Expert Systems with Applications 37 (2010) 2815–2825

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An intelligent music playlist generator based on the time parameter with artificial neural networks

Ning-Han Liu *, Shu-Ju Hsieh, Cheng-Fa Tsai

Department of Management Information Systems, National Pingtung University of Science and Technology, 1, Hseuh Fu Road, Nei Pu, Pingtung 912, Taiwan, ROC

ARTICLE INFO

Keywords:
Music playlist generator
Recommendation system
Artificial neural networks
Music database
Music features extraction

ABSTRACT

A music hobbyist listens to different types of music at different times of the day. Thus, an automatic music playlist generator that can adjust to the hobbyist's daily activities on this basis is necessary in order to generate the appropriate music to suit the user's current activity, whether it is working or driving. Although existing research has introduced various music playlist generators, there is yet a system that generates the music playlist based on time. Hence, in this paper, we present a music playlist generation system, which provides an automatic and personalized music playing service based on the time parameter. This system represents the characteristics of music from features extracted out of both the music's symbolic form and wave data. The kernel of this system is based on a modified artificial neural network. The user's music rating history and the associated time stamps in the user's profile constitute the training data of the modified artificial neural networks. A collaborative method has also been proposed to reduce the effect of the cold start problem upon system initialization. A series of experiments have been carried out to demonstrate the performance of this system.

1. Introduction

With technological advances that expand both online and physical storage media, the quantity of music that a person can store and listen to increases. For example, a person can access digital audio files from high-capacity mobile music players (e.g. the "iPod") and the internet. Nevertheless, although a person has access to large volumes of digital music (either via an internet source or shared physical storage), not all the music that the person desires can be accessed due to the unstructured manner the music has been stored. However, arranging a suitable playlist from vast amounts of music is a time-consuming and difficult problem.

Generally, previous research work has addressed this problem by generating the playlist according to user preferences and situations (Kaji, Hirata, & Nagao, 2005). These research studies have used several kinds of metadata information (e.g. artist, title, genre, etc.) to record a user's profile and generate the playlist efficiently. However, some metadata is difficult to acquire, for example, the music genre. Furthermore, in some instances, users are required to annotate the situation in order to facilitate the operation of the system, which is not always possible.

In order to generate an effective playlist generator, let us consider that the majority of people would vary the type of music they listen to depending on the current time of the day. For example, a person might listen to soft music when they wake up in the morning and might later change the music to rock-and-roll just before going off to work. A well-designed playlist generation system in this case would provide the person with appropriate music at different times automatically. To the best of our knowledge, there is currently no playlist generation system that has considered this aspect of time to produce an effective playlist generator. Such a system should include a music-filtering engine to extract music that is acceptable to the user. This engine should be able to filter from vast volumes of music stored both online and in the user's personal storage. Thus, the resultant question to be addressed is how such a playlist generator can be generated to account for the time parameter and the user recommendation function.

Fig. 1 illustrates the motivating scenario. As illustrated, user A always listens to hip-hop style music from 07:00 am to 08:00 am and subsequently Rock & Roll music. In contrast, user B likes to listen to slow tempo music from 07:00 am to 07:30 am because this duration represents her wake-up time. Both of them have different preferences to music at different times as well as different preferences in music styles (e.g. user A uses the music style, while user B uses tempo).

In this paper, we propose a novel intelligent system that provides a suitable playlist to the user depending on the time the user listens to the music. In order to represent the musical piece, this system extracts the necessary features from both the symbolic and wave forms of the music files. Mixed artificial neural networks have been proposed as the kernel of this system. The input to the

* Corresponding author. Tel.: +886 8 7703202; fax: +886 8 7740306.
E-mail address: gregliu@mail.npust.edu.tw (N.-H. Liu).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.09.009

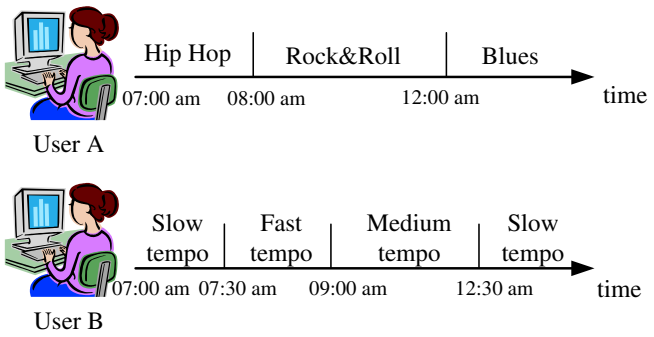


Fig. 1. An example of different users' music listening behaviors.

artificial neural networks includes the user's feedback and the time stamp. After the artificial neural networks have been trained, the system then learns the user's preference in music and predicts the playlist depending on the time parameter. Compared to other systems that have been previously proposed, our novel playlist generator generates the playlist based on the impact of time, apart from typically music content and user's rating feedback. The extensive experimental results obtained demonstrate that time is an important factor that influences the quality of the playlist.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the design of the music playlist generator and the adopted techniques. Experimental results are shown and discussed in Section 4. Finally, the conclusion is presented in Section 5.

2. Related work

Generally, a music recommendation system serves to retrieve music that the user might be interested in. As part of this process, the music playlist generator in the system sorts the music playlist arrangement according to the user's preferences and current situations. In our system, we attempt to construct a playlist generator that not only arranges the user's music collection but also predicts new music items for the user. Thus, in order to provide a general overview, this section further details existing automatic playlist generators and music recommendation systems.

Existing music recommendation systems address the prominent problem of extracting useful information from all currently available information. Generally speaking, content based filtering and collaborative filtering are the most commonly used mechanisms for this purpose. Some research works combine these two approaches in order to get more accurate solutions.

The content based filtering (CBF) mechanism predicts user preferred data items by matching the representations of the data items relevant to the user. For example, in Inforfinder (Krulwich & Burkey, 1996), and News Dude (Billsus & Pazzani, 1999), data items relevant to the user in the past are used as the user profiles. These systems then recommend new data items with higher relevance to the user profiles. The application of the CBF mechanism is dependent upon the content description. The content description describes the metadata of data items, which includes the title, artist, genre and lyrics of a musical piece. When this metadata is not available or the label of the music is not acceptable to the user (e.g. the genre of a song is labeled as hard rock), the CBF mechanism can be directly applied to the raw data in the media (Reed & Lee, 2007). For example, in Chen and Chen (2005)), the musical pieces are first categorized according to the pitch, tempo, loudness and entropy that were extracted from the raw data of the music. These features were used as the basis of recommendation. In effect, new music similar to what the user had recently listened to were recommended.

