Electronic Preprint for Journal of Information Processing Vol.26

Regular Paper

Latent Topic Similarity for Music Retrieval and Its Application to a System that Supports DJ Performance

Tatsunori Hirai[1,a)] Hironori Doi[2,b)] Shigeo Morishima[3,c)]

Received: June 11, 2017, Accepted: December 8, 2017

Abstract: This paper presents a topic modeling method to retrieve similar music fragments and its application, *MusicMixer*, which is a computer-aided DJ system that supports DJ performance by automatically mixing songs in a seamless manner. MusicMixer mixes songs based on audio similarity calculated via beat analysis and latent topic analysis of the chromatic signal in the audio. The topic represents latent semantics on how chromatic sounds are generated. Given a list of songs, a DJ selects a song with beats and sounds similar to a specific point of the currently playing song to seamlessly transition between songs. By calculating similarities between all existing song sections that can be naturally mixed, MusicMixer retrieves the best mixing point from a myriad of possibilities and enables seamless song transitions. Although it is comparatively easy to calculate beat similarity from audio signals, considering the semantics of songs from the viewpoint of a human DJ has proven difficult. Therefore, we propose a method to represent audio signals to construct topic models that acquire latent semantics of audio. The results of a subjective experiment demonstrate the effectiveness of the proposed latent semantic analysis method. MusicMixer achieves automatic song mixing using the audio signal processing approach; thus, users can perform DJ mixing simply by selecting a song from a list of songs suggested by the system.

Keywords: automatic song mixing, topic analysis, computer-aided performance

1. Introduction

Music has the power to liven up, chill out, and change the atmosphere. Not only live music performances but also recorded music has such potential. DJs are people who can maximize the potential of recorded music by playing it effectively. DJs select music and mix songs to provide the best music for the atmosphere continuously.

Playing back prerecorded music is something that anyone can do, and some who have no experience as a DJ may think that being a DJ only requires one to play music. Certainly, playing one song is easy and can be achieved by simply pressing a playback button; however, DJs mix [*1] the songs so that listeners do not notice the songs are being switched. Note that mixing songs is not the only task that DJs perform but it is the minimum and most fundamental factor for DJ performance. The inherent difficulty in song mixing is often difficult to notice because DJs attempt to make song boundaries seamless, which will not work by just switching songs using any timing. If we care about beats and sounds, the suited timing for song mixing is available only for a moment. On the other hand, there are a myriad of possibilities for song mixing timings and mixing destinations, and it is almost impossible for humans to consider all such possibilities in a limited time. More specifically, the DJ must decide and set the next song and the mixing timing before the currently playing song is finished; thus, many DJs select the next song intuitively. Selecting the best mixing option from innumerable possibilities within a limited time is difficult for DJs and is much more difficult for the inexperienced. Therefore, computationally determining candidate songs such that songs can be mixed naturally is effective for such users. Thus, we present *MusicMixer*, a DJ system that supports a user by automatic song mixing.

Given a list of songs, a DJ selects a song with beats and sounds that are similar to a specific point in the currently playing song such that the song transition is seamless. Consequently, the songs will be mixed as a consecutive song. The beats are particularly important and should be carefully considered. Maintaining stable beats during song transition is the key to achieving a seamless mix. Computers are good at searching for the most similar pairs of beats from innumerable possibilities. Hence, it is possible to solve this problem using a signal processing technique to extract beats and rely on a computer to retrieve a similar beat for effective mixing. However, computers handle audio signals numerically without considering the underlying song semantics; thus, the resulting mix will be merely a mechanical process if the system only considers beat similarity. Here, we consider what factors should be taken into consideration, in addition to the beat, during song mixing. According to the results of a preliminary survey that we performed by questionnaire, 15 out of 19 people

[1] Faculty of Global Media Studies, Komazawa University, Setagaya, Tokyo 154–8525, Japan
[2] Dwango, Chuo, Tokyo 104–0061, Japan
[3] Waseda Research Institute for Science and Engineering, Shinjuku, Tokyo 169–8555, Japan
[a)] thirai@komazawa-u.ac.jp
[b)] hironori_doi@dwango.co.jp
[c)] shigeo@waseda.jp

[*1] The word "mix" means gradually changing a song to another song.

said that the factor related to beat or tempo is important in song mixing and 13 out of 19 people said that the similarity of 2 songs during song mixing is important. From this result, we take the similarity of songs into account in addition to the similarity of beats. At this point, we aim to achieve similarity, which is not described as a numerical value.

Here, we present a method of calculating similarity based on the latent semantics of a song from the polyphonic audio signal. By employing a machine learning method called latent Dirichlet allocation (LDA) [1], we analyze the latent semantics underlying the song. This analysis method is called topic modeling and the latent topics that can be analyzed represent latent semantics on how chromatic sounds in a song are generated (i.e., the semantics related to the harmonic component). In addition to beat similarity, we consider the similarity of latent topics of songs. This process corresponds to consideration of how sound is generated from latent topics in a given song. By using this similarity in addition to the beat similarity, we aim to achieve seamless song mixing. We define the similarity measurement composed of beat similarity and latent topic similarity as "mixability" (i.e., "mix" + "ability"). We also present an interface that supports DJ performance by employing mixability. As shown in **Fig. 1**, MusicMixer automatically mixes songs as naturally as possible.

Given a collection of songs, MusicMixer calculates mixability between all possible song mixing pairs and their corresponding timings. When a user selects a song to play, MusicMixer suggests song candidates and the timing to mix the song based on the calculated mixability. The user can then semi-automatically perform DJ mixing by selecting a song from the candidates or listen to the automatic mixing that MusicMixer suggests based on the best mixability measures.

