Multimedia Tools and Applications
https://doi.org/10.1007/s11042-019-08356-9

Music recommender using deep embedding-based features and behavior-based reinforcement learning

Jia-Wei Chang[1] · Ching-Yi Chiou[2] · Jia-Yi Liao[1] · Ying-Kai Hung[1] · Chien-Che Huang[1] · Kuan-Cheng Lin[2] · Ying-Hung Pu[1]

Received: 24 May 2019 / Revised: 16 August 2019 / Accepted: 9 October 2019
Published online: 27 December 2019

Abstract
With the rapid increase of digital music on online music platforms, it has become difficult for users to find unknown but interesting songs. Although many collaborative filtering or content based recommendation methods have been proposed, they have various relatively serious some problems, including cold start, diversity of recommendations. etc. Therefore, we propose a reinforcement personal music recommendation system (RPMRS) to address these problems. RPMRS comprises two main components. First, deep representation of audio and lyrics extracted by WaveNet and Word2Vec models, respectively, and apply a proposed content based recommendation method from these. Second, we employ reinforcement learning is to learn user preferences from their song playing log. Experimental results confirm, that hybrid features are superior to audio or lyrics based features for content recommendation, largely because independent audio features significantly outperform lyrics features; and reinforcement learning improves personalized recommendations. Overall, the proposed RPMRS provides dynamic and personalized music recommendations for the user.

Keywords Music recommendation · Content based recommendation · Reinforcement learning

1 Introduction

A huge amount of digital music content has become available due to web technology advancements, e.g. Spotify, KKBOX, Apple iTunes, Amazon MP3 store, etc. Modern music listeners also have easier access to this enormous online music ensemble. However, users have widely different music tastes and preferences in different contexts or periods, i.e., user music preference is strongly dynamic. Although current music recommenders provide sing categories

✉ Ying-Hung Pu
yinghong.pu@gmail.com

[1] National Taichung University of Science and Technology, Taichung City, Taiwan

[2] National Chung Hsing University, Taichung City, Taiwan

or labels to help people select songs of interest, these are not personalized recommendations. Therefore, users find it difficult to discovering music they like among such an abundant music pool. Thus, a better recommendation approach is required to provide more accurate music selection, where the recommendation considers previous user behavior and actively provides song list they would enjoy.

Recommender systems aim to suggest accurate personalized recommendations for the user, but important recommendation factors include diversity and novelty [1]. Personalized means that the recommendations are suggested by considering user preferences, including previously heard songs and music listening behavior. Diversity means that users tend to prefer recommendation lists with high diversity, or intra-list similarity (ILS), i.e., ideal music recommendations should include different singers or music types. However, user preferences change over time and depend on multiple factors, including music style, tempo, lyrics, etc. All these factors influence user satisfaction with the recommendations.

Traditional music recommendation systems usually generate recommendation lists based on music popularity. However, such systems tend to suggest popular items rather than user preferences. Several user preference features should be considered from user prior music experience to provide personalized requirement, recommendations, including user information [2], music content [3, 4], and user playing history [5, 6]. Recommendation approaches can be broadly divided into collaborative filtering (CF) and content based methods. Although CF approaches can identify relationships among users and music items based on ratings or playing history from all users, they have a significant cold start problem, i.e., new items rarely occur in recommendations because they do not have many user ratings or interactions. In contrast, content based approaches can provide recommendations based on meta-data, such as album, genre, style, artist, audio, and lyrics; but they cannot provide dynamic recommendations over time.

This paper addresses previous approach problems, and propose the reinforcement personal music recommendation system (RPMRS). The RPMRS combines content based methods using audio and lyrics features with reinforcement learning (RL). Since considering popular music elements may capture user interest more precisely, abundant music content features may improve content based music recommendations. Data types for a popular song include acoustical, temporal, and textual features, i.e., tempo, intensity, lyrics, etc. On the other hand, RL can learn user preferences from their playing history. Thus, RPMRS dynamically suggests songs the user will like by analyzing recent playing history. The main contributions of this paper can be summarized as follows.

- We apply a WaveNet based model to extract deep audio and Word2Vec to extract deep lyrics features.
- We propose a content based recommendation method that integrates audio and lyrics features.
- We propose a reinforcement learning framework based on historical user records and music listening behavior to provide personalized music recommendation. Thus, the proposed RL mechanism provides dynamic and personalized recommendations.

The remainder of this paper is organized as follows. Section 2 discusses previous content based music recommendation, feature extraction, and reinforcement learning studies. Section 3 describes the proposed system, including system architecture, feature extraction, and recommendation algorithm. Section 4 discusses the experimental design and results. Finally, Sect. 5 summarizes and concludes the paper.

2 Related works

2.1 Content based music recommendation

Traditional music recommenders have been primarily based on collaborative filtering (CF). However, their effectiveness has been limited by insufficient information, including sparse rating data and lack of contextual information. Sparse rating data frequently occurs in real applications and can significantly distort recommendation lists. Recent studies have incorporated other information sources into the recommendation pipeline to mitigate some of these problems, e.g. a user may prefer an item is based on its content, rather than considering whether other users' ratings for the item are high.

Content based recommendation approaches analyze sets of documents and/or item descriptions, and build a model or profile of user interests based on object features. Hence, music can be recommended based on available metadata, such as album, artist, song type, year of release, audio features, etc. Most current content based methods extract traditional audio features first and then recommend based on similarities between song feature vectors. Content based approaches have some advantages.

1. User independence: each user's profile is based on their own preferences for the item.
2. Transparency: explanations for how the recommender system works can be provided from the content features.
3. New item problem: recommender systems that use overall user preferences are susceptible to the cold start problem, since the system needs time to incorporate user preferences.

However, feature limitation is a major content based recommender system drawback. If the content does not contain enough information to discriminate the items, the final recommendation list will be imprecise.

Liang et al. [6] proposed a content-aware collaborative music recommendation system that combined multi-layer neural network content and collaborative filtering models. They showed the proposed system achieved good match with user preference for music recommendation experimentally for music content and implicit user feedback. Gao et al. [7] used previous content information from location based social networks (LBSNs) for points of interest (POI) recommendations to model a recommendation framework that helped users to filter uninteresting POIs for decision making. Lu et al. [8] proposed a content based collaborative filtering approach for news recommendation, combining content based and collaborative filtering models. Soares et al. [9] studied how to use different metadata elements to help improve recommendation quality.

2.2 Deep embedding for audio features

The music listening experience is multidimensional, with different emotion perceptions commonly associated with different acoustic cue patterns [7, 8]. Feature extraction is a very important aspect of analyzing and finding relationships between different things. An audio signal can be considered as three-dimensional, with time, frequency, and amplitude axes, as shown in Fig. 1.

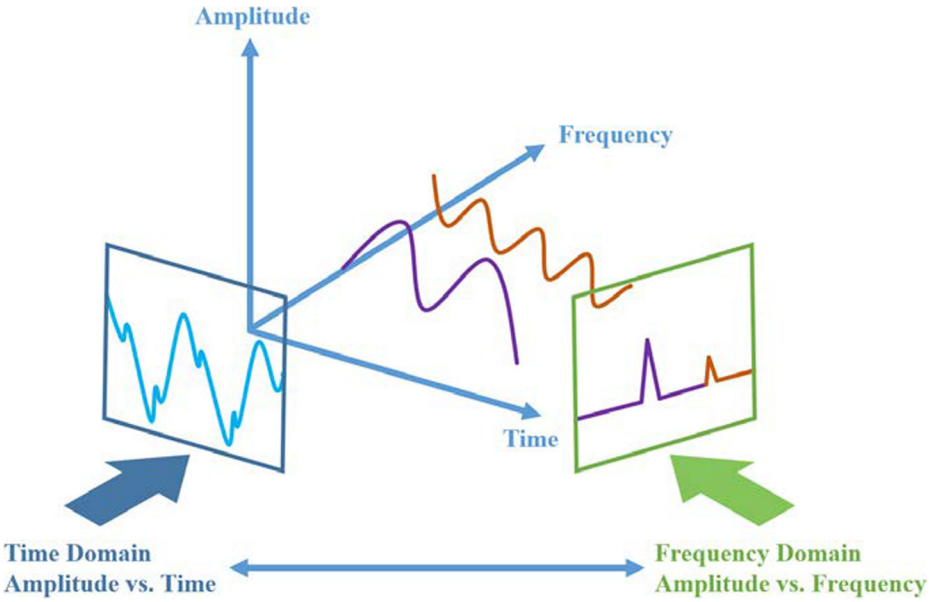


Fig. 1 Common audio signal features

The three main musical elements include rhythm, intensity and timbre.

- Rhythm is the skeleton of music, strong beats and weak beats are repeated periodically and regularly to form the structure. Yeh et al. [10] proposed a method to discriminate rhythm regularity, using a detection function to calculate spectral differences for adjacent frames, and confirm rhythm regularity from peak distance standard deviation.
- Intensity is the music strength in a score or performance, which is determined by vibration amplitude, and is also known as loudness or dynamics. Intensity is also the energy sum in the frequency domain, summing Fourier coefficients for each frame.
- Timbre is determined by the waveform, which can differ for sounds with the same pitch and loudness. Commonly used timbre features include the Mel-frequency cepstral coefficients (MFCC),

$$MFCC_m = \sqrt{\frac{2}{N}}\sum_{k=1}^{N} M_m(k)\cos\left(\frac{\pi m}{M}(k-0.5)\right) \tag{1}$$

Tzanetakis et al. [11] and Peeters [12] used classical MFCC audio features as input to classify emotions using support vector machine (SVM). Mokhsin et al. [13] proposed extracting that timbre features using spectral roll off, spectral centroid, and zero-crossing rate. Many different approaches have been proposed for a wide range of datasets and features. Kim et al. [14] collated an significant portion of these disparate approaches, and Li et al. [15] used signal processing features related to timbre, pitch, and rhythm.

