Music Playlist Recommendation with Long Short-Term Memory

Huiping Yang[1], Yan Zhao[1], Jinfu Xia[1], Bin Yao[2], Min Zhang[1], and Kai Zheng[3](✉)

[1] School of Computer Science and Technology, Soochow University, Suzhou, China
{hpyang2017,jfxia}@stu.suda.edu.cn, {zhaoyan,minzhang}@suda.edu.cn
[2] Shanghai Jiao Tong University, Shanghai, China
yaobin@cs.sjtu.edu.cn
[3] University of Electronic Science and Technology of China, Chengdu, China
zhengkai@uestc.edu.cn

Abstract. Music playlist recommendation is an important component in modern music streaming services, which is used for improving user experience by regularly pushing personalized music playlists based on users' preferences. In this paper, we propose a novel music playlist recommendation problem, namely Personalized Music Playlist Recommendation (PMPR), which aims to provide a suitable playlist for a user by taking into account her long/short-term preferences and music contextual data. We propose a data-driven framework, which is comprised of two phases: *user/music feature extraction* and *music playlist recommendation*. In the first phase, we adopt a matrix factorization technique to obtain long-term features of users and songs, and utilize the Paragraph Vector (PV) approach, an advanced natural language processing technique, to capture music context features, which are the basis of the subsequent music playlist recommendation. In the second phase, we design two Attention-based Long Short-Term Memory (AB-LSTM) models, i.e., typical AB-LSTM model and Improved AB-LSTM (IAB-LSTM) model, to achieve the suitable personalized playlist recommendation. Finally, we conduct extensive experiments using a real-world dataset, verifying the practicability of our proposed methods.

1 Introduction

With the advent of lossy compression techniques (e.g., MP3 format), the field of music distribution has changed from being medium based to being digitized, which makes the music much easier to be downloaded or received by users on their personal computers and mobile devices via Internet. However, with massive amount of music available from thousands of web sites or online services, avoiding overwhelming choices and finding the "right" music have become a challenge for users. This is calling for effective music playlist recommendation techniques that can provide suitable music playlists for users.

Most existing studies focus on user-preference-based music playlist recommendation, which infer users' preferences from past music-listening patterns or

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 416–432, 2019.
https://doi.org/10.1007/978-3-030-18579-4_25

explicit feedbacks. For example, [7] characterizes both items and users by factor vectors inferred from item-rating patterns. However, they fail to effectively incorporate users' instant preferences and historical music-listening records. The overall music-listening behavior of a user may be determined by her long-term interest. But at any given time, a user is also affected by her instant preferences due to transient events, such as issuance of new songs in the current time.

In the bulk of playlist recommendation research, music content, which is primarily extracted from the audio signal, plays a key role in generating and recommending songs for users. For instance, Cano et al. [3] automatically extract descriptions related to instrumentation, rhythm and harmony from music audio signals and design a music browsing and recommendation system based on the high-level music audio data similarity. However, content-based music recommendation has not been applied very successfully in large range systems so far [6]. Music context data, referring to all music-relevant information that is not directly extractable from the audio signal itself, is another important factor for improving the quality of music recommendation. Context-based music recommendation approaches have higher user acceptance and even outperform content-based techniques for music retrieval [6,17]. For example, Rendle et al. [14] explicitly model textual representations of musical knowledge (e.g., the pairwise interactions among users, items and tags) in music recommendation system, which performs well in runtime and achieves good recommendation quality. Cheng et al. [4] facilitate effective social music recommendation by considering users' location-related contexts as well as the global music popularity trends, which overcomes the cold-start and sparsity problems. Nevertheless, the work mentioned above ignores a crucial source of context-based data, comments of songs, which influence how a user (e.g., a listener) perceives music.

In this paper we propose a two-phase data-driven framework, namely Data-driven Music Playlist Recommendation (DMPR), which effectively combines users' long/short-term preferences and music contextual data. In the first phase, we obtain users' long-term preference features based on their favorite playlists, and songs' features (consisting of latent feature, semantic feature and category feature) based on their music context (i.e., lyrics, comments and belonging public playlists). In particular, we generate a rating matrix based on users' favorite playlists, and utilize a Matrix Factorization (MF) method to obtain users' long-term preference features for songs. The songs' latent features can be obtained by MF on the rating matrix as well. With the help of the Paragraph Vector (PV) approach [8], we can extract each song's semantic feature based on its lyrics and comments, and compute each song's category feature from its belonging public playlists. The second phase aims to combine a user's long/short-term preference features based on two Attention-based Long Short-Term Memory (AB-LSTM) models, i.e., typical AB-LSTM model and Improved AB-LSTM (IAB-LSTM) model, and recommend her a suitable playlist, which contains top-k related songs that have the highest probability of being liked.

The contributions of this paper can be summarized as follows:

(1) We provide a novel framework of Data-driven Music Playlist Recommendation (DMPR) based on users' long/short-term preferences and music contextual data, which aims to find a most suitable playlist for a user.
(2) The Matrix Factorization technique is adopted to effectively extract users' long-term preference features and songs' latent features.
(3) We introduce the Paragraph Vector (PV) approach, an advanced natural language processing technique, to extract the semantic features and category features of songs based on music context.
(4) Two Attention-based Long Short-Term Memory (AB-LSTM) models, are designed to balance the long/short-term preferences of a user in order to find the most suitable music playlist for her.
(5) We conduct extensive experiments on a real-world dataset, which empirically demonstrate the advantages of our proposed music playlist recommendation models compared to the baseline.