The collaborative filtering (CF) mechanism uses the correlations between users on the basis of their ratings to predict items for users (Rucker & Polanco, 1997). In this mechanism, users with similar preferences or profiles are grouped together to share their information in the profiles. New items are then recommended to users on the basis of similar shared information by users in the same group. Compared to CBF, CF is more likely to recommend unexpected items by manipulating shared information. As an example, in Chen and Chen (2005), the proposed CF technique captures user interests and behaviors from transactions in the user's access history. Then, the users are grouped based on these features. Lastly, the music is recommended based on the weight of music access frequency. The CF mechanism performs well in some practical areas but it requires the definition of users' distance, which can be difficult to determine.

Several research studies have combined both the CBF and CF approaches to construct recommendation systems. For example, in Konstan et al. (1997) and Schein, Popescul, Ungar, and Pennock (2001), users can specify the profiles to describe the features of data items that they are interested in. These systems allow the user to comment data items and group users according to the similarities in their ratings of data items. The recommended results are then provided on the basis of the user's personal interests (CBF) and data items that are read by other similar users (CF). In Yoshii, Goto, Komatani, Ogata, and Okuno (2006), the authors integrate both rating (CF) and content data (CBF) by using a Bayesian network to build a hybrid music recommendation system.

With regard to music playlist generation, four general methods have typically been used in the literature. The first method allows the user to specify some music as seeds. The playlist generation algorithms then search similar music to form the list, for instance the AutoDJ (Platt, Burges, Swenson, Weare, & Zheng, 2002) project developed by Microsoft Research. Another similar example is proposed in Flexer, Schnitzer, Gasser, and Widmer (2008), which concerns the creation of playlists with an inherent sequential order to create a smooth transition between the start and end song that the user has chosen. The second approach generates the playlist according to a set of user specified constraints about the user's desired musical content, for example, the system created by Sony Computer Science Laboratory (Aucouturier & Pachet, 2002). In their system, constraints are specified, for instance, "the genres of items #4 to #9 should be rock" and "the playlist should contain at least 70% of instrumental titles". The third method generates a playlist according to user preferences and situations, e.g. the system in Kaji et al. (2005). The system produces a playlist for users in similar situations by using content-based retrieval on playlists of other listeners in those situations. It can be noted that the three aforementioned methods are based on examining the metadata attached to the musical pieces and thus, when the metadata is missing, these approaches cannot be applied. The fourth method takes the listening history of the user into account and generates the listener model for predicting the user's preference. For instance, the system proposed in Andric and Haus (2006) records the frequency of two music pieces that have been played together to indicate the relevance between them. A vector space is then constructed in which each vector represents a musical piece. The vectors are then grouped so that each grouping of the pieces may indicate the different listening situations. Lastly, the music playlist is generated according to the situation indicated by the user (e.g. "party situation" or "lonely situation").

3. Smart music playlist generator

A smart music playlist generator is proposed in this paper. As shown in Fig. 2, this generator consists of several function blocks

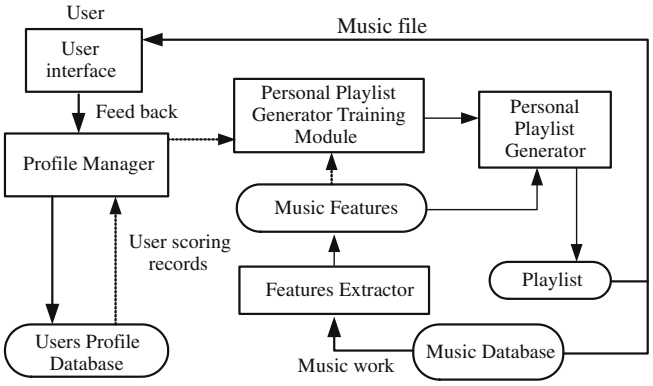


Fig. 2. The system architecture of the smart music playlist generator.

which include the music player interface, profile manager, user profile database, music database, features extractor, playlist generator training module and personal playlist generator. A new user first has to input their personal data (e.g. sex, age, job, etc.) through a registration web page. Upon completion of the registration, the user is then provided a playlist that is selected randomly or retrieved by matching profile similarity to other users. Following this, the user is then allowed to state his/her music playing preferences through our music player interface. The preferences are recorded with a time stamp, unlike previous playlist generation systems that only record the playing preference. The kernel of our personal playlist generator is an artificial neural networks (ANN) system. The ANN is trained by features of music that the user has listened to and the associated time parameter through the personal playlist generator training module. After the personal ANN is built, the user is then provided with a personalized playlist depending on the time parameter and the user's preference of music.

3.1. Features extraction

Due to varying individual preferences to music that we have identified previously, our playlist generator system extracts comparatively more features than existing systems. Although these additional features can result in a long training period for the artificial neural networks (i.e. the kernel of our system), this is justified as it would enable each individuals' music preferences to be accurately captured. Our system extracts features from both symbolic and raw audio data to provide more useful information to the user (The reader is referred to previous research work in Lidy, Rauber, Pertusa, and Inesta (2007) for further details). With respect to features extraction, each music file in the database collected or built manually includes two types of files: one is the wave file (e.g. MP3) and the other is the associated MIDI file. For the purpose of simplifying the system design and serving the preference of most people (the number of pop music listeners is much more than the one of classical music listeners), the database in our system only contains the pop music which included hip hop, rock & roll, blues, jazz and so on. The extracted features are explained in the following subsection.

3.1.1. Features from symbolic form

There are several symbolic forms of digital music, such as MIDI (MIDI Manufacturers Association, 1996) and CHARM (Wiggins, Miranda, & Harris, 1997). We adopt MIDI in our approach due to its popularity. It can be noted that our approach can also be applied to other symbolic forms to provide precise representations of the scores. A musical piece with its own style often contains particular melodies. Therefore, only melodic information is extracted from the MIDI files, i.e. pitch and duration to represent the music. Each note in a MIDI file forms a triple (v, s, e), where v is the pitch scale, s and e are the onset and offset times, respectively. Following the MIDI standard, the pitch scale is a non-negative integer smaller than 128. In this way, a monophonic melody with n notes can be represented as a note string $(v_1, s_1, e_1)(v_2, s_2, e_2)\cdots(v_n, s_n, e_n)$ where $e_i \leqslant s_{i+1}$, $1 \leqslant i < n$.

The contents of the MIDI format not only indicate the pitches and duration of notes in the musical piece precisely but also include abundant music forms and characteristics. From these contents, we then calculate the statistics derived from the MIDI file to represent the characteristics of the music. Here, the features of MIDI files are extracted from the pitches and duration of notes that are associated with the vocal sound (the main melody in the pop music is the vocal sound). These features (including some features that have been proposed in Chen and Chen (2005)) from the theme are described as follows:

- **Average pitch** (**AP**)

AP indicates the music that either goes on the higher compass or the lower compass, which is defined as follows:

$$\mathrm{AP} = \frac{\sum_{i=1}^{n} v_i}{n} \tag{1}$$

where v_i is the pitch scale of the note and n is the length of the notes sequence.

