J Supercomput (2015) 71:1933–1954
DOI 10.1007/s11227-014-1182-1

Social mix: automatic music recommendation and mixing scheme based on social network analysis

Sanghoon Jun · Daehoon Kim · Mina Jeon · Seungmin Rho · Eenjun Hwang

Published online: 26 April 2014

Abstract General preferences for music change over time. Moreover, music preferences depend on diverse factors, such as language, people, location, and culture. This dependency should be carefully considered to provide satisfactory music recommendations. Presently, typical music recommendations simply involve providing a list of songs that are then played sequentially or randomly. Recently, there has been an increasing demand for new music recommendation and playback methods. In this paper, we propose a scheme for recommending music automatically by considering both personal and general musical predilections, and for blending such music into a mixed clip for seamless playback. For automatic music recommendations, we first analyze social networks to identify a general predilection for certain music genres that depends on time and location. Songs that are generally preferred within a certain time period and location are identified through statistical analysis. This is done by analyzing, filtering, and storing massive social network streams into our own database in real time. In addition, a personal predilection for certain music genres can be inferred by analyzing similar user relationships in social network services. We selected such

S. Jun · D. Kim · M. Jeon · E. Hwang (✉)
School of Electrical Engineering, Korea University, Seoul 136-713, Korea
e-mail: ehwang04@korea.ac.kr

S. Jun
e-mail: ysbhjun@korea.ac.kr

D. Kim
e-mail: kdh812@korea.ac.kr

M. Jeon
e-mail: jmina@korea.ac.kr

S. Rho
Department of Multimedia, Sungkyul University, Anyang, Korea
e-mail: smrho@sungkyul.edu

music based on instant graphs that are generated by user relationships and underlying music information. After the songs are selected, an automatic music mixing method is used to blend those songs into a continuous music clip. We implemented a prototype system and experimentally confirmed that our scheme provides satisfactory results.

Keywords Music recommendation · Social network service · Twitter · Music structure · Music mixing

1 Introduction

Typical music recommendations merely provide a set of songs, which are played sequentially or randomly. For efficient music recommendation, there are two main issues: predicting a user's preference and searching for proper songs based on the user's preference. Presently, many studies have been conducted to identify a song search condition that will produce accurate search results. One of the most important search conditions in music recommendation is the user's musical predilection. Hence, music recommendation systems usually attempt to extract accurate predilections from a person or from the public, and find songs accordingly. One important issue in considering musical predilections is that such preferences change over time. Furthermore, music preferences exhibit different characteristics depending on people, nation, or culture. Thus, personal and public preferences should be considered as adaptive for satisfactory music recommendations.

One popular method for selecting songs in music recommendation systems is to use the statistical data from the music listening history. In other words, by analyzing the listening history of many users, the popularity scores of songs are determined, and the popular ones are provided or recommended to a user. For accurate analysis, a considerable amount of user history records are required; however, it has proven difficult to collect such an amount of records. Thus, common music recommendation systems have employed records provided by popular music streaming services, such as Last.fm. Usually, such records are extremely simple and do not contain important history data, such as location and time; this limitation has proven to be an obstacle in developing diverse user-friendly applications.

With the wide use of mobile networks and devices in recent years, various social network services (SNSs) have become available. Consequently, users can create their own content and share it through SNS without time and space restrictions. With the flood and popularity of various content, effectively searching for appropriate content has become an essential part of user satisfaction. Many researchers have observed that such content shows a sequence of temporal and spatial patterns called trends, and have attempted to accurately extract and effectively utilize such trends [1–5].

In this paper, we propose a new music recommendation scheme that calculates a sequence of songs for recommendation by considering trends and spatial–temporal user preferences for music, and then blends these songs into a single mixed clip for seamless playback. More specifically, we extract the general and personal predilections for music genres by analyzing massive user data that are collected from SNSs, and based on that, compile a list of songs for recommendation. By monitoring continuously

generated social network data in real time, we can calculate the most recent popular songs. Furthermore, by considering the time, location, nation, and language data of social networks, the accuracy of the song selections can be improved considerably. In addition, by mining underlying music preference relationships among users of SNSs, personalized music recommendations can be achieved.

In 2014, Twitter users generate 58 million tweets daily and 9,100 tweets per second [7]. To manage such an amount of tweets, effective methods for collecting and refining these tweets are required. There are two access types for collecting tweets from the Twitter server: Streaming and REST [8]. The streaming type maintains a persistent connection to the server and collects generated tweets consistently. The REST type makes a connection to the server upon request and collects information for the request. In this paper, we use both of these types; to analyze the general music preference, we consistently collect streaming tweets. In addition, because of storage requirements, we do not store all the tweets into a database. Instead, we filter music-related tweets and extract music-related information from such tweets in real time, thus allowing our system to select the songs that are generally preferred. The REST type request is used for personalized music recommendations. First, we find users who share a similar predilection and select songs that were played at least once by the users. More details are provided in Sect. 3.

On the other hand, most existing music recommendation systems play selected songs sequentially or randomly. To avoid the uninteresting playback of the traditional method, in this paper, we propose a novel method that provides a set of songs in an entirely new music format by blending the recommended songs in a continuous music clip. This blending is performed based on an analysis of songs' signal features. The blending result is similar to listening to a disk jockey's (DJ's) mixtape or radio station. Our proposed method rearranges the songs and blends them based on their signal features. This is done by segmenting music into homogeneous clips via structure analysis, extracting their musical features, and pairing the best matching segments in terms of their musical features. The neighboring songs are blended into one continuous song by one of the several mixing techniques and beat grids. More details are provided in Sect. 3.

2 Related work

To use SNS data effectively for music recommendations, it is necessary to extract trends from SNS messages. In this section, we introduce certain recent efforts in the area of SNS analysis and playlist generation. In addition, we investigate song mixing techniques and applications.

2.1 SNS analysis

Because a significant number of messages are generated every second via SNS, many works for SNS analysis have considered SNS messages as streaming data or time sequence data. By compiling statistics with various methods, such works attempt to extract trends from SNS messages. Mathioudakis et al. [1] propose TwitterMonitor to

detect trends over social network streams. The system identifies emerging topics on Twitter in real time, and provides meaningful analytics for the accurate description of each topic. However, TwitterMonitor can provide information about various events using single bursty keywords only. To overcome this limitation, Alvanaki et al. [2,3] propose an approach for automatically detecting emergent topics by detecting shifts in tag correlations as they dynamically arise. In addition, several studies have focused on extracting bursty keywords from text streams that arrive continuously over time. Benhardus et al. [4] propose a scheme for detecting trends from tweet streams over multiple timespans. To do this, Benhardus et al. performed a term frequency-inverse document frequency analysis and a relative normalized term frequency analysis that detects the unigram, bigram, and trigram of trending topics. Kleinberg [5] proposes an approach for modeling bursty keywords that suddenly increase in frequency. Similar to queuing theory models, their approach was based on an infinite-state automaton for modeling a text stream. Hierarchical structures for sets of bursts are generated for overall streams.

