User Modeling and User-Adapted Interaction manuscript No.
(will be inserted by the editor)

Feature-Combination Hybrid Recommender Systems for Automated Music Playlist Continuation

Andreu Vall · Matthias Dorfer · Hamid Eghbal-zadeh · Markus Schedl · Keki Burjorjee · Gerhard Widmer

Received: date / Accepted: date

Abstract Music recommender systems have become a key technology to support the interaction of users with the increasingly larger music catalogs of on-line music streaming services, on-line music shops, and personal devices. An important task in music recommender systems is the automated continuation of music playlists, that enables the recommendation of music streams adapting to given (possibly short) listening sessions. Previous works have shown that applying collaborative filtering to collections of curated music playlists reveals underlying playlist-song co-occurrence patterns that are useful to predict playlist continuations. However, most music collections exhibit a pronounced long-tailed distribution. The majority of songs occur only in few playlists and, as a consequence, they are poorly represented by collaborative filtering. We introduce two feature-combination hybrid recommender systems that extend collaborative filtering by integrating the collaborative information encoded in curated music playlists with any type of song feature vector representation. We conduct off-line experiments to assess the performance of the proposed systems to recover withheld playlist continuations, and we compare them to competitive pure and hybrid collaborative filtering baselines. The results of the experiments indicate that the introduced feature-combination hybrid recommender systems can more accurately predict fitting playlist continuations as a result of their improved representation of songs occurring in few playlists.

Keywords automated music playlist continuation · hybrid recommender systems · cold-start problem · music information retrieval · feature extraction

Andreu Vall · Matthias Dorfer · Hamid Eghbal-zadeh · Markus Schedl · Gerhard Widmer
Institute of Computational Perception, Johannes Kepler University Linz, Austria
E-mail: name.surname@jku.at

Gerhard Widmer
Austrian Research Institute for Artificial Intelligence, Vienna, Austria

Keki Burjorjee
Pandora Media Inc., Oakland, CA, USA
E-mail: kburjorjee@pandora.com

1 Introduction

Music recommender systems have become an important component of music platforms to assist users to navigate increasingly larger music collections. Recommendable items in the music domain may correspond to different entities such as songs, albums, or artists (Chen et al, 2016; Ricci et al, 2015, Chapter 13), and music streaming services even organize music in more abstract categories, like genre or activity.

As a consequence of the relatively short time needed to listen to a song (compared to watching a movie or reading a book) a user session in an on-line music streaming service typically involves listening to, not one, but several songs. Thus, modeling and understanding music playlists is a central research goal in music recommender systems. As in other item domains, music recommender systems often provide personalized lists of suggestions based on the users' general music preferences. This approach may work to recommend music entities such as albums, artists, or ready-made listening sessions (like curated playlists or charts) because it can be useful to provide the users with a wide choice range. However, recommendations based on the users' general music preferences may be too broad for the task of automated music playlist continuation, where it is crucial to recommend individual songs that specifically adapt to the most-recent songs played.

A common approach to explicitly address the automated continuation of music playlists consists in applying Collaborative Filtering (CF) to curated music playlists, revealing specialized playlist-song co-occurrence patterns (Aizenberg et al, 2012; Bonnin and Jannach, 2014). While this approach works fairly well, it has an important limitation: the performance of any CF system depends on the availability of sufficiently dense training data (Adomavicius and Tuzhilin, 2005). In particular, songs occurring in few playlists can not be properly modeled by CF because they are hardly related to other playlists and songs. Music collections generally exhibit a bias towards few, popular songs (Celma, 2010). In the case of collections of curated music playlists, this translates into a vast majority of songs occurring only in very few playlists. This majority of infrequent songs is poorly represented by CF.

To overcome this limitation, we observe that songs occurring rarely in the context of curated playlists are not necessarily completely unknown to us. We can often gather rich song-level side information from, e.g., the audio signal, text descriptions from social-tagging platforms, or even listening logs from music streaming services. Such additional song descriptions can be leveraged to make CF robust to infrequent songs by means of hybridization (Adomavicius and Tuzhilin, 2005; Burke, 2002).

We introduce two feature-combination hybrid recommender systems that integrate curated music playlists with any type of song feature vector derived from song descriptions. The curated music playlists provide playlist-song co-occurrence patterns as in CF approaches. The song features make the proposed systems robust to data scarcity problems. In contrast to previous hybrid playlist continuation approaches, the proposed systems are *feature-combination* hybrids (Burke, 2002), having the advantage that the collaborative information and the song features are implicitly fused into standalone enhanced recommender systems. The proposed systems can be used to play and sequentially extend music streams, resulting in a *lean-back*

listening experience similar to traditional radio broadcasting, or to assist users to find fitting songs to extend their own music playlists, stimulating their engagement.

1.1 Contributions of the paper

- We provide a unified view of music playlist continuation as a matrix completion and expansion problem, encompassing
  - pure CF systems, solely exploiting curated playlists,
  - hybrid systems integrating curated playlists and song feature vectors.
- We introduce two feature-combination hybrid recommender systems
  - readily applicable to automated music playlist continuation,
  - able to exploit any type of song feature vectors.
- Still, the proposed systems are domain-agnostic. They can generally leverage
  - collaborative implicit feedback data from any domain,
  - item feature vectors from any domain and modality.
- A thorough off-line evaluation comparing to pure and hybrid state-of-the-art CF baselines shows that, having access to comparable data, the proposed systems
  - compete to CF when sufficient training data is available,
  - outperform CF when training data is scarce,
  - compete to, or outperform the hybrid baseline.
- The proposed systems further improve their performance by considering richer song feature vectors, e.g., concatenating features from different modalities.
- The evaluation also provides a complete comparison of
  - a widely-used matrix factorization CF system (Hu et al, 2008),
  - its audio-based hybrid extension (van den Oord et al, 2013),
  - a popular playlist-neighbors CF system (Bonnin and Jannach, 2014).
- The Appendix extends the evaluation of the proposed systems with a detailed analysis of the contribution of each type of song feature vector, showing
  - the standalone performance of each type of song feature vector,
  - the incremental gains of stepwise combinations of song feature vectors.

1.2 Scope of the paper

Compiling a music playlist is a complex task. According to interviews with practitioners and postings to a playlist-sharing website, Cunningham et al (2006) found that the playlist curation process is influenced by factors such as mood, theme, or purpose. They also observed a lack of agreement on curation rules, except for loose and subjective guidelines. Krause and North (2014) studied music listening in situations. Among other conclusions, they found that participants of their study, when asked

to compile playlists for specific situations, selected music seeking to comply with perceived social norms defining what music ought to be present in each situation.

The scope of our work is restricted to machine learning approaches to music recommender systems. We focus on the exploitation of data describing playlists and the songs therein, in order to identify patterns useful to recommend playlist continuations. We acknowledge the complexity of the playlist curation process, and we are aware of the possible limitations of a pure machine-learning perspective.

1.3 Organization of the paper

The remainder of the paper is organized as follows. Section 2 reviews previous works on music playlist continuation. Section 3 formulates music playlist continuation as a matrix completion and expansion problem. Sections 4 and 5 describe the proposed systems and the baselines for music playlist continuation, respectively. The evaluation methodology is presented in Section 6. Section 7 describes the datasets of curated playlists and song features used in our experiments. Section 8 elaborates on the results. Finally, conclusions are drawn in Section 9. Additional details of each playlist continuation system, additional song feature types, and additional results are provided in Appendices A, B and C, respectively.

2 Related work

Content-based recommender systems for automated music playlist continuation generally compute pairwise song similarities on the basis of previously extracted song features and use these similarities to enforce content-wise smooth transitions. Such systems have typically relied on audio-based song features (Flexer et al, 2008; Logan, 2002; Pohle et al, 2005), possibly combined with features extracted from social tags (McFee and Lanckriet, 2011) or web-based data (Knees et al, 2006). While this approach is expected to yield coherent playlists, Lee et al (2011) actually found that recommending music with stronger audio similarity does not necessarily translate to higher user satisfaction. This limitation relates to the so-called *semantic gap* in music information retrieval, that is, the distance between the raw audio signal of a song and a listener's perception of the song (Celma et al, 2006).

Collaborative Filtering (CF) has been proven successful to reveal underlying structure from user-item interactions (Adomavicius and Tuzhilin, 2005; Ricci et al, 2015). In particular, CF has been applied to music playlist continuation by considering collections of hand-curated playlists and 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-neighbors CF systems (Bonnin and Jannach, 2014; Hariri et al, 2012; Jannach et al, 2015), but Aizenberg et al (2012) also presented 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-neighbors CF systems 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 systems can not extend playlists unseen at training time. To circumvent this issue, Aizenberg et al (2012) replaced the latent factors of unseen playlists by the latent factors of their songs, and Jannach and Ludewig (2017) showed how to efficiently implement a playlist-neighbors CF system able to extend unseen playlists in reasonable time, even for large datasets. Song-neighbors CF systems have also been investigated (Vall et al, 2017b, 2019), and Bonnin and Jannach (2014) proposed a successful variation consisting in computing similarities between artists instead of between songs, even when the ultimate recommendations were at the song level. Their system also incorporated song popularity. A common limitation of all pure CF systems is that they are only aware of the songs occurring in training playlists. Thus, songs that never occurred in training playlists, to which we refer as "out-of-set" songs, can not be recommended in an informed manner. Furthermore, songs that do occur in training playlists, but seldom, are not properly modeled by CF because they lack connections to other playlists and songs.

Other collaborative systems (i.e., systems based on the exploitation of playlist-song interactions) have been presented. Zheleva et al (2010) proposed to adapt Latent Dirichlet Allocation (LDA) (Blei et al, 2003) to modeling listening sessions. They found that a variation of LDA that specifically considers the sessions provided better recommendations than plain LDA. Chen et al (2012) presented the Latent Markov Embedding, a model that exploits radio playlists to learn an embedding of songs into a Euclidean space such that the distance between embedded songs relates to their transition probability in the training playlists. Both systems can extend playlists unseen at training time, but can only make informed recommendations for songs occurring in training playlists.

Hybrid systems combining collaborative and content information are a common approach to mitigate the difficulties of CF to represent infrequent songs. Hariri et al (2012) represented the songs in hand-curated playlists by topic models derived from social tags and then mined frequent sequential patterns at the topic level. The recommendations predicted by a playlist-neighbors CF system were re-ranked according to the next topics predicted. The approach proposed by Jannach et al (2015) pre-selected suitable next songs for a given playlist using a weighted combination of the scores yielded by a playlist-neighbors CF system and a content-based system. The candidate songs were then re-ranked to match some characteristic of the playlist being extended. In both cases the hybridization followed from the combination of independently obtained scores, by means of weighting heuristics or re-ranking. For songs occurring in few training playlists, the recommendations predicted by CF could be boosted with content information. However, the recommendations for out-of-set songs would solely rely on the content-based component of the hybrid systems.

McFee and Lanckriet (2012) proposed a hybrid system integrating collaborative and content information more closely. The system was based on a weighted song hypergraph, that is, a song graph where edges can join multiple songs, and weights define the similarity between the (possibly many) songs connected by an edge. The edges were defined by assigning the songs in hand-curated playlists to possibly overlapping song sets, which had been previously obtained through clustering of extracted multimodal song features. The weights were found in a second step as the best possible fit given a collection of training playlists and the hypergraph edges. This sys-

tem could better deal with out-of-set songs, as it would only need to assign them to appropriate edges. Not strictly applied to music playlist continuation but to music understanding and recommendation in general, van den Oord et al (2013) introduced the use of convolutional neural networks to estimate song factors from a latent-factor model, given the log-compressed mel-spectrogram of the audio signal of songs. Such networks can be essentially regarded as feature extraction tools, but further combined with latent-feature models they enable the informed recommendation of infrequent or out-of-set songs. This approach was further combined with semantic features derived from artist biographies by Oramas et al (2017).

Our own previous works on music playlist continuation focused on two main research lines. On the one hand we studied the importance of three main playlist characteristics (length, song order, and popularity of the songs included) in music playlist continuation systems (Vall et al, 2017b, 2018b, 2019). On the other hand, more related to the current paper, we analyzed the extent to which multimodal features can capture playlist-song relationships, and we designed two feature-combination hybrid recommender systems for music playlist continuation (Vall et al, 2016, 2017a, 2018a). In the current work we consolidate the second line of research. We present the two feature-combination hybrid systems in full detail. We conduct an extensive evaluation, comparing the proposed systems to four competitive playlist continuation baselines, and incorporating uncertainty estimation by means of bootstrap confidence intervals. We analyze additional audio-based features extracted applying convolutional neural networks on song spectrograms. Finally, the evaluation further provides new, insightful comparisons between well-established pure and hybrid CF systems, namely the matrix factorization model proposed by Hu et al (2008), its hybrid extension proposed by van den Oord et al (2013), and the playlist-neighbors CF system (Bonnin and Jannach, 2014; Hariri et al, 2012; Jannach et al, 2015).

3 Problem formulation

Let P be a collection of music playlists. Let S be the universe of songs available, including at least the set S_P of songs occurring in the playlists of the collection P, but possibly more (i.e., $S \supseteq S_P$). A playlist $p \in P$ is regarded as a set of songs, where the song order is ignored.[1] Playlists may have different lengths.

Since playlists are seen as song sets, any playlist p is a subset of the universe of songs S. Thus, the set difference $S \setminus p$ represents the songs 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 s in a playlist p, the set difference $p \setminus \{s\}$ removes the song s from the playlist p. The length of a playlist p is denoted by its cardinality $|p|$.

We refer to any playlist $p \in P$ as an "in-set" playlist, and to any song $s \in S_P$ as an "in-set" song. In contrast, we refer to any playlist $p \notin P$ as an "out-of-set" playlist, and to any song $s \notin S_P$ as an "out-of-set" song.

[1] Even though the process of listening to a playlist is inherently sequential, we found that considering the song order in curated music playlists is actually not crucial to extend such playlists (Vall et al, 2018b, 2019). While more research is required to fully understand the impact of the song order in music playlists, we feel confident that disregarding the song order does not harm the contribution of the current work.

