extend the playlists. Furthermore, as any pure CF model, WMF can only recommend songs occurring in training playlists. Details regarding the comparison of the proposed playlist continuation model and this CF baseline are included in Section 6.

5 DATASETS

We evaluate the proposed playlist continuation model on two datasets of hand-curated playlists, along with song features derived from the Million Song Dataset[1] (MSD) [3].

The "AotM-2011" dataset [30] is a playlists collection derived from the Art of the Mix on-line database.[2] Each playlist in the AotM-2011 dataset is represented by a list of song titles, artist names, and links to the MSD identifiers, where available. We also have access to a private playlists collection from "8tracks,"[3] an on-line platform where users share playlists and that supports listening to them through streaming. Each playlist in the 8tracks collection is represented by song titles and artist names. We use fuzzy string matching to resolve them against the MSD, adapting the code released by Jansson et al. [22] for a very similar task. The string matching results in roughly 2,5M song identifiers from the 8tracks dataset (many are spelling duplicates) resolved into 241,123 song identifiers from the MSD. Linking the 8tracks dataset to the MSD enables the extraction of song features and makes the comparison to the AotM-2011 dataset fair.

5.1 Playlists Datasets

The AotM-2011 dataset contains a considerable number of artist- and album-themed playlists. One may argue that such playlists should be considered if they are important to playlist curators, but we deem important to exclude them for two main reasons. Firstly, we presume that playlists with several songs by the same artist or from the same album may correspond to a not so careful compilation process. Secondly, and most importantly, we utilize song features that are partly derived from social tags (Section 5.2). By manual inspection we observe that some tags inform about the artist or the album. Thus, by rejecting artist- and album-themed playlists we prevent evaluation problems regarding leaking artist or album information. Note that, in any case, excluding these playlists

[1]https://labrosa.ee.columbia.edu/millionsong
[2]http://www.artofthemix.org
[3]https://8tracks.com

makes the problem harder. To discard artist- and album-themed playlists we keep only the playlists with at least 7 unique artists and with a maximum of 2 songs per artist (the thresholds were manually chosen to yield sufficient training playlists after the whole filtering process). The 8tracks dataset has not artist- or album-themed playlists because the terms of use of the 8tracks platform do not allow users to include in a playlist more than 2 songs by the same artist or from the same album. However, we apply the same filters to both datasets for the sake of consistency. To ensure that the model learns from playlists of sufficient length, we further filter both datasets by keeping only the playlists with at least 14 songs linked to the MSD. However, we also discard songs for which features can not be extracted due to missing raw data and therefore the final length of the playlists may be shortened.

In order to set up the training and the test sets, we discard playlists that have become shorter than 5 songs after the song filtering. For the weak generalization setting, we split each playlist leaving approximately 80% of the songs as a training playlist and the remaining 20% of the songs as a withheld continuation. For the strong generalization setting, we split the AotM-2011 and the 8tracks playlists collections into two playlist-disjoint subcollections each. In each case, the first playlists subcollection includes 80% of the playlists, that will be used as training playlists. The second playlists subcollection includes 20% of the playlists, that are further split leaving approximately 80% of the songs as a playlist to be extended, and the remaining 20% of the songs as a withheld continuation. Figure 2 illustrates the different playlists splits in the weak and strong generalization settings.

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. Detailed statistics regarding the distribution of unique songs per playlist, unique artists per playlist and song frequency in the dataset are included in Table 2.

5.2 Song Features

The proposed playlist continuation model requires each song in the playlists to be represented by a feature vector. We extract state-of-the-art song features, namely i-vectors from audio, word2vec semantic features from social tags and collaborative song latent factors from independent listening logs. We concatenate these features into a rich multimodal song feature vector. The choice of this specific set of features is motivated by our recent study [38], that elaborates on the performance of these and other song features for the task of music playlist continuation, including the individual performance of each type of feature, as well as their joint performance when they are combined into a multimodal feature vector. The feature extraction process is outlined below, but we refer the interested reader to [38] for a comprehensive presentation.