MusicMixer focuses on song mixing because song selection is an important task for DJs that is highly dependent on the situation, atmosphere, and meta-information of the song. Moreover, the individuality of the DJ's performance substantially affects the song selection. Song selection is what human beings are good at whereas song mixing can be performed better by a computer system. Therefore, MusicMixer takes advantage of both human song selection and automatically calculates mixability. Furthermore, MusicMixer enables automatic or semi-automatic DJ performance that can be enjoyed as a song listening experience, i.e., the system can be used as a personal DJ.

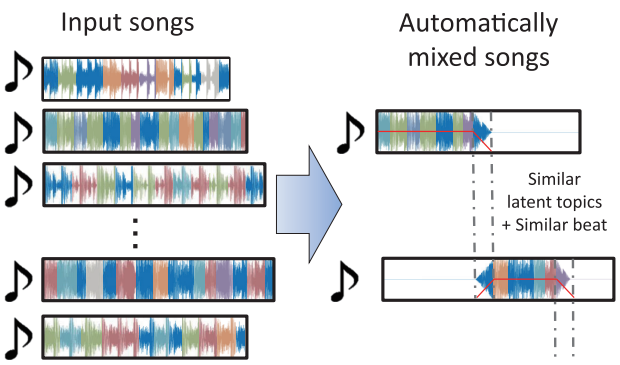


Fig. 1 Conceptual image of mixing songs with similar latent topics and beats using MusicMixer. The color of waveform indicates the latent topic.

2. Related Work

2.1 Music Mixing and Playlist Generation

Ishizaki et al. [2] proposed a DJ system that adjusts the tempo of songs in song mixing. They defined a measurement function for user discomfort relative to tempo adjustment based on a subjective experiment. Cliff [3] also presented a system that seamlessly mixes music by adjusting both tempo and beat. He also enabled users to specify the trajectory of tempo in the resulting mix such that users can influence the entire mix. However, these systems do not consider factors other than tempo or beat and do not retrieve a mixing point but instead forcibly change the song tempo.

Some studies have supported DJs not with song mixing but rather from the aspect of DJ performance. Laursen et al. [4] proposed a DJ interface for live Internet broadcasting with integrated remote audience feedback. Kapur et al. [5] presented a music retrieval tool for DJs to retrieve music through queries by *Beat-Boxing*. Many other music retrieval methods would be useful for DJs in order to search for the next song to mix. The various studies supporting DJ performance can be combined with song mixing methods.

DJ performance can be considered as making a music playlist. Several studies have focused on generating a music playlist [6], [7], [8], [9], [10]. AutoDJ [6] generates a playlist based on one or more seed songs using Gaussian process regression. The AutoDJ project team has also proposed a method of inferring the similarity between music objects and have applied this to playlist generation [9]. However, these approaches focused on playlist generation, and the importance of mixing (connecting) songs was not considered. Goto et al. [11] proposed *Musicream*, which provides a novel music listening experience, including sticking, sorting, and recalling musical pieces. It also provides a playlist generation function; however, mixing is not considered.

There is another approach to mixing songs, referred to as *mashup*. Mashup creates a single song from multiple songs. AutoMashUpper [12] generates a mashup according to a *mashability* measure. Tokui [13] proposed an interactive music mashup system called Massh. Mashups are DJ track composition styles, however not all DJs can perform mashups live without using prerecorded mashup songs. The mainstay of a DJ performance is song mixing.

Another approach to song mixing is the song morphing method proposed by Hirai et al. [14]. This method enables song mixing using a morphing algorithm; however, it can only be applied to songs in MIDI format.

There has been limited research on DJ mixing compared to research on playlist generation. We believe that a mixing method combined with playlist generation methods could be a powerful tool. Therefore, we propose a DJ system for mixing songs that considers beats and additional information that reflects the similarity of songs. Here, we consider the latent semantics, which can be analyzed by topic modeling and use them as a feature for calculating similarity of songs.

2.2 Topic Modeling

In natural language processing research, there is a method called topic modeling, which estimates the topic of a sentence from words that appear in the sentence. Those words depend on the topic of the sentence; thus, the topic can be estimated by observing the actual sentence. If the topics are the same for two sentences, the sentences will be similar at a higher semantic level even if the actual words are not similar. Sasaki et al. [15] proposed a system of analyzing latent topics of music via topic modeling of lyrics. They proposed an interface to retrieve songs based on the latent topics of lyrics.

Topic modeling can be applied to actual features. In this case, feature vectors should be quantized (e.g., a bag-of-features [16]). Nakano et al. [17] applied topic modeling to singing voice timbre. They defined a similarity measure based on the Kullback-Leibler divergence (KL-divergence) of latent topics and showed that singers with similar singing voices have similar latent topics. However, it is difficult to understand the meaning of each topic explicitly using feature vectors rather than words.

Hu et al. [18] used the pitch classes of a song as words to estimate the musical key of a song using topic modeling. This shows that topic analysis using pitch classes is effective for inferring the latent semantics of a song. Hu et al. [19] also proposed an extended method to estimate musical keys from an audio signal using a chroma vector (i.e., audio features based on a histogram of a 12 chromatic scale) rather than pitch classes. This approach shows that topic modeling using chroma vectors is useful for inferring the latent topics of songs.

3. System Overview

MusicMixer requires preprocesses to analyze song beats and latent topics. Given a collection of songs to be mixed, beat analysis is performed using an audio signal processing approach, and latent topic analysis is performed using a topic modeling approach. Beat similarity and latent topic similarity will be the criteria for the song mixing. **Figure 2** shows the system flow.

First, a low-pass filter (LPF) is applied to the input song collection to extract low-frequency signals. In the low-frequency signal, important beat information such as bass and snare drums and bass sounds, is prominent. Thus, beat information can be acquired by detecting the peaks of the envelope of audio waveform in the low-frequency signal.

The latent topic analysis is performed by using LDA [1]. First, the system constructs a topic model using a music database that includes various music genres. Thus, a model covering various music genres can be constructed. The latent topics for a new input song can be estimated using the constructed model. Our goal is to find a good mixing point rather than analyze the topics of an entire song; thus, we analyze the topics of segmented song portions.

