AUTOMATIC, PERSONALIZED, AND FLEXIBLE PLAYLIST GENERATION USING REINFORCEMENT LEARNING

Shun-Yao Shih
National Taiwan University
shunyaoshih@gmail.com

Heng-Yu Chi
KKBOX Inc., Taipei, Taiwan
henrychi@kkbox.com

ABSTRACT

Songs can be well arranged by professional music curators to form a riveting playlist that creates engaging listening experiences. However, it is time-consuming for curators to timely rearrange these playlists for fitting trends in future. By exploiting the techniques of deep learning and reinforcement learning, in this paper, we consider music playlist generation as a language modeling problem and solve it by the proposed attention language model with policy gradient. We develop a systematic and interactive approach so that the resulting playlists can be tuned flexibly according to user preferences. Considering a playlist as a sequence of words, we first train our attention RNN language model on baseline recommended playlists. By optimizing suitable imposed reward functions, the model is thus refined for corresponding preferences. The experimental results demonstrate that our approach not only generates coherent playlists automatically but is also able to flexibly recommend personalized playlists for diversity, novelty and freshness.

1. INTRODUCTION

Professional music curators or DJs are usually able to carefully select, order, and form a list of songs which can give listeners brilliant listening experiences. For a music radio with a specific topic, they can collect songs related to the topic and sort in a smooth context. By considering preferences of users, curators can also find what they like and recommend them several lists of songs. However, different people have different preferences toward diversity, popularity, and etc. Therefore, it will be great if we can refine playlists based on different preferences of users on the fly. Besides, as online music streaming services grow, there are more and more demands for efficient and effective music playlist recommendation. Automatic and personalized music playlist generation thus becomes a critical issue.

However, it is unfeasible and expensive for editors to daily or hourly generate suitable playlists for all users based on their preferences about trends, novelty, diversity, etc. Therefore, most of previous works try to deal with such problems by considering some particular assumptions. McFee *et al.* [14] consider playlist generation as a language modeling problem and solve it by adopting statistical techniques. Unfortunately, statistical method does not perform well on small datasets. Pampalk *et al.* [16] generate playlists by exploiting explicit user behaviors such as skipping. However, for implicit user preferences on playlists, they do not provide a systematic way to handle it.

As a result, for generating personalized playlists automatically and flexibly, we develop a novel and scalable music playlist generation system. The system consists of three main steps. First, we adopt Chen *et al.*'s work [4] to generate baseline playlists based on the preferences of users about songs. In details, given the relationship between users and songs, we construct a corresponding bipartite graph at first. With the users and songs graph, we can calculate embedding features of songs and thus obtain the baseline playlist for each songs by finding their k-nearest neighbors. Second, by formulating baseline playlists as sequences of words, we can pretrain RNN language model (RNN-LM) to obtain better initial parameters for the following optimization, using policy gradient reinforcement learning. We adopt RNN-LM because not only RNN-LM has better ability of learning information progresses than traditional statistical methods in many generation tasks, but also neural networks can be combined with reinforcement learning to achieve better performances [10]. Finally, given preferences from user profiles and the pretrained parameters, we can generate personalized playlists by exploiting techniques of policy gradient reinforcement learning with corresponding reward functions. Combining these training steps, the experimental results show that we can generate personalized playlists to satisfy different preferences of users with ease.

Our contributions are summarized as follows:

- We design an automatic playlist generation framework, which is able to provide timely recommended playlists for online music streaming services.
- We remodel music playlist generation into a sequence prediction problem using RNN-LM which is easily combined with policy gradient reinforcement learning method.
- The proposed method can flexibly generate suitable personalized playlists according to user profiles us-

ing corresponding optimization goals in policy gradient.

The rest of this paper is organized as follows. In Section 2, we introduce several related works about playlist generation and recommendation. In Section 3, we provide essential prior knowledge of our work related to policy gradient. In Section 4, we introduce the details of our proposed model, attention RNN-LM with concatenation (AC-RNN-LM). In Section 5, we show the effectiveness of our method and conclude our work in Section 6.