- **Pitch entropy** (**PE**)

PE indicates the variation degree of the music, which is defined as follows:

$$\mathrm{PE} = -\sum_{i=1}^{np} P_i \log P_i \tag{2}$$

where np is the number of distinct pitches that appeared in the piece of music and P_i is defined as follows:

$$P_i = \frac{N_i}{T} \tag{3}$$

where N_i is the number of notes with the same pitch and T is the total number of notes.

- **Pitch density** (**PD**)

PD can show the abundant degree of the music, which is defined as follows:

$$\mathrm{PD} = \frac{\mathrm{NP}}{128} \tag{4}$$

where NP is the number of distinct pitches in the track and 128 is based on the number of all distinct pitches in the MIDI standard.

- **Average duration** (**AD**)

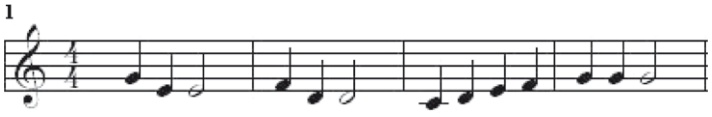
AD can be used for describing the rhythm of the music, i.e. either fast or slow rhythm, which is defined as follows:

$$\mathrm{AD} = \frac{\sum_{i=1}^{n} D_i}{n} \tag{5}$$

where $D_i = e_i - s_i$, is the duration of note i, n is the length of the notes sequence.

- **Duration entropy** (**DE**)

DE indicates the variation degree of the rhythm for the music, which is defined as follows:



Pitch string: 67, 64, 64, 65, 62, 62, 60, 62, 64, 65, 67,67,67

Pitch interval string: -3, 0, +1, -3, 0, -2, +2, +2, +1, +2, 0, 0

Fig. 3. An example of a pitch string and a pitch interval string.

$$\mathrm{DE} = -\sum_{i=1}^{nd} P_i \log P_i \tag{6}$$

where *nd* is the number of distinct durations that appeared in the piece of music and P_i is defined as follows:

$$P_i = \frac{D_i}{T} \tag{7}$$

where D_i is the number of notes with the same duration and T is the total number of notes.

We also extract features from the refrain of the music. The refrain in music can be described as the segment in the piece that repeats several times and which contains the melody that appeals to the listener. In other words, this influences whether the person would like or dislike a song. Thus, the new features also defined are: *Refrain Average Pitch*, *Refrain Pitch Entropy*, *Refrain Pitch Density*, and *Refrain Average Tempo*.

With the exception of extracting features from the absolute pitches and durations of notes, another expression method often used to support approximate matching (Suyoto & Uitdenbogerd, 2005) in the music information retrieval system is *pitch interval* (Hewlett & Selfridge-Field, 1998).

We define the *pitch interval string* as follows.

Given a pitch string $v_1 v_2 \cdots v_m$, the pitch interval string is $c_1 c_2 \cdots c_n$, where $c_i = v_{i+1} - v_i$ and $n = m - 1$.

An example of a pitch interval string is shown in Fig. 3. The pitch values in the pitch string follow the MIDI standard. The values of the pitch interval string are the differences in pitch scales between two consecutive pitches. For instance, the value −3 means that the pitch of the second note is three semitones lower than the pitch of the first note.

Extracting features from a pitch interval string is meaningful because two melodies with the same contour of pitch scales can also be recognized as being identical even though they have different values in pitch scales. The feature that we extracted from the pitch interval string is termed the *pitch interval entropy*, defined as follows:

- **Pitch interval entropy (PIE)**

PIE can be used to show the variation degree of the music under the key invariant, which is defined as follows:

$$\mathrm{PIE} = -\sum_{i=1}^{ni} P_i \log P_i \tag{8}$$

where *ni* is the number of distinct intervals that appeared in the pitch interval string and P_i is defined as follows:

$$P_i = \frac{I_i}{T} \tag{9}$$

where I_i is the number of intervals with the same value and T is the length of the interval string.

As the values of the average pitch intervals of individual music pieces are not different enough to recognize different music, this feature is not used.

3.1.2. Features extraction from vocal wave

Features from an artist's voice data are also extracted due to our observation that a user's preference of a pop music piece is often reliant upon the vocal properties of the artist. These features have been further described below (The sampling rate of the vocal sound from the audio files has been set to 16 kHz):

- Formants

Formants are the frequencies with the main bands of higher energy in the spectrum of sound. They are important features of the voice signal because there are different distributions of formants between different people's voices (Barrichelo, Heuer, Dean, & Sataloff, 2001). To reduce the amount of features, we only take the first formant (F1), the second formant (F2) and the third formant (F3) because they are more useful for distinguishing the different sound sources.

- Mel-scale frequency cepstral coefficients (MFCC)

MFCC is the most well known feature in the area of speech and speaker recognition, which takes human perception sensitivity with respect to frequencies into consideration. In our system, the input vocal signal is segmented into frames of 20 ms, with an optional overlap of 1/2 of the frame size. Only the 12 parameters of MFCC are selected as the features to avoid high dimensionality data, which increases the time complexity of the system training.

In order to simplify the system design and provide a higher accuracy of vocal features, we identified the segments with the significant singer's voice manually whereby existing methods (Feng, Nielsen, & Hansen, 2008; Tsai & Wang, 2006) could only segment vocal or non-vocal parts from the audio file. Each vocal segment is partitioned into 20 ms frames and each frame is transformed into a fifteen dimension vector (3 formants and 12 MFCC). A large number of vectors have been used as the input data for the artificial neural networks. In order to reduce the complexity of this data, two strategies have been proposed:

- Strategy one: statistics

We calculate the averages and the standard deviations of each dimension. Here, a vector with thirty dimensions is used to represent the vocal feature of a musical piece.

- Strategy two: clustering

The use of only the averages and deviations in strategy one as the feature of a vocal sound is too cursory to describe the data distribution. Thus, in strategy two, we use a clustering technique to depict the distribution of vectors in the fifteen dimension space.

The most well-known and used partitioning methods are *k-means* and *k-medoids* (Han & Kamber, 2001). Given the input parameter *k*, these methods partition a set of objects into *k* clusters. After partitioning, the intra-cluster similarity is high but the inter-cluster similarity is low, as shown as Fig. 4. We adopt the *k*-medoids to partition the vocal vector points because it is more robust than *k*-means in the presence of noise and outliers. In other words, a medoid is less influenced by outliers or other extreme

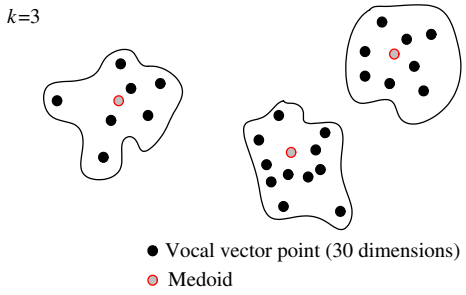


Fig. 4. An example of *k*-medoids clustering.