The remainder of this paper is organized as follows. Section 2 introduces the preliminary concepts and gives an overview of the proposed recommendation framework. Then we extract major features used in our work in Sect. 3. Two kinds of recommendation algorithms are presented in Sect. 4, followed by the experimental results presented in Sect. 5. Section 6 surveys the related works based on existing researches on music recommendation. Finally we conclude this paper in Sect. 7.

2 Problem Statement

In this section, we introduce some preliminary concepts and give an overview of the proposed recommendation framework. Table 1 summarizes the major notations used in the rest of the paper.

2.1 Preliminary Concept

Definition 1 (Song). *A song, denoted by* $s =< l, c >$, *consists of its lyrics* $s.l$ *and comments* $s.c$. *In addition, we use* S *to represent a set of songs.*

Definition 2 (User). *A user, denoted by* $u =< f, h >$, *consists of her favorite playlist* $u.f$ *and historical playlist records* $u.h$. *In particular, the favorite playlist of user* u, *denoted by* $u.f = (s_1, s_2, \ldots, s_n)$, *is a sequence of songs which have been marked as "like" by user* u, *and the playlist records history of user* u, *denoted by* $u.h = (s_1, s_2, \ldots, s_m)$, *is a finite sequence of songs sorted by time when* u *heard the songs recently. We use* U *to represent a set of users.*

Definition 3 (Public Playlist). *A public playlist, denoted by* $pl = (s_1, s_2, \ldots, s_n)$, *is a finite sequence of songs, which is created by the active users in the community.*

Table 1. Summary of notations

Notation	Definition
s	A song
$s.l$	The lyrics of song s
$s.c$	The comments of song s
S	A song set
u	A user
$u.f$	The favorite playlist of user u
$u.h$	The historical playlist records of user u
U	A user set
$r_{u,s}$	Rating between user u and song s
pl	A public playlist
$pl.v$	Vector representation of the public playlist pl
e^U	Latent preference feature matrix of user set U
e^S	Latent feature matrix of song set S
$s.v$	Distributed representation of song s
$s.v(l)$	Lyric vector representation of song s
$s.v(c)$	Comment vector representation of song s
$s.v(ca)$	Category vector representation of song s
p^u	Probability vector of user u

Note that the public playlist is different from albums, and everyone in the community can visit it without limits. For example, a HipHop fan can create a public playlist named "The Best HipHop 100", which consists of 100 HipHop songs, and each user can access to this public playlist.

Problem Statement. Given a set of users, a set of songs and public playlists, our Personalized Music Playlist Recommendation (PMPR) problem aims to provide each user an ideal playlist, which contains the top-k related songs that have the highest probability of being liked.

2.2 Framework Overview

The proposed framework shown in Fig. 1 consists of two major phases: (1) feature extraction; (2) music playlist recommendation.

Feature Extraction. This phase models long-term features of both users and songs. The features considered in our proposed Data-driven Music Playlist Recommendation (DMPR) framework can be divided into two major parts: (1) user preference feature, which describes a user's long-term preference for music; (2) music features, which describe the music context information.

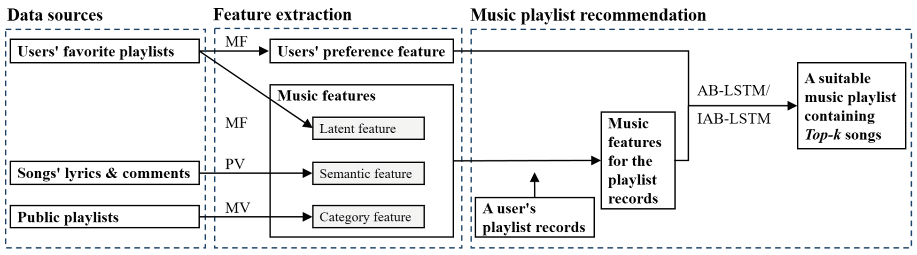
Data sources
Feature extraction
Music playlist recommendation
Users' favorite playlists
MF
Users' preference feature
Music features
MF
Latent feature
Songs' lyrics & comments
PV
Semantic feature
Public playlists
MV
Category feature
A user's playlist records
Music features for the playlist records
AB-LSTM/
IAB-LSTM
A suitable music playlist containing Top-k songs

Fig. 1. Framework overview

User preference feature represents a user's long-term preference for music, which can be obtained by Matrix Factorization (MF) based on a user-song rating matrix, which is observed from users' favorite playlists. MF maps both users and songs into a joint latent space, thus the user-song rating matrix is modeled as inner products of a user feature matrix and a song feature matrix. The user feature matrix and the song feature matrix contain latent features in this latent space, where a user's latent feature is regarded as her preference for music.

Music features comprehensively describe a song's latent feature, as well as its lyrics, comments and category information. Specifically, the music features extracted in our work consist of three parts: latent feature, semantic feature of both lyrics and comments, and category feature. Latent features of songs, which implies the ratings between users and songs, can be captured from users' favorite playlists via MF. We use a Paragraph Vector (PV) approach to obtain semantic feature from each song's lyrics and comments and design a Mean Value (MV) method to extract each song's category feature based on the public playlists, which usually contain a sequence of similar songs.