We derive the features from the MSD, together with the accompanying "Last.fm Dataset"[4] and the "Taste Profile Subset."[5] The raw data available is the following. 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. The Last.fm Dataset provides song-level social tags along with weights describing their relevance. Finally, the Taste Profile Subset includes user-song play counts derived from independent listening logs. The MSD also provides other song features such as "danceability" or "energy," that are expected to summarize aspects of the audio at a high-level. However, these features are not documented in the MSD nor were they in their original source, the now discontinued Echo Nest API.[6] Thus we do not consider them for research purposes.

The extraction of i-vectors and word2vec semantic features requires pretraining reference models on a large collection of representative songs. For both the AotM-2011 and the 8tracks datasets, 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. We assume that the unique songs in the resulting playlists are representative. For each dataset, we further exclude the songs that appear only in the corresponding weak-generalization training split to minimize leaking information in the evaluation. We refer to the final 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.

For all the feature types we extract 200-dimensional feature vectors. According to our experiments in this and our previous work [38], vectors of this dimensionality carry enough information.

5.2.1 *I-Vectors from Timbral Features.*

I-vectors were first introduced in the field of speaker verification [10], but recently they have also 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 unique songs in the playlists, we derive the total variability space yielding 200-dimensional i-vectors. Following the standard i-vector extraction pipeline, we further transform the obatained i-vectors using a linear discriminant analysis model [17] fit on the training playlists.

5.2.2 *Semantic Features from Social Tags.*

We collect the social tags of all the songs in the development song set and build a music-aware text corpus by fetching the English Wikipedia[7] pages of the collected tags. We run the continuous bag-of-words algorithm[8] on the text corpus to obtain a dictionary of 200-dimensional semantic features for the most relevant words in the corpus. For each unique song in the playlists, we look up its social tags in the dictionary. If a tag is a compound of several words (e.g., "pop rock"), we compute the average feature. Since a song may have several tags, the final semantic feature is the weighted average of all its tags' features, where the weights are provided by the Last.fm Dataset and indicate how relevant is each tag for each song.

5.2.3 *Latent Factors from Listening Logs.*

We factorize the user-song play counts from the MSD using the WMF model [18], that is specially designed for implicit feedback datasets. We use a depth of 200 factors. We discard the user latent factors, unrelated to the playlist continuation problem, and keep only the song latent factors.

5.2.4 *Multimodal Feature Vectors.*

For each song in the playlists, we concatenate its i-vector, word2vec semantic feature vector and

[4]https://labrosa.ee.columbia.edu/millionsong/lastfm
[5]https://labrosa.ee.columbia.edu/millionsong/tasteprofile
[6]http://the.echonest.com
[7]https://en.wikipedia.org
[8]https://code.google.com/p/word2vec

Table 2: Descriptive statistics for the playlists within the AotM-2011 and the 8tracks playlists datasets. 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

collaborative song factors into a multimodal feature vector. Since each individual feature vector is 200-dimensional, the final multimodal feature vector is 600-dimensional.

6 RESULTS

We evaluate the proposed playlist continuation model by conducting the off-line experiment described in Section 4.1, in both the weak and the strong generalization settings proposed in Section 4.2, using both the AotM-2011 and the 8tracks datasets. We also evaluate the CF baseline presented in Section 4.3. Since it can only extend playlists for which playlist factors have been computed at training time, we can only evaluate it in the weak generalization setting. We finally evaluate a random baseline that given any playlist-song pair predicts a random probability of it being a match. The performance of the random baseline is independent of the generalization setting.

Table 3 reports the performance metrics achieved by each of the playlist continuation models. In the weak generalization setting, the proposed playlist continuation model and the CF baseline show comparable performance in both datasets. The proposed model is able to rank the songs from the withheld continuations roughly 200 positions higher (better) than the CF baseline. On the other hand, the CF baseline achieves slightly higher recall values than the proposed model. By comparison to the random baseline, we see that the proposed model and the CF baseline reveal non trivial patterns in the data. However, we also note that the absolute performance metrics are rather low. Platt et al. [32] and McFee and Lanckriet [29] also pointed out the low performances achieved when automated music playlist continuation is evaluated as an information retrieval task. They explained it by the nature of the playlist continuation problem, namely a playlist may be extended by a number of potentially relevant songs, but information retrieval off-line metrics only accept exact matches to the withheld continuations. In this regard, it is also worth noting that related works on the AotM-2011 and the 8tracks datasets report results comparable to ours [5, 20]. We also observe that both the proposed model and the CF baseline achieve slightly worse results on the AotM-2011 dataset than in the 8tracks dataset. Since the differences are comparable for both models, we believe that this is due to the specific properties of each dataset.