2.2 Playlist generation

Playlist generation is a popular method for recommending music to users. Platt et al. [11] suggest the AutoDJ system for automatically generating music playlists based on one or more seed songs selected by a user. That is, when the user inputs one or several songs, a playlist of songs similar to the seed songs is generated. Pauws et al. [12] propose the personalized automatic track selection (PATS). PATS generates playlists that suit a particular usage context. To create playlists, PATS uses a dynamic clustering method in which songs are grouped based on their attribute similarity. Andric et al. [13] demonstrate a method to track a user's listening habits, and how it can be used to generate playlists automatically. Andric et al. focus on examining the user's listening habits, and describe two possible ways to represent this information in a compact and expressive way (listener model). Reynolds et al. [14] show an overview of existing methods for automatically generating a playlist. In particular, Reynolds et al. claim that both contextual information (such as a listener's mood) and environmental information (such as location, activity, and weather) are highly valuable and appropriate when suggesting or ordering music tracks for a listener. Although these approaches focus on user preference and contextual information, they do not consider different music preferences that depend on time and location.

2.3 Song mixing

Song mixing (DJ mixing) is the art and science of playing music [15]. Usually, song mixing is used by DJs to combine a sequence of tracks such that they appear as one continuous track. This technique is used when or where multiple songs need to be played continuously, for example, for music radio broadcasting and at nightclubs. For music mixing, we require two music players and one mixer. Music players, such as turntables and CD players, are used to play song tracks. A mixer is used to blend through volume control the songs that originate from the players. DJ mixing starts

with playing a song track on one music player. While the song track is playing, the DJ prepares the next song track on the second player. At this point, the DJ should decide the following: the song to mix, where to mix it, and when to mix it. After selecting the song track that will be mixed with the current track, the DJ decides the cue point at which the second track will start. After preparing the second track, the DJ listens to the current track and determines when to play the second track. To decide this, certain music rules, such as music phrase and structure, should be considered. To start mixing, the DJ adjusts the tempo of the second track to that of current track and starts the second music player at the cue point. Song tracks are mixed by blending volumes smoothly and gradually using cross-fader on the mixer. These steps are performed repeatedly for constructing one continuous track from multiple tracks.

Cliff [16,17] presents an automatic DJ system in which compilations of dance music can be sequenced and seamlessly mixed automatically, with minimal user involvement. The system automates a DJ's techniques, such as beat matching and cross-padding. Based on the feedback from the crowd on the dance floor, the system selects the next track to be mixed automatically.

3 Overall system architecture

In this section, we describe the overall architecture of our prototype system and some of the implementation details. Our prototype system runs on the Amazon elastic compute cloud (Amazon EC2) service [18] and was implemented using Cloudera CDH [20,21] which is an open-source Apache Hadoop distribution. We also used Cloudera Impala [19] as the main database because it supports real-time query on distributed Hadoop clusters. To refer to music information locally, we use the Musicbrainz database [22], which is updated regularly.

As shown in Fig. 1, our system consists of two main components: SNS collector and Music Recommender. The SNS collector gathers raw tweet data from the Twitter server and extracts music-related information, which is stored in a music database. The Music Recommender supports two recommendation modes for users: general recommendation and personalized recommendation. For general recommendation, the recommender selects songs from the music database based on a user's temporal and spatial properties. For personalized recommendation, the recommender selects songs, artists, or genres based on a user's seed songs. At this time, the recommender gathers tweets using keywords such as song title, artist name, and genre to identify music-related information. A music playlist is composed of the selected songs, and finally, these songs are blended into a continuous song by performing music structure analysis and mixing segments with similar musical features.

3.1 SNS collector

3.1.1 Filtering music-related tweets

Twitter users can define hashtags to indicate the keywords or topics of their tweets. A hashtag starts with the symbol # and is followed by a related keyword or phrase.

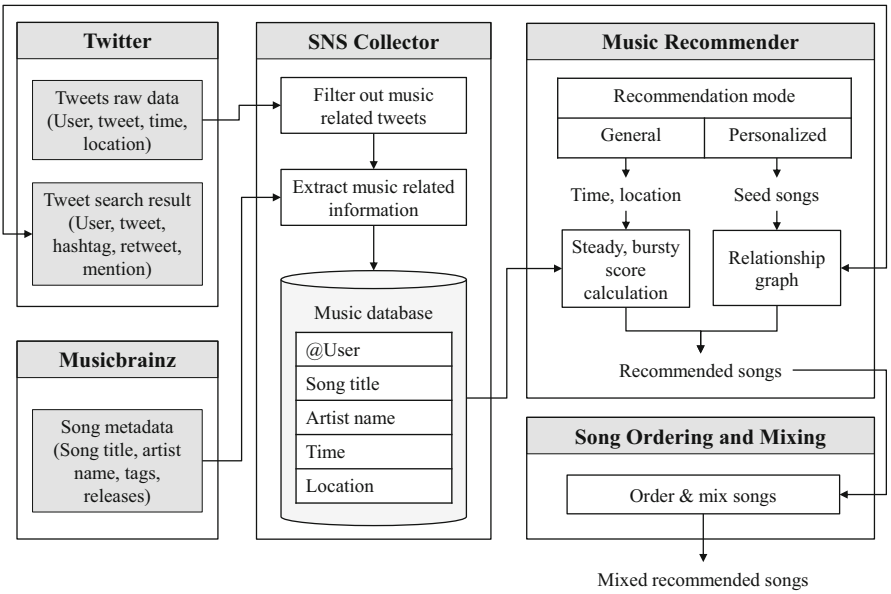


Fig. 1 Overall system architecture

Table 1 Music-related hashtags

Rank	Hashtag	Popularity
15	#nowplaying	0.010138
26	#np	0.005288
30	#music	0.004023
37	#soundcloud	0.003214
63	#musicfans	0.002219
102	#listenlive	0.001583
202	#hiphop	0.000889
219	#musicmondays	0.000827
278	#pandora	0.000637
308	#mp3	0.000595
356	#itunes	0.000480
372	#newmusic	0.000464

Hashtags are extremely useful for searching relevant tweets and for grouping tweets based on topics. In this paper, we use the hashtag as a clue for identifying music-related tweets. We first collected popular hashtags for the year 2013 from [23]. Then, we selected music-related hashtags as shown in Table 1.

Using these hashtags, we performed keyword filtering to collect relevant tweets in real time. In other words, we collected tweets that contain at least one music-related hashtag. Other tweet information we collected includes tweet user ID, user profile location, posting location, and generated time.

Table 2 Typical syntax for representing song title and artist

<Phrase A> by < Phrase B>
< Phrase A> − < Phrase B >
< Phrase A > / < Phrase B >
"< Phrase A >" − < Phrase B >

3.1.2 Extracting music-related information

Even a music-related tweet message might contain information that is not relevant to music. To remove such non-music information, we performed several preprocessing tasks to the tweets to analyze their content. The collected tweets might contain username (starts with the @ character to send messages), URLs, and special characters. Thus, we removed words that start with @, URLs, and special characters (such as $*\%!? > \$\hat{} \ \& < \{\}$).