2. RELATED WORK

Given a list of songs, previous works try to rearrange them for better song sequences [1,3,5,12]. First, they construct a song graph by considering songs in playlist as vertices, and relevance of audio features between songs as edges. Then they find a Hamiltonian path with some properties, such as smooth transitions of songs [3], to create new sequencing of songs. User feedback is also an important consideration when we want to generate playlists [6, 7, 13, 16]. By considering several properties, such as tempo, loudness, topics, and artists, of users' favorite played songs recently, authors of [6, 7] can thus provide personalized playlist for users based on favorite properties of users. Pampalk *et al.* [16] consider skip behaviors as negative signals and the proposed approach can automatically choose the next song according to audio features and avoid skipped songs at the same time. Maillet *et al.* [13] provides a more interactive way to users. Users can manipulate weights of tags to express high-level music characteristics and obtain corresponding playlists they want. To better integrate user behavior into playlist generation, several works are proposed to combine playlist generation algorithms with the techniques of reinforcement learning [11, 20]. Xing *et al.* first introduce exploration into traditional collaborative filtering to learn preferences of users. Liebman *et al.* take the formulation of Markov Decision Process into playlist generation framework to design algorithms that learn representations for preferences of users based on hand-crafted features. By using these representations, they can generate personalized playlist for users.

Beyond playlist generation, there are several works adopting the concept of playlist generation to facilitate recommendation systems. Given a set of songs, Vargas *et al.* [18] propose several scoring functions, such as diversity and novelty, and retrieve the top-K songs with higher scores for each user as the resulting recommended list of songs. Chen *et al.* [4] propose a query-based music recommendation system that allow users to select a preferred song as a seed song to obtain related songs as a recommended playlist.

3. POLICY GRADIENT REINFORCEMENT LEARNING

Reinforcement learning has got a lot of attentions from public since Silver *et al.* [17] proposed a general reinforcement learning algorithm that could make an agent achieve superhuman performance in many games. Besides, reinforcement learning has been successfully applied to many other problems such as dialogue generation modeled as Markov Decision Process (MDP).

A Markov Decision Process is usually denoted by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where

- $\mathcal{S}$ is a set of states
- $\mathcal{A}$ is a set of actions
- $\mathcal{P}(s, a, s') = \Pr[s'|s, a]$ is the transition probability that action a in state s will lead to state s'
- $\mathcal{R}(s, a) = \mathbb{E}[r|s, a]$ is the expected reward that an agent will receive when the agent does action a in state s.
- $\gamma \in [0, 1]$ is the discount factor representing the importance of future rewards

Policy gradient is a reinforcement learning algorithm to solve MDP problems. Modeling an agent with parameters θ, the goal of this algorithm is to find the best θ of a policy $\pi_\theta(s, a) = \Pr[a|s, \theta]$ measured by average reward per time-step

$$J(\theta) = \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s) \sum_{a\in\mathcal{A}} \pi_\theta(s, a)\mathcal{R}(s, a) \tag{1}$$

where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ.

Usually, we assume that $\pi_\theta(s, a)$ is differentiable with respect to its parameters θ, i.e., $\frac{\partial \pi_\theta(s,a)}{\partial \theta}$ exists, and solve this optimization problem Eqn (1) by gradient ascent. Formally, given a small enough α, we update its parameters θ by

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{2}$$

where

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s) \sum_{a\in\mathcal{A}} \pi_\theta(s, a)\nabla_\theta \pi_\theta(s, a)\mathcal{R}(s, a) \\ &= \mathbb{E}[\nabla_\theta \pi_\theta(s, a)\mathcal{R}(s, a)]\end{aligned} \tag{3}$$

4. THE PROPOSED MODEL

The proposed model consists of two main components. We first introduce the structure of the proposed RNN-based model in Section 4.1. Then in Section 4.2, we formulate the problem as a Markov Decison Process and solve the formulated problem by policy gradient to generate refined playlists.

4.1 Attention RNN Language Model

Given a sequence of tokens $\{w_1, w_2, \ldots, w_t\}$, an RNN-LM estimates the probability $\Pr[w_t|w_{1:t-1}]$ with a recurrent function

$$h_t = f(h_{t-1}, w_{t-1}) \tag{4}$$

and an output function, usually softmax,

$$\Pr[w_t = v_i|w_{1:t-1}] = \frac{\exp(W_{v_i}^\top h_t + b_{v_i})}{\sum_k \exp(W_{v_k}^\top h_t + b_{v_k})} \tag{5}$$