Music Playlist Recommendation. In this phase, we use two Attention-based Long Short-Term Memory (AB-LSTM) models to recommend a user the top-k related songs that have the highest probability of being liked based on her long/short-term preferences and songs' music features. The user preference feature extracted in the first phase represents a user's long-term preference for music, and the current songs' music features represent her short-term preference for music. Based on a user's long/short-term preferences for music, we propose two AB-LSTM model, i.e., typical AB-LSTM model and Improved AB-LSTM (IAB-LSTM) model, to recommend a suitable playlist to her.

3 Feature Extraction

In this section, we extract the main features that will be used in our work. The features can be mainly divided into two parts: (1) user preference feature, which describes a user's long-term preference for music; (2) music features, which describe the music context information.

3.1 User Preference Feature Extraction

In this section, we model users' long-term preference feature by Matrix Factorization (MF). MF performs well in learning latent features of users and songs from the observed ratings in the user-song rating matrix. Therefore we utilize the MF technique to model users' latent preference feature based on the user-song rating matrix, which can be obtained from users' favorite playlists, in order to describe the users' long-term preferences for songs.

We first generate a user-song rating matrix, $R \in \mathbb{R}^{N*M}$, which consists of N users and M songs based on users' favorite playlists. Each entry $r_{u,s}$ in matrix R denotes user u's rating on song s. For instance, if song s exists in user u's favorite playlists, then we have an indication that user u likes song s (i.e., $r_{u,s} = 1$). Otherwise, we set $r_{u,s} = 0$. We use two latent feature matrices to represent users and songs respectively, namely user feature matrix ($e^U \in \mathbb{R}^{N*d}$) and song feature matrix ($e^S \in \mathbb{R}^{M*d}$), which explain the ratings between users and songs. MF maps ratings between users and songs into a latent space, such that users' preference for songs is modeled as inner product between e^U and e^S in that latent space. The mapping of users' latent preference feature matrix e^U and songs' latent feature matrix e^S, is achieved by approximating the rating matrix by solving the following optimization problem:

$$\min_{e^U, e^S} \sum_{(u,s)\in K} (r_{u,s} - e_u^U e_s^{S^T})^2 + \lambda(||e_u^U||^2 + ||e_s^S||^2), \tag{1}$$

where K is the set of <user, song> pairs observed from users' favorite playlists, $r_{u,s}$ is the rating between user u and song s, e_u^U denotes the latent preference feature of user u, e_s^S denotes the latent feature of song s and λ is the regularization coefficient. The regularization coefficient λ is used to avoid overfitting. We apply gradient descent algorithm to solve the optimization problem in Eq. 1 and obtain the users' latent preference feature matrix e^U and songs' latent feature matrix e^S.

3.2 Music Feature Extraction

Music features comprehensively describe each song's latent feature, semantics and category. As shown in Fig. 2, the music features extracted in our work consist of three parts: (1) latent feature; (2) semantic feature of lyrics and comments; (3) category feature.

Latent Feature Extraction. Latent feature of song s, denoted as e_s^S, describes not only its musical identity but also many significant qualities that are relevant to understanding users' musical preferences. With the user-song rating matrix generated from users' favorite playlists, we can obtain each song's latent feature based on MF, following the same process as of user preference feature.

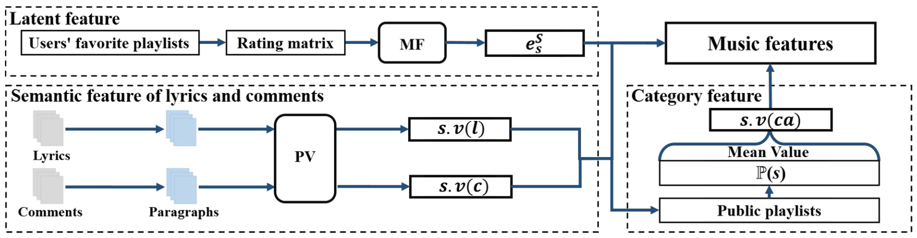

Fig. 2. Music feature extraction

Semantic Feature Extraction. In this part, we capture the songs' semantic feature of lyrics and comments to describe their music context. Lyrics is an important aspect of musical semantics since they usually imply the information about the artist/performer, e.g., cultural background, political orientation, and style of music [6]. Comments in the community are user-generated content, which have an increasingly impact on a user's preference.

We utilize the Paragraph Vector (PV) [8] technique, which is an unsupervised algorithm that learns continuous distributed vector representations for texts with any length, to obtain semantic feature of lyrics and comments. PV builds a word matrix W, where every word is mapped to a unique vector represented by a column, and builds a paragraph matrix D, where every paragraph is mapped to a unique vector represented by a column. For instance, a song's comments are considered as a sequence of words in a paragraph, denoted as $(w_1, w_2, \ldots, w_T)$. This comment paragraph is mapped into a unique vector represented by a column in matrix D and every word is mapped into a unique vector represented by a column in matrix W. In PV, the word vectors are asked to contribute to a prediction task about the next word, and the paragraph vector of this comment paragraph should also contribute to the prediction task of the next word when the contexts (sampled from the paragraph) are given. Thus, the goal of PV is to maximize the average log probability as follows:

$$\frac{1}{T}\sum_{t=k}^{T-k} \log p(w_t|w_{t-k}, \ldots, w_{t+k}), \tag{2}$$

where T is the length of the current paragraph, k controls the size of context window, and $p(w_t|w_{t-k}, \ldots, w_{t+k})$ is the probability that the predicted word is word w_t. The prediction task for the predicted word can be done via a multi-class classifier like softmax, which can be computed as follows:

$$p(w_t|w_{t-k}, \ldots, w_{t+k}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}, \tag{3}$$

where exp is the exponential function, y is a probability vector, and y_i is the un-normalized log-probability for word i to be the predicted word. y can be computed in Eq. 4.