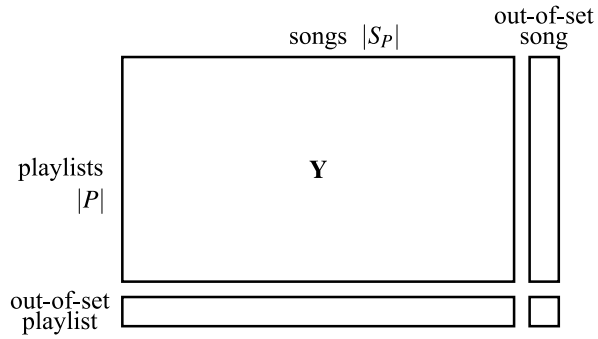


Fig. 1: Playlist continuation as a matrix completion and expansion problem. The matrix $\mathbf{Y}$ encodes the playlist collection P. CF systems discover in-set potential positive playlist-song interactions by "completing" the matrix $\mathbf{Y}$. Hybrid systems can further "expand" the matrix $\mathbf{Y}$ towards out-of-set songs by incorporating external song descriptions. Systems not specializing in the playlists of P can expand the matrix towards out-of-set playlists, possibly at the cost of slightly lower performance.

3.1 Playlist continuation as matrix completion

While a single playlist typically reflects individual preferences, a collection of playlists constitutes a source of collaborative implicit feedback (Hu et al, 2008; Pan et al, 2008) encoding rich playlist-song co-occurrence patterns. Similar to other recommendation tasks, music playlist continuation can be regarded as a matrix completion problem. The playlist collection P is arranged into a binary matrix $\mathbf{Y} \in \{0,1\}^{|P| \times |S_P|}$ of playlist-song interactions, with as many rows as playlists and as many columns as unique songs in the playlists (Fig. 1). The interaction between a playlist p and a song s indicates whether the song occurs ($y_{p,s} = 1$) or not ($y_{p,s} = 0$) in the playlist. The matrix $\mathbf{Y}$ is typically very sparse and thus it can be stored efficiently by keeping only the positive interactions (e.g., the playlist collections introduced in Section 7 have both a density rate of 0.08%).

"Completing" the matrix $\mathbf{Y}$ generally refers to discovering new potential positive playlist-song interactions. Songs identified as potential positive interactions to a given playlist are then recommended as candidates to extend the playlist. That is the approach followed by CF systems, both neighborhood-based (Bonnin and Jannach, 2014; Vall et al, 2017b, 2019) and model-based (Aizenberg et al, 2012).

We evaluate two CF baselines, one model-based and one neighbors-based. For the model-based system, we adapt the matrix factorization model for implicit feedback datasets by Hu et al (2008) for the task of music playlist continuation (Section 5.1). For the neighbors-based system, we modify the playlist-neighbors CF system (Bonnin and Jannach, 2014; Hariri et al, 2012; Jannach et al, 2015) to adapt to the challenging sparsity of the considered playlist collections (Section 5.3).

3.2 Playlist continuation as matrix expansion

The matrix completion framework is limited to playlists and songs within the matrix. However, common use cases may require extending not-yet-seen playlists, or considering candidate songs that do not occur in any of the playlists of the matrix.

3.2.1 Out-of-set songs

CF systems rely solely on playlist-song co-occurrence patterns. Therefore, they are unable to recommend out-of-set songs as candidates to extend playlists, precisely because out-of-set songs do not co-occur with the playlists in the collection. Hybrid extensions to CF overcome this limitation by incorporating external song descriptions seeking to compensate for the lack of playlist-song co-occurrences. Hybrid systems can not only enable the recommendation of out-of-set songs (Fig. 1) but also strengthen the representation of in-set but infrequent songs.

The feature-combination hybrid recommender systems proposed in this work handle out-of-set and in-set but infrequent songs by fusing any type of song feature vectors with collaborative patterns derived from hand-curated music playlists (Sections 4.1 and 4.2). We also evaluate the hybrid CF system proposed by van den Oord et al (2013), which predicts song latent factors from the audio signal and passes them to a matrix factorization model. We extend this latter approach by additionally considering song latent factors derived from independent listening logs (Section 5.2).

3.2.2 Out-of-set playlists

Model-based CF systems relying on matrix factorization are not generally able to extend playlists unseen at training time. However, we see that this limitation can be overcome for the matrix factorization model considered in this work (Hu et al, 2008), and we show how to predict continuations for out-of-set playlists (Fig. 1) provided that latent song factors are available (Section 5.1.3).

Neighbors-based CF systems can generally extend out-of-set playlists. Still, playlist-neighbors CF systems require a careful implementation to efficiently compute the similarity between out-of-set playlists and large training playlist collections (Bonnin and Jannach, 2014, Appendix A.1). This computation can be accelerated by sampling a subset of the training playlists (Jannach and Ludewig, 2017). The moderate size of the playlist collections considered in this work, however, does not make it necessary to apply such sampling (Section 5.3).

The first of the hybrid systems proposed in this work specializes towards the collection of training playlists (Section 4.1). It achieves a very competitive performance, but it is not readily able to extend out-of-set playlists. The second hybrid system is designed to generally model whether any playlist and any song fit together (Section 4.2). It achieves slightly lower performance but it can handle out-of-set playlists.

3.3 Recommending playlist continuations

A playlist continuation system has to be able to predict a score quantifying the fitness between a playlist p and a candidate song s. This score may be interpreted as a probability (e.g., in the proposed systems) or as a similarity measure (e.g., in neighbors-based CF systems). After assessing the fitness between a playlist and multiple song candidates, we select the most suitable song recommendations to extend the playlist.

4 Proposed systems

We introduce two hybrid feature-combination recommender systems. The feature-combination hybridization scheme integrates collaborative and content information treating the collaborative information as an additional feature associated to each playlist-song pair (Burke, 2002). The hybridization results in an enhanced, standalone system, informed about both types of information. This is in contrast to other hybridization schemes that simply combine the predictions of independent systems.

4.1 Profiles-based playlist continuation ("Profiles")

This system specializes towards a playlist collection by means of a song-to-playlist classifier. As a consequence of this specialization, Profiles achieves very competitive performance, but it is not readily able to extend out-of-set playlists. This system should be used to recommend songs to stable user playlists. If new playlists needed to be considered, the song-to-playlist classifier could be extended using incremental training techniques (Li and Hoiem, 2017). Profiles can deal with out-of-set songs.

4.1.1 Model definition

A song s is represented by a feature vector $\mathbf{x}_s \in \mathbb{R}^D$. We are interested in the probability of song s fitting each of the playlists of a collection P.

The system is based on a song-to-playlist classifier implemented by a neural network $\mathbf{c}\colon \mathbb{R}^D \to \mathbb{R}^{|P|}$. The network takes the song feature $\mathbf{x}_s$ as input. The output $\mathbf{c}(\mathbf{x}_s) \in \mathbb{R}^{|P|}$ is pointwise passed through logistic activation functions,[2] yielding a vector $\hat{\mathbf{y}}_s = \sigma\big(\mathbf{c}(\mathbf{x}_s)\big) \in [0,1]^{|P|}$ that indicates the predicted probability of song s fitting each of the playlists in the collection P (Fig. 2).

The song-to-playlist classifier depends on a set of learnable weights $\theta_{\mathbf{c}}$ (omitted so far for simplicity). The weights are adjusted on the basis of training examples $\{\mathbf{x}_s, \mathbf{y}_s\}$ (Section 4.1.2) by comparing the model's predicted probabilities to the actual labels $\mathbf{y}_s \in \{0,1\}^{|P|}$. Precisely, the weights $\theta_{\mathbf{c}}$ are estimated to minimize the following

[2] The song-to-playlist classifier makes as many independent decisions as playlists in the collection P. We also experimented with a softmax activation function yielding a probability distribution over playlists, but using sigmoids provided better results according to the followed evaluation methodology (Section 6).

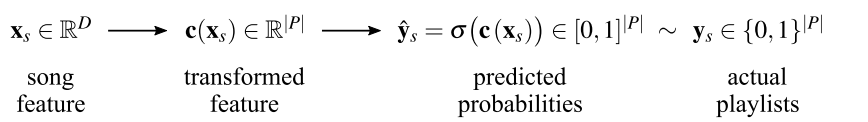


Fig. 2: Sketch of the Profiles system. A song feature vector is taken as input and processed through the network to decide the playlists of P that the song fits. The model is trained on labeled song-to-playlist examples.

binary cross-entropy cost function

$$\mathcal{L}_{\text{Profiles}}\big(\theta_{\mathbf{c}} \mid \{\mathbf{x}_s, \mathbf{y}_s\}\big) = \\ -\sum_{p,s} y_{p,s} \log\big(\hat{y}_{p,s}\big) + \big(1 - y_{p,s}\big) \log\big(1 - \hat{y}_{p,s}\big). \tag{1}$$

The terms $y_{p,s}$ and $\hat{y}_{p,s}$ denote the components of $\mathbf{y}_s$ and $\hat{\mathbf{y}}_s$ corresponding to playlist p, respectively. The dimensionality of the set of parameters $\theta_{\mathbf{c}}$ depends on the network architecture. The summation is done over all the possible playlist-song pairs, both occurring ($y_{p,s} = 1$) and non-occurring ($y_{p,s} = 0$) in the training playlists. We experimented with different weighting schemes for occurring and non-occurring pairs, as suggested by Hu et al (2008) or Pan et al (2008), but none yielded superior performance than using equal weights.

4.1.2 Song-to-playlist training examples

S_P is the set of unique songs in the playlists of P. For each song $s \in S_P$, $\mathbf{y}_s \in \{0,1\}^{|P|}$ is the column of the playlist-song interactions matrix $\mathbf{Y}$ corresponding to song s, which indicates the playlists of P to which the song s belongs. The training set consists of all the pairs of song features and playlist-indicator binary vectors $\{\mathbf{x}_s, \mathbf{y}_s\}_{s \in S_P}$.

4.2 Membership-based playlist continuation ("Membership")

This system generally models playlist-song membership relationships, that is, whether a given playlist and a given song fit together. This approach is related to the Profiles system, but here we seek to discourage the specialization towards specific playlists by generally representing any playlist by the feature vectors of the songs that it contains. In this way, Membership can deal with out-of-set playlists and out-of-set songs.

4.2.1 Model definition

A playlist p is represented by a feature matrix $\mathbf{X}_p \in \mathbb{R}^{|p| \times D}$ that contains, in each row, the feature vector of each song in the playlist. A song s is represented by a feature vector $\mathbf{x}_s \in \mathbb{R}^D$. A playlist-song pair (p,s) is then represented by the features $(\mathbf{X}_p, \mathbf{x}_s)$.

The system is based on a deep neural network with a "feature-transformation" component $\mathbf{t}\colon \mathbb{R}^D \to \mathbb{R}^H$ and a "match-discrimination" component $d\colon \mathbb{R}^{2H} \to [0,1]$.

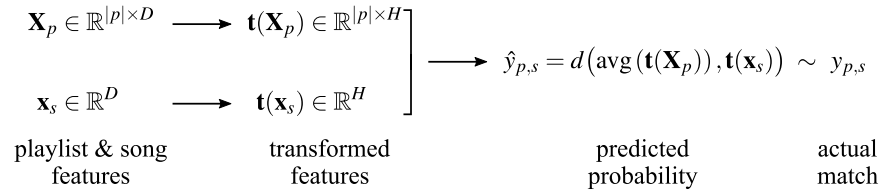


Fig. 3: Sketch of the Membership system. Given any playlist-song pair, its feature matrix and vector are transformed into hidden representations that are then used to decide if the playlist-song pair fits together. The model is trained on labeled playlist-song pairs derived from Algorithm 1.

The playlist feature matrix $\mathbf{X}_p$ is transformed song-wise (i.e., row-wise) into a hidden matrix representation $\mathbf{t}(\mathbf{X}_p) \in \mathbb{R}^{|p|\times H}$ (where we slightly abuse notation for $\mathbf{t}$). This matrix is averaged over songs yielding a summarized playlist feature vector $\text{avg}(\mathbf{t}(\mathbf{X}_p)) \in \mathbb{R}^H$. The song feature vector $\mathbf{x}_s$ is also transformed into a hidden representation $\mathbf{t}(\mathbf{x}_s) \in \mathbb{R}^H$. Both hidden representations are passed through the match-discrimination component that predicts the probability of the playlist-song pair fitting together, $\hat{y}_{p,s} = d\big(\text{avg}(\mathbf{t}(\mathbf{X}_p)), \mathbf{t}(\mathbf{x}_s)\big) \in [0,1]$ (Fig. 3).

The transformation and match-discrimination components depend on sets of learnable weights $\theta_\mathbf{t}$ and θ_d, respectively (omitted so far for simplicity). These are adjusted on the basis of training examples $\{(\mathbf{X}_p, \mathbf{x}_s), y_{p,s}\}$ (Section 4.2.2) by comparing the model's predicted probability for a pair (p,s) to the actual label $y_{p,s} \in \{0,1\}$. Precisely, the sets of weights $\theta_\mathbf{t}$, θ_d are estimated to minimize the following binary cross-entropy cost function

$$\mathcal{L}_{\text{Membership}}\Big(\theta_\mathbf{t}, \theta_d \;\Big|\; \big\{(\mathbf{X}_p, \mathbf{x}_s), y_{p,s}\big\}\Big) = \\ -\sum_{p,s} y_{p,s} \log\left(\hat{y}_{p,s}\right) + \left(1 - y_{p,s}\right) \log\left(1 - \hat{y}_{p,s}\right). \tag{2}$$

The dimensionalities of the sets of parameters $\theta_\mathbf{t}$ and θ_d depend on the network architecture.

4.2.2 Playlist-song training examples

We assume that any playlist $p \in P$ implicitly defines matches to each of its own songs. That is, each song $s \in p$ matches the shortened playlist $p_s = p \setminus \{s\}$. We further assume that any song not occurring in the playlist p is a "mismatch" to the shortened playlist p_s.[3] Thus, we can obtain a mismatch by randomly drawing a song from $S \setminus p$.

Following this procedure, Algorithm 1 details how to derive a training set with as many matching as mismatching playlist-song pairs given a playlist collection P and a universe of available songs S.

[3] In the context of implicit feedback, the term "no-match" may be preferable to "mismatch" because missing feedback does not necessarily reflect negative feedback. However, we keep the latter for simplicity.

Algorithm 1 Derive playlist-song matches and mismatches.

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

matches = [] ▷ initialize empty lists
mismatches = []
for each $p \in P$ **do**
  for each $s \in p$ **do**
    $p_s = p \setminus \{s\}$ ▷ remove s from p
    $s_+ = s$ ▷ s is a match to p_s
    $s_- = \texttt{sample}(S \setminus p)$ ▷ draw a mismatch to p_s
    matches.append$\left(\left(\mathbf{X}_{p_s}, \mathbf{x}_{s_+}\right), 1\right)$ ▷ store training examples
    mismatches.append$\left(\left(\mathbf{X}_{p_s}, \mathbf{x}_{s_-}\right), 0\right)$
  end for
end for
return matches, mismatches

4.2.3 Sampling strategy

The Membership system can be utilized as we have described so far. However, we find that applying the following sampling strategy before we derive the training playlist-song pairs and at recommendation time is necessary to obtain competitive results.