Neural networks can learn useful data representations supervised or unsupervised. In the case of supervision, features learned on large datasets are often used directly for other similar tasks that have poor data. Feature learning and deep learning are often conflated, but not all feature learning is necessarily deep; it is possible to learn shallow (i.e., single layer) data

representations, and this can offer significant advantages in terms of computation cost. This paper focused on unsupervised feature learning, i.e., where no training labels are available. Therefore, we employed an autoencoder, which use an encoding network to extract a latent representation that preserves prominent features of the original dataset.

WaveNet [16] is an autoregressive generative model for audio waveforms that has yielded state-of-art performance in text-to-speech synthesis. It can efficiently train data with tens of thousands of samples of audio per second and can also learn a multi-speaker model with conditional WaveNet. Therefore, we can learn embedding for each speaker. The WaveNet network can also be used to synthesize other audio signals, such as music. Figure 2 shows a typical WaveNet architecture.

However, although WaveNet deals well with on short range signals it relies on external conditions to achieve longer term dependencies. Therefore, Google Brain and DeepMind co-released the NSynth neural synthesizer based on the WaveNet concept [17], shown in Fig. 3. NSynth learns temporal embedding using a WaveNet-like autoencoder, removing the need for conditioning with external features. Autoencoders are just simple neural networks that are often employed for unsupervised learning. Their goal is to learn efficient encoding for a given dataset, usually for dimensionality reduction, and they consist of an encoder and decoder (Fig. 3). The goal of an automatic encoder is usually to compress the input to a smaller hidden variable, e.g. Z in Fig. 3 is a low dimensional vector, a function of the input audio.

Magenta[1] took this generative model, turned it into an autoencoder, and call resultant network NSynth. Their method utilized a pre-trained NSynth model to encode some audio tracks.[2] The encoding process can be viewed as three components:

1. segment the audio signal into 16 channels,
2. return a 3-dimensional tensor representing the encoded audio, and
3. consider these as 16 sound channels mixed together.

Figure 4 shows an example drum beat, where the encoding follow the overall audio signal pattern well.

2.3 Deep embedding for texts

Many natural language processing (NLP) applications [18–20] use semantic similarity measures [21, 22], and semantic representation techniques. Emerging NLP trends adopt semantic representation by word embedding methods to achieve better performance. Mikolov et al. [23] introduced Word2Vec 2013 at Google for efficient vector representations of words, and it has become one of the most popular techniques to learn word embedding using shallow neural networks. Word2Vec can be trained by either the continuous bag of words (CBOW) or skip gram models to capture contextual information and semantic relationships between words. Figure 4 shows the current word, $w(t)$, and context words, $w(t-2)$, $w(t-1)$, $w(t+1)$, $w(t+2)$. Given a set of sentences (also called corpus), the CBOW model takes the context for each

[1] Note: cite in Magenta | Make Music and Art Using Machine Learning. (Date Published: 2017, April 05). Retrieved from https://magenta.tensorflow.org/

[2] Pre-trained WaveNet model, downloadable from NSynth – Neural Audio Synthesis: http://download.magenta.tensorflow.org/models/nsynth/wavenet-ckpt.tar

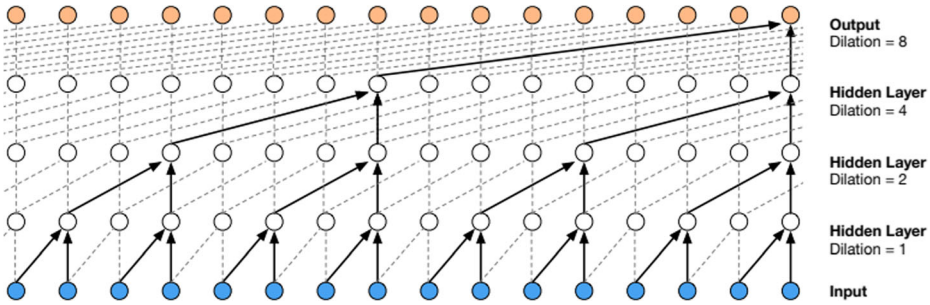


Fig. 2 NSynth architecture

word as input and tries to predict the word; whereas the skip gram model takes the current word as input and tries to predict the context.

Due to its effective capturing of object correlations, Word2Vec has been successfully applied to various applications, including feature embedding, user modeling, music/video recommendation, verbatim comment analysis, etc.

We focused on word embedding for lyrics. Previous song lyrics classification studies have used traditional frequency based text representation. However, experience driven word embedding has achieved improved performance for text classification. Choi et al. [24] proposed vector representation, an alternative and advanced method to extract higher level features from lyrics, rather than the traditional term-frequency (TF) representation. Vector representation can encode co-occurrence information for words spread across many lyrics. Delbouys et al. [25] considered multimodal music emotion prediction based on audio signals and lyrics: audio features used Mel-spectrogram input, and lyrics features used word embedding input to the network, i.e., each word was embedded in a continuous space and the vectors corresponding to each word were stacked.

2.4 Reinforcement learning

Reinforcement learning (RL) is a type of machine learning algorithm. In contrast to supervised learning, which considers only prescribed training data, the algorithm actively explores the environment to gather information many times and uses this information to make decisions or

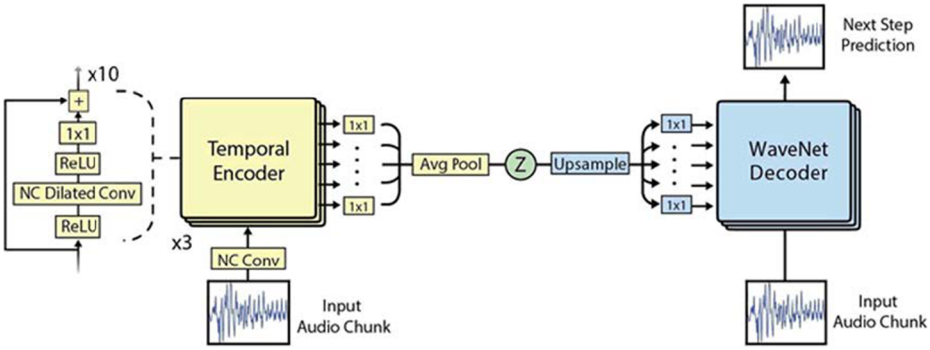


Fig. 3 NSynth architecture

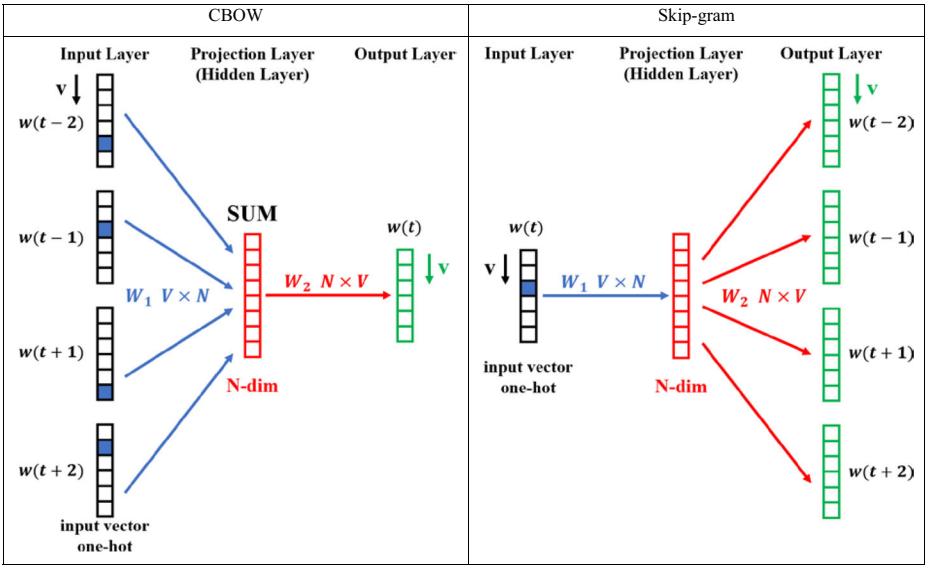


Fig. 4 CBOW and Skip-gram

predictions. It differs from other machine learning methods because the algorithm does not explicitly specify how to perform the task but solves the problem separately. RL can be understood using agent, environment, state, action, and reward concepts, as shown in Fig. 5.

Watkins [26] proposed Q-Learning, a simple RL off-policy method, i.e., it can learn different policies for behavior and estimation. The agent can be viewed as a brain, environment is the physical world the agent operates within, state is the agent's current situation, reward is feedback from the environment, and policy is the method to map the agent's state to actions.