After preprocessing, we analyzed the syntactic features of user tweets to extract music-related information, such as song title and artist. Table 2 illustrates the typical syntax used for representing song title and artist.

For instance, if a tweet text matches the syntax, the two resulting phrases could be artist and song title. The first case in the table shows that "by" can be a satisfactory clue for inferring phrases A and B as song title and artist, respectively. On the contrary, in the second and third cases, song title can be located at either phrase A or B. Hence, it is not clear which phrase is the song title in these cases. To validate song title and artist name after pattern matching, we use Musicbrainz [22], a music information database that can be migrated onto our music database. Last.fm [24] is a similar online database, but we chose Musicbrainz because it is more appropriate for processing a significant number of tweet queries in real time. After pattern matching, two phrases are validated by referencing the artist table in the local Musicbrainz database. If a phrase matches an artist name in the database, we consider the other phrase to be the song title. After validation, we can gather additional metadata from the Musicbrainz database using the song title and the artist name.

3.1.3 Music database

By performing music tweet filtering and music information extraction, our proposed system can identify tweets that contain music information in real time. Such information collected from tweets is indexed by artist name and is stored sequentially in our music database. Using the artist name as the database key, we can easily acquire relevant information from the table in the database. Each tuple in the table represents a unique song and consists of the following information:

$$\text{Song} = \{\text{Artist name, Song title, Number of occurrences, Location information}\}$$

Location information, which represents where the tweet is posted, is collected for all tweets that refer to the song. In our previous study [6], we investigated the GPS coordinates of 1.25 million tweets and analyzed them. However, according to our analysis,

Algorithm SNS Collector
Input: Pushed tweet stream $\boldsymbol{T}$

Tweet stream $\boldsymbol{T}$ = { t | t is a tweet of (*text*, *gps location* and *profile location*) }
Music database $\boldsymbol{M}$ = { m | m is a database tuple of (*artist name*, *song title*, *number of occurrence*, *location*) }
for each t in $\boldsymbol{T}$ **do**
  filter tweets having music-related hashtags in their text
  extract phrase_A and phrase_B using regular expressions
  query phrases to local Musicbrainz database
  if there exists a tuple in Musicbrainz DB with phrase_A as artist
    $m_{\text{artistname}} = \text{phrase}_A$
    $m_{\text{songtitle}} = \text{phrase}_B$
  else
    $m_{\text{artistname}} = \text{phrase}_B$
    $m_{\text{songtitle}} = \text{phrase}_A$
  end
  if t_{gps} is not null
    $m_{\text{location}} = t_{\text{gps}}$
  else
    $m_{\text{location}} = t_{\text{profilelocation}}$
  end
  if there exists a tuple in **M** with $m_{\text{artistname}}$ and $m_{\text{songtitle}}$ as its artist and title
    increase its $m_{\#\text{of occurrence}}$ by one
  else
    insert ($m_{\text{artistname}}$, $m_{\text{songtitle}}$, 1, m_{location}) into $\boldsymbol{M}$
  end
end

Fig. 2 Algorithm for collecting music information from SNS

only 1.19 % of tweets contain exact GPS coordinates. To overcome the scarcity of location information in the tweets, we consider the profile location registered by the user, from which we can obtain the location information for 65.55 % of tweets.

Tables in the music database generated through the methods described in the previous paragraphs are backed up and initialized every 2 h. As a result, the system stores tweets on a 2-h period, which makes it possible to trace song popularity over time. The overall steps for the SNS collector are described in Fig. 2.

3.2 Music recommender

3.2.1 General music recommendations

General music recommendations aim to provide a set of songs that have gained popularity. By analyzing massive music-related tweets, we can calculate a list of songs

that are preferred by many users. However, such music preferences have some dependency. For instance, music that is popular in Europe could be different from that in Asia. Hence, for more accurate and effective general music recommendations, we should consider two factors: time and location. To consider these factors, we provide several options for music recommendations to users.

First, general music preferences change over time. For example, certain people prefer songs that are steadily loved by many people. On the other hand, certain other people might prefer trendy songs that have recently become loved by many other people. Based on the time period, we defined two different types of popularity: steady and bursty.

Our proposed system estimates how bursty or steady a particular song is based on periodically collected music information by the SNS collector. For each song, we calculated two values that are between zero and one, inclusively. Steady score S_i and Bursty score B_i are calculated by the following equations.

$$S_i(t-n,t) = 1 - \frac{\sqrt{\frac{\sum_{x\in[t-n,t]}(SF_i(x)-\overline{SF}_i)^2}{n}}}{\overline{SF}_i}$$

$$B_i(t-n,t) = \frac{\sum_{x\in[t-n+1,t]} R_i(x)}{\overline{SF}_i}$$

$$\overline{SF}_i = \frac{\sum_{x\in[t-n,t]} SF_i(x)}{n},\ R_i(t) = \begin{cases} SF_i(t) - SF_i(t-1), & \text{if } SF_i(t) > SF_i(t-1) \\ 0 & \text{otherwise} \end{cases}$$

Here, $SF_i(t)$ is the occurrence count of song i. Time t can represent either the current time or a particular time in the past. If a user does not indicate a specific time, t represents the current time. n is an integer that indicates the back-up period unit of the SNS collector. The steady score indicates how steady a song has been. In essence, it represents the degree of standard deviation of occurrence counts. In other words, a higher steady score implies less fluctuation of occurrence counts in a given time period. For instance, there are two songs s_A and s_B that have the same standard deviation of 200. The average occurrence counts of s_A and s_B are 30,000 and 500, respectively. s_A has a deviation of approximately 200 near 3,000 and s_B has the same deviation near 500. In this case, we can concur that s_A has more steadiness than s_B. On the other hand, the bursty score indicates how aggressively a song's popularity fluctuates in a particular period. Hence, a higher bursty score for a song indicates that the song gains or loses popularity more abruptly in the time period. Using steady and bursty scores, candidate songs are sorted for recommendation, and the top ranked songs are selected for general recommendation.

Second, general music preferences can differ regionally. Depending on the culture and language, regions show different general musical predilections. To represent this regional difference, we divided the surface of the earth into 169 regions. Territories are grouped into six continents as described in [25]. A user can select a specific region for music recommendations. Furthermore, the region can be extended to a continent and to the entire world.