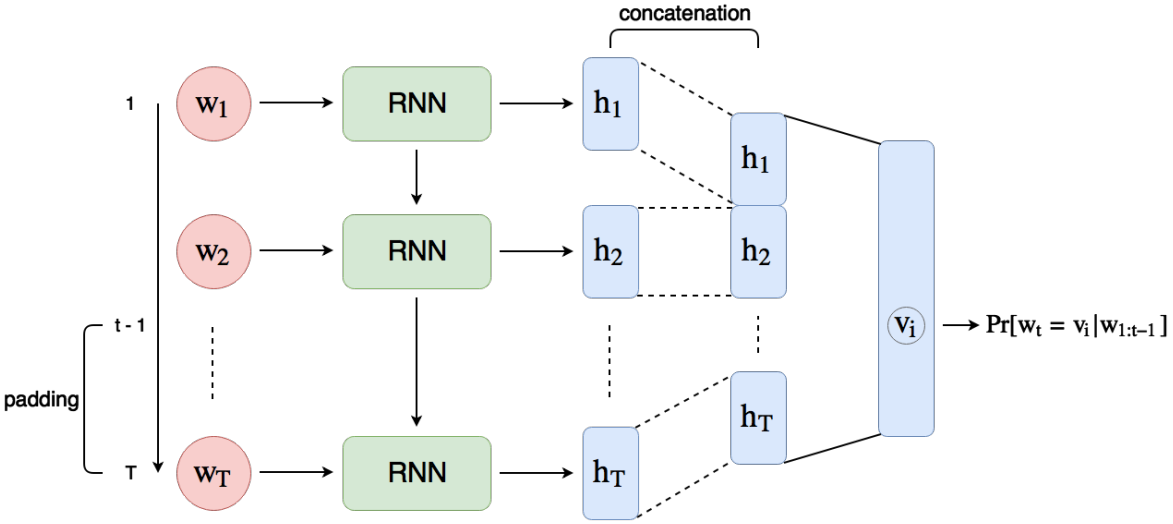

Figure 1. The structure of our attention RNN language model with concatenation

where the implementation of the function f depends on which kind of RNN cell we use, $h_t \in \mathbb{R}^D$, $W \in \mathbb{R}^{D \times V}$ with the column vector W_{v_i} corresponding to a word v_i, and $b \in \mathbb{R}^V$ with the scalar b_{v_i} corresponding to a word v_i (D is the number of units in RNN, and V is the number of unique tokens in all sequences).

We then update the parameters of the RNN-LM by maximizing the log-likelihood on a set of sequences with size N, $\{s_1, s_2, \ldots, s_N\}$, and the corresponding tokens, $\{w_1^{s_i}, w_2^{s_i}, \ldots, w_{|s_i|}^{s_i}\}$.

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=2}^{|s_n|} \log \Pr[w_t^{s_n} | w_{1:t-1}^{s_n}] \tag{6}$$

4.1.1 Attention in RNN-LM

Attention mechanism in sequence-to-sequence model has been proven to be effective in the fields of image caption generation, machine translation, dialogue generation, and etc. Several previous works also indicate that attention is even more impressive on RNN-LM [15].

In attention RNN language model (A-RNN-LM), given the hidden states from time $t - C_{ws}$ to t, denoted as $h_{t-C_{ws}:t}$, where C_{ws} is the attention window size, we want to compute a context vector c_t as a weighted sum of hidden states $h_{t-C_{ws}:t-1}$ and then encode the context vector c_t into the original hidden state h_t.

$$\beta_i = \nu^\top \tanh(W_1 h_t + W_2 h_{t-C_{ws}+i}) \tag{7}$$

$$\alpha_i = \frac{\exp(\beta_i)}{\sum_{k=0}^{C_{ws}-1} \exp(\beta_k)} \tag{8}$$

$$c_t = \sum_{i=0}^{C_{ws}-1} \alpha_i h_{t-C_{ws}+i} \tag{9}$$

$$h'_t = W_3 \begin{bmatrix} h_t \\ c_t \end{bmatrix} \tag{10}$$

where β is Bahdanau's scoring style [2], $W_1, W_2 \in \mathbb{R}^{D \times D}$, and $W_3 \in \mathbb{R}^{D \times 2D}$.

4.1.2 Our Attention RNN-LM with concatenation

In our work, $\{s_1, s_2, \ldots, s_N\}$ and $\{w_1^{s_i}, w_2^{s_i}, \ldots, w_{|s_i|}^{s_i}\}$ are playlists and songs by adopting Chen *et al.*'s work [4]. More specifically, given a seed song $w_1^{s_i}$ for a playlist s_i, we find top-k approximate nearest neighbors of $w_1^{s_i}$ to formulate a list of songs $\{w_1^{s_i}, w_2^{s_i}, \ldots, w_{|s_i|}^{s_i}\}$.