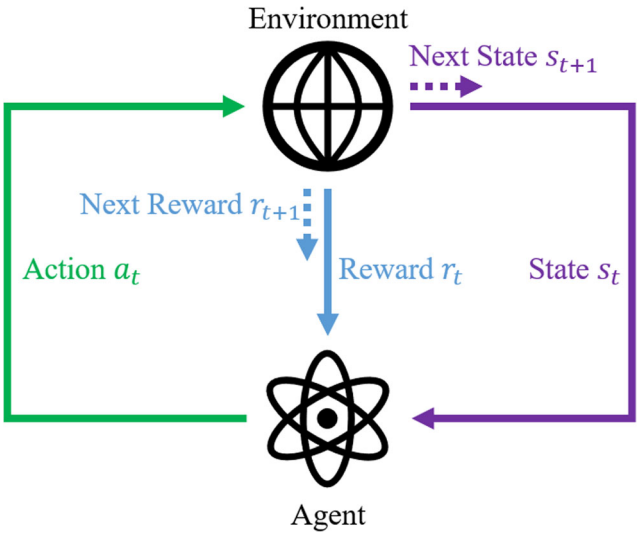


Fig. 5 Reinforcement learning concepts

The agent will take action depending on the current state, and record resulting rewards (feedback), so the process can improve the next time while still achieving the same end state. We create a Q-table for Q-Learning, presenting state and action values, then initialize these values to zero. Each time an action is executed, it updates s_t, a_t, r_{t+1} and s_{t+1}, and this Q-table becomes a reference table for the agent to select maximum Q, $Q(s_t, a_t)$ for the best action. Since a_{t+1} is always the best action, we can use $Q(s_t, a_t)$ to measure how well the current states act. Thus, the value update rule is the core of the Q-learning algorithm [30],

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \lambda \, maxQ(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{2}$$

where s_t is the agent's state at time t; a_t is the agent action at time t; r_{t+1} is the reward for state s that actions a; the user provides feedback, i.e., listen, not listen, click or not click; α is the learning rate ($0 \leq \alpha < 1$), which can be defined as the acceptance of new and old values, i.e., $\alpha = 0$ means the agent won't learn anything (old information is important), and $\alpha = 1$ means that newly discovered information is the only important information; λ is the discount rate ($0 \leq \lambda < 1$) to measure the importance of future rewards, $\lambda = 0$ means that only short-term rewards are considered, $\lambda = 1$, means paying more attention to long-term rewards [27]. The value function (referred to as Q-values) is stored in a Q-matrix.

Reinforce learning techniques can be divided into policy and value based approaches. For example, Q-Learning is value based. Zhao et al. [28] considered positive and negative user feedback to boost recommendations. Taghipour et al. [29] modeled web page recommendation as a Q-Learning problem and learned to make recommendations from web usage data. Srivihok et al. [30] proposed travel recommender based on Q-learning that considered several attributes, including trip duration, price, and country, ranked trips using a linear function, and the updated trip weights from user feedback. Zheng et al. [31] proposed an RL framework for news recommendation, using deep Q-Learning, based on user feedback (in this case whether the user clicked).

3 Methodology

3.1 System architecture

Most current recommender systems employ the collaborative filtering model. However, these systems are vulnerable to the cold-start problem, discussed above, which makes it difficult to provide reliable recommendations without sufficient information regarding new items or users. Therefore, we propose a content based music recommendation framework using audio and lyrics features, and RL on user historical records and music listening behaviors. Section 3.2 describes feature extraction methods for popular songs, and Sect. 3.3 present a content based recommendation system with hybrid features. Section 3.4 describes RL implementation for recommendation improvement, and Sect. 3.5 proposes the reinforcement personal music recommendation system (RPMRS) recommendation algorithm. Figure 6 shows the overall proposed framework.

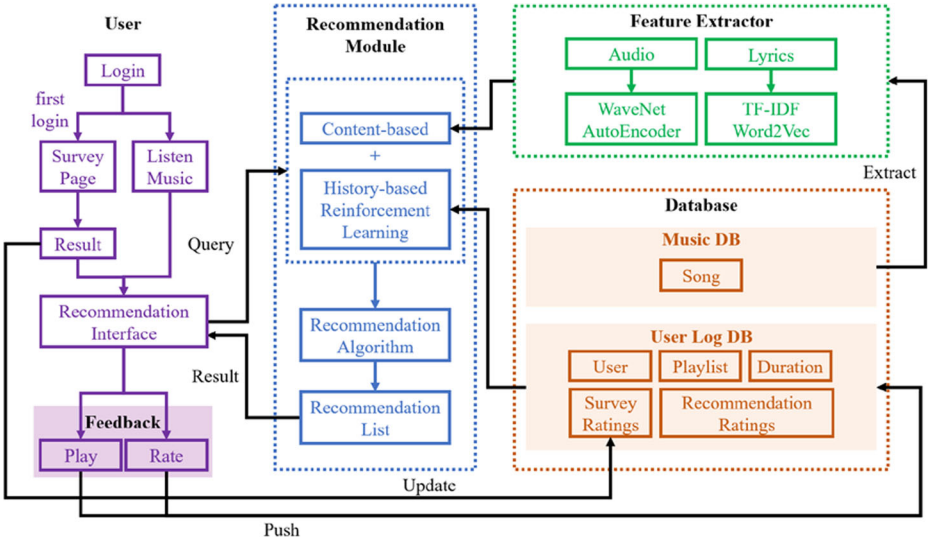


Fig. 6 Proposed reinforcement personal music recommendation system framework

3.2 Feature extraction by deep embedding

Music feature extraction is the crucial part of this study. We propose a content based recommendation model that uses features from the music audio and lyrics text. We use a feature extractor to extract song features into a, as shown in Fig. 7.

3.2.1 Audio feature extraction

We utilize the WaveNet tool created by DeepMind (a Google subsidiary) for audio feature data, which is one of the most lavish and impressive neural networks [16]. WaveNet is an

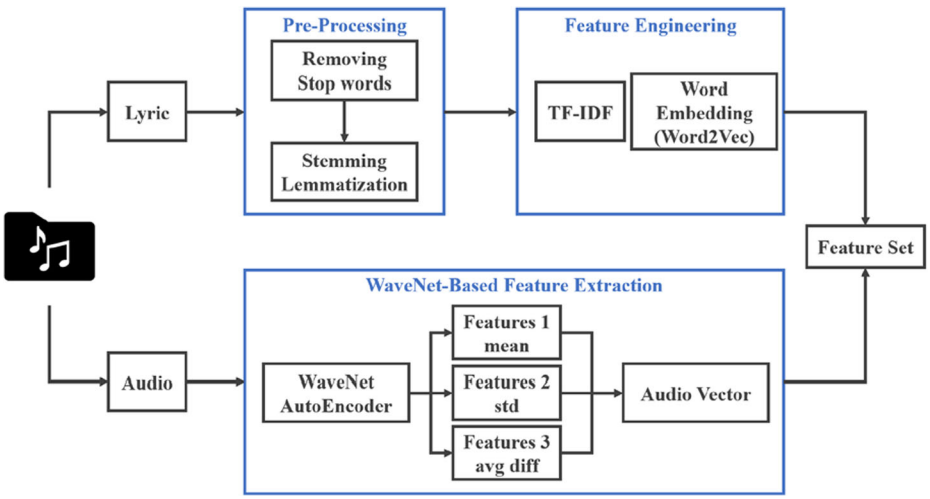


Fig. 7 Features extraction process

autoregressive model that encodes sound waves into embeddings, where sample prediction is influenced by all previous samples. Conditional probability is modelled by a stack of convolutional layers, without pooling layers. Hence network output has the same time dimensionality as the input.

We included the weights from the pre-trained WaveNet model (trained on the NSynth dataset[3]) ᵗo obtain the better performance. NSynth is a large scale and high quality dataset of annotated notes, with each musical note matched with a unique pitch, timbre, pitch, envelope, etc., and various features as shown in Table 1. We loaded WaveNet model weights, loaded and downsampled the audio, and passed the audio through the first half of the autoencoder to obtain a list of latent variables that describe the sound. We then reshape the audio into a single sound, an array of n*16 frames, to obtain a vector sequence of hidden states from the network, compressing the original signal information.

When we calculate features for these samples, using MFCCs or NSynth [17], different sample sizes produce different final feature sequence lengths. This raises the problem to take variable length features, compress them into a series of digital vectors, and finally extract a vector that better describes the sound. The final feature vector for each sound segment will be the following three parts spliced together.

1. Average feature. This provides the average over a feature sequence, i.e., average characteristics for each dimension. MFCCs have average feature dimension = 13, and NSynth = 16.
2. Standard deviation over each dimension obtained in the features. This has the same dimensionality as the average feature, and tells us the extension of the feature distribution.
3. Average difference between adjacent features. This reflects characteristic's average time. Dimensionality matches the average feature.

Finally, we concatenate the features into a single feature vector, hence any sample of any length can be compressed to a fixed length feature. We propose using a WaveNet based network, with feature dimension = 48, multiplied by 16 channels and 3 parts. Figure 8 shows the acoustic features extraction framework.

3.2.2 Lyrics feature extraction

Lyrics can provide additional valuable information regarding the song. Our goal is to detect important words in the text, identifying keywords or phrases that can represent a song. These words are important to training machine learning algorithms. Word embedding provides a dense representation of words and their relative meanings, and can also be learned as part of fitting a neural network on text data. Therefore, we not only use Word2Vec to generate neural word embeddings but also weighting word embeddings by combining Word2Vec vectors with Term frequency - inverse document frequency (TF-IDF) scores for a word during feature extraction from textual feature data. Nouns, verbs, adjectives, conjunctions, etc. may all occur within the lyrics. In

[3] Note: cite in Magenta | The NSynth Dataset. (Date Published: 2017, April 05). Retrieved from https://magenta.tensorflow.org/datasets/nsynth#files

Table 1 NSynth dataset features

Feature	Type	Description
note	int64	Unique integer identifier for the note.
note_str	bytes	Unique string identifier for the note in the format.
instrument	int64	Unique, sequential identifier for the instrument the note was synthesized from.
instrument_str	bytes	Unique string identifier for the instrument this note was synthesized from in the format.
pitch	int64	MIDI pitch in the range [0, 127].
velocity	bytes	MIDI velocity in the range [0, 127].
sample_rate	int64	Samples per second for the audio feature.
audio	int64	List of audio samples represented as floating point values in the range [−1,1].
qualities	int64	Binary vector representing which sonic qualities are present in this note.
qualities_str	bytes	List IDs of which qualities are present in this note selected from the sonic qualities list.
instrument_family	int64	Index of the instrument family this instrument is a member of.
instrument_family_str	bytes	ID of the instrument family this instrument is a member of.
instrument_source	int64	Index of the sonic source for this instrument.
instrument_source_str	bytes	ID of the sonic source for this instrument.

principle all words should be considered, but conjunctions are generally less important. We use the TF-IDF method to weight word importance for the complete song lyrics.