The length of data for latent topic analysis is set to 5 s; hence, the similarity will be calculated for all existing pairs of 5 s segment of songs. Finally, the system retrieves the most similar song fragments based on a combination of beat similarity and latent topic similarity. Once a similar pair of song fragments is retrieved, the system mixes songs at the fragment by cross-fading

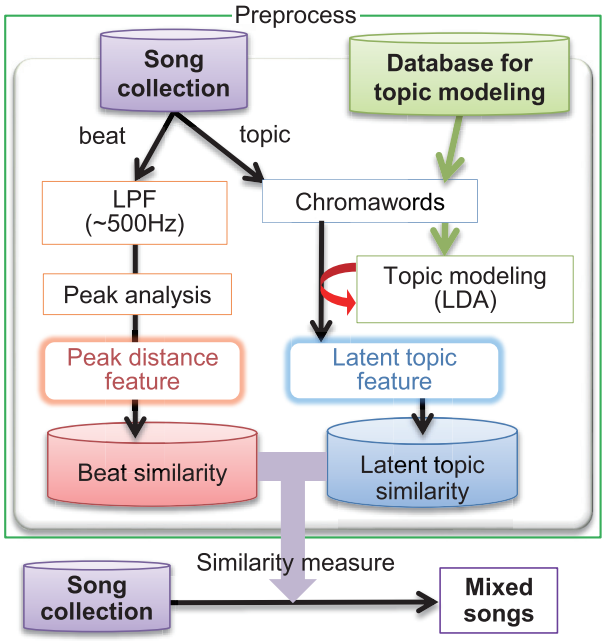


Fig. 2 System flow.

(i.e., fading in and out). Thus, the songs are mixed naturally with our method. To mix more songs for endless playback, the similarity-based retrieval is applied to the mixed song and the next song will be chosen by the system.

4. Beat Similarity

There are many factors related to beat in music. In particular, the sound of the bass drum plays a significant role (e.g., the rhythm pattern called four-on-the-floor is composed of intermittent bass drum sounds). In addition to the bass drum, the snare drum and other bass sounds are important for expressing detailed rhythm. The sound of the bass also plays an important role. These are all included in the low-frequency audio signal. Note that we assume that none of the other sounds affect the beat. To ignore other audio signals, we apply an LPF, which passes signals with frequencies below 500 Hz. This LPF passes the attack sounds of a general snare drum. By analyzing the peaks of the envelope of a low-frequency signal, dominant sound events in the low-frequency spectrum, such as the attack of drums can be detected.

For the low-pass-filtering and peak picking process, we used mirToolbox 1.6.1, which is a musical signal processing tool provided by Lartillot et al. [20]. The LPF signal is extracted from an original audio waveform by using Gammatone filterbank decomposition. In this LPF signal, the envelope is extracted by the infinite impulse response (IIR) filtering process. Peaks are picked by calculating the local maxima. At this point, the local maxima will be considered as a peak if the difference with respect to both the previous and successive local minima is higher than the threshold. We set the threshold value as 10% of the local maximum value which is focused on. The distances between peaks correspond to the length of the beat. **Figure 3** shows an example of the envelope of low-frequency audio signal.

Beat similarity is calculated by comparing the distances between N peaks of the envelope. Here, N is the number of distances to consider. The peak distance feature D_{peak} is an N-

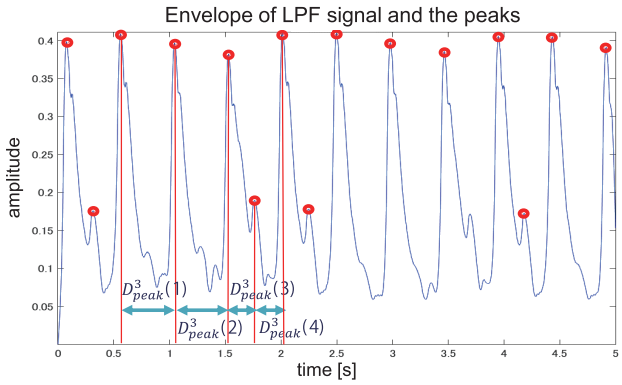


Fig. 3 Extracting peak distance features from the envelope of low-frequency audio signal. The red circle indicates the extracted peaks.

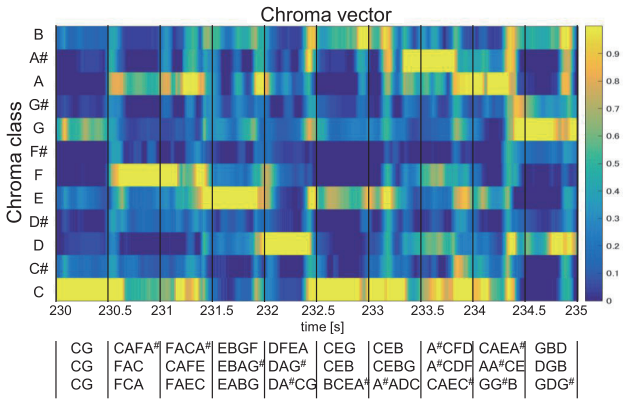


Fig. 4 Extraction of ChromaWords from chroma vector.

dimensional vector composed of distances between peaks. The beat similarity S_{beat} between fragment i and fragment j is calculated as follows:

$$S_{beat}(i,j) = \frac{1}{\sum_{k=1}^{N-1} \|D^i_{peak}(k) - D^j_{peak}(k)\| + 1}. \quad (1)$$

Here, larger N values attain better matching relative to beats. However, the number of candidate songs to be mixed will be excessively reduced if the N value is extremely large. At the current implementation, the parameter N is user-defined.

5. Latent Topic Similarity

This section describes the method of analyzing a latent topic of a song using topic modeling. In particular, we propose a topic modeling method that considers the latent topic of a song by expressing the audio signal symbolically.