We set a fix playlist length n given by the length of the shortest playlist in a collection P. Given a playlist $p \in P$, we derive all the sub-playlists p' that result from drawing n songs from p without replacement. However, the number of possible draws can be large. To keep the approach computationally tractable, if the number of possible draws is larger than $|p|$, we select only $|p|$ sub-playlists by randomly drawing n songs from p without replacement $|p|$ times.[4]

We apply this procedure to each playlist of P, thus obtaining a modified playlist collection P' with many more, but shorter fix-length playlists. Then, we apply Algorithm 1 to the modified collection P' to derive training playlist-song pairs.

Once Membership is trained, we also apply the sampling strategy to predict the match probability of an unseen playlist-song pair (p,s). We derive no more than $|p|$ sub-playlists out of p as described above. We let Membership predict the match probability of (p',s) for each derived sub-playlist p'. Then we average the probabilities.

[4] The number of possible sub-playlists is $\binom{|p|}{n}$. For example, we could sample 2,002 sub-playlists of length 5 out of a playlist of length 14. In this case, we would randomly draw 14 sub-playlists of length 5.

5 Baseline systems

5.1 Matrix-factorization-based playlist continuation ("MF")

This is a purely collaborative system based on the weighted matrix factorization model proposed by Hu et al (2008). As any pure CF system, it is unable to recommend out-of-set songs. In principle it is also unable to extend out-of-set playlists, but we see how to overcome this limitation with a fast one-step factorization update (Section 5.1.3).

5.1.1 Model definition

We factorize the matrix of playlist-song interactions $\mathbf{Y} \in \{0,1\}^{|P|\times|S_P|}$ into two low-rank matrices $\mathbf{u} \in \mathbb{R}^{|P|\times D}$, $\mathbf{v} \in \mathbb{R}^{|S_P|\times D}$ of playlist and song latent factors, respectively, where D is the depth of the factorization and the product $\hat{\mathbf{Y}} = \mathbf{u} \cdot \mathbf{v}^T$ approximately reconstructs the original matrix $\mathbf{Y}$. Precisely, the latent factors are estimated to minimize the following weighted least squares cost function

$$\mathcal{L}_{\mathrm{MF}}\big(\mathbf{u},\mathbf{v} \;\big|\; \mathbf{Y}\big) = \sum_{p,s} w_{p,s}\left(y_{p,s} - \mathbf{u}_p \cdot \mathbf{v}_s^T\right)^2, \tag{3}$$

where $w_{p,s}$ is the weight assigned to the playlist-song pair (p,s). Following Hu et al (2008), we define the weights by $w_{p,s} = 1 + \alpha y_{p,s}$, where α is a parameter adjusted on a validation set. However, since the matrix $\mathbf{Y}$ is binary, the weighting scheme is reduced to

$$w_{p,s} = \begin{cases} w_1 & \text{if } y_{p,s} = 1 \\ 1 & \text{if } y_{p,s} = 0, \end{cases}$$

and the weight w_1 is adjusted on a validation set.

5.1.2 Minimization via Alternating Least Squares

The cost function (3) is minimized via Alternating Least Squares (ALS), an iterative optimization procedure consisting in subsequently keeping one of the factor matrices fixed while the other is updated. The initial factor matrices $\mathbf{u}^0$, $\mathbf{v}^0$ are set randomly. At iteration k, the song factors $\mathbf{v}^k$ are obtained by minimizing an approximation of the original cost function where the playlist factors have been fixed to $\mathbf{u}^k$:

$$\widetilde{\mathcal{L}}_{\mathrm{MF}}\big(\mathbf{v} \;\big|\; \mathbf{u} = \mathbf{u}^k, \mathbf{Y}\big) = \sum_{p,s} w_{p,s}\left(y_{p,s} - \mathbf{u}_p^k \cdot \mathbf{v}_s^T\right)^2. \tag{4}$$

The playlist factors $\mathbf{u}^{k+1}$ for the next iteration are obtained analogously, by minimizing the approximate cost function where the song factors have been fixed to $\mathbf{v}^k$:

$$\widetilde{\mathcal{L}}_{\mathrm{MF}}\big(\mathbf{u} \;\big|\; \mathbf{v} = \mathbf{v}^k, \mathbf{Y}\big) = \sum_{p,s} w_{p,s}\left(y_{p,s} - \mathbf{u}_p \cdot \mathbf{v}_s^{kT}\right)^2. \tag{5}$$

The approximate cost functions (4) and (5), where one of the factor matrices has been fixed, become quadratic on the other, unknown factor matrix. Thus, they have a unique minimum and it can be found exactly. At each iteration, the original cost function (3) is expected to move closer to a local minimum and the procedure is repeated until convergence.

5.1.3 Extension of out-of-set playlists

In principle, CF systems based on matrix factorization can only extend in-set playlists, for which latent playlist factors have been pre-computed at training time. However, we observe that ALS enables a fast procedure to obtain reliable playlist factors for out-of-set playlists.

Firstly, we have to insist that ALS is an iterative optimization procedure whose updates are solved exactly. Given the song factors matrix $\mathbf{v}^*$, one update solving for cost function (5) yields the playlist factors matrix $\mathbf{u}^*$ deterministically. As an example, imagine two independent optimization processes factorizing the same matrix but initialized differently. If, by chance, both processes reached the same song factors matrix $\mathbf{v}^*$ at whichever iteration, then both processes would derive $\mathbf{u}^*$ as the next playlist factors matrix, regardless of when and how they had arrived at $\mathbf{v}^*$ in the first place. A simple corollary of this observation is that, given the song factors matrix $\mathbf{v}^*$, the playlist factors matrix $\mathbf{u}^*$ derived next is always an equally good solution, regardless of how many ALS iterations had occurred before arriving at $\mathbf{v}^*$.

Assume that the playlist collections P and P' are disjoint. We are interested in predicting continuations for the playlists in the collection P', but at training time we only have access to the collection P. Even though the playlist collections are disjoint, the songs within them are likely not. We arrange the collections P and P' into respective matrices $\mathbf{Y}$ and $\mathbf{Y}'$ of playlist-song interactions. We factorize the matrix $\mathbf{Y}$ until convergence and keep only the song factors $\mathbf{v}^*$. We can now perform one ALS update solving for the following cost function, which is similar to cost function (5) but combines the song factors $\mathbf{v}^*$ (derived from $\mathbf{Y}$) with the matrix $\mathbf{Y}'$:

$$\widetilde{\mathcal{L}}_{\text{Out-of-set}}\left(\mathbf{u} \;\middle|\; \mathbf{v}=\mathbf{v}^*, \mathbf{Y}'\right) = \sum_{p,s} w_{p,s} \left(y'_{p,s} - \mathbf{u}_p \cdot \mathbf{v}_s^{*T}\right)^2. \tag{6}$$

This yields playlist factors $\mathbf{u}'$ for the playlists in P'. We can finally predict extensions for the playlists in P' by reconstructing $\hat{\mathbf{Y}}' = \mathbf{u}' \cdot \mathbf{v}^{*T}$.

Even though the playlist factors are the result of a single ALS update, they are as reliable as the song factors used to derive them. This follows from the reasoning presented above, together with the condition that matrix $\mathbf{Y}'$ is not much more sparse than $\mathbf{Y}$, as this could degrade the results.

5.2 Hybrid matrix-factorization-based playlist continuation ("Hybrid MF")

This is a hybrid extension to the just-presented weighted matrix factorization model for implicit feedback datasets (Section 5.1). It is based on the exploitation of song latent factors derived from sources other than the playlist-song matrix $\mathbf{Y}$, and it is

enabled by an appropriate application of the ALS procedure. This is a hybrid system because the song latent factors are derived from independent song descriptions, such as independent listening logs (Section 7.2.1) or the audio signal (Section 7.2.2).

Let $\mathbf{v}^e$ be a song factors matrix corresponding to the songs in $\mathbf{Y}$ but derived from external song descriptions. We perform one ALS update solving for cost function (5) but replacing $\mathbf{v}^k$ by $\mathbf{v}^e$. This yields a playlist factors matrix $\mathbf{u}$ corresponding to the playlists in $\mathbf{Y}$. We can then predict recommendations by reconstructing $\hat{\mathbf{Y}} = \mathbf{u} \cdot \mathbf{v}^{eT}$.

Using this system is not advised when the matrix $\mathbf{Y}$ contains sufficient training data. However, it can be helpful to deal with infrequent songs (poorly represented by pure CF), and it enables the recommendation of out-of-set songs. The distinction between in-set and out-of-set playlists is not meaningful for this system because the song latent factors are derived independently from any playlist collection. Then, one ALS iteration adapts to whichever playlist collection is being considered.

5.3 Playlist-neighbors-based playlist continuation ("Neighbors")

This is a CF system based on playlist-to-playlist similarities. A playlist p is represented by a binary vector $\mathbf{s}_p \in \{0,1\}^{|S|}$ indicating the songs that it includes. The similarity of a pair of playlists p,q is computed as the cosine between $\mathbf{s}_p$ and $\mathbf{s}_q$, i,.e.,

$$\text{sim}(p,q) = \cos(\mathbf{s}_p, \mathbf{s}_q) = \frac{\mathbf{s}_p \cdot \mathbf{s}_q}{\|\mathbf{s}_p\| \|\mathbf{s}_q\|}. \tag{7}$$

Given a reference playlist collection P, the score assigned to a song s as a candidate to extend a playlist p (which need not belong to P) is computed as

$$\text{score}(s,p) = \sum_{q \in P(s)} \text{sim}(p,q), \tag{8}$$

where $P(s)$ are the playlists from the collection P that contain the song s. The system considers the song s to be a suitable continuation for the playlist p if s has occurred in playlists of the collection P that are similar to p.

This system is closely related to the playlist-based k-nearest neighbors system (Bonnin and Jannach, 2014; Hariri et al, 2012; Jannach et al, 2015). The difference is that we consider the whole collection P as the neighborhood of p instead of considering only the k playlists most-similar to p. We found in preliminary experiments that, given the sparsity of the playlist collections, considering as many neighbors as possible is beneficial for the computation of the playlist-song scores.

5.4 Popularity-based playlist continuation ("Popularity")

This system computes the popularity of a song s according to its relative frequency in a reference playlist collection P, i.e.,

$$pop(s) = \frac{|P(s)|}{|P|}, \tag{9}$$

where $P(s)$ are the playlists from the collection P that contain the song s. Candidate songs to extend a playlist are ranked by their popularity.

5.5 Random playlist continuation ("Random")

This is a dummy system included as a reference. The fitness of any playlist-song pair (p, s) is randomly drawn from a uniform distribution $\mathcal{U}[0, 1]$.

6 Evaluation

We conduct off-line experiments to assess the performance of the playlist continuation systems. Following the evaluation approaches used in the literature (Aizenberg et al, 2012; Bonnin and Jannach, 2014; Hariri et al, 2012; Jannach et al, 2015), we devise a retrieval-based task to measure the ability of the systems to recover withheld playlist continuations. Even though off-line experiments can not directly assess the user satisfaction as user experiments do, they provide a controlled and reproducible approach to compare different systems.

6.1 Off-line experiment

Given a playlist p, we assume that a continuation p_c, proportionally shorter than p, is known and withheld for test. For example, if continuations were set to have a length of 25% their original playlist length, two playlists of 8 and 12 songs would have continuations of 2 and 3 songs, respectively. This follows the evaluation methodology used by Aizenberg et al (2012) but differs from the one used by Hariri et al (2012), Bonnin and Jannach (2014), and Jannach et al (2015), where the withheld continuations have always one song regardless of the length of the playlist p.

We let the system under evaluation predict the fitness of the playlist-song pair (p, s) for each song $s \in S \setminus p$. The set of recommendable songs is restricted to $S \setminus p$ not to recommend songs from the very playlist p. We rank the candidate songs in the order of preference to extend p given by the system predictions. On the basis of this ordered list of song candidates, we compute rank-based metrics reflecting the ability of the system to recover the songs from the playlist continuation p_c. We find the rank that each song in the withheld continuation p_c occupies within the ordered list of song candidates. We compute two additional metrics for the continuation p_c as a whole: the reciprocal rank, i.e., the inverse of the top-most rank achieved by a song from p_c within the ordered list of song candidates, and the recall@100, i.e., the amount of songs from p_c within the top 100 positions of the ordered list of song candidates (Fig. 4) (Manning et al, 2009, Chapter 8).

This process is repeated for all the playlists we set to extend. We finally report the median rank over all the songs in all the continuations, the mean reciprocal rank (MRR) over all the continuations, and the mean recall@100 (R@100) over all the continuations. We construct 95% basic bootstrap confidence intervals for each of the reported metrics (DiCiccio and Efron, 1996). Since these are not necessarily symmetric, to avoid clutter in the tables, we will show the nominal metric value plus/minus the largest margin. For example, a median rank of 1091 with a confidence interval of (1001, 1162) will not be reported as $1001 \pm {}^{71}_{90}$, but as 1001 ± 90.

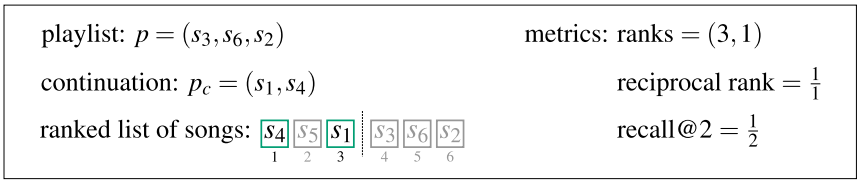


Fig. 4: Illustration of the off-line experiment for one playlist. A system extends the playlist $p = (s_3, s_6, s_2)$. It ranks all the songs available according to its predictions, leaving out the songs in the playlist p. The songs in the continuation $p_c = (s_1, s_4)$ attain, respectively, ranks 3 and 1 in the ordered list of candidate songs. The first hit is at rank 1, thus the continuation's reciprocal rank is $\frac{1}{1}$. If we recommend the top-2 results, one of the two is a hit, therefore the continuation's recall@2 is $\frac{1}{2}$.