$$y = b + Uh(w_{t-k}, \ldots, w_{t+k}; W; D), \tag{4}$$

where U and b are the softmax parameters, h is constructed by a concatenation of the word vectors of $(w_{t-k}, \ldots, w_{t+k})$ and the paragraph vector extracted from word matrix W and paragraph matrix D.

After training, we get word matrix W, softmax weights U and b, and paragraph matrix D on comments and lyrics. Then we add new lyrics paragraphs and comment paragraphs as columns in D and use gradient descendent on D while holding W, U, b fixed. Finally, we obtain the lyric vector representation $s.v(l)$ and comment vector representation $s.v(c)$ of song s from paragraph matrix D.

Category Feature Extraction. As mentioned before, a public playlist consists of a sequence of similar songs, which can be regarded as the same category. Therefore, we obtain the songs' category information based on public playlists.

With the lyrics vector representation $s.v(l)$ and comment vector representation $s.v(c)$ for each song s contained in the public playlist pl, the vector representation of pl, denoted by $pl.v$, can be formulated as follows:

$$pl.v = \frac{1}{n} \sum_{i=1}^{n} g(s_i.v(l), s_i.v(c)), \tag{5}$$

where n is the amount of songs in the public playlist pl, g is constructed by a concatenation of $s_i.v(l)$ and $s_i.v(c)$.

It is worth noting that a song may be contained in multiple public playlists, such that we calculate the song's category information by combining all its belonging public playlists' vectors as follows:

$$s.v(ca) = \frac{\sum_{pl \in \mathbb{P}(s)} pl.v}{|\mathbb{P}(s)|}, \tag{6}$$

where $s.v(ca)$ is song s's category vector representation, $\mathbb{P}(s)$ is a set of public playlists that contain song s and $|\mathbb{P}(s)|$ is the size of $\mathbb{P}(s)$. In addition, we concatenate semantic feature of lyrics $s.v(l)$ and comments $s.v(c)$, and category feature $s.v(ca)$ as the distributed representation of song s, denoted by $s.v \in \mathbb{R}^k$.

Given a song s, we can describe its music features by a combination of the latent feature e_s^S, and the distributed representation $s.v$.

4 Music Playlist Recommendation

In this section, we introduce our Attention-based Long Short-Term Memory (AB-LSTM) and Improved Attention-based Long Short-Term Memory (IAB-LSTM), which generate personalized playlists based on users' long/short preferences for songs and music context features.

LSTM is a variant of RNN, which is effective and scalable for sequential prediction problems [16]. Considering the time-ordered playlist records as sequential data, we adopt the Long Short-Term Memory (LSTM) to generate a suitable playlist that fits a user's musical interests. Recently, attention-based neural networks have been successfully used in many tasks like machine translation.

For example, [1] calculates weight attention score for each word in original sentences during the translation of the word. In our model, the attention mechanism can be used to calculate the weight attention scores for users' long/short-term preferences, which helps in recommendation performance. As a result, we use attention-based LSTM model to recommend suitable playlists for users. In particular, we design two models with different attention mechanisms. Firstly we adopt a typical Attention-Based Long Short-Term Memory (AB-LSTM) model to recommend playlists. Then we make some modification based on AB-LSTM, namely Improved Attention-Based Long Short-Term Memory (IAB-LSTM) model, to get better performance in recommendation.

4.1 AB-LSTM-Based Music Playlist Recommendation

In this section, we apply AB-LSTM model to recommend suitable songs for an individual user.

The architecture of AB-LSTM model is shown in Fig. 3, which contains three layers: input layer, hidden layer and output layer. The input layer contains the latent preference feature matrix of the user set U ($e^U \in \mathbb{R}^{N*d}$, N is the quantity of users in U), and user u's historical playlist records $u.h = (s_1, \ldots, s_t, \ldots, s_m)$. At each time step t, we concatenate the latent feature of song s_t, denoted by $e^S_{s_t} \in \mathbb{R}^d$, and the distributed representation of song s_t, denoted by $s_t.v \in \mathbb{R}^k$, as the music feature of song s_t, and input the music feature of song s_t and user preference feature matrix e^U to the hidden layer respectively.