First, we pre-process the lyrics to remove words that add little meaning to a sentence, which can be safely ignored without sacrificing sentence meaning, e.g. “the”, “is”, “have”, “be”, etc. Lyrics commonly use different word forms for grammatical reasons, and there are families of derivationally related words with similar meanings. We reduce derivationally related words to a common base form using stemming and lemmatization, e.g. “the girl’s clothes are different styles” becomes “the girl cloth be different style”.

The TF-IDF score is the weight for each word in terms of its importance within the dataset of documents (song lyrics), where word importance decreases with increasing number of documents it appears in, since it does not identify a specific document. TF-DIF weights typically incorporate two terms as follows.

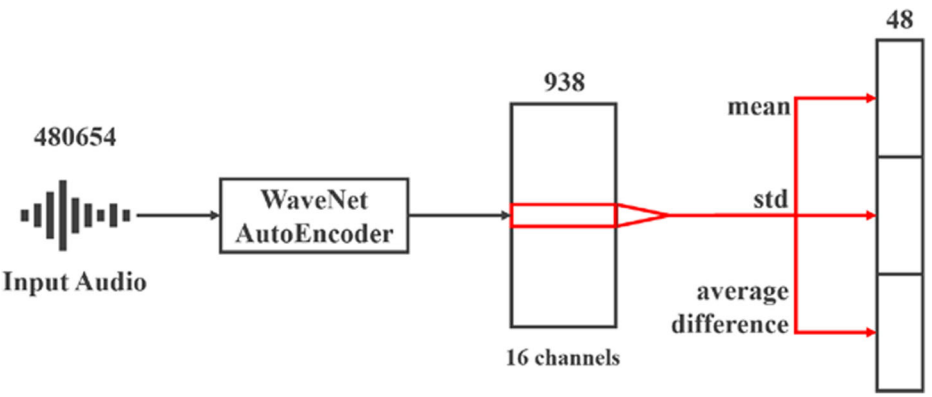


Fig. 8 Acoustic feature extraction framework

4 The normalized term frequency, i.e., the number of times a word appears in a document

$$tf_{i,j} = \frac{f_{i,j}}{\sum_{i' \in j} f_{i',j}}, \tag{3}$$

where i is the term, j is the document, and $f_{i,j}$ is the number of times term i appears in document j. All terms are viewed as equally important when computing tf, but some terms, e.g. "a", "of", "and", "that", etc. may appear many times but have little importance. Therefore, we need to give lower weight for frequent terms and higher weight for rare ones.

5 The inverse document frequency, i.e., how important the term is

$$idf_i = log \frac{N}{df_i}, \tag{4}$$

where J is the set of all documents, N is the total number of documents, and df_i is the number of documents the specific term i appears in.

Once tf and idf have been calculated, we can multiply them to produce a composite TF-IDF weight for each term in each document,

$$w_{i,j} = tf_{i,j} \times idf_i. \tag{5}$$

The TF-IDF process can select words with higher weights and filter common words. Feature dimension after TF-IDF selection is relatively low, but various depth features between these original features need to be further extracted and reduced. Therefore, we employed Word2Vec to convert text into a vector form that deep learning can understand, mapping each word in the lyrics into the vector space with semantically similar words mapped to nearby points. We take the top 30 TF-IDF words for each song and express them as vectors using Word2Vec. Mikolov et al. [23] showed that Word2Vec with 300 dimensions achieves the best performance.

Iyyer et al. [32] proposed deep averaging networks that generate vector representations for input text by averaging word vectors, rather than passing the representation directly to the output layer. We used a similar technique to generate lyrics vectors from the 30 word vectors for a song. We trained the word vectors in 300 dimensions, with each song represented by the average of 30 word vectors,

$$\begin{bmatrix} W_{11} \\ W_{12} \\ \vdots \\ W_{1n} \end{bmatrix} + \begin{bmatrix} W_{21} \\ W_{22} \\ \vdots \\ W_{2n} \end{bmatrix} + \ldots + \begin{bmatrix} W_{n1} \\ W_{n2} \\ \vdots \\ W_{nn} \end{bmatrix} = \begin{bmatrix} \frac{W_{11} + W_{21} + \ldots + W_{n1}}{n} \\ \frac{W_{12} + W_{12} + \ldots + W_{12}}{n} \\ \vdots \\ \frac{W_{1n} + W_{2n} + \ldots + W_{nn}}{n} \end{bmatrix} \tag{6}$$

5.1 Content based recommendation with hybrid features

Content based recommendation can recommend items that have not been rated by anyone but are close music contents users have previously liked. To achieve better recommendations, we tested content based methods using audio, lyrics, and hybrid audio and lyrics features. These features used 48, 300 and 348 (concatenating audio and lyric features) dimensional vectors, respectively, to suggest the 20 most similar songs. We compared the three feature vectors for recommendation satisfaction, and adopted the best model to generate initial recommendations and as the RPMRS similarity measurement.

5.2 Recommendation improvement by reinforcement learning

Recently studies have proposed state of art methods for reinforcement recommenders, but several problems remain.

1. Dynamic changes in music recommendations are difficult to handle. Music is very time-sensitive, and can quickly become obsolete, and individual user listening interest also changes dynamically. Previous studies focused only on static models, whereas the current study focused on dynamic changes in music features and user interests.
2. Previous recommendation systems usually only considered whether the user listened or did not listen to music. In particular, they did not consider how long they listened. Suppose user A listened to music A for 10 s, whereas user B listened to music A for 30 s. This implies user B has higher preference for music A. Thus, we can learn important aspects of user behavior through playing duration history.
3. Current recommendation methods tend to recommend popular items to users, which can decrease user interest in similar music, since users are not necessarily only interested in popular music.

In the standard reinforcement-learning model, an agent is connected to its environment via an action. Considering music recommendation dynamics and the need to estimate future reward, we applied Q-Learning to model the probability that one user may listen on one specific piece of music. Table 2 defines the notations used in the remainder of this paper.

A state, S, is defined as the music playing history for a user, i.e., the user listened to N songs, $s = [\ s_1, s_2, \ldots, s_N\ ] \in S$. An action, A, is to recommend a music list containing K songs based on s, $a = [\ a_1, a_2, \ldots, a_K\ ] \in A$. A reward ($R$) means that

Table 2 Reinforcement learning notations used in this paper

Notation	Meaning
$\hat{A}$	Agent
S	State
A	Action
R	Reward
P	Policy
L	Music recommendation list
F	Users' feedback
Q	Q-Learning

the agent takes action a at state s, $r = [\ r_1, r_2, \ldots, r_K\ ] \in R$, i.e., recommending music list L to the user, the user then listens to the music in L and provides feedback to system. The user can listen or not listen to the music and click the like or dislike button, and the agent receives reward $R = r(s, a)$ according to the user's feedback.

Figure 9 shows the proposed recommendation system incorporating Q-Learning (a simple and efficient model free reinforcement RL algorithm) to provide some memory for the agent. Green arrows represent recommender information flow and blue arrows represent user preference information flow.

To ensure good recommendations, we need to know the relationship between different music, i.e., what the user is interested in. We use user music information for this purpose, such as their playing history. Given the playing history, $\{S, A, R\}$, our goal is to find a recommendation policy $P : S \rightarrow A$, which maximizes total (future) reward. Therefore, the Q-Learning algorithm calculates state-action combination quality, $Q : S \times A \rightarrow R$, where Q are initialized to arbitrary values, and updated while the agent accesses the environment and receives different rewards by applying different actions,

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \, max \, Q'\left(s', a'\right) - Q(s,a) \right], \tag{7}$$

where $Q(s, a)$ is the current values, which are updated every iteration; α is the learning rate; r is the reward for taking action a at state s; and γ is the discount factor, which determine the degree of importance we give to future rewards. For this paper, we set $\alpha = 0.1$ and $\gamma = 0.9$.

The second problem is that previous methods were often short-sighted, i.e., they did not focus on long term rewards, only capturing recent playing history one time, with little or no facility to dynamically capture user playing history. We use user log data to consider long term reward, including how long users played the various songs. We derive 20 recently played

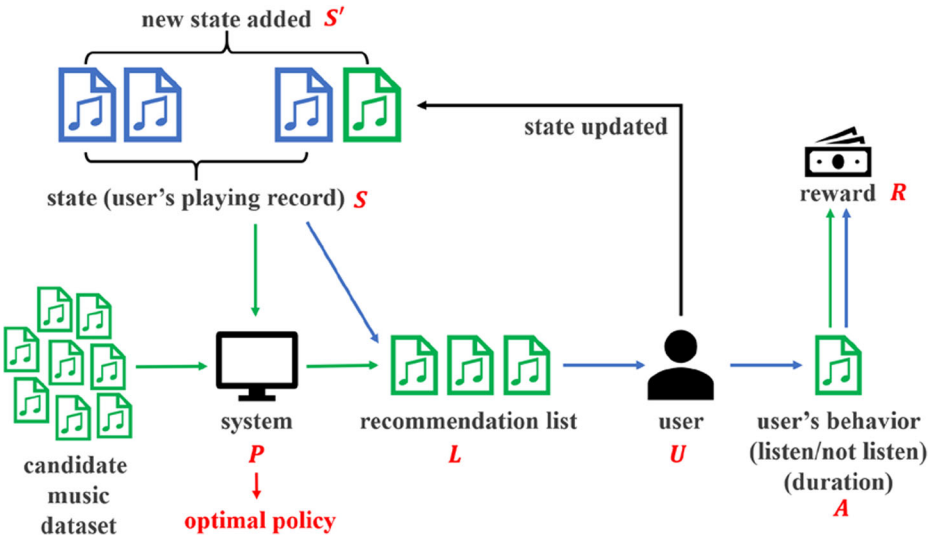


Fig. 9 Proposed recommendation system with Q-Learning

songs from the log data, including listening duration. However, we need to develop standards to determine user behavior. We divided the playing records into 5 levels depending on listening time, as follows.