As we have discussed in Section 4.2, the proposed playlist continuation model can evaluate any playlist-song pair, regardless of whether the playlist and the song were observed before. This flexibility enables the proposed model to extend playlists in the strong generalization setting. However, the question is whether the proposed model's performance is harmed in this evaluation setting. Still in Table 3, we observe that the median rank in the strong generalization setting is roughly 150 positions lower (worse) than in the weak generalization for the AotM-2011 dataset. However, the median rank remains stable for the 8tracks dataset. Similarly, the recall values in the strong generalization setting are slightly lower in the AotM-2011 dataset, but comparable for the 8tracks dataset. Overall, even though the strong generalization setting poses a much harder task, the proposed model shows a performance comparable to its own performance in the weak generalization setting.

We now analyze the robustness of the proposed playlist continuation model to recommend rare and out-of-set songs. Figure 3 shows the results from the off-line experiments discussed above as a function of how often the songs in the withheld continuations occurred in training playlists. We first focus on the weak generalization results, where we can compare the proposed model and the CF baseline. As expected, both models achieve best performances when they recommend songs that occurred in 5 or more training playlists. As we know, the CF baseline can not recommend out-of-set songs, but we also observe that is has severe difficulties to recommend songs occurring rarely in training playlists. The performance of the proposed model on rare and out-of-set songs is not as good as on songs occurring in five or more playlists. Nevertheless, the proposed model is able to rank such songs much better than a random model would (Table 3) and clearly better than the CF baseline.

We finally focus on the robustness of the proposed playlist continuation model to recommend rare and out-of-set songs in the strong generalization setting. As before, we want to investigate if the proposed model's performance is harmed in this evaluation setting, compared to its performance in the weak generalization setting. Still in Figure 3, we observe that the behavior of the proposed model in the weak and strong generalization settings is fairly similar, in line with the overall results discussed above (Table 3). This is an interesting finding because it provides an empirical indication that regarding playlist-song pairs exclusively in terms of their feature representations favors generalization and discourages the specialization towards particular training playlists.

7 CONCLUSION

We have introduced a novel hybrid playlist continuation model based on the general notion of playlist-song membership. The

Table 3: Results achieved by the proposed playlist continuation model, the CF baseline and a random baseline on the off-line evaluation experiment. We report the median rank, the MAP and the recall (R) at lists of top10, top30 and top100 song candidates. 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 R@{10, 30, 100} higher is better.

dataset	generalization	model	med rank	MAP	R@10	R@30	R@100
AotM-2011	weak	proposed	1230	1.35%	2.62%	6.32%	13.89%
		CF	1444	1.96%	3.99%	7.84%	14.56%
	strong	proposed	1395	1.10%	1.93%	4.63%	11.65%
		CF	—	—	—	—	—
	weak & strong	random	6087	0.11%	0.20%	0.28%	0.79%
8tracks	weak	proposed	726	2.31%	4.34%	9.39%	20.02%
		CF	1000	2.65%	5.06%	10.14%	19.60%
	strong	proposed	706	2.41%	4.20%	9.58%	19.36%
		CF	—	—	—	—	—
	weak & strong	random	7320	0.09%	0.12%	0.23%	0.66%