5.1 Topic Modeling

The topic model is constructed by extracting the features of songs and applying LDA [1] to the features. We extract the chroma vector from the audio signal and represent the feature symbolically, which we refer to as "ChromaWords."

5.1.1 Feature Extraction

Topic analysis using audio feature values makes it difficult to understand the meaning of topics explicitly. For example, if the chroma vector value is 1.0, 0.1, 0.1, 0.0, 0.2, 0.1, 0.0, 0.4, 0.2, 0.1, 0.0, 0.3, ChromaWords expresses the feature as "CGB," whose meaning is much easier to understand. It is difficult to determine the meaning from a high-dimensional raw feature value; therefore, conventional topic modeling methods could not always describe the meaning of each topic clearly. To improve the interpretability of topic clustering results, we express the audio signal symbolically and represent them as a maximum of 4-letter word. There are symbols in music that are represented in a musical score, i.e., pitch classes. Because a letter is assigned to each musical note, we can use the letters to construct a word for topic modeling. Here, we employ an audio feature referred to as a chroma vector, which is a histogram of 12 musical scales. Each bin of the chroma vector represents a musical note. By sorting the chroma vector by dominant notes, a word can be generated (e.g., [CADE], [BAD#]) which we refer to as ChromaWords. Typically, a chroma vector includes noise caused primarily by dissonant sounds. To avoid the effects of noise, we use the top 70% power of notes. Here, we set the maximum length of the word to four letters. Thus, we can represent polyphonic audio signals symbolically with natural language processing. **Figure 4** shows an example of ChromaWords (bottom) acquired from the chroma vector of an actual song (top). Note that "#" is not counted as a single letter. Because of space limitations, we only display three ChromaWords per 0.5 s. These three ChromaWords are sampled at equal intervals. The leftmost letter is the most dominant component, and the less dominant components are to the right.

ChromaWords are acquired per audio frame. Here, the audio sampling rate is 16,000 Hz monaural, and the frame length is 200 ms, shifting every 10 ms. One hundred words can be acquired from 1 second audio signal.

5.1.2 Latent Dirichlet Allocation

By acquiring ChromaWords from a song, topic modeling can be applied similar to the methods in natural language processing. MusicMixer employs LDA [1] for training of latent topic analysis. The number of topics is set to 100 in order to express semantics that are more complex than those of basic Western tonality. The vocabulary of ChromaWords is 13,345 ($= {}_{12}P_4 + {}_{12}P_3 + {}_{12}P_2 + {}_{12}P_1 + 1$), including perfect silence.

Training is required prior to latent topic analysis. We use 100 songs from the RWC music genre database [21], which comprises songs of various genres. The parameters and algorithm for LDA is the same as the topic modeling method employed for the latent topic analysis of lyrics by Sasaki et al. [15].

Table 1 shows the top 5 representative ChromaWords for each topic learned from the RWC music database (12 topics out of 100, sorted by probability). These 12 topics are the representative topics learned with the RWC music database. The leftmost letter in a ChromaWord indicates the dominant note in the sound. Because many initial letters in ChromaWords for the same topic are the same, the topic model constructed by LDA reflects the semantics of chromatic notes, which is difficult for conventional methods to explicitly express.

Using the constructed model, the latent topics for a new input song can be estimated by calculating a predictive distribution. The latent topics for the new input song are represented as a mixing ratio of all 100 topics. **Figure 5** shows an example of topic analysis for the first 7 s fragment of the song "Let It Be" by The Beatles. In this result, topic 41 includes ChromaWords "DB," "GB," and "AC" as dominant words, and the dominant letters in

Table 1 Examples of top 5 ChromaWords allocated to each topic.

Topic22	Topic90	Topic7	Topic98	Topic78	Topic52	Topic9	Topic79	Topic43	Topic80	Topic50	Topic63
CBC#A	silent	AG	CFG	AA#BC	ED	AFB	silent	AEA#	DAA#C	DGG#	EB
CC#BA	GDCA	AA#G	CFGD	ABA#C	E	AA#BF	GA	AEA#B	DACA#	CGB	BE
CC#AB	GCDA	AGA#	CGFA	BAA#C	F	silent	CF	silent	ADA#C	AEA#	BEG
CBAA#	GDAC	ADGA#	CGFD	AA#CB	DE	AA#GB	CFAG	AEA#G	DACC#	CFA	EBG
CABA#	DA	ADA#G	CGF	ABCA#	EF	AA#FB	FC	AEA#C	AA#BC	CGE	EA

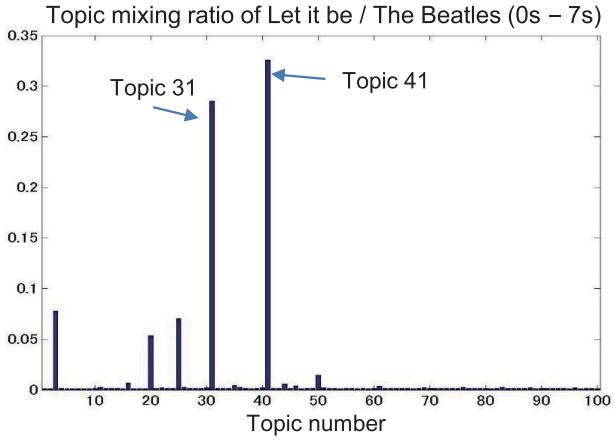


Fig. 5 Results of latent topic analysis applied to the first 7 s fragment of the song "Let It Be" by The Beatles.

the ChromaWords of topic 31 are "F," "C," and "A." In fact, the chord progression for this part of the song is "C, G, Am, F," which shows that topics mostly reflect the notes in these chords. This indicates the relevance between chords and ChromaWords. Note that chords or harmony affects the ChromaWords, but the topics themselves do not directly represent chords or harmony.