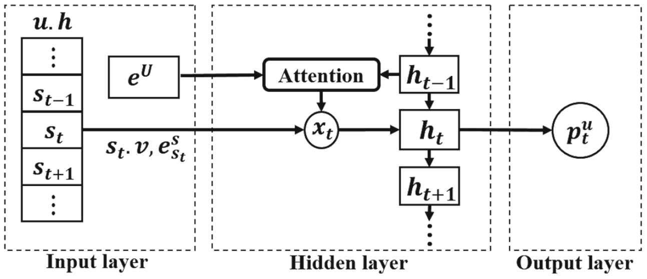

Fig. 3. Diagram of attention-based long short-term memory

The hidden layer, consisting of LSTM cells, is the key component of the AB-LSTM model, in which the hidden state can be computed as:

$$h_t = LSTM(h_{t-1}, x_t), \tag{7}$$

where h_t is the hidden state at time step t in the hidden layer, and x_t is a concatenation of the latent feature of song s_t ($e^S_{s_t}$), song s_t's distributed representation ($s_t.v$) and the context vector of user u at time t c^u_t. The context vector c^u_t is calculated in the attention part based on the user latent preference feature and the hidden state in the previous time step, which can be considered as an extra input to help AB-LSTM to fully get long-term preference of the user. c_t^u is calculated in Eq. 8.

$$c_t^u = \sum_{v}^{U} a_{v,t}^u e_v^U, \tag{8}$$

where $a_{v,t}^u$ is the attention weight of user u on user v at time step t, e_v^U represents the latent preference feature of user v. The attention weight $a_{v,t}^u$ is computed as follows:

$$a_{v,t}^u = \frac{\exp(\sigma(h_{t-1}, e_v^U))}{\sum_{v'}^{U} \exp(\sigma(h_{t-1}, e_{v'}^U))}, \tag{9}$$

where σ is a feed-forward neural network to produce a real-valued score. The attention weight $a_{v,t}^u$ determines which user's latent preference feature should be selected to generate user u's probability vector of predicted songs.

The probability vector of M predicted songs for user u at time step t, denoted as $p_t^u \in \mathbb{R}^M$, can be calculated by a single layer neural network activated by softmax in the output layer. The output of AB-LSTM model, p_t^u, can be computed as:

$$p_t^u = softmax(g(h_t)) = softmax(Wh_t + b), \tag{10}$$

where W and b are parameters of single layer neural network g, softmax function is used to squash the probability vector into a vector where each entry is in the range (0, 1), and all the entries add up to 1. In this model, we usually use p_m^u at the last time step m as the probability vector of the predicted songs for user u, denoted by p^u.

Finally, we generate the top-k related songs that have the highest probability of being liked by the user. As mentioned before, user u's probability vector of the predicted songs, denoted by p^u, is a M-length vector, where M is the amount of predicted songs. The value of p_j^u corresponds to the predicted probability that user u likes the j-th predicted song. Given a specified playlist length k, we select the top-k songs with the k greatest probability values from user u's probability vector (p^u) as the recommendation result.

4.2 IAB-LSTM-Based Music Playlist Recommendation

We further propose a novel model, namely Improved Attention-based Long Short-Term Memory (IAB-LSTM), with a new designed attention mechanism to recommend suitable songs for an individual user. We utilize an attention layer between hidden layer and output layer of IAB-LSTM, to replace the attention weight calculation during each time step of AB-LSTM's hidden layer, which helps in improving recommendation effectiveness. More importantly, IAB-LSTM focuses on calculating the attention weight of hidden states (implying the user's recent listened songs' music features) while AB-LSTM calculates the attention weight of user preference feature, which makes IAB-LSTM better in capturing the user's short-term preference.

The architecture of IAB-LSTM model is shown in Fig. 4, which contains four layers: input layer, hidden layer, attention layer and output layer. The

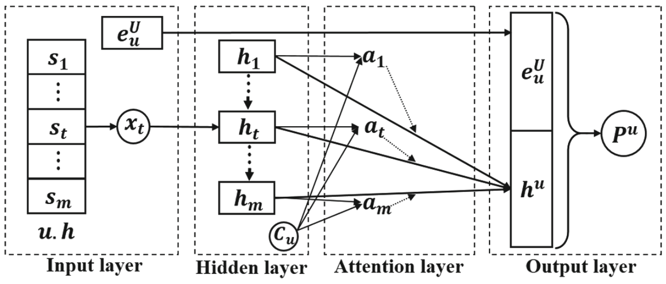

Fig. 4. Diagram of improved attention-based long short-term memory

input layer contains the latent preference feature of the user u, denoted by $e_u^U \in \mathbb{R}^d$, and her historical playlist records $u.h = (s_1, \ldots, s_t, \ldots, s_m)$. And at each time step t, we concatenate the latent feature of song s_t, denoted by $e_{s_t}^S \in \mathbb{R}^d$, and the distributed representation of song s_t, denoted by $s_t.v \in \mathbb{R}^k$, as an input x_t to the hidden layer. The hidden layer, consisting of LSTM cells, is an important component of the IAB-LSTM model, in which the hidden state can be computed as:

$$h_t = LSTM(h_{t-1}, x_t), \tag{11}$$

where h_t is the hidden state at time step t in the hidden layer.

The attention layer is the key component of IAB-LSTM, in which we summarize all hidden states as h^u, which is computed as follows:

$$h^u = \sum_{t=1}^{m} a_t^u h_t, \tag{12}$$

where m is the quantity of hidden states, a_t^u is the attention weight of user u on hidden state h_t at time step t, which is computed as follows:

$$a_t^u = \frac{\exp(g(h_t)^T C_u)}{\sum_{t'} \exp(g(h_{t'})^T C_u)}, g(h_t) = tanh(W h_t + b), \tag{13}$$

where W and b are parameters of single layer neural network g, and context vector C_u is randomly initialized and jointly learned during the training process. The context vector C_u can be seen as a high level representation of a fixed query "what is the informative song" in the user u's historical playlist records.