Algorithm Generating Relationship Graph
Input: Seed songs $\boldsymbol{SS} = \{ss_1, \ldots, ss_n\}$
Output: Related songs(nodes) $\boldsymbol{S} = \{s_1, \ldots, s_m\}$
Weight vector(edges) $\boldsymbol{W} = \{w_1, \ldots, w_m\}$

initialize score list of related songs $\boldsymbol{C}$
for each seed song in $\boldsymbol{SS} = \{ss_1, \ldots, ss_n\}$ **do**
  query tweets by music hashtag(Table 1), song title and artist
  for each user of retrieved tweets **do**
    query other tweets having music hashtag from the user
    for each retrieved tweet **do**
      extract music information s from the tweet
      if s does not exist in $\boldsymbol{S}$
        append s to $\boldsymbol{S}$
        append c = (1+ no. of retweets) to $\boldsymbol{C}$
      else if s exists in $\boldsymbol{S}$ at position j
        increase its c_j **by** (1+ no. of retweets)
    end
  end
end
for $i = 1$ to m **do**
  set $w_i = c_i \div m$ to $\boldsymbol{W}$
end
return $\boldsymbol{S}$, $\boldsymbol{W}$

Fig. 3 Generating relationship graph algorithm

3.2.2 Personalized music recommendations

For personalized music recommendations, our proposed system selects songs by revealing the underlying song relationships within the ever-changing social network data. To do this, seed songs are required to establish the basis for finding-related songs. Seed songs can be chosen by a user or collected automatically from the user's previous playback history. In this paper, we propose a graph generation method that infers underlying relationships among social network data. The detailed algorithm is described in Fig. 3.

First, we search for users R from Twitter who played at least one seed song. Subsequently, we search for other songs that users R have played previously. To perform this efficiently, we use the hashtags that we described previously for finding the users' tweets. Music information such as song title and artist name is extracted from the tweets. The related scores for each song are calculated by adding up the occurrence count and the retweet count of the tweets. By dividing the counts into the sum of related scores, the related weights are calculated as the weight of edges in the graph. A related weight represents the probability of being recommended. In other words,

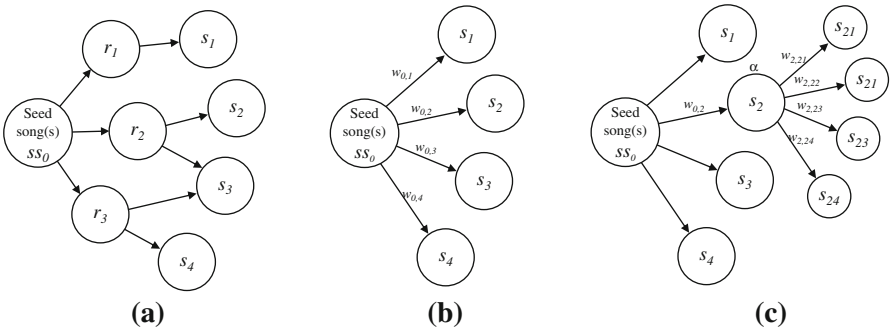


Fig. 4 Relationship graphs between songs

the song that has a higher related weight is more likely to be selected for recommendation. The overall process for generating the relationship graph is shown in Fig. 4a. The process can be abstracted as shown in Fig. 4b. The generated graph is cached in the music database for a short time to avoid additional contact with the Twitter server.

Although the recommendation method that is based on the relationship graph can utilize social network data generated in real time, its results could still be limited depending on the number of relationships. To remedy this problem, we extend this method such that extended relationship graphs can be considered simultaneously. First, we generate an initial relationship graph for a seed song. Based on the weights of other related songs, one of the songs is selected for recommendation. At the same time, we assign a seed probability α to the selected song. The selected song can become a new seed song depending on the seed probability. In that case, we generate another relationship graph for the selected song, which provides an extended relationship graph. Otherwise, the song is counted as a candidate song for recommendation. By repeating all the steps from generation of the initial relationship graph to song selection N times, we can generate N songs for recommendation. The detailed algorithm is shown in Fig. 5.

By adjusting the seed probability, a user can control the diversity and relevance of the recommendation results. That is, a higher seed probability provides more diverse recommendation results with lower relevance to the seed songs.

3.3 Song ordering and mixing

In this section, we describe a novel music ordering and mixing method for blending selected songs into a single continuous music clip, as shown in Fig. 6. The overall steps for music ordering and mixing are shown in Fig. 7. The music recommendation method described previously returns a set of songs for playback. Even though these songs can be played in a traditional way, our proposed method plays the songs similar to the way professional DJs play songs. To do this, first, the set of songs are analyzed and segmented. Features that represent song characteristics are extracted for ordering and mixing. The set of songs should be ordered such that neighboring songs are well overlapped and hence, well mixed. Using song mixing methods, overlapped songs are blended by finding well-matched parts.

Algorithm Recommending N songs using extended relation graph
Input: Seed songs $\boldsymbol{SS} = \{ss_1, \ldots, ss_n\}$, number of recommendation N
Output: Recommended songs $\boldsymbol{RC} = \{rc_1, \ldots, rc_N\}$

generate relation graph RG_0 using seed songs $\boldsymbol{SS}$
related songs(nodes) $\boldsymbol{S} = \{s_1, \ldots, s_m\}$
weight vector(edges) $\boldsymbol{W} = \{w_1, \ldots, w_m\}$
for $i = 1$ to N **do**
select a song s in $\boldsymbol{S}$ based on probability in $\boldsymbol{W}$
determine whether to re-seed or not based on seed probability α
if (re-seed) or (s exists in $\boldsymbol{RC}$)
generate a relationship graph RG' using seed song s
repeat the loop using RG'
else
add s into recommended songs $\boldsymbol{RC}$
end
end
return $\boldsymbol{RC}$