Table 1
Score table in user profile.

Music ID	Score	Time stamp (am)	Date
3	10	00:06	2008/11/5
100	8	01:22	2008/11/5
10	3	01:30	2008/11/5

values compared to a mean (Han & Kamber, 2001). After using *k*-medoids to partition the vector points into *k* clusters, the averages and the standard deviations of each dimension for each cluster are calculated. As shown in Fig. 4, the vector with $k * 30$ dimensions becomes the representative feature of vocal sound in a musical piece. Nevertheless, the setting of the *k* value represents a trade-off problem, i.e. a bigger *k* value could represent the distribution of the vectors precisely but produces larger input units of artificial neural networks. Therefore, we have opted to select a smaller *k* value by considering the response time of the playlist generation system.

The comparison of the above strategies and decision of a suitable *k* value are further discussed in the experimental results section.

3.2. The playlist generation engine

The input data to our playlist generation engine includes the attributes of the music features and the time parameter. After the engine evaluates these inputs through an intelligent mechanism, it outputs the predicted value that dictates the user's preference in the musical piece. This value can be regarded as the class label in the prediction process as a type of classification. We have adopted artificial neural networks as the kernel of the playlist generator due to its ability to classify continuous values (i.e. the value of input features).

3.2.1. Training data of artificial neural networks

In order to enable the neural networks to learn the user's preference in music and the relationship between this preference and the time parameter, the system collects the scores and time stamp with the associated music file name (or id) after they have listened to the music. Table 1 shows the user's scores.

The user's preference to music ranges from one to ten. A higher score implies that the user feels the music played is suitable for that particular time frame and vice-versa. The time parameter is discussed below.

3.2.1.1. Time parameter. A full day is regarded as one period in the system, while the time of a musical piece is represented in minutes. This is because musical pieces are typically in the length of minutes and users do not require accuracy to seconds in this instance. The time parameter displays the minutes counting from 00:00 am. For example, a musical piece listened to by the user at time 00:06 am can be represented by a time parameter value of 6 units. Similarly, if the musical piece has been listened to at time 01:22 am, the time parameter value is $1 * 60 + 22 = 82$.

For each user, the time parameter value of the musical piece listened to and the associated music features are the training data inputs to the neural networks and the scores are the output of the training data.

3.2.2. The kernel of the playlist generator – a mixed ANN

The training time of the ANN can be higher than normal in two cases. Firstly, when the training set data for an ANN has a higher dimensionality or consists of a larger amount of training samples, more time would be spent to complete the training. Secondly, when the prediction does not satisfy the user's need, the user's new feedbacks are recorded to retrain the ANN. Retraining the system further consumes time if all of the user's data is used as the training samples. Thus, to reduce the retraining time, we propose a new structure as shown in Fig. 5. This structure includes two ANNs: one termed as the *short term ANN* which indicates the user's recent behavior (Chen & Chen, 2005) to the music selection, while the other ANN is termed as the *long term ANN* which is used to record the user's long term preference in music.

Both long term and short term ANN are multilayer feed-forward neural networks with two hidden layers. To predict the user's preference to a musical piece *Mx* at time *t*, the features of *Mx* and time *t* form the input vector of the input units for the two ANNs. Through computations using the two ANNs, two output values are obtained namely $Output_{ANN1}$ and $Output_{ANN2}$. Following this, the two output values are then combined together to form a single value by a fusion function. The output of this fusion function is the prediction of the user's score for the music that has been played at time *t*. The fusion function is defined as follows:

$$F(Output_{ANN1}, Output_{ANN2}) = w_1 \times Output_{ANN1} + w_2 \times Output_{ANN2} \quad (10)$$

where $w_1 + w_2 = 1$.

The two weight parameters w_1 and w_2 in this fusion function can be adjusted by the user as follows. The user increases the value of w_2 if preference is given to musical pieces in which the user has recently heard and gives higher scores to. Conversely, the user increases the value of w_1 if preference is given to musical pieces that the user has enjoyed in the past and gives higher scores to.

Our system builds a personal mixed ANN for each user. The scores recorded in the user's profile provide the basis of the training data for the personal mixed neural networks, i.e. the long term and short term ANN. To elaborate, let us refer back to the description of the score record. The score record contains the music ID, score, playing time, and date. Through a database search using the music ID, the system first obtains the music features that are associated with the music ID. These features provide the training data input, which include the basic music features and playing time of the music that the user has assigned the score to. Intuitively, the output is the associated score value. The date of the training data for the long term ANN belongs to a longer time, e.g. a half year whereas, the date of training data for the short term ANN belongs to a shorter time and is a recent date, e.g. one week. The training process familiarizes supervised learning and the bias weights adjustment in networks uses error back-propagation.

However, when the trained personal mixed ANN does not satisfy the user's need and the user decides to rebuild the bias weights in the ANNs, the short term ANN is first retrained using the user modified score table. The long term ANN will only be retrained if the resultant rebuilt system from training the short term ANN cannot satisfy the user's need. The details of this training process (e.g. error back-propagation) will be not discussed here.

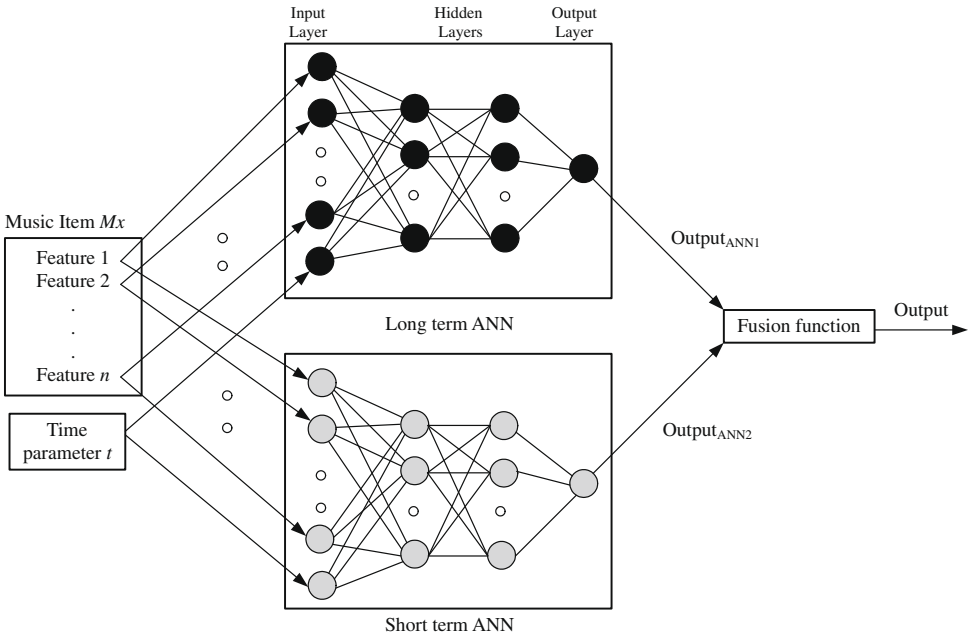


Fig. 5. The structure of the mixed artificial neural networks.