6.2 Weak and strong generalization

We consider two evaluation settings as proposed by Aizenberg et al (2012). The first setting, or "weak generalization" setting, assumes that only one playlist collection P is available. The playlists are used to train a playlist continuation system. Then, the system recommends continuations to the very training playlists. The second setting, or "strong generalization" setting, assumes that two disjoint playlist collections P and P' are available. The playlists in the collection P are used to train a playlist continuation system. Then, the system recommends continuations to the playlists in the collection P', which it has not seen before.

7 Datasets

We compile two datasets, each consisting of a collection of hand-curated music playlists and feature vectors for each of the songs in the playlists.

The playlists are derived from Art of the Mix[5] and 8tracks,[6] two on-line platforms where music aficionados can publish their playlists. Previous research in automated music playlist continuation has focused on these databases precisely because of the presumable careful curation of their playlists (Bonnin and Jannach, 2014; Hariri et al, 2012; Jannach et al, 2015; McFee and Lanckriet, 2011, 2012).

The song feature vectors are extracted from song audio clips gathered from the content provider 7digital,[7] and from social tags and listening logs obtained from the Million Song Dataset (MSD)[8] (Bertin-Mahieux et al, 2011), a public database providing an heterogeneous collection of data for a million contemporary songs.

[5] `http://www.artofthemix.org`
[6] `https://8tracks.com`
[7] `https://www.7digital.com`
[8] `https://labrosa.ee.columbia.edu/millionsong`

7.1 Playlist collections

For Art of the Mix, we use the playlists published in the AotM-2011 dataset, a publicly available corpus of playlists crawled by McFee and Lanckriet (2012). The songs in the playlists that also belong to the MSD come properly identified. For 8tracks, we are given access to a private corpus of playlists. These playlists are represented by plain-text song titles and artist names. We match them against the MSD to get access to song-level descriptions and for comparability with the AotM-2011 dataset. The songs that are not present in the MSD are dropped from both playlist collections because we can not extract feature vectors without their song-level descriptions.

7.1.1 Playlist filtering

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 (e.g., saving a full album as a playlist). We also observe that social tags, which we use for feature extraction, can contain artist or album information. Therefore, we decide not to consider artist- and album-themed playlists to ensure the quality of the playlists and to prevent leaking artist or album information into the evaluation. We keep only playlists with at least 7 unique artists and with a maximum of 2 songs per artist (the thresholds were manually chosen to yield sufficient playlists after the whole filtering process).

This type of filtering, which we already proposed in our previous works (Vall et al, 2017a,b, 2018a,b, 2019), has also been adopted in the RecSys Challenge 2018.[9] On the other hand, other previous works have typically not filtered the playlists by such criteria (Hariri et al, 2012; McFee and Lanckriet, 2011, 2012) and have even investigated the exploitation of artist co-occurrences (Bonnin and Jannach, 2014). We believe that either approach conditions the type of patterns that playlist continuation systems will identify. Thus, filtering the playlists or not can be regarded as a design choice depending on the use case and the target users.

To ensure that the playlist continuation systems learn from playlists of sufficient length, we further keep only the playlists with at least 14 songs. The final length of the playlists may still be shortened because we drop songs missing some type of song-level description, for which we can not extract all the feature vector types. Finally, in order to set up training and evaluation playlist splits, we discard playlists that have become shorter than 5 songs after the song filtering.

The filtered AotM-2011 dataset has 2,711 playlists with 12,286 songs by 4,080 artists. The filtered 8tracks dataset has 3,269 playlists with 14,552 songs by 5,104 artists. Detailed statistics for the final playlist collections are provided in Table 1.

7.1.2 Playlist splits

We create training and test playlist splits for the weak and strong generalization settings. For the weak generalization setting, we split each playlist leaving approximately the final 20% of the songs as a withheld continuation. For the strong generalization setting, we split each playlist collection into 5 disjoint sub-collections for

[9] `https://recsys-challenge.spotify.com/details`

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

dataset	statistic	min	1q	med	3q	max
AotM-2011	Playlist length	5	7	9	11	26
	Artists per playlist	4	7	9	11	26
	Song frequency	1	1	1	2	42
8tracks	Playlist length	5	8	10	12	38
	Artists per playlist	3	8	10	11	34
	Song frequency	1	1	1	2	140

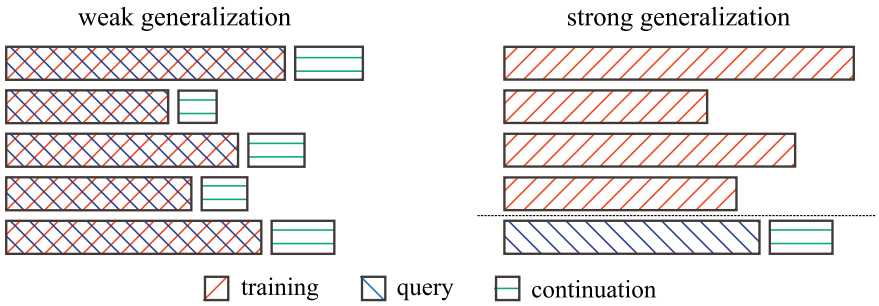


Fig. 5: Illustration of the playlist splits. In the weak generalization setting every playlist is split withholding the last songs as a continuation. In the strong generalization setting the playlist collection is split into disjoint sub-collections. One sub-collection is used to train the system. Every playlist in the other sub-collection is split withholding the last songs as a continuation. The red stripes indicate the playlists used to train the systems. The blue stripes indicate the playlists that the systems have to extend. The green stripes indicate the withheld continuations used for evaluation.

cross validation. At each iteration, 4 disjoint sub-collections are put together for training and the playlists therein are not split. The playlists in the remaining sub-collection are used for evaluation and are split as in the weak generalization setting (Fig. 5). The playlists in the training splits of both generalization settings are further split leaving approximately the final 20% of the songs as withheld continuations for validation.

7.2 Song feature extraction

For all the feature types we extract 200-dimensional vectors. According to our experiments, feature vectors of this dimensionality carry enough information.

7.2.1 Latent factors from independent listening logs ("Logs")

The Echo Nest Taste Profile Subset[10] is a dataset of user listening histories from undisclosed partners. It contains *(user, song, play-count)* triplets for songs included in the MSD. We factorize the triplets using the already discussed weighted matrix factorization model for implicit feedback datasets (Hu et al, 2008) with a factorization depth of 200 dimensions. However, the weighting scheme now depends on the play-counts. We use the obtained song latent factors as song feature vectors.

7.2.2 Latent factors from audio signal ("Audio2CF")

Following the work by van den Oord et al (2013), we build a feature extractor to predict collaborative filtering song factors from song spectrograms. We use a convolutional neural network inspired by the VGG-style architecture (Simonyan and Zisserman, 2014) consisting of sequences of 3×3 convolution stacks followed by 2×2 max pooling. To reduce the dimensionality of the network output to the predefined song factor dimensionality, we insert, as a final building block, a 1×1 convolution having 200 feature maps followed by global average pooling (Lin et al, 2013).

We assemble a training set for the feature extractor using the latent factors of the songs from the Echo Nest Taste Profile Subset (Section 7.2.1) and the corresponding audio previews downloaded from 7digital. To prevent leaking information, we discard the songs present in the playlist collections. We use the trained feature extractor to predict song latent factors for the songs in the playlist collections, given audio snippets that we also download from 7digital.

7.2.3 Semantic features from social tags ("Tags")

The Last.fm Dataset[11] gathers social tags that users of the on-line music service Last.fm[12] assigned to songs included in the MSD. Along with the tag strings, the dataset provides relevance weights describing how well a particular tag applies to a song, as returned by the `tracks.getTopTags` function of the Last.fm API.[13]

We extract semantic features from the tags assigned to a song using *word2vec* (Mikolov et al, 2013). Even though we have experimented with *word2vec* models trained on very large text corpora (e.g., on GoogleNews[14]), we obtain best results using models trained on custom, smaller but music-informed text corpora (more details can be found in our previous works (Vall et al, 2017a, 2018a)).

For each unique song in the playlists, we look up its social tags in the music-informed *word2vec* model. 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 the relevance weights provided by the Last.fm Dataset.

[10] `https://labrosa.ee.columbia.edu/millionsong/tasteprofile`
[11] `https://labrosa.ee.columbia.edu/millionsong/lastfm`
[12] `https://www.last.fm`
[13] `https://www.last.fm/api/show/track.getTopTags`
[14] `https://code.google.com/archive/p/word2vec`

8 Results

Tables 2 and 3 report the results achieved in the weak and strong generalization settings, respectively. The Profiles system can only operate in weak generalization and therefore it only appears in Table 2. The purely collaborative systems, i.e., MF and Neighbors, can not predict scores for out-of-set songs. During the evaluation of these systems, if a withheld continuation contains an out-of-set song, it is simply ignored. Thus, the overall performance of MF and Neighbors is not directly comparable to the performance of the other systems. To make this information clear, Tables 2 and 3 report the number N of songs in the withheld continuations that each system could consider, and the results corresponding to MF and Neighbors are displayed in italics. A fair comparison of the hybrid systems and MF is provided in Section 8.4, where the performance of each system is shown as a function of how often the songs in the withheld continuations occurred in training playlists.

8.1 Interpreting the results

Figure 6 displays the complete recall curve achieved by the playlist continuation systems on the AotM-2011 dataset in the weak generalization setting (all the hybrid systems use the Logs features). To highlight that MF and Neighbors can only deal with in-set songs, their recall curves are represented with dashed lines. Profiles, Membership, MF and Hybrid MF bend considerably to the upper left corner. This shows that they keep on predicting relevant songs as their recommendation lists grow. Neighbors starts similarly, but it quickly flattens, not finding additional relevant recommendations as its recommendation list grows. However, if we look closer at the top 200 results (detail box in Fig. 6), we find that Neighbors actually starts comparably well to MF, and even better than Profiles and Membership. Its quick decline is likely explained by the high sparsity of the datasets: half of the songs occur in one training playlist, and three quarters of the songs occur in no more than 2 training playlists (Table 1). Neighbors is only able to successfully predict recommendations for the most frequent songs, which co-occur often, but it is unable to do so for the vast majority of infrequent songs. Similarly, Popularity performs reasonably well for the top-most positions of the recommendation list, but its performance quickly degrades.

The median rank (corresponding approximately to the dots at 50% recall in Fig. 6) summarizes the overall distribution of ranks attained by a system. The MRR and R@100 capture the performance at the top positions of the recommendation lists. Focusing on the overall performance of the systems, Profiles, Membership and MF are clearly preferable over Neighbors. Looking at the top positions only, Neighbors might appear preferable. It could be argued that only the top positions matter, because a user could not possibly look further than the top 10 recommendations. While this reasoning seems valid in on-line systems, where users react to the predicted recommendations, we believe that it is inaccurate in the context of off-line experiments. Assessing the usefulness of music recommendations is highly subjective. Given a playlist, there are multiple songs that a user could accept as relevant continuations. However, off-line experiments only accept exact matches to the withheld ground-truth continuation

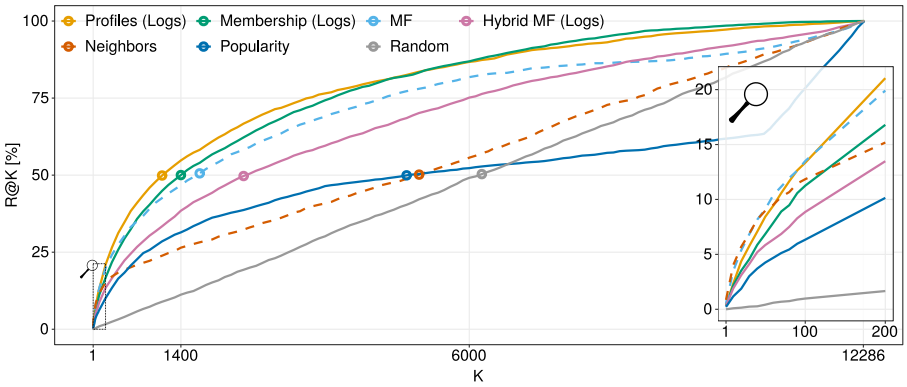


Fig. 6: Recall curve on the AotM-2011 playlist collection in weak generalization. The top 200 positions are detailed in the box. MF and Neighbors, only evaluated for in-set songs, are displayed with a dashed line.

(McFee and Lanckriet, 2011; Platt et al, 2002). For this reason, measuring the performance of a system focusing only on the top positions of recommendation lists can be misleading. Off-line experiments should be regarded as approximations of the final system performance, and the performance should be measured by the system's global merits. Throughout this section, we will mostly rely on the median rank as the metric to assess the global behavior of the playlist continuation systems.

8.2 Overall performance of the playlist continuation systems

For now we only let the proposed systems use the Logs and Audio2CF features, which the Hybrid MF system can also utilize. This makes the comparison fair.

8.2.1 Weak generalization

Profiles and Membership obtain lower (better) median rank than Hybrid MF using Logs and Audio2CF features, respectively (Table 2). Using Logs features, Profiles and Membership obtain higher (better) R@100 than Hybrid MF, but their MRR is comparable. Using Audio2CF features, the MRR and R@100 of Profiles, Membership and Hybrid MF are comparable in the AotM-2011 dataset, but Membership is clearly better than Profiles and Hybrid MF in the 8tracks dataset.

For all the hybrid systems (Profiles, Membership and Hybrid MF) the Logs features yield better results than the Audio2CF features, regardless of the metric considered. The improvements are not equally pronounced in all the combinations of systems and datasets, but the gains are clear. This is expected because Audio2CF features are an approximation of Logs features derived from the audio signal of songs. Despite its remarkable results, the Audio2CF features can not bridge the music semantic gap (Celma et al, 2006; van den Oord et al, 2013).

The performance of Profiles and Membership is comparable. Overall, Profiles can achieve a higher performance when using Logs features, but Membership is superior using Audio2CF features, especially in the 8tracks dataset.

Finally, we comment on the performance of the pure CF baselines. MF obtains a clearly lower (better) median rank than Neighbors. On the other hand, Neighbors obtains MRR and R@100 comparable to MF. The reason for this apparent mismatch between median rank, MRR and R@100 was just exposed in Section 8.1.

8.2.2 Strong generalization

Membership obtains a clearly lower (better) median rank than Hybrid MF, regardless of the feature used (Table 3). Using Logs features, Membership and Hybrid MF obtain comparable MRR and R@100. Using Audio2CF features, the MRR and R@100 of Membership and Hybrid MF are comparable in the AotM-2011 dataset, but Membership is clearly better in the 8tracks dataset.

For the hybrid systems, again the Logs features yield better results than the Audio2CF features. Indeed, the different information that these two features carry is independent of the generalization setting used to evaluate the systems.