Here, our goal is to find a good mixing point rather than analyze the topics of an entire song. Therefore, MusicMixer analyzes latent topics every 5 s to acquire the temporal transition of the topic ratio.

5.2 Calculation of Latent Topic Similarity

The mixing ratio of latent topics for each 5 s song fragment is acquired by the above-mentioned method. The mixing ratio is extracted as 100-dimensional feature vectors, and we use the mixing ratio as the latent topic feature f.

The latent topic similarity S_{topic} between fragment i and fragment j is calculated in the same form as the beat similarity:

$$S_{topic}(i, j) = \frac{1}{\sum_{k=1}^{K} \|f_i(k) - f_j(k)\| + 1}, \quad (2)$$

where K is the number of topics (100).

5.3 Evaluation

We performed a subjective evaluation experiment to evaluate the effectiveness of latent topic analysis using ChromaWords. We compared the proposed method to a latent topic analysis method using mel-frequency cepstral coefficient (MFCC) feature values and raw chroma vector feature values to investigate the effectiveness of our ChromaWords expression. MFCCs are commonly used audio features for measuring similarity. The topic modeling method for the compared methods is based on the method proposed by Nakano et al. [17], which uses k-means clustering to describe feature values in a bag-of-features expression. The number of clusters for k-means is 100. Note that we do not use a similarity calculated from raw feature values so that we can focus on the effects of ChromaWords.

Table 2 Evaluation items and the corresponding scores.

Evaluation item	Score
Pair B is more similar than A	5
Pair B is bit similar than A	4
both pairs are equal in terms of similarity	3
Pair A is bit similar than B	2
Pair A is more similar than B	1

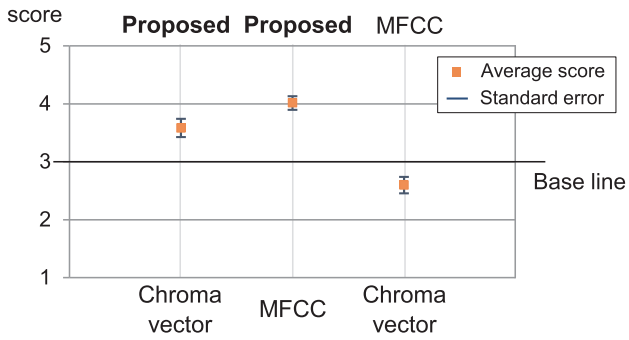


Fig. 6 Result of subjective evaluation experiment.

A total of pop, rock, and dance songs were used in the experiment, and 2,192 segments were generated by cutting the songs into 5 s fragments. We calculated the latent topic similarity for all pairs between the 2,192 fragments.

The subjects were asked to listen to two pairs of songs and indicate which pair was more similar. A pair was generated based on the latent topic similarity using each feature. We selected three pairs of songs per method (nine pairs in total). The three pairs were selected from the top 30 latent topic similarity. Note that song repetition was avoided in this experiment. To avoid the effects of beat, we did not mix the songs but played each one separately.

Eight subjects (ages 22 to 24) with no DJ experience participated in the experiment. The subjects listened to a 2 song-fragment pair generated by method A first, and then a 2 song-fragment pair generated by method B. They then rated the pairs from 1 (Pair A is more similar than B) to 5 (Pair B is more similar than A) as indicated in **Table 2**. A score of 3 is the baseline, indicating that "both pairs are equal in terms of similarity." The pairs were presented randomly. Two out of three methods will be compared by one score.

Figure 6 shows the results of the experiment. Each score is the average of all eight subjects and all nine compared pairs (72 scores per comparison). Comparing the proposed topic modeling method with that using raw chroma vector feature resulted in a score of 3.58, which indicates that the proposed method expresses similarity better. Comparing the proposed method with

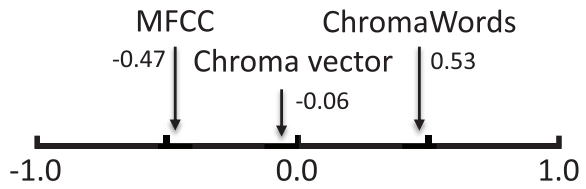


Fig. 7 Psychological scale constructed using Scheffe's method of paired comparison.

one with MFCC resulted in a score of 4.01, which also indicates that the proposed method performs better. The rightmost plot shows a comparison of MFCC and chroma vector methods, which are not related to the proposed method. As can be observed, the topic model using the chroma vector outperforms the MFCC method. We also tested the result using Scheffe's multiple comparison method. As a result, we obtained a psychological scale as shown in **Fig. 7**. The psychological yardstick for the constructed scale was 0.361 ($p < 0.01$). This result shows the following statistical significance: proposed > chroma, proposed > MFCC, and chroma > MFCC. In other words, the proposed method using ChromaWords outperformed the other methods in terms of music fragment similarity.

6. Mixing Songs

MusicMixer mixes songs based on the similarity measurements described above. The combined similarity S between fragments i and j can be calculated as follows:

$$S(i, j) = w \times S_{beat}(i, j) + (1 - w) \times S_{topic}(i, j), \quad (3)$$

where w denotes the weight parameter used to change the balance of the beat and latent topic similarity, which was defined previously as *mixability*.

The length of each song fragment depends on the number of peaks in the beat similarity calculation, i.e., the parameter N. Although the length of each fragment differs, beat similarity ensures that fragments with similar lengths are selected as similar beat fragments. In addition, the fragment lengths are not 5 s (the length for topic analysis). Therefore, we assume that the fragment within a 5 s fragment is similar relative to the latent topic feature of 5 s fragments. Thus, we use the same latent topic feature even though the length of the fragment is not 5 s.

A user can specify the scope as to when mixing can occur. For example, we do not want to switch to a new song at the beginning of the currently playing song or start the next song at the end. In the current implementation, a song will not change until the latter half, and a song will start no later than the first half.