3.2.2.1. Reduction of training time. It is computationally heavy for the system to support every user with a personal mixed ANN. Therefore, to reduce the training time, both long and short term ANNs use the following activation function (Guyon, 1991)

$$Y = \frac{2a}{1 + e^{-bx}} - a \tag{11}$$

where Y is the output of the neural unit and x is the summation of inputs to the neural unit. The constants a, b are set to 1.716, 0.667, respectively, which depends on the reference Guyon (1991).

We also note that, due to the subjective nature of a user's music preference, a user may give totally different scores to the same musical piece at different dates. In the extreme case, this can result in the system temporarily halting due to the process stop criterion, e.g. the constraint on sum of squared errors cannot be reached. To avoid this situation, we adopt the random restart strategy. Using this strategy, when the mixed ANNs are unable to finish the training process after a preset system time, the bias weights are reset randomly and the training process is restarted. The constraint on the sum of squared errors r' is adjusted as follows:

$$r' = r \times 2 \tag{12}$$

where r is the previous constraint on the sum of squared errors.

3.2.2.2. Continuity problem of the time attribute. The value of the time parameter is reset to zero when the time is 00:00 am. However, this would not capture the fact that the time should be continuous and as a result, an error can arise when the score is predicted under a special data distribution. As an example, a user's score table might be empty for a long segment of time from 00:00 am, though a lot of musical pieces have been rated before and close to 00:00 am. As a result, the trained mixed ANNs will be working on a non-representative data distribution when a user uses the system to make a playlist after and close to 00:00 am. This sudden and unexpected change when time passes through the 00:00 am boundary is referred to as the *Cinderella phenomenon*.

This phenomenon is further illustrated in Fig. 6. As shown in Fig. 6a, a user gives high scores to three *A* types musical pieces before 00:00 am. The user subsequently rates a high score to one *B* type of music at 03:00 am (i.e. no rating data appears from 00:00 am to 03:00 am). Thus, the problem arises when the user decides to listen to music at time 00:05 am. In this event, the trained playlist generator would recommend a *B* type musical piece to the user, even when the time slot is closer to the *A* type musical pieces. Fig. 6b shows another inverse example.

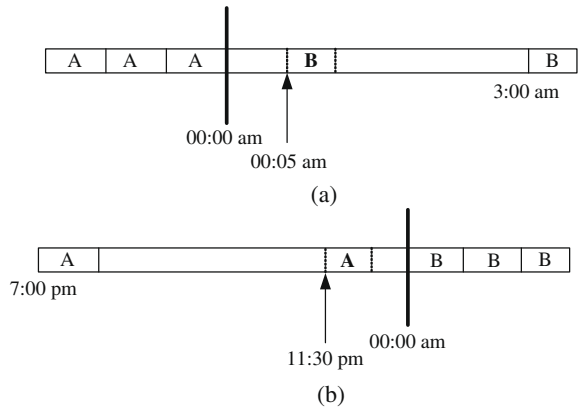


Fig. 6. The Cinderella phenomenon example.

To solve the Cinderella phenomenon problem, the first step is to check the user's score table to determine whether there is scoring data for a long segment of time on one side of the 00:00 am boundary. If no scoring data is present, then the data on the other side of the boundary is copied into the time segment. The data copied is selected randomly in a specified time segment before or after the boundary. In order to avoid producing too many simulated training samples, the time segment and the amount of copied data are restricted. In this playlist generator prototype, the length of time to check is set to 1 h to check the hour before or after 00:00 am. The amount of randomly selected data is half of the total data inside the 1-h segment of the other side. Tables 2a and 2b illustrate our example.

With reference to Table 2a, it can be observed that there is no rating data from 00:00 am to 01:00 am in this user's score table. In this instance, the scoring data 1 h before 00:00 am is copied into

Table 2a
Original score table.

Music ID	Score	Time stamp	Date
3	10	11:05 pm	2008/11/5
100	8	11:09 pm	2008/11/5
10	7	05:14 am	2008/11/6
5	9	11:10 pm	2008/11/6
65	8	11:09 pm	2008/11/6
20	9	05:10 am	2008/11/7

Table 2b
Extended score table.

Music ID	Score	Time stamp	Date
3	10	11:05 pm	2008/11/5
100	8	11:09 pm	2008/11/5
100	8	00:01 am	2008/11/6
10	7	05:14 am	2008/11/6
5	9	11:10 pm	2008/11/6
65	8	11:09 pm	2008/11/6
5	9	00:01 am	2008/11/7
20	9	05:10 am	2008/11/7

the empty time segment. Particularly, the data copied pertain to the two data items with time stamps located in the same duration for the current day. As only half the amount of data has been selected randomly, one data item is copied into the empty time segment for the other days. The duplicated data items are shown in underline in Table 2b. The effect of scoring data before the boundary is extended to the other side of the boundary.

3.2.3. Music selection procedure

After the personal mixed ANNs has been trained, the user is then allowed to set the time interval to play the music in the playlist. The scheduling module in the system takes the features of each musical piece and the playing time as the inputs of the mixed ANNs. Then, the mixed ANNs computes an output value as the predicted value of the user's set score for each musical piece. The higher the value indicated, the greater is the suitability of the musical piece to the user to be played in the time frame. However, the selection of musical pieces, the quantity of music to play and the orderly arrangement of the musical pieces are all difficult issues to address. In particular, for each possible selection of music and arrangement of schedule, there is an associated sum of the scores. As a matter of fact, the playlist with the highest sum of the scores should be returned as the recommendation for the playlist. Unfortunately, the optimal solution will consume too much computational time. Therefore, a greedy method is used to expedite the process of music selection and scheduling.

Firstly, the musical piece with the highest score at the starting time of the interval is selected. Then, the starting time of the next musical piece is the sum of the starting time of the selected music and duration of the music. The next music piece (excluding the music that has been selected) with the highest score is then selected. These steps are repeated until the stopping time of the musical piece exceeds the end of the time interval.

However, this procedure has a serious problem in that through the trained mixed ANNs, the playlist would always be the same at the same time interval even on a different day. Therefore, the probability method is adopted to revise the strategy of playlist generation. In this method, the computed score of a musical piece is formulated as the probability of being selected. This implies that the musical piece with a higher set score has a higher possibility to be selected into the list. Nevertheless, this value is relatively smaller than the total number of music pieces available. In order to formulate a probability that weights this value appropriately, we restrict only musical pieces with a score larger than five as candidates to be added into the music playlist. The probability of a music playlist candidate m_x being selected at time t is defined as follows:

$$P(m_x|t) = \frac{S(m_x|t)}{\sum_{i=1}^{k} S(m_i|t)} \tag{13}$$

where m is the candidate with a score larger than 5, $S(m|t)$ is the score of music m at the time t and k is the number of music candidates.