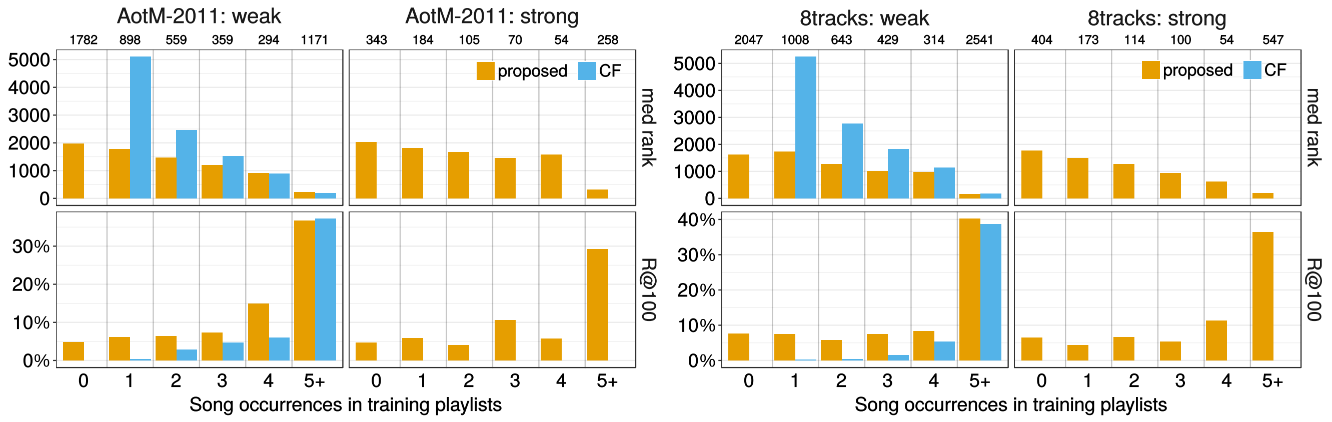

Figure 3: Results achieved by the proposed playlist continuation model and the CF baseline on the off-line experiment as a function of how often the songs in the withheld continuations occurred in training playlists. We report the median rank (lower is better) and the recall at 100 (R@100, higher is better). The text annotations on top of each panel indicate the number of observations in each bar, which are the same for the proposed model and the CF baseline.

model integrates collaborative information encoded in curated playlists with state-of-the-art multimodal song features derived from audio, social tags and independent listening logs. By design, it regards playlists-song pairs exclusively in terms of their feature vectors, seeking to discourage the specialization towards specific playlists and songs. As a consequence, in contrast to CF models limited to extending previously profiled playlists with songs seen at training time, the proposed playlist continuation model can flexibly decide the fitness of any playlist-song pair, regardless of whether the playlist and the song were observed at training time. Furthermore, we follow a feature-combination hybrid approach, that is, the different sources of information are implicitly fused into a standalone enhanced playlist continuation model. According to our experimental results, the proposed model compares to a state-of-the art CF model for the task of extending profiled playlists, when sufficient training data is available. In contrast to the CF model, the proposed model can additionally extend non-profiled playlists without a significant performance loss, and recommend rare and out-of-set songs with fair performance. We believe that the current work provides a proof of concept for this newly proposed approach to playlist modeling. The natural next step is the comprehensive benchmarking of the proposed approach against other competing techniques and the identification of improvable aspects.

8 ACKNOWLEDGMENTS

We thank Christine Bauer, Hamid Eghbal-Zadeh and Rocío del Río Lorenzo for their valuable comments and the helpful discussions.

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] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.

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

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

[9] 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*.

[10] Najim Dehak, Patrick J Kenny, RÃĩda 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.

[11] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gábor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degrave. 2015. Lasagne: First release. (Aug. 2015). http://dx.doi.org/10.5281/zenodo.27878

[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*. 131–131.

[17] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2008. *The elements of statistical learning*. Springer series in statistics.

[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 preprint 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] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proc. RecSys*. 306–310.

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

[23] Irving Kaplansky. 1972. *Set theory and metric spaces*. Allyn and Bacon, Inc.

[24] 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.

[25] Jin Ha Lee, Bobby Bare, and Gary Meek. 2011. How similar is too similar?: Exploring users' perceptions of similarity in playlist evaluation. In *Proc. ISMIR*. 109–114.

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

[27] Oded Maimon and Lior Rokach (Eds.). 2010. *Data mining and knowledge discovery handbook*. Springer US, Boston, MA.

[28] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2009. *An introduction to information retrieval*. Cambridge University Press.

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

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

[31] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proc. ICDM*. 502–511.

[32] John C. Platt, Christopher JC Burges, Steven Swenson, Christopher Weare, and Alice Zheng. 2002. Learning a Gaussian process prior for automatically generating music playlists. In *Advances in Neural Information Processing Systems*. 1425–1432.

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

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

[35] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proc. WWW*. 285–295.

[36] 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.

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

[38] 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: Alleviating the cold-start problem for rare and out-of-set songs. In *Proc. Workshop on Deep Learning for Recommender Systems, DLRS@RecSys*. Como, Italy, 46–54.

[39] 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*.

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

[41] 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.