7. Implementation of MusicMixer

7.1 System Design

We implemented a DJ system based on the aforementioned automatic song mixing method. As described above, the objective of the MusicMixer is to support a DJ primarily from the aspect of song mixing. Therefore, as shown in **Fig. 8**, we designed the MusicMixer system to automatically mix songs; however, we allow users to select a song from songs that the system considers as well mixed. As shown in Fig. 8, once the similarity matrix is obtained, MusicMixer works in real-time.

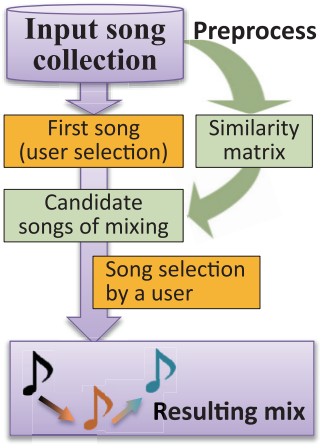


Fig. 8 The system design of MusicMixer.

Input songs are songs that a user might play as a DJ. In this situation, we need to determine the order in which we play these songs and how we mix each pair of songs. The input songs are preprocessed and the system generates a similarity matrix that contains information regarding good mixing points. After preprocessing, DJs can select the first song from a list of input songs. The system then immediately suggests song candidates and corresponding timings. This candidate song suggestion corresponds to supporting DJ mixing with MusicMixer. From the candidates, a user selects the next song. This is the song selection task that the user takes charge of. Note that song selection is not mandatory because the system can automatically mix the next song based on calculated mixability if the user does not directly interact with the system. Thus, MusicMixer supports fully automated DJ performance as well as song mixing support.

7.2 Interface

Figure 9 shows the interface of MusicMixer. The leftmost song list is where users select the first song to play. The selected song then starts immediately; on the timeline at the bottom of the interface, five timings with good mix destinations are displayed. Selected based on point-wise summation of mixability, these five timings will be candidates when mixing occurs. For each mixing timing, the top five mixing destination songs are then displayed as candidates for the next song to be played. Users can either select the next song from the candidates or simply leave the system alone so that the next song will be automatically selected and mixed based on the maximum similarity value. After mixing, the next set of candidate songs will be suggested and the same steps are repeated until the user stops the music. To immediately suggest mixing candidates from such a large similarity matrix, MusicMixer preconstructs the underlying database.

To prevent the five mixing timings from gathering too close to a specific point, candidates are not selected if there is already a candidate that exists within 5 s of the point. Moreover, to prevent the same songs with different destinations within the suggested song from appearing in the same mixing timing, we have restricted the display to no more than three destinations within the same song. Most songs have a repeating structure. Therefore, such a bias often occurs and these constraints are necessary. We also prevent a

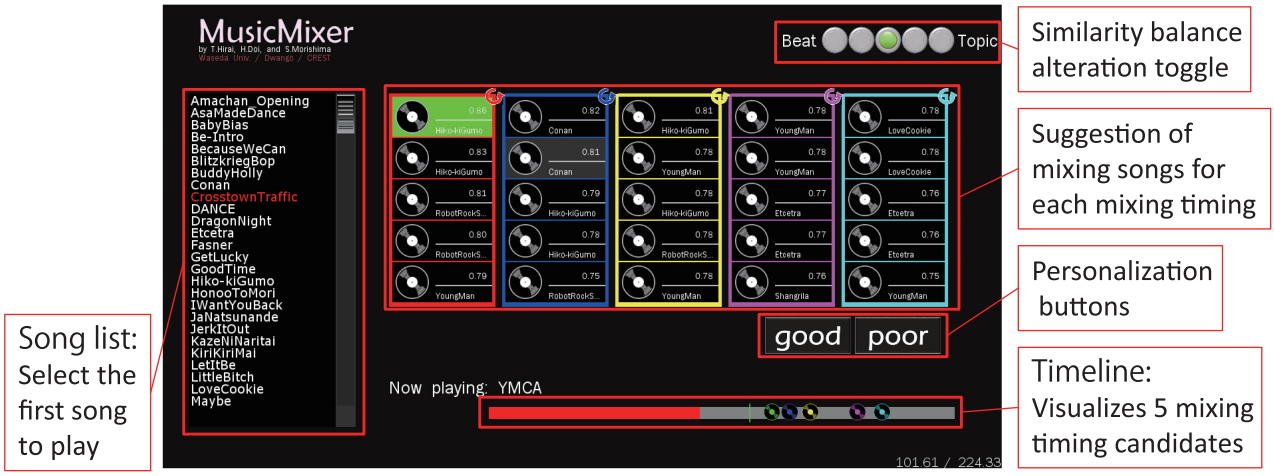


Fig. 9 MusicMixer user interface.

song from changing until somewhere within the latter half of the song; similarly, we prevent a song from starting in the latter half. This will prevent songs from switching in too short a time.

The upper-right toggle in the interface enables the user to alternate the weight between beat similarity and latent topic similarity (the parameter w in Eq. (3)). If a user selects the leftmost toggle, candidate songs will be suggested based only on beat similarity, which is effective for songs that have clear beats, such as dance music; conversely, topic similarity is effective for songs without beats, such as classical music.

Finally, the *good* and *poor* buttons shown on the interface enable users to personalize their automatic song mixing results from user mixing preferences, which we describe in the next section.

7.3 Personalization to the Mixing Preference

Because MuxicMixer does not consider factors other than similarity, it might suggest weird song mixes in some instances. To address this, we include a functionality for remembering "good" mixing and ignoring "bad" mixing; this should help the system become a good automatic DJ.

Good and poor buttons are the means by which a user can provide feedback regarding mixing. If a user selects good, the mixability value for the corresponding mix will be increased, and vice versa, if the poor button is pressed. This function is achieved by preparing a bias matrix. The size of the matrix is the same as the similarity matrix and the values are 0 at the initial state. When a user pushes good button, the bias value will be added to the bias matrix for the mixing. This bias matrix will be added to the similarity matrix when the user wants to consider the preferences.