Fig. 5 Recommending N songs using extended relationship graph algorithm

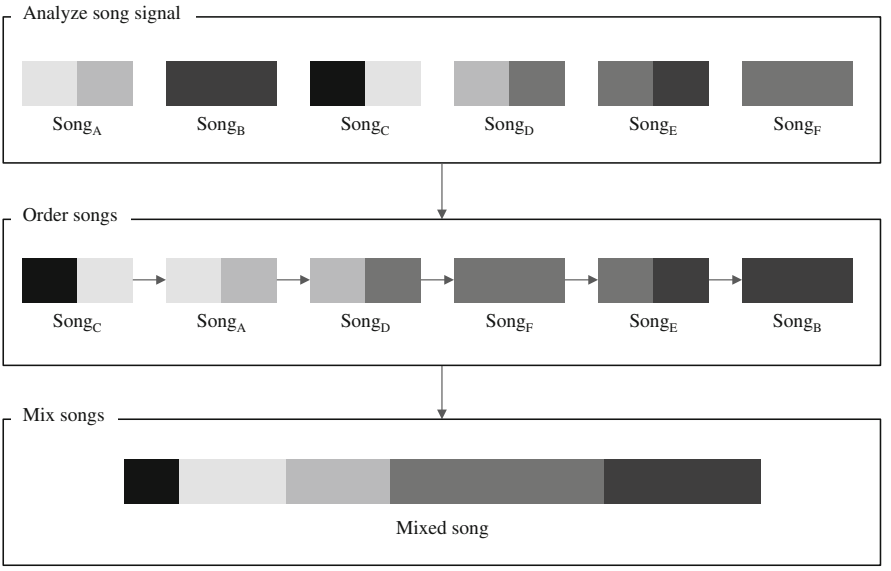


Fig. 6 Example of song ordering and mixing

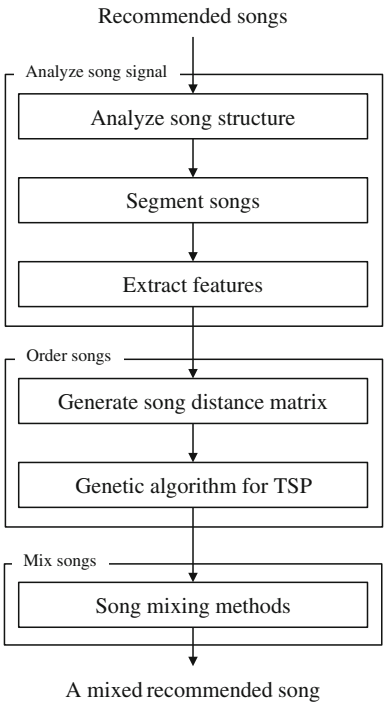


Fig. 7 Overall process for song ordering and mixing

3.3.1 Signal analysis

A song usually consists of a few musically distinct segments. Such distinction can be identified by analyzing various musical features. Music structure analysis divides music clips into such distinct segments based on their musical feature.

Jun et al. [26] present a method for segmenting and summarizing music based on its structure analysis. To do this, we construct a self-similarity matrix. The matrix can be obtained in a 2D domain by measuring cosine similarities between the successive music signals. The matrix can be constructed based on a beat-based frame as described above. By finding patterns within the matrix, segment boundaries where music timbre changes significantly can be recognized. Figure 8 shows the overall steps for music structure analysis using a self-similarity matrix. Greater details are provided in [26].

Typically, a song track is divided into five to six segments by structure analysis. Segments longer than 30 s are considered. This is because short segments usually represent a musical transition.

To represent music characteristics, we use ten features, including intensity, mel-frequency cepstral coefficients, spectral centroid, roll-off, flux, and zero crossing rate features. A ten-dimension feature vector is normalized to zero-mean and uniform standard deviation. Then, the principal coefficient analysis (PCA) method is applied to reduce the dimensionality of vectors. In addition, for harmonic mix, the keys of each track are extracted by chromagram [27]. They are all stored in our music database.

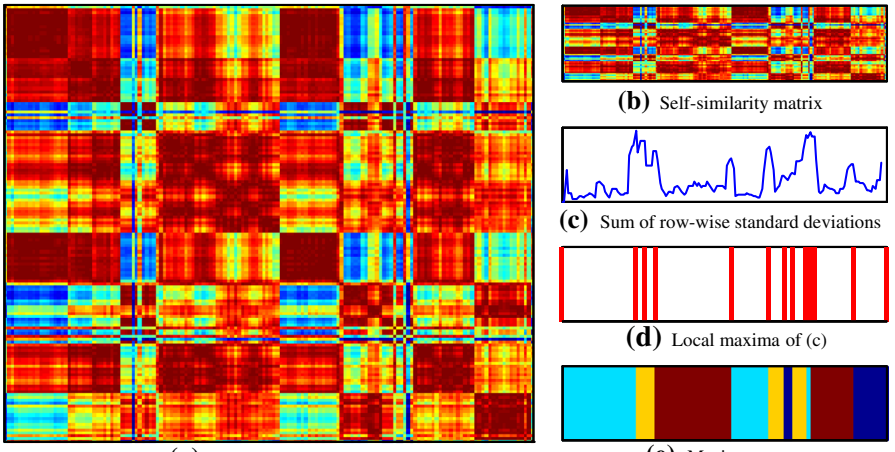


Fig. 8 Steps for music structure analysis

3.3.2 Ordering recommended songs

For smooth song mixing, a pair of songs that will be mixed should have similar music characteristics. To find the best ordering of a fixed number of songs for mixing, signal or harmonic similarity should be considered. To do this, we extract the musical features of songs and construct a song relationship matrix that represents the similarities among songs. The $N \times N$ song distance matrix SDM is defined by the Euclidean distance of feature vectors,

$$\mathrm{SDM}(i, j) = \mathrm{dist}(f_{\text{last sement of song}}\, i,\ f_{\text{first sement of song}}\, j) \quad (i, j = 1, \ldots, N),$$

where f is the feature vector of a segment and dist is the Euclidean distance measurement function. Because our mixing method occurs in a unit of segment, we calculate the similarity of the last and first segments of neighboring songs.

Sometimes, the Euclidian distance metric on the feature vector might not be the best choice. That is, it might be better to mix tracks with harmonic keys, even though their signal similarity is low. Harmonic mixing is an advanced DJ technique to mix tracks. By mixing tracks that are in the same or related keys, harmonic mixing could provide a well-blended mixset. The Camelot wheel (cycle of fifths) illustrates related keys as circular table [28]. In this paper, we consider that two songs have zero-distance if they contain related keys.

$$\mathrm{SDM}(i, j) = 0 \text{ (if songs } i \text{ and } j \text{ contain related keys)}$$

The song distance matrix is used to determine the ordering of songs. Ordering songs is similar to the traveling salesman problem (TSP) in that it finds the optimal ordering that has the shortest total distance. In TSP, a single salesman (player) should complete the route (playlist) by traveling to each of the cities (songs). Each city (song) should

be visited by the salesman (song player) exactly once. The popular method for solving TSP is to use a genetic algorithm for finding a near exact solution [29]. The genetic algorithm selects a few orders among the initial population. Through evolution, the song order is modified by cross-over, mutation, and survival processes. In this paper, we limit the number of iterations to 10,000 and the size of the population to 60. Using our song distance matrix and the genetic algorithm, the order of the songs that minimizes the distance is approximated.

3.3.3 Mixing songs

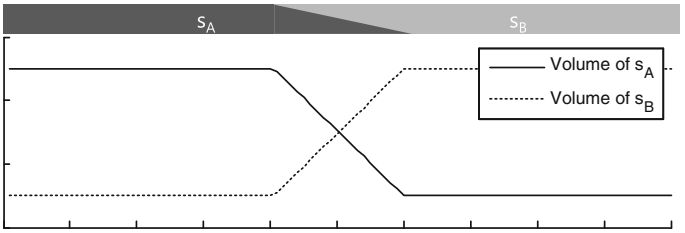
After songs are ordered, they are synchronized by tempo, and then mixed by beat. There are two methods for song mixing: cross-fading and cutting. Suppose that we mix two songs s_A and s_B in that order. In the cross-fading mixing method, s_A is played first and in the mean time its tempo is calculated. At the beginning of the last segment, the method plays s_B at that tempo with the minimum volume. By gradually decreasing the volume of s_A and increasing that of s_B, the method shows a smooth transition from s_A to s_B. When the transition is completed, s_A is stopped with the minimum volume and $s_B$is played on. On the other hand, cutting mixing method is very simple. s_A is played first. On its last segment, s_A is stopped and s_B is played on. Hence, the cutting mixing method shows an abrupt transition between two songs. In cross-fading mixing, two songs are mixed gradually by volume transition over time. This method can be used when two songs have a similar tempo and a related key (harmonic mixing). In the cutting mixing, a new song starts as soon as the current song ends. This method is used for mixing music when two songs have extremely different tempos. Figure 9 shows examples of song mixing using cross-fading and cutting mixing.