This probability method can also be used to address the problem that arises when a user typically avoids listening to the same musical piece repeatedly in a short time period. In this event, the selection of a musical piece can be made on the basis of the calculated probability rather than the absolute score.

Users may also change their music preference after using the system for a period of time, which is reflected by the lower level rating set towards the recommended music. At this point, the mixed ANNs should be retrained using the newer user's score table. In this instance, the retraining time can be reduced by collecting training data with mainly positive scores (i.e. score larger than 5). In other words, it should be possible to rate music that has not been considered as candidate. Therefore, in the music selection module, when a user always gives a low score to the recommended music in a specific time period continuously for several days, the module would choose some musical piece from the music database randomly to be placed into the playlist during that time period.

3.3. Cold start problem

A common problem for music recommendation systems is the cold start problem, whereby the number of the response data from a new user is insufficient to produce a satisfactory answer. This is because the time parameter divides the score table, causing the rating data stored to become sparser than the ones of the other systems. This is also a prominent problem faced by our system, i.e. insufficient training data results in a playlist that is generated no different to a random selection. This problem only disappears when the user has used the system for a prolonged time period.

In order to minimize the effect of the cold start problem, a collaborative method is adopted for our system. In this method, the rating records held by similar users are treated as a part of the training data, whereby users are deemed similar when they have a similar background and have listened to the same musical piece with similar rating scores.

Users with a similar background are determined via a questionnaire that requires any new user to reply to the relevant questions upon registration with the system. These questions include the classifications of their jobs, ages, sexes, tastes of music, etc. The higher the number of identical answers between any two users, the higher the degree of background similarity between them.

The background similarity measure and similar rating scores form integral parts of the similarity degree function. In the latter, as the rating scores for distinct users are possibly different, a predefined score threshold (e.g. 5 in our previous example) has been used to separate the score table into two portions. One portion contains the records with scores larger than the threshold, implying that the music content is acceptable to the user while the other portion is unacceptable to the user. Therefore, when the system computes the similarity degree between two users, the rating of the music is simplified within the system as either acceptable or unacceptable. The similarity degree between user x and user y is defined as follows:

Table 3a
The partial data extracted from the profile of new user *x*.

Question	Answer	Music ID	Score	Time stamp	Date
Sex	Female	3	10	01:00 pm	2008/11/2
Jobs	Student	4	7	01:04 pm	2008/11/2
Age	15–20	5	3	08:14 am	2008/11/3
Devotee of rock	Yes	7	4	08:05 am	2008/11/5
Devotee of pop	Yes	8	7	11:09 am	2008/11/5
Devotee of hip hop	No	10	3	11:14 am	2008/11/5

Table 3b
The partial data extracted from the profile of user *y*.

Question	Answer	Music ID	Score	Time stamp	Date
Sex	Female	3	10	01:05 pm	2008/11/4
Jobs	Student	4	8	01:09 pm	2008/11/4
Age	15–20	5	3	09:14 am	2008/11/5
Devotee of rock	No	7	2	09:05 am	2008/11/6
Devotee of pop	Yes	9	8	09:09 am	2008/11/6
Devotee of hip hop	Yes	10	3	09:14 am	2008/11/6

$$Sim(x,y) = \frac{Same(x,y)}{n} + \frac{Num(Pos(x) \cap Pos(y))}{Num(Pos(x) \cup Pos(y))} + \frac{Num(Neg(x) \cap Neg(y))}{Num(Neg(x) \cup Neg(y))} \quad (14)$$

where $Same(x, y)$ is the number of identical responses between two users x and y, n is the number of questions in the questionnaire. $Pos(x)$ and $Pos(y)$ are the music objects in the users' score table of x and y with the scores higher than a predefined threshold. $Neg(x)$ and $Neg(y)$ are the music objects in users' score table of x and y with scores less than or equal to the predefined threshold. $Num(S)$ function returns the number of items in set S. The value of $Sim(x, y)$ ranges between 0 and 3. The value implies the similarity degree between user x and user y.

We use the following simplified table as an example:

Observing Tables 3a and 3b, the similarity degree between new user x and user y is computed as:

$$Sim(x,y) = 4/6 + 2/4 + 3/3 = 2.17$$

After computing the similarity degree between the new user and another user, the system searches k most similar users (i.e. kNN search) and then copies the k users' records into the new user's score table for those records the music IDs are not contained in the new user's table. The copied records are used to increase the number of the new user's training data. In our system, the collaborative method is not only provided to the new user but also to previous users when they want to involve the other user's opinion into their decision mechanisms.

4. Evaluation

The implementation of the playlist generation system is illustrated in Section 4.1. The results of a series of experiments are shown and explained in Section 4.2.

4.1. Implementation

The playlist generation system is installed at both the client and the server sites. The server is based on Microsoft's ASP.Net where an SQL server records the user's profiles. The user plays the music and rates the scores of music through the program installed at the client site. The user's interface is shown in Fig. 7.

Fig. 7. The user's interface.

4.2. Experimental results

In the following experiments, we have allowed the system to run for a sufficiently long period of time in order to train the kernel adequately for learning the relations between the user's music preference and the associated time parameter. Only results pertaining to the performance of our system are shown as no other systems in the literature has factored in the time parameter, to the best of our knowledge.

In the initial stage of experimentation, 300 MP3 music files and associated MIDI files are loaded into the music database. In order to satisfy users' demands for new music, new incoming music files are also processed accordingly during the experimental trials.

Users are divided into two main groups. The first group consists of graduate students who are familiar with the proposed method of the playlist generator system. The assignment to users of group one is to find out the suitable parameters for the extraction of wave data. The second group consists of university students who are the users to test the performance of the system under different parameter settings.

Although every user uses a different rating scale, a common rule has been enforced to all users such that a score larger than five implies that the musical piece is acceptable. We define the scheduling *quality* to measure the system performance. The function of this quality is shown as follows:

$$Quality = \frac{N_A}{N} \quad (15)$$

where N is the number of music objects in the playlist and N_A is the number of music objects in the playlist with a score higher than 5.

In the experiments, the weights between long term ANN and short term ANN are decided by the user, with the time period of training data for the short term ANN set to one week. The user also controls the timing to retrain the personal mixed ANNs.

In the first experiment, the two strategies previously discussed are used to extract the features of vocal waves. The k values in strategy 2 are 2, 3, and 4. All the users do not use the collaborative method and the experimental days are 8 weeks.

From Fig. 8, the result shows that strategy 2 (i.e. clustering method) outperforms strategy 1 (i.e. only statistics). Specifically, the higher the k value, the better the average quality. However, the higher k value also results in a longer training time due to the high dimensionality of the training data. We can also observe from the experimental results that the different k values have little divergence. Therefore, considering the load on the system, we set the k value as 2 in the later experiments.