The proposed attention RNN-LM with concatenation (AC-RNN-LM) is shown in Figure 1. We pad $w_{1:t-1}$ to $w_{1:T}$ and concatenate the corresponding $h'_{1:T}$ as the input of our RNN-LM's output function in Eqn (5), where T is the maximum number of songs we consider in one playlist. Therefore, our output function becomes

$$\Pr[w_t = v_i | w_{1:t-1}] = \frac{\exp(W_{v_i}^\top h' + b_{v_i})}{\sum_k \exp(W_{v_k}^\top h' + b_{v_k})} \tag{11}$$

where $W \in \mathbb{R}^{DT \times V}$, $b \in \mathbb{R}^V$ and

$$h' = \begin{bmatrix} h'_1 \\ h'_2 \\ \vdots \\ h'_T \end{bmatrix} \in \mathbb{R}^{DT \times 1} \tag{12}$$

4.2 Policy Gradient

We exploit policy gradient in order to optimize Eqn (1), which is formulated as follows.

4.2.1 Action

An action a is a song id, which is a unique representation of each song, that the model is about to generate. The set of actions in our problem is finite since we would like to recommend limited range of songs.

4.2.2 State

A state s is the songs we have already recommended including the seed song, $\{w_1^{s_i}, w_2^{s_i}, \ldots, w_{t-1}^{s_i}\}$.

4.2.3 Policy

A policy $\pi_\theta(s, a)$ takes the form of our AC-RNN-LM and is defined by its parameters θ.

4.2.4 Reward

Reward $\mathcal{R}(s, a)$ is a weighted sum of several reward functions, i.e., $\mathcal{R}_i : s \times a \mapsto \mathbb{R}$. In the following introductions, we formulate 4 important metrics for playlists generation. The policy of our pretrained AC-RNN-LM is denoted as $\pi_{\theta_{RNN}}(s, a)$ with parameters θ_{RNN}, and the policy of our AC-RNN-LM optimized by policy gradient is denoted as $\pi_{\theta_{RL}}(s, a)$ with parameters θ_{RL}.

Diversity represents the variety in a recommended list of songs. Several generated playlists in Chen *et al.*'s work [4] are composed of songs with the same artist or album. It is not regarded as a good playlist for recommendation system because of low diversity. Therefore, we formulate the measurement of the diversity by the euclidean distance between the embeddings of the last song in the existing playlist, $w^s_{|s|}$, and the predicted song, a.

$$\mathcal{R}_1(s, a) = -\log(|d(w^s_{|s|}, a) - C_{\text{distance}}|) \quad (13)$$

where $d(\cdot)$ is the euclidean distance between the embeddings of $w^s_{|s|}$ and a, and C_{distance} is a parameter that represents the euclidean distance that we want the model to learn.

Novelty is also important for a playlist generation system. We would like to recommend something new to users instead of recommend something familiar. Unlike previous works, our model can easily generate playlists with novelty by applying a corresponding reward function. As a result, we model reward of novelty as a weighted sum of normalized playcounts in periods of time [19].

$$\mathcal{R}_2(s, a) = -\log(\sum_t w(t)\frac{\log(p_t(a))}{\log(\max_{a' \in A} p_t(a'))}) \quad (14)$$

where $w(t)$ is the weight of a time period, t, with a constraint $\sum_t w(t) = 1$, $p_t(a)$ is playcounts of the song a, and A is the set of actions. Note that songs with less playcounts have higher value of $\mathcal{R}_2$, and vice versa.

Freshness is a subjective metric for personalized playlist generation. For example, latest songs is usually more interesting for young people, while older people would prefer old-school songs. Here, we arbitrarily choose one direction for optimization to the agent $\pi_{\theta_{RL}}$ to show the feasibility of our approach.

$$\mathcal{R}_3(s, a) = -\log(\frac{Y_a - 1900}{2017 - 1900}) \quad (15)$$

where Y_a is the release year of the song a.