By mixing songs with MusicMixer and providing such feedback, the mixing made by MusicMixer will be personalized to the user's preferences. Because the sense of good and bad music differs from one individual to another, this functionality helps the system adapt to the user's likes and dislikes.

Moreover, this personalization function can be used to learn the preferences of other DJs. If a professional DJ uses the MusicMixer system, inexperienced individuals can then play the music just as a DJ would by applying the learned preferences. Currently, this functionality depends on a database. We intend to learn the user's preferences by modeling the relations between audio features and user feedback in a future implementation. Liebman et al. [22] proposed a reinforcement learning to recommend music playlists that are adapted to users and such an approach might also be effective for our task.

7.4 Users' Feedback

We demonstrated our system at a workshop of a special interest group for music and computers in Japan. More than 20 people enjoyed the resulting mixes that the system automatically played. Some users however found it difficult to understand how to operate the system without any explanation. Moreover, some felt that the system would be more attractive if there were functions for performing more complicated DJ techniques such as scratching or tempo modifications.

Some individuals noted that the effect of latent topic similarity was not clear compared to the effect of beat similarity. In addition, many people pointed out that they did not know what latent semantics in music meant. This indicates that the latent semantics our method acquires are still not explicit enough. Further user study will be conducted in our future research.

8. Limitations and Applications

8.1 Limitations

MusicMixer considers both beat and latent semantics. However, latent semantics are limited to the chromatic audio signal. Therefore, other types of high-level information such as instrument variations or dynamics within a song cannot be considered in the current implementation. In the future, we will explore the possibility of semantic topic analysis using the symbolic representation acquired from audio signals.

MusicMixer does not consider lyrics or their semantics. Therefore, a summer song may be selected after a winter song, which is undesirable. A user may possibly compensate for such flaws by utilizing a user interface.

As MusicMixer focuses primarily on song mixing, there are limitations in terms of DJ performance. For example, DJ scratching, which is another key DJ technique, cannot be performed using MusicMixer. We currently assume that MusicMixer users are people with no experience on DJ mixing. Therefore, after acquiring a sense of song selection and song mixing with MusicMixer,

it might be easier to use actual turntables.

One strict limitation on the current version of MusicMixer is that the user cannot control the timing of song mixing. The switching timing is disregarded since the five mixing candidates are displayed based on similarity. To make the interface closer to the actual DJ turntable, we will make it possible for the user to control the timing of song mixing by visualizing the similarity information more effectively in future implementation.

Another limitation as an interface is that we do not change the tempo of existing songs. In practice, DJs often change the playback speed to make natural mixes, even though the tempo of two songs may be drastically different. We will implement such tempo alterations to MusicMixer in the future update.

8.2 Applications of MusicMixer

MusicMixer's applications are not limited to being an automatic DJ tool. There is a style of DJ performance referred to as "back-to-back," which is collaborative based on play among multiple DJs. In a back-to-back session, a partner DJ selects the next song while one DJ's song is playing. Thus, the partner DJ's play may be unpredictable. Although the back-to-back style cannot be performed alone, a DJ system such as MusicMixer could act as a partner DJ for a back-to-back performance. This is similar to playing a video game against the computer serves to improve the player's technique. Furthermore, collaboration with a computer might produce a new or unexpected groove.

It is also possible for inexperienced people to practice DJ performance using MusicMixer. For example, mixing songs is the difficult part of a DJ's performance, but song selection might be easier for inexperienced people. In this case, the connection of songs could be performed by the system, and the user can focus on song selection. A DJ performance requires significant skills that can only be acquired from practical experience.

9. Conclusion

We present a topic modeling method to retrieve similar music fragments and its application. Our method allows mixing songs naturally by considering both beat and latent topic similarity. Our main contribution is the application of topic modeling using ChromaWords, which are audio signal-based symbolic representations. Previous topic modeling methods have analyzed the latent topics of audio or images using features represented as a bag-of-features; hence, the meaning of the topic was not clear. We have improved its interpretability by using ChromaWords. Furthermore, the results from a subjective evaluation indicate that our topic modeling method outperforms other methods in terms of music similarity. Topic modeling is primarily used to analyze latent semantics in observed data. The proposed method makes it possible to employ the latent semantics of chromatic sounds. However, the semantics of chromatic sounds do not cover all the semantics of a song. Thus, in the future, we plan to consider other semantics such as timbre. We have also presented MusicMixer, which is a computer-aided DJ system based on the music retrieval using above-mentioned similarity.

This study focuses on song mixing without changing the original songs. In a future implementation, the proposed system will do song modulation to allow free connection of any type of song pairs. For example, by using a song morphing method [14], it may be possible to embed such a function.

MusicMixer takes advantage of computing power to calculate good mixing points of input songs and enables the user to select the next song from the suggestions provided by the system. This can be regarded as one style of collaborative content generation between humans and computers.

MusicMixer not only supports people who are performing DJ mixing but also enhances the listening experience. As it is difficult to hire personal DJs for all events, particularly on a day-to-day basis, MusicMixer can serve as a user's personal DJ fully integrated within an existing music player as an application.

We plan to explore further possibilities of human–computer collaboration for DJ performance as well as the possibility of human–computer complementation.

References

[1] Blei, D.M., Ng, A.Y. and Jordan, M.I.: Latent Dirichlet allocation, *Journal of Machine Learning Research*, Vol.3, pp.993–1022 (2003).

[2] Ishizaki, H., Hoashi, K. and Takishima, Y.: Full-Automatic DJ Mixing System with Optimal Tempo Adjustment based on Measurement Function of User Discomfort, *Proc. 10th International Society for Music Information Retrieval*, pp.135–140 (2009).