The experimental results also demonstrate an interesting phenomenon. The average qualities are down from week 5 onwards regardless of the strategy and the k values used. Upon questioning the users, we find that a lot of users are tired of listening to musical pieces in the database after a long period of time. To address this issue, new musical pieces are added into the database in every week for the later experiments.

In the second experiment, the collaborative method is compared with the non-collaborative method. Six groups of users participated in this experiment. The first group of users is provided a playlist obtained through random selection while the second group of users is provided a playlist obtained through the non-collaborative method. For the rest of the groups, the collaborative method is used to generate the playlists for the 2, 3, 4 and 5 users with similar settings, respectively. This experiment is run for a longer duration (12 weeks) than the first experiment due to our earlier observation that the users, as university students, are unable to listen to music for a long time everyday. This has been reflected in the result of the previous experiment, which shows the number of clusters to be 2 (i.e. $k = 2$). The experimental result is shown in Fig. 9.

According to the curve of random selection in Fig. 9, the average qualities are always below 0.2. It can be noted that the gradient of the curve increases slowly in the initial weeks. The reason for this is that the users that are initially not familiar with some musical pieces soon become fond of these musical pieces after listening to them for several times. Nevertheless, the same does not hold for the average qualities when the random method is used to select the music.

By comparing the curves of the non-collaborative and the collaborative methods, we discover that the average quality is the highest in the curve at the seventh week but the average qualities of the collaborative methods get their highest points quickly at the fourth week. From this, we can conclude that the collaborative method can limit the effect of a cold start. Furthermore, the average qualities of the collaborative methods are higher than one of the non-collaborative method from the second week. The reason is that the collaborative users could obtain other similar users' score records, which reduces the probability of rating the user's unfavorable music.

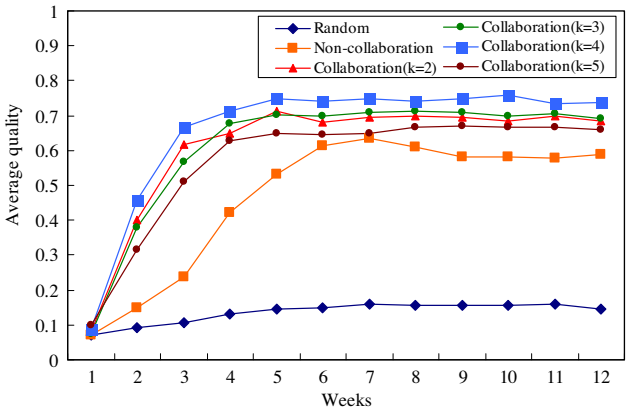


Fig. 9. Average qualities of random, non-collaborative, and collaborative methods.

Fig. 9 also shows the relationship between the preference settings of similar users and the average quality curves in the collaborative methods. We observe that the increase in the number of similar users (i.e. k from 2 to 4) also increases the gradient of the curve and enhances the average quality until a certain value (i.e. k is 5). This happens when the value of k is too large, which increases the number of scoring records from dissimilar users and negatively affects the results of the playlist generation.

In the third experiment, we investigate the relationship between the average quality and a user with a regular daily life (also termed a regular user). We determine the regular users through a questionnaire that lets the user decide if he/she regards himself/herself as a person with a regular daily life. According to the answers we have collected from our questionnaires, 42% of the questioned individuals have a regular life while the others do not. This data has been obtained from experiment two. In the following experiment, we divide the data into two classes, i.e. regularity and irregularity. For visual clarity in the graphs, only data using the collaborative method with $k = 4$ has been re-evaluated. The results obtained are shown in Fig. 10.

From the curves in Fig. 10, we observe that the average quality of users without regularity has increased to the highest point at the third week and stabilizes at the fifth week. After the fifth week, there are no evident differences between the two curves. As stated earlier, the music listened to by irregular user is irrelevant with the time parameter, the mixed ANNs only has to learn the user's preference in music regardless of the playing time (i.e. the system becomes a general playlist generator without the time parameter). Therefore, the necessary number of training data is less than the

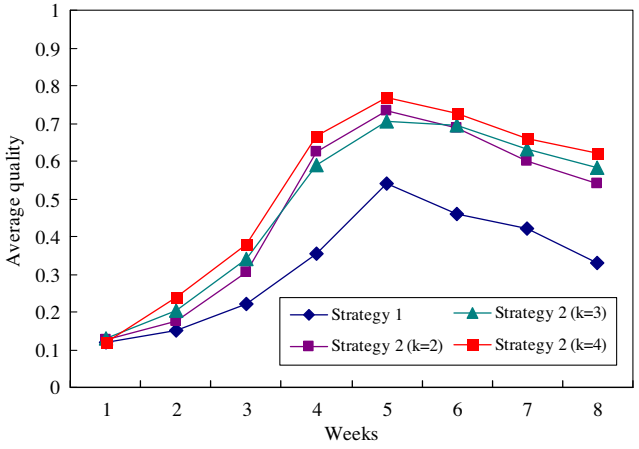


Fig. 8. Average quality with different strategies and parameters.

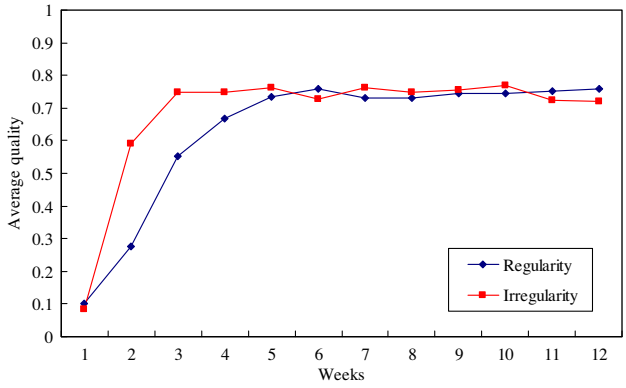


Fig. 10. Average qualities of a regular user and an irregular user.

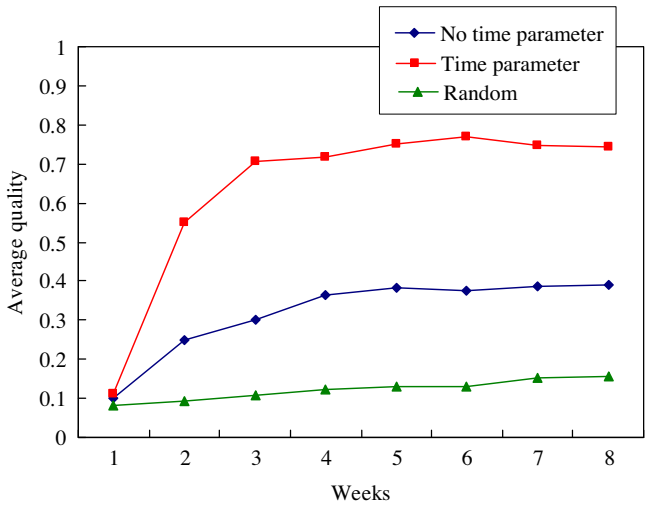


Fig. 11. Average qualities using no time parameter, time parameter, and random methods.

one for the regular user and consequently reduces the training time of the mixed ANNs.