Compared to one another, the pure CF baselines behave similarly as in weak generalization. MF obtains lower (better) median rank than Neighbors. However, Neighbors obtains R@100 comparable to MF and MRR superior than MF, especially in the AotM-2011 dataset.

8.2.3 Robustness to strong generalization

We analyze the robustness of each system (but Profiles) to the strong generalization setting by comparing whether its performance degrades from Table 2 to Table 3.

Membership performs comparably well in both generalization settings regardless of the feature utilized, the metric considered, and the dataset. This result indicates that regarding playlist-song pairs exclusively in terms of feature vectors (the key characteristic of Membership) does favor generalization and discourages the specialization towards particular training playlists.

The performance of MF is also comparable in weak and strong generalization. This supports the approach detailed in Section 5.1.3, by which latent song factors derived from the factorization of a collection of training playlists can be successfully utilized to extend out-of-set playlists.

We pointed out in Section 5.2 that Hybrid MF does not distinguish between in-set and out-of-set playlists. Now we observe that the performance of Hybrid MF is identical in both generalization settings. Only the confidence intervals are slightly different due to the randomness involved in bootstrap resampling.

The performance of Neighbors is not harmed in strong generalization. In fact, it is slightly superior for the 8tracks dataset and superior for the AotM-2011 dataset. Popularity is also not affected when recommending out-of-set playlists.

Table 2: Weak generalization results. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 12,286 and 14,552 unique songs for the AotM-2011 and the 8tracks datasets, respectively. Lower is better. For MRR and R@100 higher is better. Systems clearly outperforming the best result of Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

AotM-2011 - weak generalization

system	feature	N	med rank	MRR [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	4473	891 ± 63	2.49 ± 0.41	15.91 ± 1.23
	Logs	4473	1091 ± 90	1.91 ± 0.33	13.34 ± 1.12
	Audio2CF	4473	2298 ± 127	0.75 ± 0.20	6.68 ± 0.83
Membership	Audio2CF + Tags + Logs	4473	1052 ± 73	2.18 ± 0.37	14.88 ± 1.16
	Logs	4473	1391 ± 101	1.71 ± 0.34	11.25 ± 1.01
	Audio2CF	4473	2042 ± 113	1.08 ± 0.25	7.56 ± 0.88
MF	—	2835	1572 ± 150	2.21 ± 0.45	13.50 ± 1.36
Hybrid MF	Logs	4473	2404 ± 164	1.47 ± 0.32	8.86 ± 0.92
	Audio2CF	4473	2921 ± 164	0.74 ± 0.21	5.60 ± 0.77
Neighbors	—	2835	5112 ± 321	2.66 ± 0.55	11.84 ± 1.28
Popularity	—	4473	5217 ± 570	1.03 ± 0.27	6.34 ± 0.80
Random	—	4473	6163 ± 218	0.10 ± 0.03	0.98 ± 0.34

8tracks - weak generalization

system	feature	N	med rank	MRR [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	6289	556 ± 38	3.90 ± 0.43	23.51 ± 1.21
	Logs	6289	718 ± 58	$3.62 \pm 0.44^{\star}$	20.03 ± 1.12
	Audio2CF	6289	1988 ± 90	1.13 ± 0.24	8.21 ± 0.80
Membership	Audio2CF + Tags + Logs	6289	652 ± 46	3.73 ± 0.44	20.43 ± 1.12
	Logs	6289	986 ± 68	2.89 ± 0.37	16.92 ± 1.04
	Audio2CF	6289	1149 ± 69	2.49 ± 0.35	15.50 ± 1.02
MF	—	4299	1198 ± 145	4.04 ± 0.56	18.62 ± 1.29
Hybrid MF	Logs	6289	1677 ± 138	2.79 ± 0.41	15.88 ± 1.05
	Audio2CF	6289	2636 ± 142	1.09 ± 0.24	7.03 ± 0.75
Neighbors	—	4299	3566 ± 318	4.93 ± 0.64	19.50 ± 1.32
Popularity	—	6289	3352 ± 272	1.80 ± 0.33	9.71 ± 0.81
Random	—	6289	7094 ± 234	0.15 ± 0.06	0.78 ± 0.24

* The 95% bootstrap CIs for the MRR of Profiles (Logs) and Hybrid MF (Logs) do not overlap. However, this is not apparent from the table because, for readability, we only display the largest margin of the CIs.

Table 3: Strong generalization results. The first two columns indicate the system and feature names (if any). The third column indicates the number *N* of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 12,286 and 14,552 unique songs for the AotM-2011 and the 8tracks datasets, respectively. Lower is better. For MRR and R@100 higher is better. Systems clearly outperforming the best result of Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

AotM-2011 - strong generalization

system	feature	N	med rank	MRR [%]	R@100 [%]
Membership	Audio2CF + Tags + Logs	4473	1150 ± 72	1.88 ± 0.33	13.69 ± 1.13
	Logs	4473	1552 ± 96	1.42 ± 0.29	10.71 ± 1.04
	Audio2CF	4473	2065 ± 120	0.93 ± 0.21	7.05 ± 0.85
MF	—	2849	1428 ± 135	2.72 ± 0.53	15.43 ± 1.44
Hybrid MF	Logs	4473	2404 ± 164	1.47 ± 0.31	8.86 ± 0.94
	Audio2CF	4473	2921 ± 164	0.74 ± 0.21	5.60 ± 0.78
Neighbors	—	2849	4502 ± 304	4.89 ± 0.82	15.16 ± 1.43
Popularity	—	4473	5143 ± 583	1.03 ± 0.27	7.28 ± 0.85
Random	—	4473	6161 ± 188	0.13 ± 0.05	1.23 ± 0.37

8tracks - strong generalization

system	feature	N	med rank	MRR [%]	R@100 [%]
Membership	Audio2CF + Tags + Logs	6289	710 ± 51	3.46 ± 0.43	20.40 ± 1.12
	Logs	6289	1038 ± 78	2.85 ± 0.39	15.90 ± 1.03
	Audio2CF	6289	1317 ± 86	2.58 ± 0.37	14.54 ± 0.99
MF	—	4298	997.5 ± 109.5	4.48 ± 0.61	20.32 ± 1.35
Hybrid MF	Logs	6289	1677 ± 138	2.79 ± 0.41	15.88 ± 1.04
	Audio2CF	6289	2636 ± 142	1.09 ± 0.24	7.03 ± 0.75
Neighbors	—	4298	3045.5 ± 390.5	5.27 ± 0.68	20.43 ± 1.41
Popularity	—	6289	3293 ± 278	1.76 ± 0.32	9.32 ± 0.78
Random	—	6289	6876 ± 158	0.14 ± 0.06	0.73 ± 0.25

8.3 Combined features

Until now we only considered the proposed systems with Logs and Audio2CF features to make the comparison to Hybrid MF fair. However, Profiles and Membership can flexibly exploit any type of song feature vector. In particular, they can utilize feature vectors resulting from the concatenation of other feature vectors. This simple approach already yields performance gains.

To illustrate this effect, we now consider the Tags features as well. For each song, we create a combined feature by concatenating its Audio2CF, Tags and Logs feature vectors. Since each individual feature vector has 200 dimensions, the resulting feature vector is 600-dimensional. Profiles and Membership achieve clearly better results

with the combined feature than with the Logs feature in terms of median rank and R@100, and modest but visible improvements in terms of MRR (Tables 2 and 3).

We have just exposed an example of a combined feature vector to illustrate the capability of the proposed systems. Appendix B introduces additional feature types, and Appendix C provides an exhaustive evaluation of the results achieved using all the feature types and their combinations.

8.4 Infrequent and out-of-set songs

We analyze the performance of the hybrid systems Profiles, Membership and Hybrid MF, as well as the performance of the purely collaborative system MF, as a function of how often the songs in the withheld continuations occurred in training playlists. We restrict the analysis to the weak generalization setting, but the results are comparable in the strong generalization setting. Figure 7 reports the results. Profiles and Membership can use Audio2CF, Logs, or the combined feature described in Section 8.3. Hybrid MF can only use Audio2CF and Logs features. For the sake of space, MF is represented together with Hybrid MF in the figure. The legend, which indicates the color associated to each feature, points to MF with a dummy feature called "None."

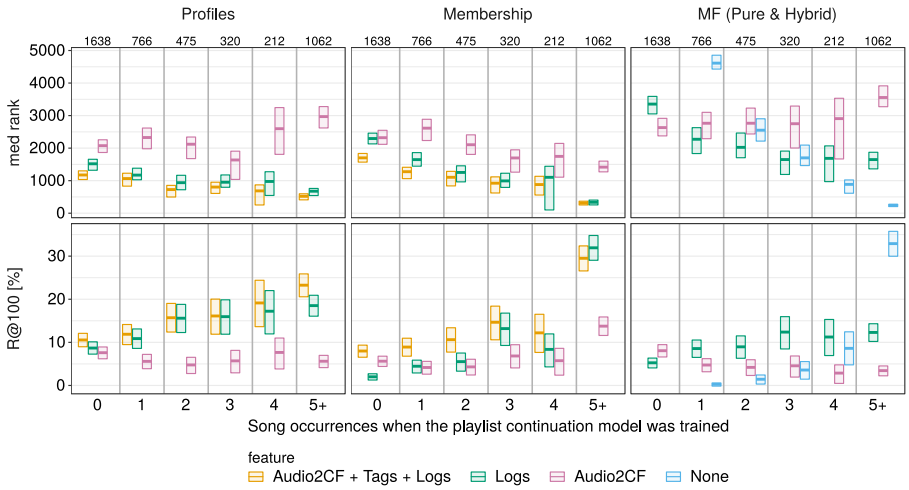
MF can not recommend out-of-set songs, and it achieves very low performance for songs that occurred in only one training playlist. This is expected because purely collaborative systems can not derive patterns in absence of sufficient playlist-song co-occurrences. MF steadily improves its performance as songs become more frequent in training playlists, until it achieves a very competitive performance for songs occurring in 5+ training playlists.

Hybrid MF with Logs features outperforms MF for very infrequent songs. Its performance improves as songs become more frequent, but it does not achieve the high performance of MF for frequent songs. Hybrid MF with Audio2CF features only outperforms MF for very infrequent songs, but MF quickly becomes better.

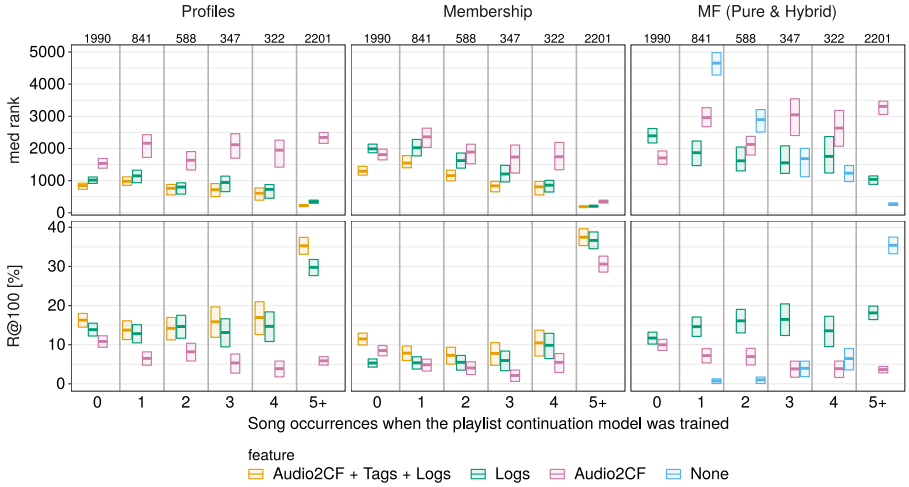
Profiles and Membership compete with Hybrid MF in terms of R@100, both using Logs and Audio2CF features. In terms of the median rank, Profiles and Membership compete with Hybrid MF using Audio2CF features, and they are generally superior using Logs features. Profiles and Membership further improve their performance using the combined feature, with which they perform reasonably well even for out-of-set and very infrequent songs. Using the combined features or Logs features, and despite some fluctuations, the proposed systems improve their performance as songs become more frequent, with results competitive with those of MF for songs occurring in 5+ training playlists, especially in the 8tracks dataset.

8.5 Additional remarks on the sparsity of playlist collections

The hybrid systems discussed throughout the paper mitigate the sparsity of the playlist collections by introducing inherent song relationships derived from content information. In this way, they provide performance improvements over systems based exclusively on the playlist collections. An alternative, compatible approach to reduce



(a) AotM-2011: weak generalization



(b) 8tracks: weak generalization

Fig. 7: Weak generalization results as a function of how often the songs in the withheld continuations occurred in training playlists. Left, center and right panels correspond to different systems. Upper and lower panels report the median rank (lower is better) and the R@100 (higher is better). The central values in the boxes correspond to the nominal metric value, and the ends correspond to the 95% CI. Each color corresponds to a feature type (MF is labeled as "None"). The text annotations on top indicate the number of songs in the withheld continuations falling in each box.

Table 4: Selection of weak generalization results in the AotM-2011 dataset for in-set songs only. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. The median rank is relative to 12,286 unique songs. Lower is better. For MRR and R@100 higher is better.

AotM-2011 - weak generalization on in-set songs

system	feature	N	med rank	MRR [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	2835	735 ± 76	2.64 ± 0.50	17.76 ± 1.53
Membership	Audio2CF + Tags + Logs	2835	756 ± 73	2.39 ± 0.45	18.08 ± 1.54
CAGH	—	2835	821 ± 89	1.78 ± 0.36	15.18 ± 1.41
Artists	—	2835	1463 ± 140	0.76 ± 0.20	8.33 ± 1.13

the sparsity of playlist collections consists in representing songs by their artists. Bonnin and Jannach (2014) proposed the Collocated Artists Greatest Hits (CAGH) system, a song-neighbors CF system where the pairwise song similarities are replaced by the pairwise similarities of their artists, and the obtained playlist-song scores are further scaled by the frequency of the songs in the training playlists. We have experimented with CAGH, as well as with a variation of CAGH where the playlist-song scores are not scaled by the song frequency, which we name "Artists." For comparison, we also evaluate Profiles and Membership for in-set songs only (Table 4). By design, CAGH is likely suffering from a bias towards popular songs that should be investigated in more detail. That is, the apparent outstanding performance of CAGH, even if only for in-set songs, may be the result of averaging accurate predictions for few but frequent songs, with poor predictions for many but infrequent songs (Vall et al, 2017b, 2019). In any case, Artists should not be affected by such bias and also provides competitive results that seem to validate the assumption that songs can be successfully approximated by their artists. This points to an interesting line for future work, namely the combination of hybridization and artist-level representations to reduce the sparsity in playlist collections.