4 Results and discussion

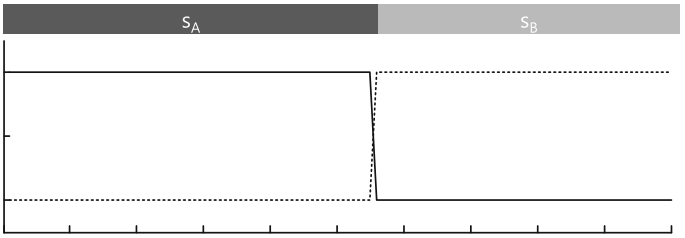
4.1 Tweet stream statistics

Before we show the performance of our scheme, we first investigate the number of tweets and the storage requirements to perform automatic music recommendations. As mentioned previously, we gather tweet streams from the Twitter server in real time, collect music information from twitters, and store the information in the music database. During one week, we collected 4.57 million tweets that had the hashtags listed in Table 1. The distribution of music-related hashtags is shown in Fig. 10. The most popular music-related hashtag is "#nowplaying" and the distribution of hashtags is well matched with the popularity orders in Table 1.

When we stored 4.57 million tweets in our Hadoop database, raw tweets occupied 587 MB. After filtering the tweets through regular expressions, the number of tweets was reduced to 1.56 million (34 %). By extracting and storing music-related information, the total storage requirement was reduced to 131 MB (22 %). Figure 11 shows storage consumptions and number of tweets after filtering and extraction.



(a) Example of cross-fading mixing



(b) Example of cutting mixing

Fig. 9 Examples of song mixing using cross-fading and cutting mixing

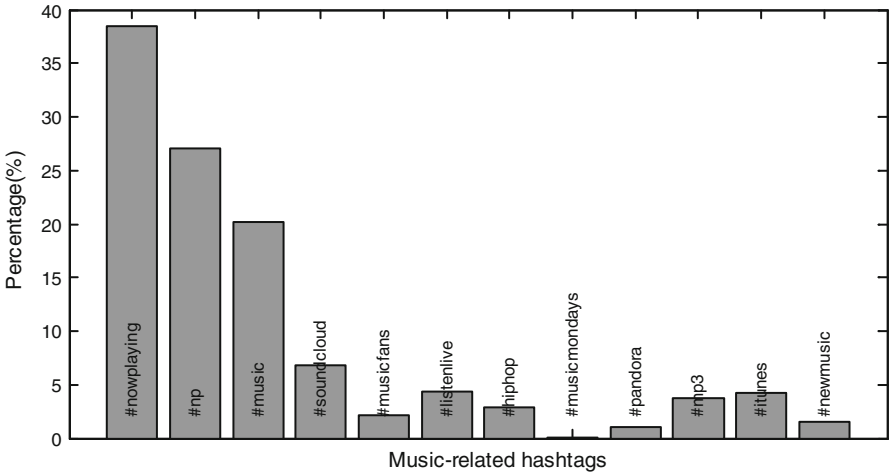


Fig. 10 Music-related hashtag distribution

4.2 Music information extraction accuracy

In this experiment, we evaluate whether our proposed method identifies tweets that have music information and extracts song titles and artist names accurately. After collecting 1,000 tweets, we classified the tweets manually depending on whether they contain music information, such as song title and artist name, and extracted the tweets that have music information. Table 3 lists the precision and recall of our identification

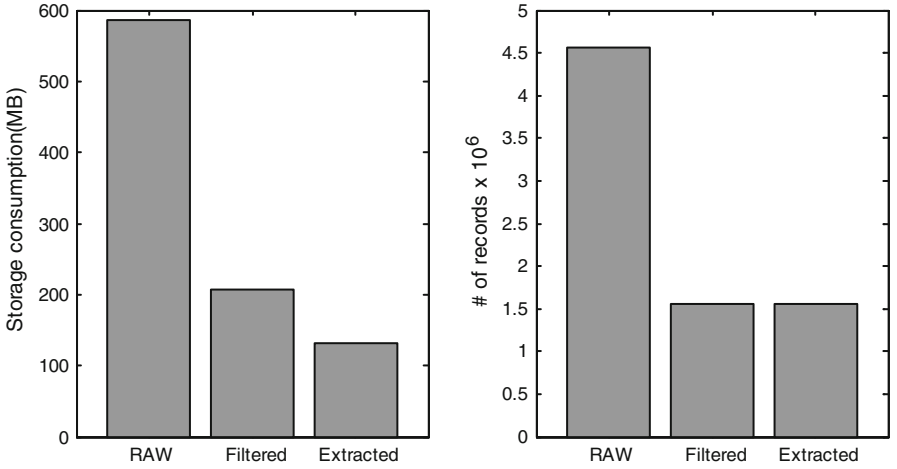


Fig. 11 Storage consumptions and number of tweets

Table 3 Accuracies of identifying and extracting music information

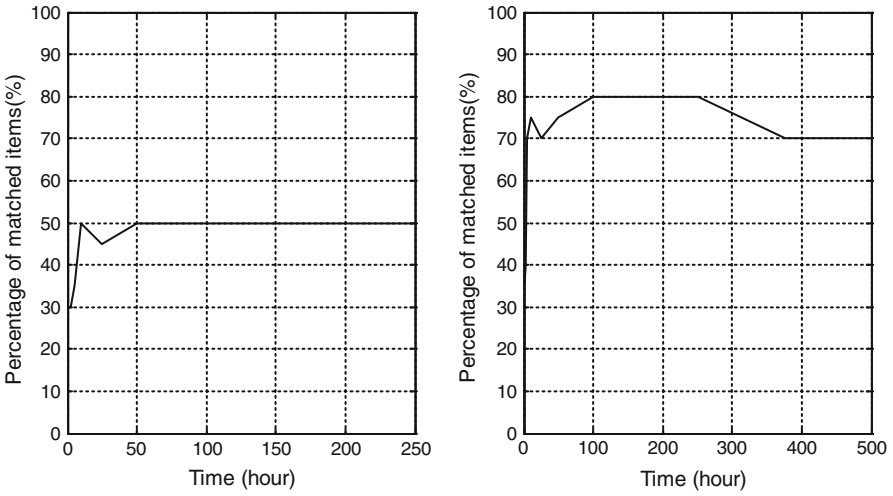
Identifying tweets containing song title and artist name	Precision	86.8 %
	Recall	96.8 %
Extracting correct song title and artist name	Regular expressions	89.4 %
	Regular expressions + Musicbrainz	96.8 %

method using pattern matching. We observed that tweets that contain many special characters or Twitter ID (@justinbieber) instead of artist name disturb the matching.