Coherence is the major feature we should consider to avoid situations that the generated playlists are highly rewarded but lack of cohesive listening experiences. We therefore consider the policy of our pretrained language model, $\pi_{\theta_{RNN}}(s, a)$, which is well-trained on coherent playlists, as a good enough generator of coherent playlists.

$$\mathcal{R}_4(s, a) = \log(\Pr[a|s, \theta_{RNN}]) \quad (16)$$

Combining the above reward functions, our final reward for the action a is

$$\begin{aligned}\mathcal{R}(s, a) =& \gamma_1\mathcal{R}_1(s, a) + \gamma_2\mathcal{R}_2(s, a) + \\ & \gamma_3\mathcal{R}_3(s, a) + \gamma_4\mathcal{R}_4(s, a)\end{aligned} \quad (17)$$

where the selection of γ_1, γ_2, γ_3, and γ_4 depends on different applications.

Note that although we only list four reward functions here, the optimization goal $\mathcal{R}$ can be easily extended by a linear combination of more reward functions.

5. EXPERIMENTS AND ANALYSIS

In the following experiments, we first introduce the details of dataset and evaluation metrics in Section 5.1 and training details in Section 5.2. In Section 5.3, we compare pretrained RNN-LMs with different mechanism combination by perplexity to show our proposed AC-RNN-LM is more effectively and efficiently than others. In order to demonstrate the effectiveness of our proposed method, AC-RNN-LM combined with reinforcement learning, we adopt three standard metrics, diversity, novelty, and freshness (cf. Section 5.1) to validate our models in Section 5.4. Moreover, we demonstrate that it is effortless to flexibly manipulate the properties of resulting generated playlists in Section 5.5. Finally, in Section 5.6, we discuss the details about the design of reward functions with given preferred properties.

5.1 Dataset and Evaluation Metrics

The playlist dataset is provided by KKBOX Inc., which is a regional leading music streaming company. It consists of $10,000$ playlists, each of which is composed of 30 songs. There are $45,243$ unique songs in total.

For validate our proposed approach, we use the metrics as follows.

Perplexity is calculated based on the song probability distributions, which is shown as follows.

$$perplexity = e^{\frac{1}{N}\sum_{n=1}^{N}\sum_x -q(x)\log p(x)}$$

where N is the number of training samples, x is a song in our song pool, p is the predicted song probability distribution, and q is the song probability distribution in ground truth.

Diversity is computed as different unigrams of artists scaled by he total length of each playlist, which is measured by Distinct-1 [9]

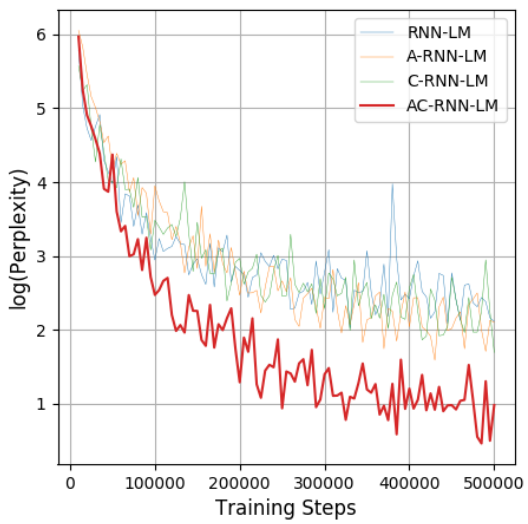

Figure 2. Log-perplexity of different pretrained models on the dataset under different training steps

Novelty is designed for recommending something new to users [19]. The more the novelty is, the lower the metric is.

Freshness is directly measured by the average release year of songs in each playlist.

5.2 Training Details

In the pretraining and reinforcement learning stage, we use 4 layers and 64 units per layer in all RNN-LM with LSTM units, and we choose $T = 30$ for all RNN-LM with padding and concatenation. The optimizer we use is Adam [8]. The learning rates for pretraining stage and reinforcement learning stage are empirically set as 0.001 and 0.0001, respectively.

5.3 Pretrained Model Comparison

In this section, we compare the training error of RNN-LM combining with different mechanisms. The RNN-LM with attention is denoted as A-RNN-LM, the RNN-LM with concatenation is denoted as C-RNN-LM, and the RNN-LM with attention and concatenation is denoted as AC-RNN-LM. Figure 2 reports the training error of different RNN-LMs as log-perplexity which is equal to negative log-likelihood under the training step from 1 to $500,000$. Here one training step means that we update our parameters by one batch. As shown in Figure 2, the training error of our proposed model, AC-RNN-LM, can not only decrease faster than the other models but also reach the lowest value at the end of training. Therefore, we adopt AC-RNN-LM as our pretrained model.