9 Conclusion

We have introduced Profiles and Membership, two feature-combination hybrid recommender systems for automated music playlist continuation. The proposed systems extend collaborative filtering by not only considering hand-curated playlists but also incorporating any type of song feature vector. We have designed feature-combination hybrids, that is, systems that consolidate collaborative and content information into enhanced, standalone systems. Even though we have focused on music playlist continuation, the proposed systems are domain-agnostic and can be applied to other item domains.

We have conducted an exhaustive off-line evaluation to assess the ability of the proposed systems to retrieve withheld playlist continuations, and to compare them to the state-of-the-art pure and hybrid collaborative systems MF and Hybrid MF. The

results of the off-line experiments indicate that Profiles and Membership compete with MF when sufficient training data is available and outperform MF for infrequent and out-of-set songs. Profiles and Membership compete to, or outperform Hybrid MF when using Logs or Audio2CF features. The flexibility of the proposed systems to exploit any type of song feature vector provides a straightforward means of further improving their performance by simply combining different song feature vectors.

We have also evaluated MF, Hybrid MF and Neighbors thoroughly. Hybrid MF outperforms MF only for very infrequent songs. Thus, if we were restricted to use MF and Hybrid MF, it would seem advisable to use Hybrid MF for infrequent songs (occurring in up to 3 training playlists in our experiments) and switch to MF for more frequent songs. It should be noted that the proposed systems, Profiles and Membership, take care of this switch automatically. We have also investigated why Neighbors obtains competitive MRR and R@100 metrics but very poor median rank, and we have discussed the interpretation of these metrics.

We have observed the low predictive performance of Audio2CF features, derived exclusively from audio. While this can be explained by the music semantic gap, using Audio2CF features (or other purely audio-based features) as standalone features should be restricted to the recommendation of new releases, where the audio signal is the only song information available.

Preliminary experiments point to building artist-level representations into hybrid recommender systems as a promising line of future work.

Acknowledgements We thank Christine Bauer, Rainer Kelz, and Rocío del Río Lorenzo for their careful reading and valuable feedback during the preparation of this paper. This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 670035 (Con Espressione).

References

Adomavicius G, Tuzhilin A (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):734–749

Aizenberg N, Koren Y, Somekh O (2012) Build your own music recommender by modeling internet radio streams. In: Proc. WWW, pp 1–10

Bertin-Mahieux T, Ellis DP, Whitman B, Lamere P (2011) The million song dataset. In: Proc. ISMIR, University of Miami, pp 591–596

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. Journal of Machine Learning Research 3:993–1022

Bonnin G, Jannach D (2014) Automated generation of music playlists: Survey and experiments. ACM Computing Surveys 47(2):1–35

Bottou L, Curtis FE, Nocedal J (2018) Optimization methods for large-scale machine learning. arXiv preprint arXiv:160604838

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

Celma O (2010) Music recommendation and discovery. Springer

Celma O, Herrera P, Serra X (2006) Bridging the music semantic gap. In: Proc. ESWC Workshop on Mastering the Gap: From Information Extraction to Semantic Representation, Budva, Montenegro

Chen CM, Tsai MF, Lin YC, Yang YH (2016) Query-based music recommendations via preference embedding. In: Proc. RecSys, pp 79–82

Chen S, Moore JL, Turnbull D, Joachims T (2012) Playlist prediction via metric embedding. In: Proc. SIGKDD, pp 714–722

Choi K, Fazekas G, Sandler M (2016) Automatic tagging using deep convolutional neural networks. In: Proc. ISMIR

Cunningham SJ, Bainbridge D, Falconer A (2006) "More of an art than a science": Supporting the creation of playlists and mixes. In: Proc. ISMIR

Dehak N, Kenny PJ, Dehak R, Dumouchel P, Ouellet P (2011) Front-end factor analysis for speaker verification. IEEE Transactions on Audio, Speech, and Language Processing 19(4):788–798

DiCiccio TJ, Efron B (1996) Bootstrap confidence intervals. Statistical science pp 189–212

Dieleman S, Schlüter J, Raffel C, Olson E, Sønderby SK, Nouri D, Maturana D, Thoma M, Battenberg E, Kelly J, Fauw JD, Heilman M, Almeida DMd, McFee B, Weideman H, Takács G, Rivaz Pd, Crall J, Sanders G, Rasul K, Liu C, French G, Degrave J (2015) Lasagne: First release. URL `http://dx.doi.org/10.5281/zenodo.27878`

Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research 12:2121–2159

Eghbal-zadeh H, Lehner B, Schedl M, Widmer G (2015a) I-vectors for timbre-based music similarity and music artist classification. In: Proc. ISMIR

Eghbal-zadeh H, Schedl M, Widmer G (2015b) Timbral modeling for music artist recognition using i-vectors. In: Proc. EUSIPCO, pp 1286–1290

Flexer A, Schnitzer D, Gasser M, Widmer G (2008) Playlist generation using start and end songs. In: Proc. ISMIR, pp 173–178

Frederickson B (2018) Fast python collaborative filtering for implicit datasets. URL `https://github.com/benfred/implicit`

Hariri N, Mobasher B, Burke R (2012) Context-aware music recommendation based on latent topic sequential patterns. In: Proc. RecSys, pp 131–131

Hastie T, Tibshirani R, Friedman J (2008) The elements of statistical learning. Springer series in statistics

Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Proc. ICDM, pp 263–272

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167

Jannach D, Ludewig M (2017) When recurrent neural networks meet the neighborhood for session-based recommendation. In: Proc. RecSys, pp 306–310

Jannach D, Lerche L, Kamehkhosh I (2015) Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks. In: Proc. RecSys

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Proc. ICLR, arXiv: 1412.6980

Knees P, Pohle T, Schedl M, Widmer G (2006) Combining audio-based similarity with web-based data to accelerate automatic music playlist generation. In: Proc. International Workshop on Multimedia IR, pp 147–154

Krause AE, North AC (2014) Contextualized music listening: Playlists and the mehrabian and russell model. Psychology of Well-Being 4(1):22

Lee JH, Bare B, Meek G (2011) How similar is too similar?: Exploring users' perceptions of similarity in playlist evaluation. In: Proc. ISMIR, pp 109–114

Li Z, Hoiem D (2017) Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence

Liang D, Zhan M, Ellis DP (2015) Content-aware collaborative music recommendation using pre-trained neural networks. In: Proc. ISMIR, pp 295–301

Lin M, Chen Q, Yan S (2013) Network in network. arXiv preprint arXiv:13124400

Logan B (2002) Content-based playlist generation: Exploratory experiments. In: Proc. ISMIR

Manning CD, Raghavan P, Schütze H (2009) An introduction to information retrieval. Cambridge University Press

McFee B, Lanckriet GR (2011) The natural language of playlists. In: Proc. ISMIR, pp 537–542

McFee B, Lanckriet GR (2012) Hypergraph models of playlist dialects. In: Proc. ISMIR, pp 343–348

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119

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

van den Oord A, Dieleman S, Schrauwen B (2013) Deep content-based music recommendation. In: Advances in Neural Information Processing Systems, pp 2643–2651

Oramas S, Nieto O, Sordo M, Serra X (2017) A deep multimodal approach for cold-start music recommendation. In: Proc. DLRS@RecSys, pp 32–37, DOI 10.1145/3125486.3125492, arXiv: 1706.09739

Pan R, Zhou Y, Cao B, Liu NN, Lukose R, Scholz M, Yang Q (2008) One-class collaborative filtering. In: Proc. ICDM, pp 502–511

Platt JC, Burges CJ, Swenson S, Weare C, Zheng A (2002) Learning a Gaussian process prior for automatically generating music playlists. In: Advances in Neural Information Processing Systems, pp 1425–1432

Pohle T, Pampalk E, Widmer G (2005) Generating similarity-based playlists using traveling salesman algorithms. In: Proc. DAFx, pp 220–225

Rabiner LR, Juang BH (1993) Fundamentals of speech recognition. PTR Prentice Hall

Ricci F, Rokach L, Shapira B (2015) Recommender Systems Handbook, 2nd edn. Springer US

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. Proc ICLR

Sonnleitner R, Widmer G (2016) Robust quad-based audio fingerprinting. IEEE/ACM Transactions on Audio, Speech, and Language Processing 24(3):409–

421
Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1):1929–1958
Team TD (2016) Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688
Vall A, Eghbal-zadeh H, Dorfer M, Schedl M (2016) Timbral and semantic features for music playlists. In: Machine Learning for Music Discovery Workshop at ICML, New York, NY, USA
Vall A, Eghbal-zadeh H, Dorfer M, Schedl M, Widmer G (2017a) 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, pp 46–54
Vall A, Schedl M, Widmer G, Quadrana M, Cremonesi P (2017b) The importance of song context in music playlists. In: RecSys 2017 Poster Proceedings
Vall A, Dorfer M, Schedl M, Widmer G (2018a) A hybrid approach to music playlist continuation based on playlist-song membership. In: Proc. SAC, Pau, France
Vall A, Quadrana M, Schedl M, Widmer G (2018b) The importance of song context and song order in automated music playlist generation. In: Proc. ICMPC-ESCOM, Graz, Austria
Vall A, Quadrana M, Schedl M, Widmer G (2019) Order, context and popularity bias in next-song recommendations. International Journal of Multimedia Information Retrieval
Zheleva E, Guiver J, Mendes Rodrigues E, Milić-Frayling N (2010) Statistical models of music-listening sessions in social media. In: Proc. WWW, pp 1019–1028

A Additional system details

A.1 Profiles

A.1.1 System configuration

We conducted an initial, non-exhaustive exploration of architectures on a validation set 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.

Given the results of the initial exploration, we systematically explored all the combinations of networks with $\{2,3\}$ hidden layers and with $\{50,100,200\}$ hidden units. We decided to fix the number of layers to 3 and the number of units to 100. We further fixed the learning rate to 0.5, the batch size to 50 songs, and the hyperbolic tangent as the activation function for hidden layers. The output layer of the network is passed through logistic functions by design (Section 4.1.1).

We used batch normalization (Ioffe and Szegedy, 2015). We also experimented with different dropout probabilities (Srivastava et al, 2014) 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 feature vectors were standardized and $L2$-normalized.

The networks were optimized to minimize the cost function (1) using AdaGrad (Duchi et al, 2011) with Nesterov momentum (Nesterov, 1983). We trained for a maximum of 1,000 epochs but stopped before if the value of the cost function on the validation set did not decrease for 50 epochs. The cost function drove

Table 5: Membership architecture. The input to the system are the features $(\mathbf{X}_p, \mathbf{x}_s)$ of a playlist-song pair (p, s). $\mathbf{X}_p^t$ denotes the t-th row of the feature matrix $\mathbf{X}_p$, i.e., the t-th song of the playlist p. The upper part of the table corresponds to the feature transformation component $\mathbf{t}$, and the lower part of the table corresponds to the match-discrimination component d. The boldface layers $\mathrm{DE_k}$, $\mathrm{BN_k}$ in the transformation component $\mathbf{t}$ share their weights for all the songs in the playlist p and for song s (Section 4.2.1). The dimensionality of each layer is annotated in parentheses. DE: Dense layer, RE: Rectifier activation function, BN: Batch Normalization (Ioffe and Szegedy, 2015), DR: Dropout (Srivastava et al, 2014).

	$\mathbf{X}_p^1$ (D)	...	$\mathbf{X}_p^{\lvert p\rvert}$ (D)	$\mathbf{x}_s$ (D)
$\mathbf{t}$:	**$\mathrm{DE_1}$** + RE (512)	...	**$\mathrm{DE_1}$** + RE (512)	**$\mathrm{DE_1}$** + RE (512)
	$\mathrm{BN_1}$ + DR (512)	...	**$\mathrm{BN_1}$** + DR (512)	**$\mathrm{BN_1}$** + DR (512)
	$\mathrm{DE_2}$ + RE (512)	...	**$\mathrm{DE_2}$** + RE (512)	**$\mathrm{DE_2}$** + RE (512)
	$\mathrm{BN_2}$ + DR (512)	...	**$\mathrm{BN_2}$** + DR (512)	**$\mathrm{BN_2}$** + DR (512)
	Average (512)			
	Concatenate (1024)			
d:	$\mathrm{DE_3}$ + RE (512)			
	$\mathrm{BN_3}$ + DR (512)			
	$\mathrm{DE_4}$ + RE (512)			
	$\mathrm{BN_4}$ + DR (512)			
	$\mathrm{DE_5}$ + RE (1)			
	Cost			

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. We implemented the networks using Lasagne (Dieleman et al, 2015), which is built on top of Theano (Team, 2016).

A.1.2 Computational requirements

The instance of this system trained on the AotM-2011 dataset using Logs features has 323,144 learnable weights (almost 4 and 10 times fewer than Membership and MF, respectively). The system is trained with any variant of stochastic gradient descent (Bottou et al, 2018) using mini-batches. That is, even if a much larger playlist collection were considered, the system could still be trained efficiently in the same manner. Training the aforementioned system until it achieved the reported results required 666.07 seconds (roughly 11 minutes) on a desktop computer with an Intel Core i5-4570 CPU. The system required 4.46 seconds to make the predictions necessary for the weak generalization evaluation. The implementation of this system was not optimized to obtain fast training and prediction times.

A.2 Membership

A.2.1 System configuration

We conducted an heuristic exploration of architectures and hyperparameters. We chose the architecture detailed in Table 5, which we found to provide good performance. We selected the hyperparameters that yielded lowest cost on the validation set. We used an initial learning rate of 0.001 and a batch size of 100 playlist-song pairs (50 matches and 50 mismatches). The features were not standardized nor normalized.

The networks were optimized to minimize the cost function (2) using Adam (Kingma and Ba, 2015). We trained for a maximum of 1,000 epochs but stopped before if the value of the cost function on the validation set did not decrease for 25 epochs. We implemented the networks using Lasagne (Dieleman et al, 2015) and Theano (Team, 2016).

A.2.2 Computational requirements

Membership is computationally more demanding than Profiles, meaning that we need to use a larger neural network to obtain competitive results. This is reasonable after all, because by not specializing towards the training playlists, Membership accomplishes a much more general task. The instance of this system trained on the AotM-2011 dataset using Logs features has 1,159,681 learnable weights (almost 3 times fewer than MF). The system is trained with any variant of stochastic gradient descent (Bottou et al, 2018) using mini-batches. That is, even if a much larger playlist collection were considered, the system could still be trained efficiently in the same manner. Training the aforementioned system until it achieved the reported results required 3649.90 seconds (roughly 1 hour) on a GeForce GTX 1080 Ti GPU. On the same GPU, the system required 1193.88 seconds (roughly 20 minutes) to make the predictions necessary for the weak generalization evaluation. The implementation of this system was not optimized to obtain fast training and prediction times.