[3] Cliff, D.: Hang the DJ: Automatic Sequencing and Seamless Mixing of Dance-Music Tracks, *HP LABORATORIES TECHNICAL REPORT HPL*, No.104 (2000).

[4] Laursen, L.F., Goto, M. and Igarashi, T.: A Multi-Touch DJ Interface with Remote Audience Feedback, *Proc. 22nd ACM International Conference on Multimedia*, pp.1225–1228 (2014).

[5] Kapur, A., Benning, M. and Tzanetakis, G.: Query-by-beat-boxing: Music retrieval for the DJ, *Proc. 5th International Conference on Music Information Retrieval*, pp.170–177 (2004).

[6] Platt, J.C., Burges, C.J., Swenson, S., Weare, C. and Zheng, A.: Learning a Gaussian Process Prior for Automatically Generating Music Playlists, *Proc. 14th Advances in Neural Information Processing Systems*, pp.1425–1432 (2001).

[7] Aucouturier, J.-J. and Pachet, F.: Scaling Up Music Playlist Generation, *Proc. IEEE International Conference on Multimedia and Expo 2002*, Vol.1, pp.105–108 (2002).

[8] Pampalk, E., Pohle, T. and Widmer, G.: Dynamic Playlist Generation Based on Skipping Behavior, *Proc. 6th International Conference for Music Information Retrieval*, Vol.5, pp.634–637 (2005).

[9] Ragno, R., Burges, C.J. and Herley, C.: Inferring Similarity Between Music Objects with Application to Playlist Generation, *Proc. 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pp.73–80 (2005).

[10] Nakano, T., Kato, J., Hamasaki, M. and Goto, M.: PlaylistPlayer: An Interface Using Multiple Criteria to Change the Playback Order of a Music Playlist, *Proc. 21st ACM International Conference on Intelligent User Interfaces*, pp.186–190 (2016).

[11] Goto, M. and Goto, T.: Musicream: Integrated Music-Listening Interface for Active, Flexible, and Unexpected Encounters with Musical Pieces, *Information and Media Technologies*, Vol.5, No.1, pp.139–152 (2010).

[12] Davies, M.E., Hamel, P., Yoshii, K. and Goto, M.: AutoMashUpper: Automatic Creation of Multi-Song Music Mashups, *IEEE/ACM Trans. Audio, Speech, and Language Processing*, Vol.22, No.12, pp.1726–1737 (2014).

[13] Tokui, N.: Massh!: A Web-based Collective Music Mashup System, *Proc. 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, pp.526–527, ACM (2008).

[14] Hirai, T., Sasaki, S. and Morishima, S.: MusicMean: Fusion-Based Music Generation, *Proc. 12th Sound and Music Computing Conference*, pp.323–327 (2015).

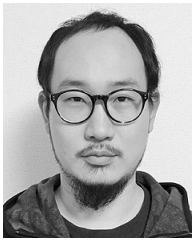
[15] Sasaki, S., Yoshii, K., Nakano, T., Goto, M. and Morishima, S.: LyricsRadar: A Lyrics Retrieval System Based on Latent Topics of Lyrics, *Proc. 15th International Society for Music Information Retrieval*, pp.585–590 (2014).

[16] Sivic, J. and Zisserman, A.: Video Google: A Text Retrieval Approach to Object Matching in Videos, *Proc. International Conference on Computer Vision 2003*, Vol.2, pp.1470–1477 (2003).

[17] Nakano, T., Yoshii, K. and Goto, M.: Vocal Timbre Analysis using

Latent Dirichlet Allocation and Cross-Gender Vocal Timbre Similarity, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing 2014*, pp.5202–5206 (2014).
[18] Hu, D.J. and Saul, L.K.: A Probabilistic Topic Model for Unsupervised Learning of Musical Key-Profiles, *Proc. 10th International Society for Music Information Retrieval*, pp.441–446 (2009).
[19] Hu, D.J. and Saul, L.K.: A Probabilistic Topic Model for Music Analysis, *Proc. 22nd Advances in Neural Information Processing Systems*, Vol.9 (2009).
[20] Lartillot, O., Toiviainen, P. and Eerola, T.: A Matlab Toolbox for Music Information Retrieval, *Data Analysis, Machine Learning and Applications*, pp.261–268 (2008).
[21] Goto, M.: Development of the RWC music database, *Proc. 18th International Congress on Acoustics*, Vol.1, pp.553–556 (2004).
[22] Liebman, E., Saar-Tsechansky, M. and Stone, P.: Dj-mc: A reinforcement-learning agent for music playlist recommendation, *Proc. International Conference on Autonomous Agents & MultiAgent Systems*, pp.591–599 (2015).

Tatsunori Hirai was born in 1988. He received his B.S., M.S. and Ph.D. degrees from Waseda University in 2011, 2012, and 2015, respectively. He joined the Information Processing Society of Japan in 2010. He is currently an assistant professor at faculty of global media studies, Komazawa University, Tokyo, Japan. His research interest is multimedia content processing.



Hironori Doi graduated from the Department of Information and Image Science, Faculty of Engineering, Chiba University, Japan in 2008. He received his M.E. and D.E. degrees from the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Japan, in 2009 and 2012, respectively. He currently works for DWANGO Co., Ltd., Japan. He mainly studies singing voice conversion and speaking-aid systems.

Shigeo Morishima was born in 1959. He received his B.S., M.S. and Ph.D. degrees, all in Electrical Engineering from the University of Tokyo, Tokyo, Japan, in 1982, 1984, and 1987, respectively. Currently, he is a professor at School of Advanced Science and Engineering, Waseda University, Tokyo, Japan. His research interests include 3D Reconstruction and Modeling of Face, Motion Analysis and Synthesis of Human Body, Analysis and Synthesis of Facial Expression and Retargetting, and all concerning about Future Interactive Entertainment using speech and image processing.