Worth noting that the pretrained model is developed for two purposes. One is to provide a good basis for further optimization, and the other is to estimate transition probabilities of songs in the reward function. Therefore, we simply select the model with the lowest training error to be optimized by policy gradient and an estimator of $\Pr[a|s, \theta_{RNN}]$ (cf. Eqn (16)).

Table 1. Weights of reward functions for each model

Model	γ_1	γ_2	γ_3	γ_4
RL-DIST	0.5	0.0	0.0	0.5
RL-NOVELTY	0.0	0.5	0.0	0.5
RL-YEAR	0.0	0.0	0.5	0.5
RL-COMBINE	0.2	0.2	0.2	0.4

Table 2. Comparison on different metrics for playlist generation system (The bold text represents the best, and the underline text represents the second)

Model	Diversity	Novelty	Freshness
Embedding [4]	0.32	0.19	2007.97
AC-RNN-LM	0.39	0.20	2008.41
RL-DIST	<u>0.44</u>	0.20	2008.37
RL-NOVELTY	0.42	**0.05**	2012.89
RL-YEAR	0.40	0.19	**2006.23**
RL-COMBINE	**0.49**	<u>0.18</u>	<u>2007.64</u>

5.4 Playlist Generation Results

As shown in Table 2, to evaluate our method, we compare 6 models on 3 important features, diversity, novelty, and freshness (cf. Section 5.1), of playlist generation system. The details of models are described as follows. Embedding represents the model of Chen *et al.*'s work [4]. Chen *et al.* construct the song embedding by relationships between user and song and then finds approximate k nearest neighbors for each song. RL-DIST, RL-NOVELTY, RL-YEAR, and RL-COMBINE are models that are pretrained and optimized by the policy gradient method (cf. Eqn (17)) with different weights, respectively, as shown in Table 1.

The experimental results show that for single objective such as diversity, our models can accurately generate playlists with corresponding property. For example, RL-Year can generate a playlist which consists of songs with earliest release years than Embedding and AC-RNN-LM. Besides, even when we impose our model with multiple reward functions, we can still obtain a better resulting playlist in comparison with Embedding and AC-RNN-LM. Sample result is shown in Figure 3.

From Table 2, we demonstrate that by using appropriate reward functions, our approach can generate playlists to fit the corresponding needs easily. We can systematically find more songs from different artists (RL-DIST), more songs heard by fewer people (RL-NOVELTY), or more old songs for some particular groups of users (RL-YEAR).

5.5 Flexible Manipulating Playlist Properties

After showing that our approach can easily fit several needs, we further investigate the influence of γ to the re-

Figure 3. Sample playlists generated by our approach. The left one is generated by Embedding [4] and the right one is generated by RL-COMBINE.

sulting playlists. In this section, several models are trained with the weight γ_2 from 0.0 to 1.0 to show the variances in novelty of the resulting playlists. Here we keep $\gamma_2 + \gamma_4 = 1.0$ and $\gamma_1 = \gamma_3 = 0$ and fix the training steps to $10,000$.

As shown in Figure 4, novelty score generally decreases when γ_2 increases from 0.0 to 1.0 but it is also possible that the model may sometimes find the optimal policy earlier than expectation such as the one with $\gamma_2 = 0.625$. Nevertheless, in general, our approach can not only let the model generate more novel songs but also make the extent of novelty be controllable. Besides automation, this kind of flexibility is also important in applications.

Take online music streaming service as an example, when the service provider wants to recommend playlists to a user who usually listens to non-mainstream but familiar songs (i.e., novelty score is 0.4), it is more suitable to generate playlists which consists of songs with novelty scores equals to 0.4 instead of generating playlists which is composed of 60% songs with novelty scores equals to 0.0 and 40% songs with novelty scores equals to 1.0. Since users usually have different kinds of preferences on each property, to automatically generate playlists fitting needs of each user, such as novelty, becomes indispensable. The experimental results verify that our proposed approach can satisfy the above application.