Finally, the probability vector of M predicted songs for user u, denoted as $p^u \in \mathbb{R}^M$, can be calculated by a single layer neural network activated by softmax in the output layer. The output of IAB-LSTM model, p^u, can be computed as:

$$p^u = softmax(g'(h^u, e_u^U)) = softmax(W'(h^u \oplus e_u^U) + b'), \tag{14}$$

where p^u is user u's probability vector of the predicted songs, e_u^U is the latent preference feature of user u, W' and b' are parameters of single layer neural network g', $\oplus$ is defined as a concatenation and h^u is calculated in Eq. 12. Getting the probability vector, we generate a suitable playlist for the user by the same way as AB-LSTM.

5 Experiments

In this section, we conduct extensive experiments on a real-world dataset to study the performance of the proposed models. All the algorithms are implemented on an Intel(R) Xeon(R) CPU E7-4860 v2 @ 2.60 GHz with 64 G RAM.

5.1 Experiment Setup

Dataset. We use a real-world dataset generated by Netease Cloud Music, which is a freemium music streaming service. Netease Cloud Music has 300 million users and a music database consisting of more than over 10 million songs. Specifically, we crawl a dataset containing 35365 users, 1496 public playlists, 35469 songs and 377194 comments.

Evaluation. We study and compare the performance of the following algorithms:

(1) CF: a model-based Collaborative Filtering approach [7], which utilizes matrix factorization to calculate the ratings between all the users and songs and recommends the top-k songs with highest rating values for each user.
(2) BPR: Bayesian Personalized Ranking [13] ranks each user's preference for songs and provides a top-k recommendation.
(3) P-AB-LSTM: Preference-Based AB-LSTM, an AB-LSTM without song's distributed representation (i.e., a song's semantics and category information).
(4) P-IAB-LSTM: Preference-Based IAB-LSTM without song's distributed representation (i.e., a song's semantics and category information).
(5) AB-LSTM: Attention-Based Long Short-Term Memory model (based on both the user's preference and song's distributed representation).
(6) IAB-LSTM: Improved Attention-Based Long Short-Term Memory model (based on both the user's preference and song's distributed representation).

Four widely-used metrics, Precision@k (P@k, the accuracy rate of top-k recommendation), Normalized Discounted Cumulative Gain@k (NDCG@k), Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR), are used to evaluate the recommendation precision of the above methods. The greater values of the above metrics mean the better performance. CPU time is given by the average time cost of recommending the top-k songs by the recommendation models. We also evaluate the recommended playlists based on the number of recalled songs. Specifically, for each user, we use CF, AB-LSTM and IAB-LSTM to generate a playlist containing 10 songs and compare it with her real playlist records of next 10 songs. In addition, CF selects top-10 related songs with highest ratings as the generated playlist. A greater number of recalled songs mean better performance.

Table 2 shows our experimental settings, where the default values of all parameters are underlined.

Table 2. Experiment parameters

Parameters	Values
Size of ratings	25%, 50%, 75%, <u>100%</u>
Length of historical playlist records	20, <u>30</u>, 40, 50
Amount of features extracted via MF	20, <u>40</u>, 60, 80

5.2 Experiment Results

We first compare our proposed models with two baseline methods including CF and BPR. The experimental results are summarized in Table 3. Our models consistently outperform the baseline methods by a noticeable margin. Taking the P@10 as an example, our IAB-LSTM achieves the most accurate recommendation result, whose accuracy is improved by around 83% compared with CF and 152% compared with BPR.

Table 3. Results of models

	P@3	P@5	P@10	NDCG@3	NDCG@5	NDCG@10	MAP	MRR
CF	0.0509	0.0517	0.0580	0.0846	0.1024	0.1376	0.1104	0.1181
BPR	0.0474	0.0453	0.0421	0.0856	0.1009	0.1150	0.0996	0.1077
P-AB-LSTM	0.0838	0.0822	0.0919	0.1047	0.1154	0.1742	0.1752	0.1920
P-IAB-LSTM	0.0891	0.0930	0.1020	0.1159	0.1440	0.2279	0.1840	0.2001
AB-LSTM	0.0817	0.0832	0.1061	0.1113	0.1281	0.1859	0.1858	0.2040
IAB-LSTM	**0.0962**	**0.0966**	**0.1356**	**0.1416**	**0.1640**	**0.2430**	**0.1916**	**0.2151**

Effect of the Size of Ratings. In this part of experiments, we change the size of the ratings used in feature extraction and study their effects on music playlist recommendation. As shown in Fig. 5, the precision of all approaches increases (with MAP and P@10 increase) when more ratings are used. Among all the methods, IAB-LSTM achieves the highest precision since IAB-LSTM effectively captures users' preferences and music context features, which demonstrates the robustness of our proposed algorithms. The CPU cost of all methods is not apparently affected by the size of the ratings since the prediction phrase is not directly computed from the ratings. Moreover, AB-LSTM (IAB-LSTM) runs slower than P-AB-LSTM (P-IAB-LSTM) because of the extra time cost for integrating the song's semantics and category features into the recommendation.