The extraction of music information for the identified tweets is based on regular expressions, which are used for matching the song title and artist name. As we described earlier, we adapt the Musicbrainz database to validate the result matched by the regular expressions. Accuracies of music information extraction with and without validation are listed in Table 3.

4.3 General recommendation accuracy

In this experiment, we evaluate the performance of our general music recommendation method by comparing the recommendation results with commercial music streaming charts. First, we collected daily popular US music charts from iTunes. For the US area, we calculated the general music recommendations included 20 bursty songs and 20 steady songs. We selected the top 20 songs that have the highest feature score. By changing the time window of bursty and steady recommendations, we show that the recommendation results correlate to the real-world chart. Figure 12a shows the percentage of matched songs between the chart and bursty songs. We observed that the songs whose rank increased in a week matched our bursty songs, and that they occupy



(a) Top 20 burst songs **(b)** Top 20 steady songs

Fig. 12 Percentage of matched songs and artist using bursty and steady

50 % of the bursty songs. Figure 12b shows the percentage of matched songs between the chart and the steady songs. Steady songs are matched differently depending on the time window. We observed that the music recommendation that used a 100–200 h time window best matched the chart. The experimental results show that songs in the music chart overlap songs with a high bursty or steady score.

4.4 Personalized recommendation

In this experiment, we evaluate the performance of our personalized recommendation method by comparing selected songs with seed songs. For simplicity, we assume that users listen to music of one particular genre. We consider four music genres: pop/rock, jazz, rap, and electronica as categorized in [30]. Seed songs are collected for each genre using the genre top tracks that are described in Last.fm [24]. For each genre, ten songs of the genre are used as seed songs. Our method returns 20 songs for recommendation. We evaluate the number of songs that have the same genre among the selected songs. Moreover, by varying the seed probability α, we show its effect on the recommendation performance in terms of diversity and relevance. The results are shown in Fig. 13.

A higher seed probability provides lower recommendation accuracy. We observed that the higher the seed probability, the more songs belong to different genres.

4.5 Scalability of tweet analysis and recommendation

In this experiment, we measured the time required to analyze tweets and the scalability of our proposed scheme. Because the proposed system retrieves tracks in real time, the overall retrieval process should be performed extremely fast. On average, 29,285 music-related tweets are collected per hour, and the number can be increased to 80,000 in some cases. To ensure real-time analysis and recommendation, we used a distributed

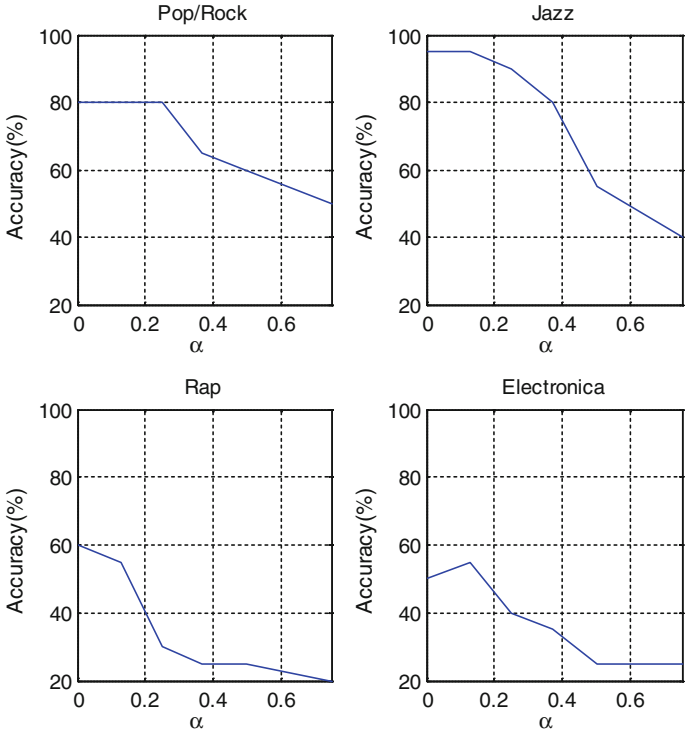


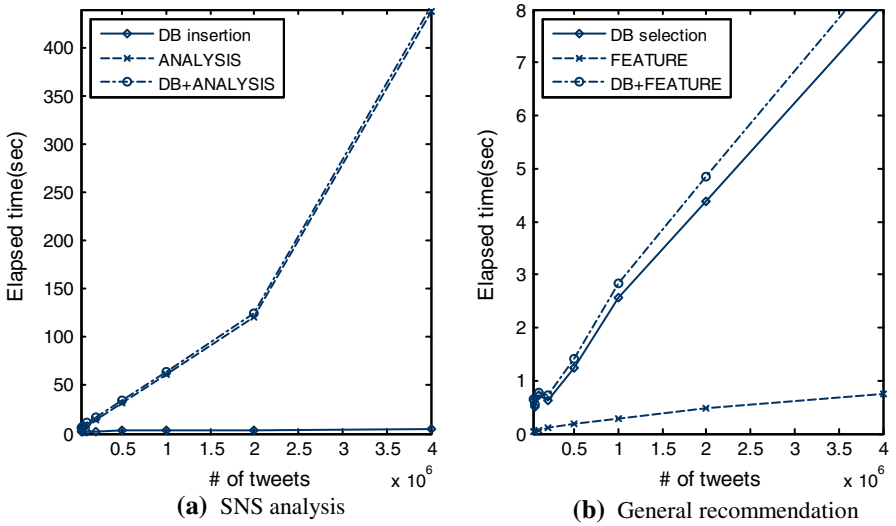
Fig. 13 Recommendation accuracies of four genres

database and query engine [19,20]. The average elapsed times for SNS analysis and general recommendation are shown in Fig. 14 and listed in Table 4.

As shown in the result, our proposed method has enough scalability in both analysis and recommendation in real time.

4.6 Comparison of music mixing methods

In this experiment, the quality of the mixing result (mixset) that uses the proposed scheme is evaluated by comparing the results of our mixset with two other mixing methods: manual mixing and random mixing. Especially, manual mixing was done by an amateur DJ. That is, we had an amateur DJ manually mix songs that were acquired for recommendation. For comparison, 20 subjects were asked to listen to three mixing results in random order and to indicate whether each mixing result is satisfactory. The comparison result is shown in Fig. 15. The comparison shows that mixing results with the proposed mixing method are more satisfactory than random mixing and almost equal to that of the manual mixing method.