- duration <5 s: very dislike
- 6 < duration <15 s: dislike
- 16 < duration <20 s: so-so
- 21 < duration <27 s: like
- duration >28 s: very like

Table 3 shows a typical user playing history section.

The proposed RPMRS makes recommendations based on user listening duration. First, we build a Q-table, using states from 20 recent songs and 5 listening levels. We use a ϵ-greedy strategy in the learning process, trading-off between exploration and exploitation. We then estimate Q-values for performing specific behaviors in each state, which not only considers direct, but also cumulative reward for actions. Reward functions describe how the agent "should" behave. We designed the reward mechanism $R(realA, S, A, state\_L)$ for reward r, $realA$ to represent real user listening levels, where S represents the states for 20 recent songs, A represents listening levels from the ϵ-greedy strategy, $state\_L$ represents states where users have "very like" listening. The agent receives reward +1 when $realA = A$, and 0 otherwise. The system then integrates Q-learning as shown in Eq. (2).

Table 3 allows us to discover user preferences. Music 4, 5, 12, 18, and 20 have "very like" listening behavior, i.e., relatively high preferences for these sings recently. Therefore, we give lager reward to those songs. Considering dynamic music recommendations (discussed above) and the need to handle different recent 20 songs, conjecture two circumstances here: the most recent 20 songs contain "very like" songs or they don't. IN the former case, we just need to

Table 3 Example playing history

Time	Music	Listening time (s)	Levels
2019-04-30 23:30	1	5	very dislike
2019-04-30 20:25	2	10	dislike
2019-04-30 22:15	3	12	dislike
2019-04-30 12:30	4	29	very like
2019-04-30 10:10	5	30	very like
2019-04-27 08:20	6	8	dislike
2019-04-27 06:00	7	15	dislike
2019-04-22 23:30	8	9	dislike
2019-04-21 23:30	9	20	so-so
2019-04-21 23:30	10	8	dislike
2019-04-20 15:30	11	11	dislike
2019-04-20 23:30	12	30	very like
2019-04-20 23:30	13	7	dislike
2019-04-19 23:30	14	27	like
2019-04-19 23:30	15	10	dislike
2019-04-18 23:30	16	15	dislike
2019-04-17 23:30	17	21	like
2019-04-17 23:30	18	28	very like
2019-04-17 13:30	19	25	like
2019-04-17 10:00	20	29	very like

pick songs the user likes most (duration >28 s) set the reward mechanism, and let the song state accumulate maximum reward. For the latter circumstance, we should select four songs in each listening level to get 20 recent songs. The recommendation system performance can be continuously optimized through the reward mechanism, ensuring the recommended result aligns with user preferences.

5.3 Recommendation algorithm

Listening to music on the music platform is an interactive process. The music platform recommends a song, and we can choose a series of actions as feedback, including skip, stop, listen, etc. This level of interaction generates many records, hence we can obtain the interaction sequence. We need to create a recommendation list before collecting log data. Therefore, the proposed RPMRS combines content based and reinforcement learning models with feedback, i.e., hybrid personalized music recommendation. The content based model incorporates NLP models to analyze text and audio from raw audio tracks. The RL model analyzes user preferences and provides personalized recommendations using past user behavior records. Thus, we can train the recommender system to recommend songs that match user preferences.

Table 4 shows the notation used in the proposed RPMRS development

Let the set of users be $U = [\ u_1, u_2, u_3, \ldots, u_{|U|}\ ]$, and the set of music be $M = [\ m_1, m_2, m_3, \ldots, m_{|M|}\ ]$, where $|U|$, $|M|$ are the total number of users and music respectively. The proposed RPMRS has three main modules: system database, feature extraction, and recommendation module, as shown in Fig. 10. The database includes music and access history databases, storing music used in the recommendation system and user playing history (including song title, duration, survey results, etc.), respectively. The recommendation module creates a recommendation list when the system receives the request.

Content based recommendation is usually adopted to solve cold-start problems, typically without involving other users. The recommendation algorithm simply picks items with similar content based on user preferences. In contrast the proposed RPMRS recommends music the user may have never heard before. For example, a user may listen to many songs, hence more than one piece of information resides in the playing record. Given a new song, predicting whether the user would listen to this song can be estimated by calculating the similarity between new song features and those for songs in user playing record, which can be represented by a group of features associated with all of songs the user has heard previously.

Table 4 Notation in the proposed system

Notations	Meaning
u, U	User, user set
m, M	Music, music set
s, S	Sequence, sequence set
L	Recommended music list

When user u sends a music request to the recommendation interface, the algorithm selects list L of top-K appropriate music for this user from a candidate music set, M

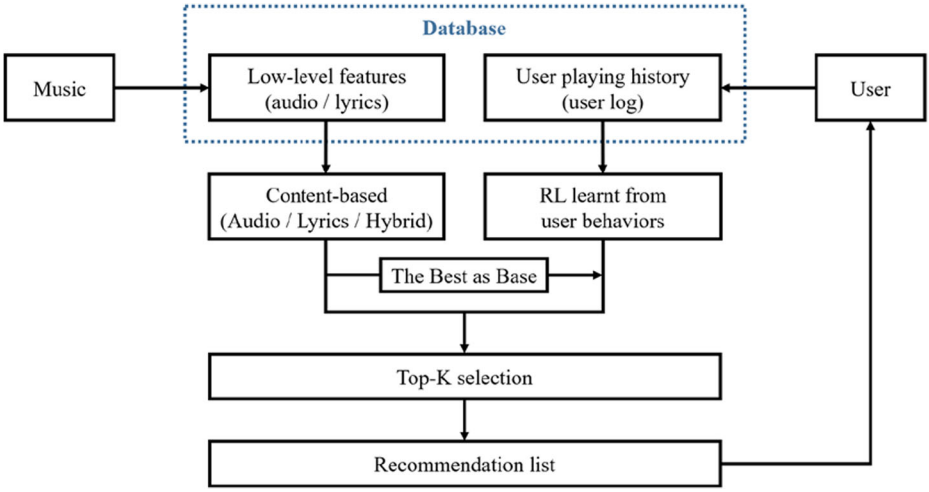


Fig. 10 Proposed recommendation module structure

Therefore, we need a similarity calculation to know which song the user may like. The RPMRS uses Euclidean distance similarity, i.e., the geometric distance between two n-dimensional points. Thus, we consider song feature vectors to calculate geometric distance,

$$\begin{aligned} & D_{Euclidean}\left(Song_x, Song_y\right) \\ & = \sqrt{\left(Song_{x1}-Song_{y1}\right)^2 + \left(Song_{x2}-Song_{y2}\right)^2 + \ldots + \left(Song_{xn}-Song_{yn}\right)^2} \\ & = \sqrt{\sum_{i=1}^{n}\left(Song_{xi}-Song_{yi}\right)^2}. \end{aligned} \tag{8}$$

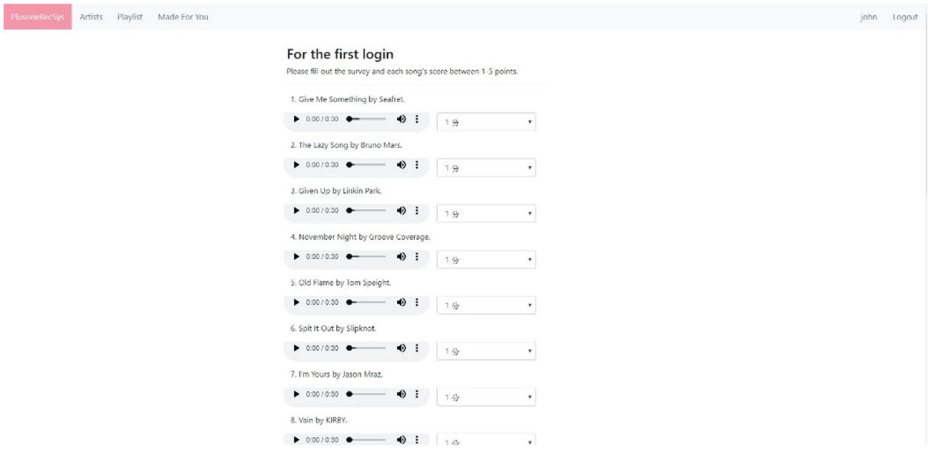


Fig. 11 User initial survey

Fig. 12 Content based recommendation lists

However, historical preference dynamics remain largely undeveloped. We propose to make recommendation integrating the user listening record, i.e., user playing history. Thus, we must collect user activity records before making recommendations, including what kind of music has the user heard, what kind do they like, etc. We will also use user feedback after receiving the recommendation to refine which music is preferred, hence we record user feedback in the user log database. If we use a sequence to represent the playing history of a user, then sequential past can be seen as user state which means the user's past activity record, and sequential future can be seen as reward which means the user's feedback after the recommendation.