A.3 MF and Hybrid MF

A.3.1 System configuration

We used the validation set to experiment with different values for the weight w_1 (Section 5.1.1). We found that using $w_1 = 2$ yielded best results. MF and Hybrid MF use $L2$ regularization to prevent overfitting (Hu et al, 2008). We did not describe the regularization in Section 5.1.1 for simplicity, but the discussion and properties of ALS remain valid. We decided to use a weight of 10 for the $L2$ regularization term based on experiments on the validation set.

A.3.2 Computational requirements

The instance of MF trained on the AotM-2011 dataset has 2,999,400 learnable weights. The system is trained efficiently with ALS. Training MF until it achieved the reported results required 190.08 seconds (roughly 3 minutes) on a desktop computer with an Intel Core i5-4570 CPU. MF required 0.22 seconds to make the predictions necessary for the weak generalization evaluation. Hybrid MF required 2.48 seconds to complete the weak generalization evaluation, including the ALS update and the predictions. We used the publicly available implementation of MF provided by Frederickson (2018), which is highly optimized to obtain fast training times. The predictions, which simply consist in the multiplication of low rank matrices, are very fast.

A.4 Neighbors

This system does not require adjusting any hyperparameters or learning any weights. The weak evaluation of Neighbors on the AotM-2011 dataset using a desktop computer with an Intel Core i5-4570 CPU required 0.81 seconds to compute the pairwise playlist similarities given by equation (7) and 1.38 seconds to compute the playlist-song scores given by equation (8). Given the moderate size of the considered AotM-2011 and 8tracks playlist collections, both operations could be implemented as fast matrix multiplications.

B Additional song features

B.1 Semantic features from audio signal (“Audio2Tag”)

The on-line music service Jamendo[15] hosts songs under Creative Commons[16] licenses and allows free downloads. We obtain a dump of the Jamendo database, that includes, among other types of information,

[15] `https://www.jamendo.com`

[16] `https://creativecommons.org`

the social tags assigned to songs hosted in Jamendo. Then, we use the code released by Sonnleitner and Widmer (2016) to collect the audio files corresponding to the database dump. This results in a collection of roughly 200k songs independent of the playlists collections and the MSD, for which both a full-length audio file and a list of social tags is known.

We build an auto-tagger system based on a convolutional neural network. The network takes a song spectrogram as input, processes it, and predicts the probability of this song being labeled with each of a series of known tags. We train this network using the audio files and the social tags from the Jamendo database. We use only the tags that have been assigned to at least 1,000 songs, resulting in 156 unique tags and 187,663 songs.

The tagging network is similar to the Audio2CF feature extractor (Section 7.2.2). It is inspired by the VGG-style architecture (Simonyan and Zisserman, 2014) and consists of sequences of 3×3 convolution stacks followed by 2×2 max pooling. To reduce the dimensionality of the network output to the predefined number of tags, we insert, as a final building block, a 1×1 convolution having 156 feature maps followed by global average pooling (Lin et al, 2013). Each output neuron, which corresponds to one of the tags, is followed by a logistic activation function to obtain a valid tag probability. However, we use the 156-dimensional output layer before it is passed through the activation function as the Audio2Tag feature.

We use the trained auto-tagger to predict Audio2Tag features for the songs in the playlist collections, given audio snippets downloaded from 7digital. This approach is related to the systems presented by Liang et al (2015) and Choi et al (2016).

B.2 I-vectors from timbral features

I-vectors were first introduced in the field of speaker verification (Dehak et al, 2011), but recently they have also been successfully utilized for music similarity and music artist recognition tasks (Eghbal-zadeh et al, 2015a,b).

The MSD splits songs into segments of variable length (typically under a second) and provides 12-dimensional timbral coefficients for each segment similar to MFCCs (Rabiner and Juang, 1993). We build a Gaussian mixture model with 1,024 components using the segment-level timbral features of a collection of representative songs (more details can be found in our previous works (Vall et al, 2017a, 2018a)). 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 obtained i-vectors using a linear discriminant analysis model (Hastie et al, 2008) fit on the training playlists.

C Additional results

We provide additional results to demonstrate the performance of Profiles and Membership with different types of song features, and with stepwise combinations of two or three types of song features. Tables 6 and 7 report weak generalization results for the AotM-2011 and the 8tracks datasets, respectively. Tables 8 and 9 report strong generalization results for the AotM-2011 and the 8tracks datasets, respectively.

We also analyze the performance of Profiles, Membership and Hybrid MF simulating the recommendation of new song releases. To this end, we pretend that the audio signal is the only type of song description available, and we only consider songs that occurred in 4 or less training playlists. Tables 10 and 11 report the results for the AotM-2011 and the 8tracks datasets, respectively.

Besides the median rank, the MRR and the R@100, Tables 6–11 also report the Mean Average Precision (MAP), the R@10 and the R@30.

Table 6: Weak generalization results for the AotM-2011 dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 12,286 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

AotM-2011 - weak generalization

system	**feature**	**N**	**med rank**	**MRR [%]**	**MAP [%]**	**R@10 [%]**	**R@30 [%]**	**R@100 [%]**
Profiles	Audio2CF + Tags + Logs	4473	891 ± 63	2.49 ± 0.41	1.75 ± 0.28	3.47 ± 0.61	7.21 ± 0.85	15.91 ± 1.23
	Tags + Logs	4473	947 ± 74	2.01 ± 0.33	1.42 ± 0.22	2.82 ± 0.55	6.59 ± 0.84	14.81 ± 1.17
	Audio2CF + Logs	4473	1010 ± 86	1.98 ± 0.33	1.44 ± 0.27	2.57 ± 0.54	6.17 ± 0.80	14.29 ± 1.16
	Logs	4473	1091 ± 90	1.91 ± 0.33	1.31 ± 0.24	2.33 ± 0.49	5.75 ± 0.74	13.34 ± 1.12
	Audio2CF + Tags	4473	1340 ± 95	1.65 ± 0.31	1.22 ± 0.24	2.32 ± 0.52	5.23 ± 0.77	12.15 ± 1.08
	Tags	4473	1453 ± 114	1.45 ± 0.26	1.05 ± 0.20	1.84 ± 0.44	5.80 ± 0.80	11.78 ± 1.05
	Audio2CF	4473	2298 ± 127	0.75 ± 0.20	0.54 ± 0.14	0.80 ± 0.31	2.38 ± 0.52	6.68 ± 0.83
	i-vectors	4473	2820 ± 167	0.59 ± 0.17	0.41 ± 0.13	0.60 ± 0.25	1.45 ± 0.40	4.45 ± 0.71
	Audio2Tag	4473	3315 ± 172	0.31 ± 0.08	0.23 ± 0.06	0.25 ± 0.18	0.98 ± 0.33	3.14 ± 0.57
Membership	Audio2CF + Tags + Logs	4473	1052 ± 73	2.18 ± 0.37	1.52 ± 0.26	2.95 ± 0.58	6.45 ± 0.81	14.88 ± 1.16
	Audio2CF + Logs	4473	1144 ± 78	1.97 ± 0.34	1.35 ± 0.24	2.34 ± 0.50	5.43 ± 0.73	12.88 ± 1.09
	Tags + Logs	4473	1145 ± 94	2.20 ± 0.37	1.54 ± 0.28	2.68 ± 0.52	6.09 ± 0.78	14.11 ± 1.15
	Tags	4473	1201 ± 104	1.62 ± 0.29	1.16 ± 0.20	2.08 ± 0.48	4.92 ± 0.71	12.55 ± 1.08
	Audio2CF + Tags	4473	1243 ± 94	1.53 ± 0.26	1.12 ± 0.20	2.08 ± 0.48	5.70 ± 0.78	12.76 ± 1.09
	Logs	4473	1391 ± 101	1.71 ± 0.34	1.15 ± 0.22	2.16 ± 0.47	4.56 ± 0.69	11.25 ± 1.01
	Audio2CF	4473	2042 ± 113	1.08 ± 0.25	0.77 ± 0.19	1.28 ± 0.37	2.96 ± 0.58	7.56 ± 0.88
	i-vectors	4473	2632 ± 179	1.01 ± 0.22	0.72 ± 0.16	1.32 ± 0.37	3.26 ± 0.57	8.13 ± 0.88
	Audio2Tag	4473	2887 ± 163	0.57 ± 0.18	0.40 ± 0.12	0.69 ± 0.29	1.66 ± 0.44	4.48 ± 0.70
MF	—	2835	1572 ± 150	2.21 ± 0.45	1.73 ± 0.37	3.49 ± 0.74	6.83 ± 1.01	13.50 ± 1.36
Hybrid MF	Logs	4473	2404 ± 164	1.47 ± 0.32	1.03 ± 0.22	1.90 ± 0.47	4.12 ± 0.66	8.86 ± 0.92
	Audio2CF	4473	2921 ± 164	0.74 ± 0.21	0.54 ± 0.15	0.91 ± 0.33	2.33 ± 0.52	5.60 ± 0.77
Neighbors	—	2835	5112 ± 321	2.66 ± 0.55	2.04 ± 0.43	4.02 ± 0.79	6.88 ± 1.04	11.84 ± 1.28
Popularity	—	4473	5217 ± 570	1.03 ± 0.27	0.67 ± 0.17	1.17 ± 0.36	3.01 ± 0.57	6.34 ± 0.80
Random	—	4473	6163 ± 218	0.10 ± 0.03	0.07 ± 0.02	0.09 ± 0.13	0.24 ± 0.18	0.98 ± 0.34

Table 7: Weak generalization results for the 8tracks dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 14,552 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

8tracks - weak generalization

system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	6289	556 ± 38	3.90 ± 0.43	2.46 ± 0.27	4.79 ± 0.62	11.46 ± 0.89	23.51 ± 1.21
	Tags + Logs	6289	587 ± 42	3.85 ± 0.45	2.48 ± 0.29	4.84 ± 0.61	10.65 ± 0.88	22.37 ± 1.20
	Audio2CF + Logs	6289	659 ± 45	3.40 ± 0.41	2.17 ± 0.26	4.09 ± 0.56	9.42 ± 0.83	20.62 ± 1.15
	Logs	6289	718 ± 58	3.62 ± 0.44*	2.31 ± 0.28	4.30 ± 0.58	9.92 ± 0.83	20.03 ± 1.12
	Audio2CF + Tags	6289	965 ± 61	2.84 ± 0.40	1.76 ± 0.25	3.22 ± 0.49	7.28 ± 0.75	15.74 ± 1.03
	Tags	6289	1084 ± 65	2.54 ± 0.36	1.63 ± 0.25	2.86 ± 0.47	6.55 ± 0.71	14.64 ± 0.98
	Audio2CF	6289	1988 ± 90	1.13 ± 0.24	0.69 ± 0.13	0.99 ± 0.29	2.75 ± 0.47	8.21 ± 0.80
	i-vectors	6289	2164 ± 114	1.34 ± 0.25	0.73 ± 0.12	1.32 ± 0.31	3.02 ± 0.46	7.10 ± 0.70
	Audio2Tag	6289	2946 ± 138	0.57 ± 0.15	0.35 ± 0.09	0.45 ± 0.18	1.26 ± 0.32	4.13 ± 0.56
Membership	Audio2CF + Tags + Logs	6289	652 ± 46	3.73 ± 0.44	2.35 ± 0.28	4.52 ± 0.60	9.58 ± 0.83	20.43 ± 1.12
	Audio2CF + Logs	6289	691 ± 49	3.33 ± 0.38	2.16 ± 0.25	4.20 ± 0.56	9.55 ± 0.83	20.05 ± 1.09
	Tags + Logs	6289	719 ± 53	3.47 ± 0.40	2.18 ± 0.26	4.68 ± 0.58	9.45 ± 0.80	19.59 ± 1.12
	Audio2CF + Tags	6289	762 ± 59	3.56 ± 0.43	2.19 ± 0.27	4.18 ± 0.56	9.42 ± 0.84	19.58 ± 1.08
	Tags	6289	804 ± 54	3.31 ± 0.40	2.05 ± 0.25	4.09 ± 0.56	8.88 ± 0.79	18.53 ± 1.10
	Logs	6289	986 ± 68	2.89 ± 0.37	1.78 ± 0.22	3.79 ± 0.52	8.00 ± 0.75	16.92 ± 1.04
	Audio2CF	6289	1149 ± 69	2.49 ± 0.35	1.47 ± 0.20	2.71 ± 0.46	6.44 ± 0.67	15.50 ± 1.02
	i-vectors	6289	1441 ± 86	2.23 ± 0.34	1.31 ± 0.19	2.31 ± 0.41	5.72 ± 0.61	12.76 ± 0.92
	Audio2Tag	6289	1841 ± 87	2.23 ± 0.36	1.33 ± 0.22	2.43 ± 0.42	5.63 ± 0.63	11.52 ± 0.88
MF	—	4299	1198 ± 145	4.04 ± 0.56	2.77 ± 0.42	5.27 ± 0.72	10.38 ± 1.01	18.62 ± 1.29
Hybrid MF	Logs	6289	1677 ± 138	2.79 ± 0.41	1.74 ± 0.25	3.19 ± 0.50	7.39 ± 0.75	15.88 ± 1.05
	Audio2CF	6289	2636 ± 142	1.09 ± 0.24	0.73 ± 0.18	1.06 ± 0.30	2.58 ± 0.46	7.03 ± 0.75
Neighbors	—	4299	3566 ± 318	4.93 ± 0.64	3.49 ± 0.48	6.94 ± 0.86	12.11 ± 1.08	19.50 ± 1.32
Popularity	—	6289	3352 ± 272	1.80 ± 0.33	1.05 ± 0.19	1.79 ± 0.36	4.14 ± 0.53	9.71 ± 0.81
Random	—	6289	7094 ± 234	0.15 ± 0.06	0.11 ± 0.06	0.15 ± 0.13	0.31 ± 0.17	0.78 ± 0.24

* The 95% bootstrap CIs for the MRR of Profiles (Logs) and Hybrid MF (Logs) do not overlap. However, this is not apparent from the table because, for readability, we only display the largest margin of the CIs.