In the second stage of this experiment, we investigate the effect of the time parameter on the result of playlist generation. Here, a similar system is constructed but excludes the use of the time parameter in the mixed ANNs. Only regular users are used and they are divided into three groups. The first group of users uses the system with random music selection. The second group of users uses the system that has excluded use of the time parameter. The third group of users uses the original complete system. The second and third groups adopt the collaborative method ($k = 4$). Referring to the previous experiments, the curve of average quality is stable after five weeks and thus, this experiment has only been made to run for eight weeks. The experimental results are shown in Fig. 11.

From the experimental results obtained, we observed that the system using the time parameter outperformed the other systems in general and that the system without the time parameter only performs slightly better than the system utilizing random selection. Furthermore, the result of random selection is worse than the results obtained from experiment two. One reason could be that the regular user wanted to listen to some particular type of music at a specified time (i.e. the number of preferred music pieces is less due to random selection) and as a result, the probability of a good selection is lower than the one for irregular users. Thus, we can conclude that the time parameter is an important element for an automatic playlist generator for regular users.

5. Conclusion

Automatic music playlist generators provide a more effective mechanism to schedule music that the user desires to play. Nevertheless, existing systems have only scheduled the music playlist based on certain restrictions or use some musical pieces as seeds to find similar music. The common drawback in these systems is that they only schedule musical pieces that are only temporarily favorable. In this paper, we present a novel playlist generator which is based on artificial neural networks. In this generator, the generation of the playlist is not limited to only the user's preference in music but also the changes in user preferences over time. The inclusion of the time parameter in the playlist generator is integral to the user with a regular daily life. Through utilizing the time parameter, it enables the system to produce different playlists suitable at different times of the day. That is the core contribution in this paper.

Apart from this, in order to satisfy the needs of different users, we extract multiple features from both the symbolic form and the wave data for classification. These features are extracted from the refrain of music due to the refrain being the impressible characteristic to the listener. From our experimental results, we have demonstrated that the proposed intelligent system can provide a satisfactory service for users with or without a regular daily life.

Another important contribution in this paper is our definition of the Cinderella phenomenon and the proposal of a method to reduce the effect of this phenomenon that generally appears in any recommendation system that uses the time parameter. We have also discovered that in order to comply with users that prefer listening to new pop musical pieces, the music database should continuously be updated with new musical pieces to ensure the performance of the system is upheld.

We also note that it is time-consuming to train the artificial neural networks and that a real time system cannot be built easily when the user needs to retrain the kernel. Hence, in the future, we wish to evaluate other methods (e.g. decision trees) to construct the playlist generator and involve dimension reduction methods to reduce the construction time. More time parameters such as the season, festival, date and so on can also be used in the system, which can in effect generate a more preferable playlist for the user.

References

Andric, A., & Haus, G. (2006). Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications, 29*(2), 127–151.

Aucouturier, J. J., & Pachet, F. (2002). Scaling up music playlist generation. In *Proceedings of the IEEE international conference on multimedia and expo*.

Barrichelo, V. M. O., Heuer, R. J., Dean, C. M., & Sataloff, R. T. (2001). Comparison of singer's formant, speaker's ring, and LTA spectrum among classical singers and untrained normal speakers. *Journal of Voice, 15*(3), 344–350.

Billsus, D., & Pazzani, M. (1999). A hybrid user model for news story classification. In *Proceedings of international conference on user modeling* (pp. 99–108).

Chen, H.-C., & Chen, A. L. P. (2005). A music recommendation system based on music and user grouping. *Intelligent Information Systems, 24*(2/3), 113–132.

Feng, L., Nielsen, A. B., & Hansen, L. K. (2008). Vocal segment classification in popular music. In *Proceedings of the 9th international conference on music information retrieval* (pp. 121–126).

Flexer, A., Schnitzer, D., Gasser, M., & Widmer, G. (2008). Playlist generation using start and end songs. In *Proceedings of the 9th international conference on music information retrieval* (pp. 173–178).

Guyon, I. P. (1991). Applications of neural networks to character recognition. *International Journal of Pattern Recognition and Artificial Intelligence, 5*, 353–382.

Han, J., & Kamber, M. (2001). *Data mining concepts and techniques*. Morgan Kaufmann.

Hewlett, W., & Selfridge-Field, E. (1998). *Melodic similarity concepts, procedures, and applications. Computing in musicology* (Vol. 11). Cambridge: MIT Press.

Kaji, K., Hirata, K., & Nagao, K. (2005). A music recommendation system based on annotations about listeners' preferences and situations. In *Proceedings of 1st international conference on automated production of cross media content for multi-channel distribution* (pp. 231–234).

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM, 40*(3), 77–87.

Krulwich, B., & Burkey, C. (1996). Learning user information interests through extraction of semantically significant phrases. In *Proceedings of AAAI spring symposium on machine learning in information access*.

Lidy, T., Rauber, A., Pertusa, A., & Inesta, J. M. (2007). Improving genre classification by combination of audio and symbolic descriptors using a transcription system. In *Proceedings of the 8th international conference on music information retrieval*.

MIDI Manufacturers Association (1996). Complete Detailed MIDI 1.0 Specification, Los Angeles, California.

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

Reed, J., & Lee, C.-H. (2007). A study on attribute-based taxonomy for music information retrieval. In *Proceedings of the 8th international conference on music information retrieval*.

Rucker, J., & Polanco, M. J. (1997). Personalized navigation for the web. *Communications of the ACM, 40*(3), 73–75.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2001). Generative models for cold-start recommendations. In *Proceedings of SIGIR workshop on recommender systems* (pp. 253–260).

Suyoto, I. S. H., & Uitdenbogerd, A. L. (2005). Effectiveness of note duration information for music retrieval. In *Proceedings of database systems for advanced applications* (pp. 265–275).

Tsai, W.-H., & Wang, H.-M. (2006). Automatic singer recognition of popular music recording via estimation and modeling of solo vocal signals. *IEEE Transactions on Audio, Speech and Language Processing, 14*(1), 330–341.

Wiggins, G. E., Miranda, A. S., & Harris, M. (1997). A framework for the evaluation of music representation systems. *Computer Music Journal, 17*(3), 31–42.

Yoshii, K., Goto, M., Komatani, K., Ogata, T., & Okuno, H. G. (2006). Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *Proceedings of the 7th international conference on music information retrieval.*