(a) SNS analysis **(b)** General recommendation

Fig. 14 Elapsed times for SNS analysis and general recommendation

Table 4 Average elapsed times of SNS analysis and general recommendation

Average number of streamed tweets in an hour = 29,285 (100 %)		
Elapsed time for SNS analysis of n tweets		
n = 30,000	102 %	5.40 s
n = 50,000	171 %	7.53 s
n = 100,000	341 %	11.07 s
n = 200,000	683 %	17.08 s
n = 500,000	1,707 %	33.90 s
n = 1,000,000	3,415 %	63.97 s
n = 2,000,000	6,829 %	124.99 s
n = 4,000,000	13,659 %	442.91 s
Elapsed time for general recommendation of n tweets		
n = 30,000	102 %	0.65 s
n = 50,000	171 %	0.55 s
n = 100,000	341 %	0.77 s
n = 200,000	683 %	0.73 s
n = 500,000	1,707 %	1.42 s
n = 1,000,000	3,415 %	2.84 s
n = 2,000,000	6,829 %	4.84 s
n = 4,000,000	13,659 %	8.86 s

5 Conclusion

In this paper, we propose a scheme for generating automatic music recommendations as a mixed music clip for seamless playback. For automatic music recommendations,

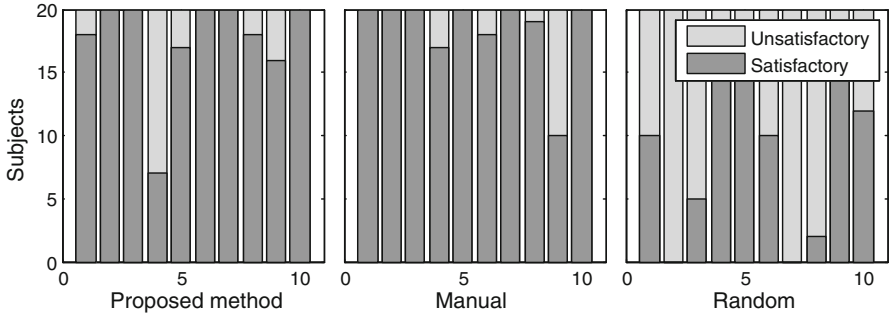


Fig. 15 Comparison of three mixing methods

we collected a significant amount of SNS documents and analyzed them to extract general and personalized musical predilections, music-related information such as title and artist, and other metadata such as time and location. Based on the analysis, a collection of songs is calculated for recommendation. These songs are, in turn, analyzed structurally and reordered based on their musical features for smooth mixing. Finally, reordered songs are blended together using two mixing methods. We implemented a prototype system and performed various experiments to measure the system's performance. We showed that our system can manage massive SNS data efficiently for music recommendation, and that the recommendations made are well matched with commercial music stream charts. We also showed that our system is well scalable with increasing data size. Finally, our final music mixing gained high satisfaction from users. Overall, our music recommendation and mixing schemes can offer new possibilities in the field of music.

Acknowledgments This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2013R1A1A2012627) and the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014-H0301-14-1001) supervised by the NIPA (National IT Industry Promotion Agency).

References

1. Mathioudakis M, Koudas N (2010) TwitterMonitor: trend detection over the Twitter stream. In: Proceedings of the 2010 ACM SIGMOD international conference on management of data, New York, USA, pp 1155–1158
2. Alvanaki F, Michel S, Ramamritham K, Weikum G (2012) See what's enBlogue: real-time emergent topic identification in social media. In: Proceedings of the 15th international conference on extending database technology, New York, USA, pp 336–347
3. Alvanaki F, Sebastian M, Ramamritham K, Weikum G (2011) EnBlogue: emergent topic detection in Web 2.0 streams. In: Proceedings of the 2011 ACM SIGMOD international conference on management of data, New York, USA, pp 1271–1274
4. Benhardus J, Kalita J (2013) Streaming trend detection in Twitter. Int J Web Based Communities 9(1):122–139
5. Kleinberg J (2002) Bursty and hierarchical structure in streams. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, New York, USA, pp 91–101

6. Kim D, Kim D, Jun S, Rho S, Hwang E (2013) TrendsSummary: a platform for retrieving and summarizing trendy multimedia contents. Multimed Tools Appl
7. Statistic Brain (http://www.statisticbrain.com/twitter-statistics)
8. Twitter developer documentation (https://dev.twitter.com/docs)
9. Yokomoto D, Makita K, Suzuki H, Koike D, Utsuro T, Kawada Y, Fukuhara T, (2012) LDA-based topic modeling in labeling blog posts with wikipedia entries. In: Proceedings of the 14th international conference on web technologies and applications, Berlin, Heidelberg, pp 114–124
10. Quercia D, Askham H, Crowcroft J (2012) TweetLDA: supervised topic classification and link prediction in Twitter. In: Proceedings of the 3rd annual ACM web science conference, New York, USA, pp 247–250
11. Platt J, Burges C, Swenson S, Weare C, Zheng A (2002) Learning a Gaussian process prior for automatically generating music playlists. Adv Neural Inf Process Syst 2:1425–1432
12. Pauws S, Eggen B (2002) PATS: realization and user evaluation of an automatic playlist generator. In: Proceedings of ISMIR, pp 222–230
13. Andric A, Haus G (2006) Automatic playlist generation based on tracking user's listening habits. Multimedia Tools Appl 29(2):127–151
14. Reynolds G, Barry D, Burke T, Coyle E (2007) Towards a personal automatic music playlist generation algorithm: the need for contextual information. In: Proceedings of 2nd conference on interaction with sound, Ilmenau, Germany, pp 84–89
15. Broughton F, Brewster B (2007) How to DJ right: the art and science of playing records. Grove Press, New York
16. Cliff D (2006) hpDJ: an automated DJ with floorshow feedback. In: O'Hara K, Brown B (eds) Consuming music together. Springer, Netherlands, pp 241–264
17. Cliff D (2000) Hang the DJ: automatic sequencing and seamless mixing of dance-music tracks. HP Labs technical report
18. Amazon EC2 (http://aws.amazon.com/en/ec2)
19. Cloudera impala (http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html)
20. Hadoop (http://hadoop.apache.org)
21. CDH (http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html)
22. Musicbrainz (http://musicbrainz.org)
23. Statweestics (http://statweestics.com)
24. Last.fm (http://www.last.fm)
25. World Atlas (http://www.worldatlas.com/cntycont.htm)
26. Jun S, Rho S, Hwang E (2013) Music structure analysis using self-similarity matrix and two-stage categorization. Multimedia Tools Appl
27. Fujishima T (1999) Realtime chord recognition of musical sound?: a system using common lisp music. Proc ICMC 1999:464–467
28. How-to guide from Harmonic-Mixing.com (http://www.harmonic-mixing.com/HowTo.aspx)
29. Grefenstette J, Gopal R, Rosmaita B, Van Gucht D (1985) Genetic algorithms for the traveling salesman problem. In: Proceedings of the first international conference on genetic algorithms and their applications, Lawrence Erlbaum, New Jersey (160–168)
30. AllMusic (http://www.allmusic.com)