Effect of the Length of Historical Playlist Records. Figure 6 illustrates the effect of the length of historical playlist records on the performance of our models. Naturally the precision of all model gradually increases (with MAP and P@10 increase) as the length of historical playlist records grows. Obviously the performance of IAB-LSTM is better than P-IAB-LSTM, which proves the effectiveness of song's distributed representation. The CPU cost increases when the length of historical playlist records increases, since the size of input in our models is determined by the length of historical playlist records.

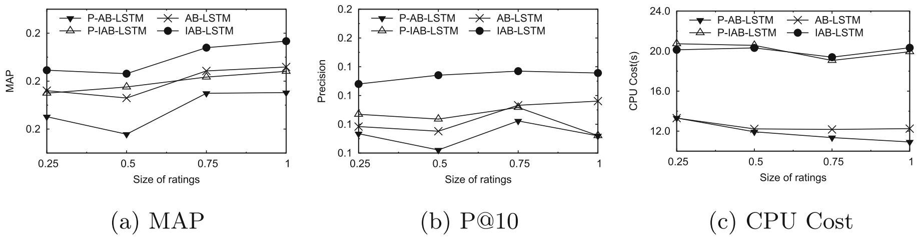

Fig. 5. Effect of the size of ratings

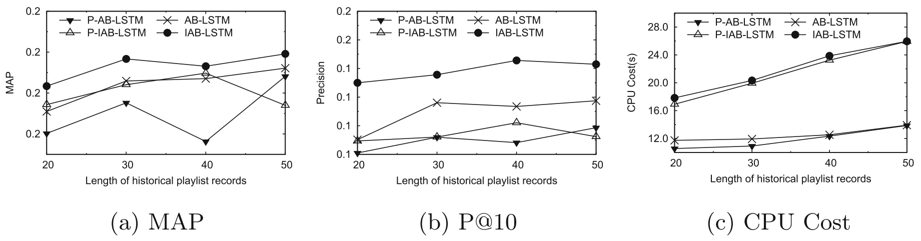

Fig. 6. Effect of the length of historical playlist records

Effect of the Amount of Features Extracted via MF. Next we study the effects of the amount of features extracted via MF. From Fig. 7 we can see that the precision of all algorithms increases (with MAP and P@10 increase) when the amount of features extracted via MF grows. The CPU cost slightly increases when the amount of features extracted via MF increases, since the greater size of the latent preference feature of the user increases computational complexity.

Recalled Songs. Finally we provide the recommended music playlists generated by some algorithms (i.e., CF, AB-LSTM and IAB-LSTM) for two users who are randomly selected from the dataset. The predicted songs existing in the user's real playlist records will be marked by ✓ in Table 4, which shows that our proposed IAB-LSTM can better capture the preferences of users than other methods. Taking user 16596 as an example, IAB-LSTM recalls 4 songs and

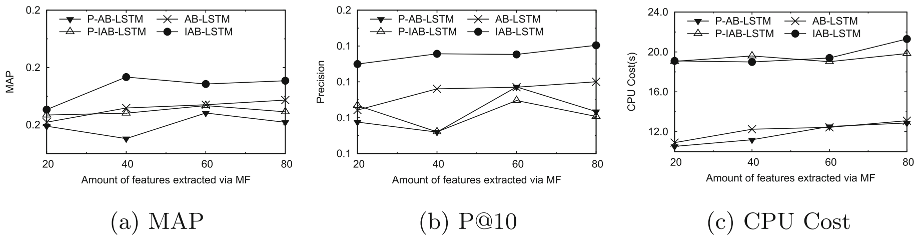

Fig. 7. Effect of the amount of features extracted via MF

AB-LSTM recalls 2 songs, while CF recalls 0 song. This means our model can make better music playlist recommendation with users' long/short term preferences and music context features.

Table 4. The recalled songs from Top-10 candidates

User Id	CF	AB-LSTM	IAB-LSTM
16596	We Don't Talk Anymore	Lie ✓	She Say
	Fade	Willow	Lie ✓
	Into A River	Iron Heart	I Am You
	Something	Old Street	In Fact, No
	Make You Mine	Travel	Along The Way
	A Half	Be What You Wanna Be	Cheng Du
	East Of Eden	Chasing The Wind	Outside The Light Years
	Es rappelt im Karton	Young And Young ✓	Lone Ranger ✓
	Counting Stars	Yesterday	Young And Young ✓
	I Am You	Star Falling	Trap ✓
8759	Say Honestly	Those Were The Days	Stupid ✓
	Special	Waste	Lie ✓
	Lover Boy 88	When You	Blue Lotus
	Jocelyn Flores	Lie ✓	Don't Understand
	Complete	Iron Heart	Vincent
	Regret	Young And Young	Don't Talk
	Meet	Yesterday	Young And Young
	Destination	Trap	Mind
	My Heart Will Go On	Calorie	Trap
	No Sad No Bad	Star Falling	Best

6 Related Work

The music playlist recommendation has attracted a number of researchers in recent years. In this section, we categorize the major methodologies used by recommendation systems as being based on: (1) music content; (2) music context.