Table 8: Strong generalization results for the AotM-2011 dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 12,286 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

AotM-2011 - strong generalization

system	**feature**	**N**	**med rank**	**MRR [%]**	**MAP [%]**	**R@10 [%]**	**R@30 [%]**	**R@100 [%]**
Profiles	—	—	—	—	—	—	—	—
Membership	Audio2CF + Tags + Logs	4473	1150 ± 72	1.88 ± 0.33	1.37 ± 0.25	2.53 ± 0.51	5.91 ± 0.78	13.69 ± 1.13
	Tags + Logs	4473	1223 ± 77	1.82 ± 0.33	1.23 ± 0.21	2.20 ± 0.48	5.66 ± 0.74	13.50 ± 1.13
	Audio2CF + Logs	4473	1248 ± 95	1.86 ± 0.33	1.25 ± 0.23	2.31 ± 0.48	5.41 ± 0.75	12.64 ± 1.08
	Audio2CF + Tags	4473	1285 ± 89	1.73 ± 0.31	1.26 ± 0.23	2.57 ± 0.55	5.42 ± 0.77	12.54 ± 1.10
	Tags	4473	1423 ± 106	1.79 ± 0.35	1.27 ± 0.26	1.97 ± 0.46	4.95 ± 0.73	12.33 ± 1.07
	Logs	4473	1552 ± 96	1.42 ± 0.29	0.98 ± 0.20	1.88 ± 0.47	4.18 ± 0.64	10.71 ± 1.04
	Audio2CF	4473	2065 ± 120	0.93 ± 0.21	0.62 ± 0.12	1.40 ± 0.39	3.18 ± 0.58	7.05 ± 0.85
	i-vectors	4473	2768 ± 185	0.93 ± 0.20	0.63 ± 0.15	1.11 ± 0.35	2.99 ± 0.58	7.45 ± 0.85
	Audio2Tag	4473	2900 ± 151	0.51 ± 0.14	0.35 ± 0.09	0.57 ± 0.26	1.29 ± 0.37	4.00 ± 0.64
MF	—	2849	1428 ± 135	2.72 ± 0.53	2.12 ± 0.42	4.03 ± 0.81	8.41 ± 1.10	15.43 ± 1.44
Hybrid MF	Logs	4473	2404 ± 164	1.47 ± 0.31	1.03 ± 0.23	1.90 ± 0.47	4.12 ± 0.67	8.86 ± 0.94
	Audio2CF	4473	2921 ± 164	0.74 ± 0.21	0.54 ± 0.14	0.91 ± 0.33	2.33 ± 0.52	5.60 ± 0.78
Neighbors	—	2849	4502 ± 304	4.89 ± 0.82	4.26 ± 0.73	6.54 ± 1.03	10.31 ± 1.24	15.16 ± 1.43
Popularity	—	4473	5143 ± 583	1.03 ± 0.27	0.67 ± 0.17	1.09 ± 0.34	3.07 ± 0.57	7.28 ± 0.85
Random	—	4473	6161 ± 188	0.13 ± 0.05	0.10 ± 0.05	0.13 ± 0.15	0.35 ± 0.22	1.23 ± 0.37

Table 9: Strong generalization results for the 8tracks dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 14,552 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

8tracks - strong generalization

system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Profiles	—	—	—	—	—	—	—	—
Membership	Audio2CF + Tags + Logs	6289	710 ±51	3.46 ±0.43	2.19 ±0.27	4.31 ±0.58	9.77 ±0.84	20.40 ±1.12
	Audio2CF + Logs	6289	773 ±52	3.28 ±0.42	2.04 ±0.27	3.57 ±0.52	8.68 ±0.78	18.38 ±1.07
	Tags + Logs	6289	792 ±51	3.41 ±0.44	2.10 ±0.26	3.69 ±0.53	8.83 ±0.80	18.90 ±1.11
	Audio2CF + Tags	6289	849 ±52	3.23 ±0.39	2.04 ±0.25	4.17 ±0.57	8.58 ±0.78	18.94 ±1.12
	Tags	6289	869 ±73	3.31 ±0.42	1.99 ±0.25	3.87 ±0.53	7.93 ±0.76	17.76 ±1.05
	Logs	6289	1038 ±78	2.85 ±0.39	1.70 ±0.22	3.10 ±0.47	7.36 ±0.71	15.90 ±1.03
	Audio2CF	6289	1317 ±86	2.58 ±0.37	1.52 ±0.22	2.72 ±0.44	5.96 ±0.64	14.54 ±0.99
	i-vectors	6289	1455 ±91 *	2.01 ±0.29	1.22 ±0.18	2.32 ±0.40	5.71 ±0.64	12.83 ±0.94
	Audio2Tag	6289	2093 ±125	1.91 ±0.31	1.11 ±0.19	2.15 ±0.38	4.61 ±0.57	9.68 ±0.81
MF	—	4298	997.5 ±109.5	4.48 ±0.61	3.06 ±0.42	5.41 ±0.75	11.02 ±1.05	20.32 ±1.35
Hybrid MF	Logs	6289	1677 ±138	2.79 ±0.41	1.74 ±0.26	3.19 ±0.50	7.39 ±0.73	15.88 ±1.04
	Audio2CF	6289	2636 ±142	1.09 ±0.24	0.73 ±0.18	1.06 ±0.30	2.58 ±0.47	7.03 ±0.75
Neighbors	—	4298	3045.5 ±390.5	5.27 ±0.68	3.87 ±0.54	7.11 ±0.85	12.76 ±1.14	20.43 ±1.41
Popularity	—	6289	3293 ±278	1.76 ±0.32	1.04 ±0.20	1.75 ±0.36	4.49 ±0.55	9.32 ±0.78
Random	—	6289	6876 ±158	0.14 ±0.06	0.10 ±0.05	0.15 ±0.13	0.21 ±0.14	0.73 ±0.25

* The 95% bootstrap CIs for the median rank of Membership (i-vectors) and Hybrid MF (Logs) do not overlap. However, this is not apparent from the table because, for readability, we only display the largest margin of the CIs.

Table 10: Results on the AotM-2011 dataset recommending songs that occurred in 4 or less training playlists using only features derived from the audio signal. The first three columns indicate the generalization setting, the system name, and the feature name (if any). The fourth column indicates the number N of songs in the withheld continuations involved in each experiment. The median rank is relative to 12,286 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold.

AotM-2011 - recommending songs that occurred in 4 or less training playlists with audio-based features only

generalization	system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Weak	Profiles	Audio2CF	3411	2130 ± 140	0.58 ± 0.16	0.46 ± 0.13	0.61 ± 0.30	2.23 ± 0.55	6.65 ± 0.94
		i-vectors	3411	2738 ± 177	0.49 ± 0.15	0.35 ± 0.10	0.57 ± 0.29	1.33 ± 0.43	4.06 ± 0.73
		Audio2Tag	3411	3141 ± 189	0.32 ± 0.08	0.25 ± 0.07	0.29 ± 0.20	1.07 ± 0.40	3.53 ± 0.68
	Membership	Audio2CF	3411	2264 ± 135	0.55 ± 0.17	0.46 ± 0.16	0.68 ± 0.33	1.75 ± 0.50	5.33 ± 0.84
		Audio2Tag	3411	3106 ± 174	0.26 ± 0.05	0.21 ± 0.04	0.28 ± 0.22	0.96 ± 0.39	3.36 ± 0.69
		i-vectors	3411	3514 ± 154	0.10 ± 0.01	0.09 ± 0.01	0.00 ± 0.00	0.06 ± 0.10	0.75 ± 0.34
	Hybrid MF	Audio2CF	3411	2718 ± 205	0.75 ± 0.22	0.62 ± 0.19	1.10 ± 0.40	2.74 ± 0.63	6.12 ± 0.92
	Random	—	3411	6108 ± 224	0.10 ± 0.03	0.08 ± 0.03	0.13 ± 0.17	0.30 ± 0.24	1.10 ± 0.41
Strong	Profiles	—	—	—	—	—	—	—	—
	Membership	Audio2CF	3456	2257.0 ± 134.5	0.43 ± 0.08	0.35 ± 0.07	0.68 ± 0.33	1.90 ± 0.52	5.20 ± 0.83
		Audio2Tag	3456	3029 ± 218	0.25 ± 0.07	0.19 ± 0.04	0.21 ± 0.17	0.53 ± 0.25	2.82 ± 0.62
		i-vectors	3456	3578.5 ± 194.5	0.11 ± 0.01	0.09 ± 0.01	0.00 ± 0.00	0.08 ± 0.11	0.80 ± 0.34
	Hybrid MF	Audio2CF	3456	2705 ± 202	0.75 ± 0.21	0.63 ± 0.20	1.11 ± 0.41	2.74 ± 0.63	6.12 ± 0.92
	Random	—	3456	6118 ± 214	0.12 ± 0.05	0.10 ± 0.05	0.15 ± 0.17	0.43 ± 0.26	1.23 ± 0.42

Table 11: Results on the 8tracks dataset recommending songs that occurred in 4 or less training playlists using only features derived from the audio signal. The first three columns indicate the generalization setting, the system name, and the feature name (if any). The fourth column indicates the number N of songs in the withheld continuations involved in each experiment. The median rank is relative to 14,552 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold.

8tracks - recommending songs that occurred in 4 or less training playlists with audio-based features only

generalization	system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Weak	Profiles	Audio2CF	4088	1778 ± 152	0.99 ± 0.22	0.71 ± 0.17	1.13 ± 0.36	2.94 ± 0.58	8.79 ± 1.01
		i-vectors	4088	2279.5 ± 145.5	0.71 ± 0.17	0.48 ± 0.10	0.86 ± 0.32	2.14 ± 0.50	5.48 ± 0.77
		Audio2Tag	4088	2727.0 ± 156.5	0.56 ± 0.17	0.38 ± 0.11	0.51 ± 0.23	1.48 ± 0.41	4.62 ± 0.72
	Membership	Audio2CF	4088	1922 ± 118	0.57 ± 0.15	0.43 ± 0.09	0.59 ± 0.27	1.96 ± 0.49	6.60 ± 0.89
		i-vectors	4088	2596.5 ± 119.0	0.32 ± 0.12	0.23 ± 0.07	0.20 ± 0.16	0.66 ± 0.27	2.35 ± 0.51
		Audio2Tag	4088	2779.5 ± 155.5	0.26 ± 0.04	0.21 ± 0.03	0.17 ± 0.13	0.93 ± 0.33	3.31 ± 0.64
	Hybrid MF	Audio2CF	4088	2180 ± 187	1.15 ± 0.28	0.84 ± 0.21	1.33 ± 0.40	3.03 ± 0.61	8.09 ± 0.94
	Random	—	4088	7168 ± 256	0.10 ± 0.04	0.07 ± 0.03	0.10 ± 0.12	0.20 ± 0.15	0.70 ± 0.27
Strong	Profiles	—	—	—	—	—	—	—	—
	Membership	Audio2CF	4134	2002 ± 136	0.58 ± 0.15	0.43 ± 0.11	0.51 ± 0.23	1.55 ± 0.43	6.03 ± 0.85
		i-vectors	4134	2522.5 ± 123.0	0.23 ± 0.05	0.17 ± 0.02	0.05 ± 0.06	0.55 ± 0.24	2.11 ± 0.50
		Audio2Tag	4134	2747.5 ± 152.5	0.35 ± 0.11	0.25 ± 0.06	0.30 ± 0.18	0.83 ± 0.32	3.59 ± 0.64
	Hybrid MF	Audio2CF	4134	2202.5 ± 190.0	1.16 ± 0.29	0.84 ± 0.21	1.35 ± 0.41	3.03 ± 0.60	8.11 ± 0.96
	Random	—	4134	6921.5 ± 220.5	0.08 ± 0.03	0.06 ± 0.02	0.02 ± 0.04	0.05 ± 0.06	0.61 ± 0.27

Author biographies

Andreu Vall received his Licentiate Degree (MSc) in Mathematics from the Technical University of Catalonia in 2008. From 2009 to 2013 he worked as a university research assistant conducting research first in the field of geomatics, and later in operations research applied to the fashion retail industry. Since 2014 he is a PhD student at the Institute of Computational Perception of the Johannes Kepler University Linz. His current research centers on machine learning approaches to hybrid music recommender systems, with a special focus on automated music playlist continuation.

Matthias Dorfer studied medical informatics at Vienna University of Technology, Austria, with a focus on computational image analysis and machine learning. After finishing his master's degree studies, he worked for two years as an industrial researcher in the field of medical image analysis. Since April 2015, he has been a Ph.D. student in the Institute of Computational Perception, Johannes Kepler University, Linz, Austria, under the supervision of Prof. Gerhard Widmer. His research interests are artificial neural networks, especially multimodality deep learning, and audiovisual representation learning.

Hamid Eghbal-zadeh is a PhD student at Johannes Kepler University Linz, Austria. He received his Master's degree in Computer Engineering from the Shiraz State University, Iran in 2012. In 2014, he joined the Institute of Computational Perception at Johannes Kepler University. During the following years, he worked as a researcher on the topic of representation learning for sequential data including audio, music and genomic data. He is currently pursuing his PhD and his focus of research is on representation learning with incomplete information.

Markus Schedl is Associate Professor at the Johannes Kepler University Linz / Institute for Computational Perception. He graduated in Computer Science from the Vienna University of Technology and earned his Ph.D. in Computer Science from the Johannes Kepler University Linz. He (co-)authored almost 200 refereed conference papers and journal articles and is Associate Editor of the Springer International Journal of Multimedia Information Retrieval. His main research interests include web and social media mining, data analytics, information retrieval, recommender systems, multimedia, and music information research.

Keki Burjorjee holds a Ph.D. in Computer Science from Brandeis University, USA, and is currently a Staff Scientist at Pandora Media Inc., the largest music streaming service in the USA. His research interests include large scale factorization based recommendation systems, probabilistic modeling, and evolutionary computation. At Pandora he has worked on station, playlist, and track recommendation, automatic genre tagging, and automatic playlist extension.

Gerhard Widmer is Professor and Head of the Institute of Computational Perception at Johannes Kepler University, Linz, Austria, and leads the Intelligent Music Processing and Machine Learning Group at the Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria. His research interests include AI, machine learning, and intelligent music processing, and his work is published in a wide range of scientific fields, from AI and machine learning to audio, multimedia, musicology, and music psychology. He is a Fellow of the European Association for Artificial Intelligence (EurAI), has been awarded Austrias highest research awards, the START Prize (1998) and the Wittgenstein Award 2009), and currently holds an ERC Advanced Grant for research on computational models of expressivity in music.