Figure 11 shows the user survey we designed to help achieve our goal of recommending potential favored music. Users can rate each recommended song on a 5 point scale. This helps solving the cold start problem, i.e., get user preferences without requiring their access history, and provides a basic user survey. The proposed methods can adapt to user interest variations

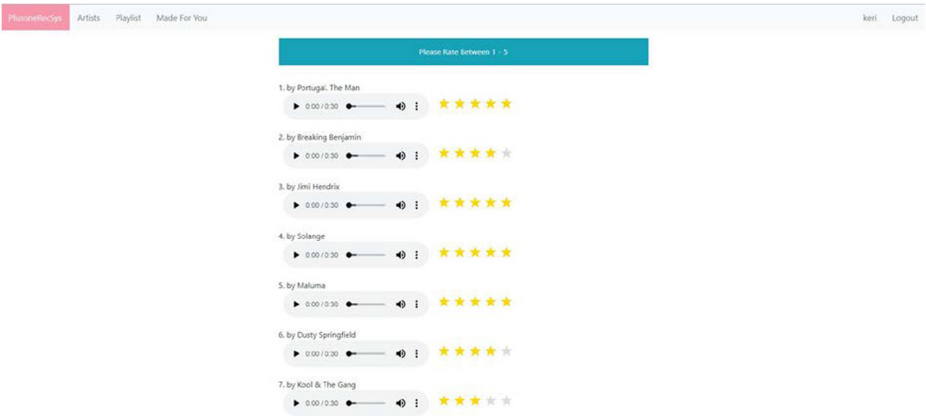


Fig. 13 Rating content based recommendation lists

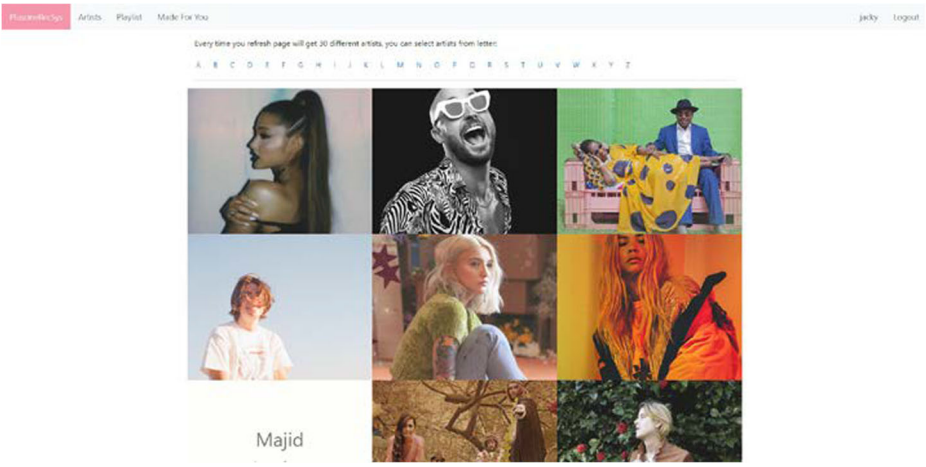


Fig. 14 Singer selection screen

using the survey feedback, and hence promptly avoid recommending disfavored music. The survey collects basic user information and provides 20 initial songs for the user to score. Users need to complete this survey the first time they use the proposed system. The recommendation system incorporates the survey ratings to help provide suitable music recommendations.

Survey ratings and similarities are used to develop recommendations. Larger Euclidean distance implies less similar songs. Therefore, we use calculate the similarity of two songs as

$$Similarity\left(Song_x, Song_y\right) = 1 - D_{Euclidean}\left(Song_x, Song_y\right), \tag{9}$$

and multiply survey rating to obtain a suitable metric, V, to select and rank the top 20 songs for recommendations,

$$V = rating_{u,i} \times Similarity\left(Song_i, Song_j\right) \quad 1 \le i \le 20, 1 \le j \le 20, \tag{10}$$

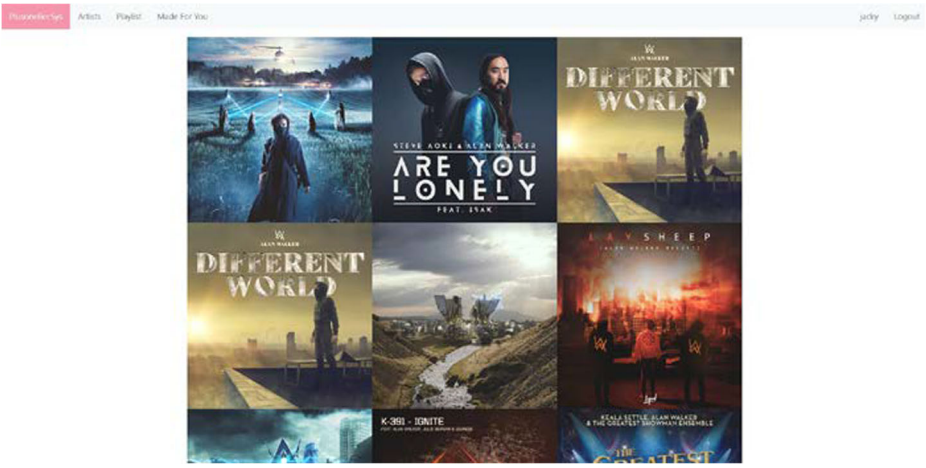


Fig. 15 Album selection screen

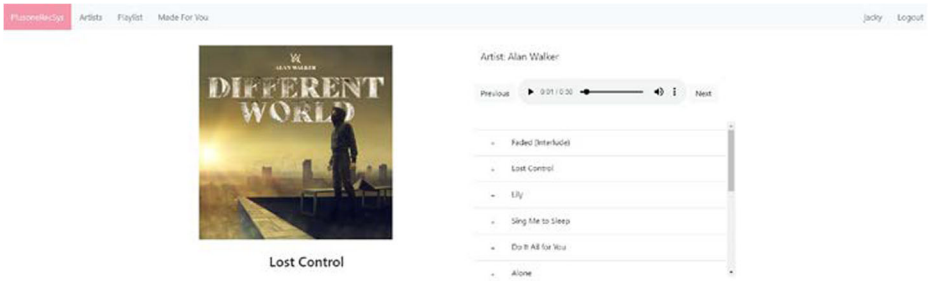


Fig. 16 Song selection screen

where $Song_i$ is the ith survey song from first login, $Song_j$ is the jth song among the top 20 similar songs, $rating_{u,\,i}$ is the rating of $Song_i$ from user u, and $Similarity(Song_i, Song_j)$ is the Euclidean similarity between $Song_i$ and $Song_j$.

We compared three baseline content based methods to separately generate different music recommendation lists, each containing 20 songs. For their first recommendation, users can listen to the music and rate them, as shown in Figs. 12 and 13. The recommendation system web page does not specify which method generated the recommendation to avoid users having preconceived ideas. After rating the songs, the backend system calculates average rating from the three methods.

We select the best content based method as RPMRS base method for more advanced recommendations. Users can search for favorite music based on the singer using the music platform, with access to all albums from the selected singer. The backend system starts recording user playing behavior when a song is selected, for future recommendations. Figures 14, 15 and 16 show screen the user operates on the platform. The proposed RPMRS can use these playing histories to generate suitable music lists once sufficient play records are logged. Q-learning evaluates true rewards for performing each action in each state, providing Q-value lists for each state action. Thus, we need to design a recommendation strategy to choose songs specific users may prefer. The strategy is to provide larger reward for songs users listen longer. The proposed model can recommend a song list using several strategies. The basic concept is to choose songs with maximum Q-value in the "very like" field of the Q-table. We apply nearest-neighbor based method using calculated similarity to

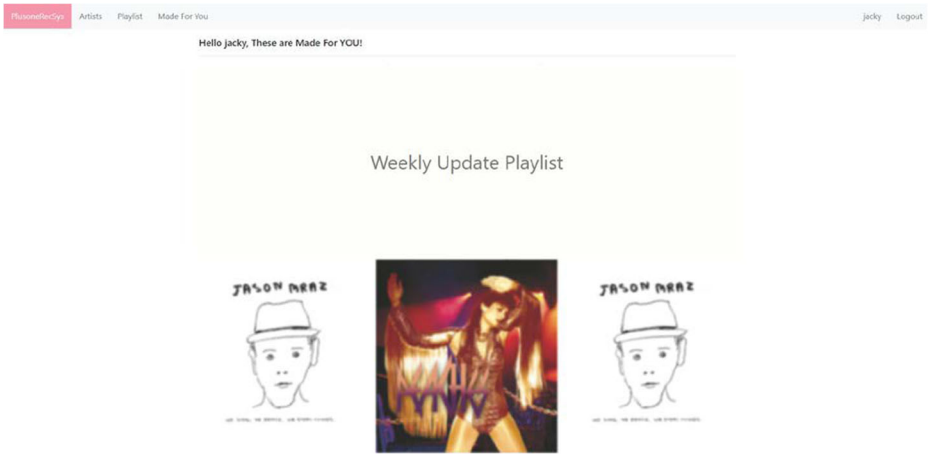


Fig. 17 Proposed RPMRS recommendation lists

discover which song is most similar to songs the user may like, and hence recommend these similar songs. Finally, ranking the similarity of each song to select the top-20 to generate the recommendation list, the list of reinforcement learning recommendation named as "Weekly Update Playlist", as shown in Fig. 17.

When the user first uses the proposed recommendation system, the system provides a survey investigate user music preference. Actual user playing history is subsequently collected and stored in the user log database. When the user clicks on the RPMRS interface, the system presents a song recommendation specific to the user, generated from content based and RL models using extracted audio and lyrics features, Euclidean similarity between songs. User feedback on the recommendation list is also recorded, to continuously optimize recommendation accuracy, implementing the RL concept.