5.6 Limitation on Reward Function Design

When we try to define a reward function $\mathcal{R}_i$ for a property, we should carefully avoid the bias from the state s. In other words, reward functions should be specific to the corresponding feature we want. One common issue is that

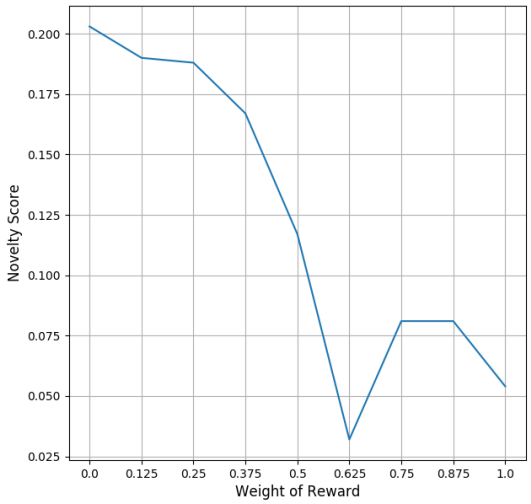

Figure 4. Novelty score of playlists generated by different imposing weights

the reward function may be influenced by the number of songs in state s. For example, in our experiments, we adopt Distinct-1 as the metric for diversity. However, we cannot also adopt Distinct-1 as our reward function directly because it is scaled by the length of playlists which results in all actions from states with fewer songs will be benefited. Therefore, difference between cR_1 and Distinct-1 is the reason that RL-DIST does not achieve the best performance in Distinct-1 (cf. Table 1). In summary, we should be careful to design reward functions, and sometimes we may need to formulate another approximation objective function to avoid biases.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we develop a playlist generation system which is able to generate personalized playlists automatically and flexibly. We first formulate playlist generation as a language modeling problem. Then by exploiting the techniques of RNN-LM and reinforcement learning, the proposed approach can flexibly generate suitable playlists for different preferences of users.

In our future work, we will further investigate the possibility to automatically generate playlists by considering qualitative feedback. For online music streaming service providers, professional music curators will give qualitative feedback on generated playlists so that research developers can improve the quality of playlist generation system. Qualitative feedback such as 'songs from diverse artists but similar genres' is easier to be quantitative. We can design suitable reward functions and generate corresponding playlists by our approach. However, other feedback such as 'falling in love playlist' is more difficult to be quantitative. Therefore, we will further adopt audio features and explicit tags of songs in order to provide a better playlist generation system.

7. REFERENCES

[1] Masoud Alghoniemy and Ahmed H. Tewfik. A network flow model for playlist generation. In *Proceedings of International Conference on Multimedia and Expo*, pages 329–332, 2001.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[3] Rachel M. Bittner, Minwei Gu, Gandalf Hernandez, Eric J. Humphrey, Tristan Jehan, P. Hunter McCurry, and Nicola Montecchio. Automatic playlist sequencing and transitions. In *Proceedings of the 18th International Conference on Music Information Retrieval*, pages 472–478, 2017.

[4] Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. Query-based music recommendations via preference embedding. In *Proceedings of the ACM Conference Series on Recommender Systems*, pages 79–82, 2016.

[5] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In *Proceedings of the 9th International Society for Music Information Retrieval Conference*, pages 173–178, 2008.

[6] Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-aware music recommendation based on latent-topic sequential patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 131–138, New York, NY, USA, 2012. ACM.

[7] Dietmar Jannach, Lukas Lerche, and Iman Kamehkhosh. Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 187–194, New York, NY, USA, 2015. ACM.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *CoRR*, abs/1510.03055, 2015.

[10] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541, 2016.

[11] Elad Liebman and Peter Stone. DJ-MC: A reinforcement-learning agent for music playlist recommendation. *CoRR*, abs/1401.1880, 2014.

[12] Beth Logan. Content-based playlist generation: Exploratory experiments. 2002.

[13] François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, 2009.

[14] Brian McFee and Gert Lanckriet. The natural language of playlists. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 537–542, 2011.

[15] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. Coherent dialogue with attention-based language models. In *The Thirty-First AAAI Conference on Artificial Intelligence*, 2016.

[16] Elias Pampalk, Tim Pohle, and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. In *Proceedings of the 6th International Society for Music Information Retrieval Conference*, pages 634–637, 2005.

[17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 2017.

[18] Saúl Vargas. New approaches to diversity and novelty in recommender systems. In *Proceedings of the Fourth BCS-IRSG Conference on Future Directions in Information Access*, FDIA'11, pages 8–13, Swindon, UK, 2011. BCS Learning & Development Ltd.

[19] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: Introducing serendipity into music recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 13–22, New York, NY, USA, 2012. ACM.

[20] Xinxi Wang Zhe Xing and Ye Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 445–450, 2014.