Music Content-Based Recommendation. Music content-based features refer to tonality, pitch, and beat, symbolic features extracted from the music scores [9]. Existing research in the area of audio content-based music recommendation usually focuses on measuring music similarity [2]. For example, McFee et al. [10] treat music similarity learning as an information retrieval problem, where the music similarity is learned to optimize the ranked list of results in response to a query song. Dieleman et al. [11] use content-based latent factors to produce sensible recommendations, ignoring the fact that there is a large semantic gap between the song's characteristics and its corresponding audio signal. Wang et al. [19] simultaneously learn features from audio content and make personalized recommendation, which performs well on both the warm-start and cold-start problems. However, content-based music recommendation has not been applied very successfully in large range systems so far [6].

Music Context-Based Recommendation. In music recommendation field, contextual data refers to all music-relevant information that is not included in the audio signal itself. In particular, we review three main types of context-based approaches: (1) text-retrieval-based approaches; (2) co-occurrence-based approaches; (3) user-preference-based approaches.

Text-retrieval-based approaches exploit textual representations of musical knowledge originating from web pages, user tags, or song lyrics. Oramas et al. [12] exploit tags and textual descriptions to extract and link entities to external graphs which are in turn used to semantically enrich the initial data in music recommendation. Schedl et al. [15] address the problem of similarity measurement among music artists via text-based features extracted from Web pages.

Co-occurrence-based approaches follow an immediate mechanism to estimate similarity based on the occurrence of two music pieces or artists within the same context like web pages, microblogs, playlists, and Peer-to-Peer (P2P) networks. Zangerle et al. [20] use the absolute numbers of co-occurrences between songs in order to measure the similarities between songs and artists, which helps in music recommendation systems.

User-preference-based approaches usually estimate music context similarity based on users' feedbacks. Cheng et al. [5] present a venue-aware music recommender system that recommends music to match different types of common venues in user's everyday life. Wang et. al. [18] learn the low dimensional representations of music pieces from users' music listening sequences using neural network models. Cheng et al. [4] develop an effective social music recommendation system by considering users' location-related contexts as well as the global music popularity trends.

7 Conclusions

In this paper, we study the problem of Personalized Music Playlist Recommendation, where each user can receive a personalized music playlist based on her historical playlist records and music context. To settle this problem, we propose a novel Data-driven Music Playlist Recommendation (DMPR) framework, which incorporates long/short-term preferences of users and music features to improve the performance of recommendation. We address a few challenges by proposing different strategies to extract the long-term features of users and songs and designing effective AB-LSTM models to recommend a personalized music playlist (including top-k related songs that have the highest probability of being liked) for each user by obtaining her short-term preference. Extensive empirical study based on a real dataset demonstrates our proposed models can effectively capture long/short-term preferences of users via attention mechanisms, and recommend suitable personalized playlists to users.

Acknowledgement. This work was supported by the NSFC (61832017, 61532018, 61836007, 61872235, 61729202, U1636210), and The National Key Research and Development Program of China (2018YFC1504504).

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR (2014)
2. Bogdanov, D., Haro, M., Fuhrmann, F., Xambó, A., Gómez, E., Herrera, P.: Semantic audio content-based music recommendation and visualization based on user preference examples. IPM **49**(1), 13–33 (2013)
3. Cano, P., Koppenberger, M., Wack, N.: Content-based music audio recommendation. In: ACM Multimedia, pp. 211–212 (2005)
4. Cheng, Z., Shen, J.: Just-for-me: an adaptive personalization system for location-aware social music recommendation. In: ICMR, p. 185 (2014)
5. Cheng, Z., Shen, J.: On effective location-aware music recommendation. ACM TOIS **34**(2), 13 (2016)
6. Knees, P., Schedl, M.: A survey of music similarity and recommendation from music context data. TOMCCAP **10**(1), 2 (2013)
7. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
8. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: ICML, pp. 1188–1196 (2014)
9. Li, T., Ogihara, M., Li, Q.: A comparative study on content-based music genre classification. In: SIGIR, pp. 282–289 (2003)
10. McFee, B., Barrington, L., Lanckriet, G.: Learning content similarity for music recommendation. TASLP **20**(8), 2207–2218 (2012)
11. Van den Oord, A., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. In: NIPS, pp. 2643–2651 (2013)
12. Oramas, S., Ostuni, V.C., Noia, T.D., Serra, X., Sciascio, E.D.: Sound and music recommendation with knowledge graphs. ACM TIST **8**(2), 21 (2017)
13. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: AUAI, pp. 452–461 (2009)
14. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: WSDM, pp. 81–90 (2010)
15. Schedl, M., Pohle, T., Knees, P., Widmer, G.: Exploring the music similarity space on the web. ACM TOIS **29**(3), 14 (2011)
16. Schmidhuber, J., Wierstra, D., Gomez, F.J.: Evolino: hybrid neuroevolution/optimal linear search for sequence prediction. In: IJCAI (2005)
17. Slaney, M.: Web-scale multimedia analysis: does content matter? IEEE Multimed. **18**(2), 12–15 (2011)
18. Wang, D., Deng, S., Xu, G.: Sequence-based context-aware music recommendation. Inf. Retr. J. **21**(2–3), 230–252 (2018)
19. Wang, X., Wang, Y.: Improving content-based and hybrid music recommendation using deep learning. In: ACM Multimedia, pp. 627–636 (2014)
20. Zangerle, E., Gassler, W., Specht, G.: Exploiting twitter's collective knowledge for music recommendations. In: #MSM, pp. 14–17 (2012)