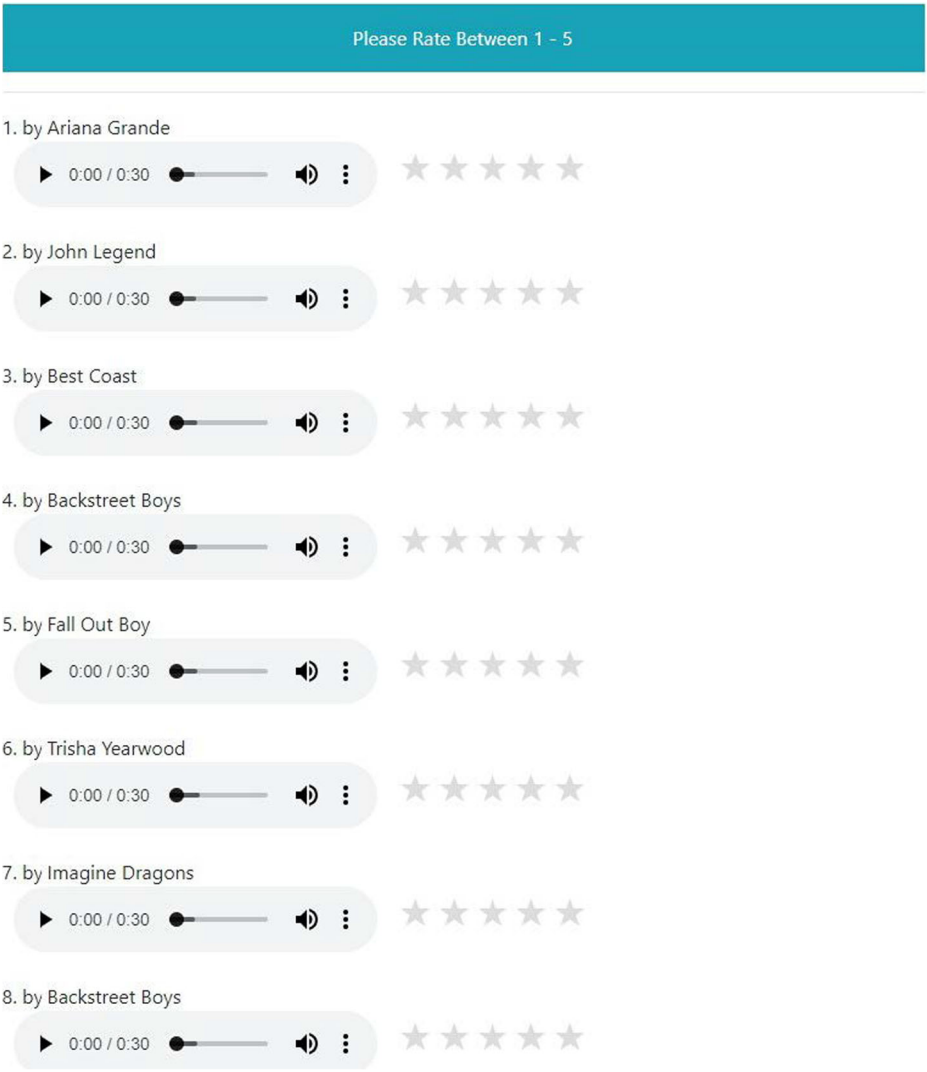


Fig. 18 Recommendation list

6 Experiments

6.1 Dataset

We performed an experiment using a real music and user log dataset, including song data and user playing history. This music dataset included 11,394 songs with corresponding essential features, including artist, album, lyrics, music track link, album image link, artist image link, etc. The user log dataset comprised 10 users with 4000 feedback points. Audio, lyrics, and hybrid feature vectors were calculated for each song, and collated with user playing history, and survey ratings.

6.2 Baseline methods

We constructed a system to recommend music to users after mining their survey ratings. The baseline methods recommended 20 music selections separately and users can choose to listen to the music or otherwise by pressing the appropriate buttons, as shown in Fig. 18. User then selected the star rating to reflect their preference for the recommended song.

Each user received a recommendation list derived using three different methods. Table 5 shows user satisfaction results. The hybrid music recommender provided superior recommendation performance than audio or lyrics based recommenders. Therefore, we selected the hybrid music recommendation method as the basis for RPMRS music recommendation.

6.3 Experimental evaluation

We adopted the hybrid feature system to boost recommendation performance and solve the cold-start problem. This section implements RL to generate personalized satisfactory music playlists. We performed the same experiments and compared user satisfaction with the 20 song playlist for each recommendation round. The proposed RPMRS incorporated user feedback from each recommendation to help provide better-personalized recommendations subsequently. We ran the experiment for 20 music recommendation rounds for each user.

Figure 19 shows that individual user ratings did not stably grow through early rounds, but generally improved significantly for later rounds. Figure 20 shows average ratings of all users. Overall, recommendations tended to provide stable and high rating performance within the

Table 5 Baseline method satisfaction scores

Method User	Audio	Lyrics	Hybrid
1	3.45	2.4	4
2	3.15	2.95	4.25
3	2.35	1.3	3.5
4	2.3	1.5	3.6
5	3.25	1.55	3.9
6	3.3	1.6	3.75
7	3	1.9	3.5
8	2.25	2.1	4
9	3.45	1.6	3.7
10	3.95	1.9	4.2
Average Rating	3.045	1.88	3.84

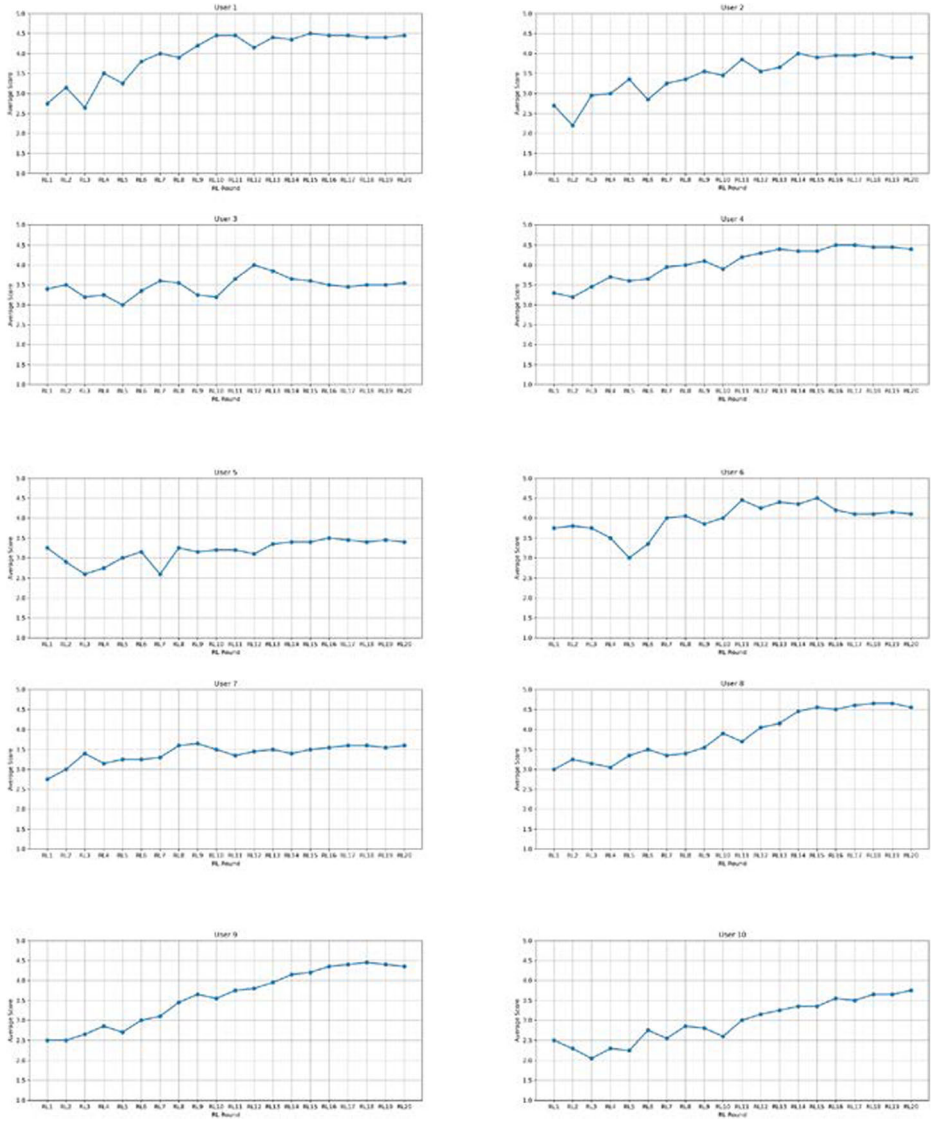


Fig. 19 Individual user ratings over 20 rounds

final five rounds. Thus, RPMRS recommendations were more personalized and satisfactory with incorporating RL.

The proposed RPMRS model provided significant improvement through the content based method with hybrid deep features for initial recommendations, and improved over time as RL benefits helped to ensure recommendation quality. The RPMRS continuously considers user preferences from recent songs they have listened to. Therefore, RPMRS can learn user preferences without user inputs, successfully generating more closely personalized playlists. Experimental results confirmed recommendation performance was significantly enhanced by the RL mechanism.

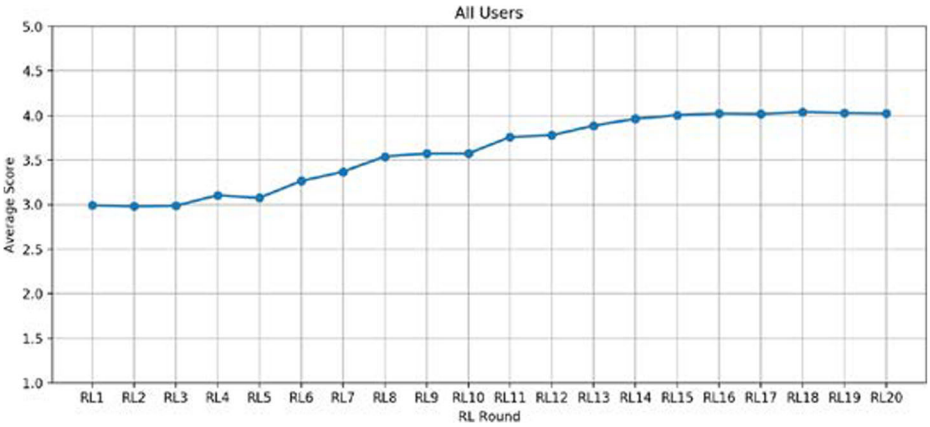


Fig. 20 The average rating of all users with 20 rounds

7 Conclusions

We proposed a content based music recommendation system combined with history based reinforcement learning (RL). We extracted features using unsupervised learning for the content based model, learned from unlabeled input data using audio and lyrics features. We removed stop words and simplified the lyrics in pre-processing, applied TF-IDF to select the top 30 words, and Word2Vec to produce representative song vectors. Audio features were extracted using the WaveNet Autoencoder to embed and select 3 features, concatenating them into a single feature vector. We used the extracted feature vectors to calculate Euclidean song similarity, and hence help recommend favored music not previously listened to by the user, but also to avoid repeatedly recommending disfavored music.

We design the reinforcement personal music recommendation system (RPMRS) to generate music playlists, learning recommendations from user playing history. RL based recommender systems can continuously update strategies using user feedback, and learn strategies that maximize cumulative reward from users. The RPMRS continuously self-optimizes, automatically adjusting recommendation lists to address user dynamic preferences. Experimental outcomes on real datasets verified the effectiveness of the proposed model.

Acknowledgements This work was supported by the Ministry of Science and Technology, Taiwan, R.O.C. [grant number MOST 108-2218-E-025-002-MY3], [grant number MOST 108-2218-E-001-001].

References

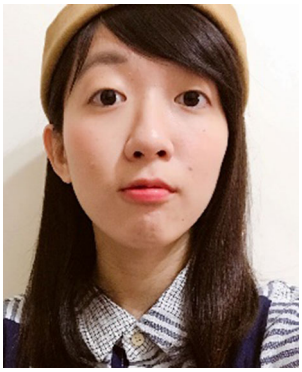
1. Hurley N, Zhang M (2011) Novelty and diversity in top-n recommendation – analysis and evaluation. ACM Trans Internet Technol (TOIT) 10(4):14
2. Cheng Z, Shen J, Nie L, Chua TS, Kankanhalli M (2017) Exploring user-specific information in music retrieval. In: Proceedings of the 40th international ACM SIGIR conference on Research and Development in Information Retrieval. ACM, pp 655–664

3. Wang X, Wang Y (2014) Improving content-based and hybrid music recommendation using deep learning. In: Proceedings of the 22nd ACM international conference on Multimedia. ACM, pp 627–636
4. Logan B (2004) Music recommendation from song sets. In: ISMIR, pp 425–428
5. Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. In: Recommender systems handbook. Springer, Boston, pp 1–35
6. Liang D, Zhan M, Ellis DP (2015) Content-aware collaborative music recommendation using pre-trained neural networks. In: ISMIR, pp 295–301
7. Gao H, Tang J, Hu X, Liu H (2015) Content-aware point of interest recommendation on location-based social networks. In: Twenty-ninth AAAI conference on Artificial Intelligence
8. Lu Z, Dou Z, Lian J, Xie X, Yang Q (2015) Content-based collaborative filtering for news topic recommendation. In: Twenty-ninth AAAI conference on Artificial Intelligence
9. Soares M, Viana P (2015) Tuning metadata for better movie content-based recommendation systems. Multimed Tools Appl 74(17):7015–7036
10. Yeh CH, Tseng WY, Chen CY, Lin YD, Tsai YR, Bi HI, Lin YC, Lin HY (2014) Popular music representation: chorus detection & emotion recognition. Multimed Tools Appl 73(3):2103–2128. https://doi.org/10.1007/s11042-013-1687-2
11. Tzanetakis G (2007) Marsyas submissions to MIREX 2007. In: Proceedings of the international conference on Music Information Retrieval
12. Peeters G (2008) A generic training and classification system for MIREX08 classification tasks: audio music mood, audio genre, audio artist and audio tag. In: Proceedings of the international conference on Music Information Retrieval
13. Mokhsin MB, Rosli NB, Wan Adnan WA, Abdul Manaf N (2014). Automatic music emotion classification using artificial neural network based on vocal and instrumental sound timbres. In: New trends in software methodologies, tools and techniques – Proceedings of the 13th SoMeT 2014. Frontiers in artificial intelligence and applications, vol 265, IOS Press, pp 3–14. https://doi.org/10.3233/978-1-61499-434-3-3
14. Kim Y, Schmidt E, Migneco R, Morton B, Richardson P, Scott J et al (2010) Music emotion recognition: a state of the art review. In: Proceedings of the 11th international Society for Music Information Retrieval Conference (ISMIR). ISMIR, Utrecht, pp 255–266
15. Li T, Ogihara M (2003) Detecting emotion in music
16. Van Den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kavukcuoglu K (2016) WaveNet: a generative model for raw audio. SSW 125
17. Engel J, Resnick C, Roberts A, Dieleman S, Norouzi M, Eck D, Simonyan K (2017) Neural audio synthesis of musical notes with wavenet autoencoders. In: Proceedings of the 34th international conference on Machine Learning-Volume 70, pp 1068–1077. JMLR.org
18. Huang JW, Chiang CW, Chang JW (2018) Email security level classification of imbalanced data using artificial neural network: the real case in a world-leading enterprise. Eng Appl Artif Intell 75:11–21
19. Chang JW, Lee MC, Su CY, Wang TI (2017) Effects of using self-explanation on a web-based Chinese sentence-learning system. Comput Assist Lang Learn 30(1–2):44–63
20. Chang JW, Lee MC, Wang TI (2016) Integrating a semantic-based retrieval agent into case-based reasoning systems: a case study of an online bookstore. Comput Ind 78:29–42
21. Huang PS, Chiu PS, Chang JW, Huang YM, Lee MC (2019) A study of using syntactic cues in short-text similarity measure. J Internet Technol 20(3):839–850
22. Lee MC, Chang JW, Hsieh TC (2014) A grammar-based semantic similarity algorithm for natural language sentences. Sci World J:2014
23. 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
24. Choi K, Downie JS (2018) Exploratory investigation of word embedding in song lyric topic classification: promising preliminary results. In: JCDL, pp 327–328
25. Delbouys R, Hennequin R, Piccoli F, Royo-Letelier J, Moussallam M (2018) Music mood detection based on audio and lyrics with deep neural net. arXiv preprint arXiv:1809.07276
26. Watkins CJCH (1989) Learning from delayed rewards
27. King J, Imbrasaitė V (2015) Generating music playlists with hierarchical clustering and Q-learning. In: European conference on Information Retrieval. Springer, Cham, pp 315–326
28. Zhao X, Zhang L, Ding Z, Xia L, Tang J, Yin D (2018) Recommendations with negative feedback via pairwise deep reinforcement learning. In: Proceedings of the 24th ACM SIGKDD international conference on Knowledge Discovery & Data Mining. ACM, pp 1040–1048
29. Taghipour N, Kardan A (2008) A hybrid web recommender system based on q-learning. In: Proceedings of the 2008 ACM symposium on Applied Computing. ACM, pp 1164–1168
30. Srivihok A, Sukonmanee P (2005) E-commerce intelligent agent: personalization travel support agent using Q Learning. In: Proceedings of the 7th international conference on Electronic Commerce. ACM, pp 287–292

31. Zheng G, Zhang F, Zheng Z, Xiang Y, Yuan NJ, Xie X, Li Z (2018). DRN: a deep reinforcement learning framework for news recommendation. In Proceedings of the 2018 World Wide Web conference on World Wide Web. International World Wide Web Conferences Steering Committee, pp 167–176
32. Iyyer M, Manjunatha V, Boyd-Graber J, Daumé III H (2015) Deep unordered composition rivals syntactic methods for text classification. In: Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on Natural Language Processing (Volume 1: Long Papers), vol 1, pp 1681–1691

Jia-Wei Chang is an assistant professor in Department of Computer Science and Information Engineering at National Taichung University of Science and Technology. He is the Chair of Young Professionals, IET Taipei Local Network. He was a data scientist and project manager at IoT BU, Nexcom during 2016–2017. He received the Ph.D. degree from Department of Engineering Science, National Cheng Kung University in 2017. His research interests include natural language processing, internet of things, artificial intelligence, data mining, and e-learning technologies.

Ching-Yi Chiou is studying master program in Department of Information Management at National Chung Hsing University. Her research interests include data mining, artificial intelligence, music recommendation, and music generation.



Jia-Yi Liao is an undergraduate student in Department of Computer Science and Information Engineering at National Taichung University of Science and Technology. Her research interests include data mining, artificial intelligence, music recommendation, and music generation.

Ying-Kai Hung is an undergraduate student in Department of Computer Science and Information Engineering at National Taichung University of Science and Technology. His research interests include data mining, artificial intelligence, audio analysis, and audio generation.

Chien-Che Huang is an associate professor in Department of Leisure and Recreation Management at National Taichung University of Science and Technology. He has received several master degrees including Computer Science, Special Education and Leisure Management and PhD degree in Industrial Education from USA and

Taiwan universities and has over 20 years of industry experience. He has also awarded Australia Endeavour Executive Fellowship in 2016 as visiting scholar concentrate on Skill Competency Platform (SCP) when he was an assistant professor in Department of Computer Science and Information Engineering at National Taichung University of Science and Technology (NUTC). Current, he is a director of Institutional Research Center in NUTC. His research interests include IOT, machine learning, data mining, and e-learning technologies.

Kuan-Cheng Lin is a professor in Department of Information Management at National Chung Hsing University. His research interests include natural language processing, internet of things, artificial intelligence, data mining, and e-learning technologies.

Ying-Hung Pu is an assistant professor in College of Languages at National Taichung University of Science and Technology. He received the Ph.D. degree from Department of Engineering Science, National Cheng Kung University in 2017. His research interests include Educational Technology, Mobile and Authentic Learning, Learning Behavioral Analysis, Computer-assisted Learning, STEM and Maker education and Evaluation Method.